OpenSTAAD

V8*i* (SELECTseries 4)

Reference Manual

DAA039740-1/0002
Last updated: 17 May 2012

Copyright Information

Trademark Notice

Copyright Notice

Acknowledgments

Restricted Rights Legends

End User License Agreement

To view the End User License Agreement for this product, review: **eula_en.pdf**.

Table of Contents

Section 1
Getting Started

1.1 Introduction

OpenSTAAD is a library of exposed functions allowing engineers access to STAAD.Pro's internal functions and routines as well as its graphical commands. With OpenSTAAD, you can use Visual Basic for Applications (VBA) macros to perform such tasks as automating repetitive modeling or post-processing tasks or embedding customized design routines. Following an open architecture paradigm, OpenSTAAD was built using **ATL**[1] **COM**[2] and COM+ standards as specified by Microsoft. This allows OpenSTAAD to be used in a macro application like Microsoft Excel or Autodesk AutoCAD. OpenSTAAD can also be used to link STAAD data to Web-based applications using ActiveX, **HTML**[3] and **ASP**[4].

OpenSTAAD allows engineers and other users to link in-house or third-party applications with STAAD.Pro. For example, a user might create a spreadsheet in Excel to analyze and design a circular base plate using support reactions from STAAD. With OpenSTAAD, a simple macro can be written in Excel or within the STAAD environment to retrieve the appropriate STAAD data and automatically link the results. If the STAAD file changes, so will the Excel sheet!

[1]Active Template Library
[2]Component Object Model
[3]Hyper Text Markup Language
[4]Active Server Pages

With a built-in VBA editor, macros can be written inside STAAD using VBA to create new dialog boxes or menu items which run design codes or specific structural components (like certain connections) that automatically link to STAAD's familiar reporting tables. A cumbersome export/import link between two or three software is not required.

1.2 Help and Documentation

In an effort to provide you with the best application support in the industry, OpenSTAAD documentation is provided electronically in the form of HTML help files. This important decision was made to provide a method of quickly updating users with the latest program additions or modifications. Since this information is provided electronically, users can simply download the latest help files from our web site, without the delay to update and reprint hard copy documentation.

1.2.1 HTML Help Files

In addition to new dynamic help features built into OpenSTAAD, help files are provided in Adobe® Acrobat PDF formats. To view the PDF files, you must have Adobe Acrobat Reader v.5.x or later installed and functioning. The online help is viewed externally of the program using MadCap Help Viewer v4.x or later, which is installed along with OpenSTAAD. You can download the most recent versions of these applications via the following urls:

- Adobe Acrobat Reader: **http://www.adobe.com/products/acrobat/**

These applications provide a **Help** > **Contents** menu selection, which will display the relevant help for that application. In most instances, help files for the primary application may be launched under that applications sub-menu in your Windows Start menu.

1.2.2 OpenSTAAD Help Organization

The OpenSTAAD documentation is organized in the following manner:

- New Features
- Getting Started
- Fundamentals
- OpenSTAAD Application Objects
- Troubleshooting
- Examples

1.2.3 Printing Help and Documentation Files

Topics in the OpenSTAAD help system may be printed when they are opened in the Help Viewer application. To print a topic:

1. Select the topic you wish to print from the Table of Contents.
2. Select **File** > **Print**.

or

Select **File** > **Print Preview** to review the output before printing.

Note: Help topics displayed in the context sensitive Help Window embedded within the application cannot be printed directly from that window.

PDF files may be printed by selecting **File** > **Print** from the PDF reader application, then selecting the range of pages to print.

1.3 Documentation Conventions

A number of typographical conventions are maintained throughout Bentley documentation, which makes it easier to identify and understand the information presented.

Notes, Hints, and Warnings

Items of special note are indicated as follows:

Note: This is an item of general importance.

Hint: This is optional time-saving information.

Warning: This is information about actions that should not be performed under normal operating conditions.

File Path/File Name.extension

A fixed width typeface is used to indicate file names, file paths, and file extensions (e.g., `C:/SProV8i/STAAD/Staadpro.exe`)

Interface Control

A bold typeface is used to indicate user controls. Menu and sub-menu items are indicated with a series of > characters to distinguish menu levels. (e.g., **File** > **Save As...**).

User Input

A bold, fixed width typeface is used to indicate information which must be manually entered. (e.g., Type **`DEAD LOAD`** as the title for Load Case 1).

STAAD Page Controls

A " | " character is used to represent the page control levels between pages and sub-pages. (e.g., Select the **Design** | **Steel** page).

1.3.1 Terminology

- Click - This refers to the action of pressing a mouse button. When not specified, click means to press the left mouse button.
- Select - Synonymous with Click. Used when referring to an action in a menu, drop-down list, list box, or other control where multiple options are available to you.
- pop-up menu - A pop-up menu is displayed typically with a right-click of the mouse on an item in the interface.
- Window - Describes an on screen element which may be manipulated independently. Multiple windows may be open and interacted with simultaneously.
- Dialog - This is an on screen element which (typically) must be interacted with before returning to the main window.

1.3.2 Mathematical Notation

Similar to spelling conventions, American mathematical notation is used throughout the documentation. A serif typeface is typically used to clarify numbers or letters which might otherwise appear similar.

- Numbers greater than 999 are written using a comma (,) to separate every three digits. For example, the U.S. value of Young's Modulus is taken as 29,000,000 psi.

  Warning: Do not use commas or spaces to separate digits within a number in a STAAD input file.

- Numbers with decimal fractions are written with a period to separate whole and fraction parts. For example, a beam with a length of 21.75 feet.
- Multiplication is represented with a raised, or middle, dot (·). For example, P = F·A.
- Operation separators are used in the following order:

  1. parenthesis ( )
  2. square brackets [ ]
  3. curly brackets (i.e., braces) { }

  For example, $F_a = [1 - (Kl/r)^2/(2 \cdot C_c^2)]F_y / \{5/3 + [3(Kl/r)/(8 \cdot C_c)] - [(Kl/r)^3/(8 \cdot C_c^3)]\}$

1.4 Function List Organization

The list of functions is formatted in the following way:

- The name of each function appears in large, bold typeface at the top of the page.
- A general description follows which contains the purpose of the function, and provides general comments, constraints, and recommendations for using it.
- The Visual Basic (VB) syntax for the function appears next in bold typeface. All function

syntax definitions consist of the function name and then the comma separated parameters, e.g.,

```
functionname param1, param2, … , paramn
```

The function and all its parameters are usually written on a single line, but for long lines of code, a line continuation character may be used to make the code easier to read. In VB, the line continuation character is a space followed by an underscore (_).

- Following the function syntax definition is a description of each parameter required by the function. Note that the names of the parameters are just example names. If the parameter is a variable name for storing results returned by the function, or if you are passing a parameter required by the function as a variable, you can give it any legal variable name.
- If the function directly returns a value (often a boolean value indicating if the function was successful or not), a section on interpreting the return value is included.
- Then an example of the function is provided. In some cases, the example is just a short code snippet; in other cases a fully viable macro or program is given. All code provided will be able to work within the STAAD macro editor or an external VBA editor (e.g., Microsoft Office® Excel®).
- Finally, a list of related functions are listed, where applicable.

Section 2
Fundamentals of OpenSTAAD

2.1 Programming Languages

Although *OpenSTAAD* supports all major programming languages used today capable of calling COM components, it is impractical to document the usage of each and every function in all of these languages. Most of the example programs or code snippets for each documented *OpenSTAAD* function are written in *VBA for Excel* or *AutoCAD VBA*. The reason is that with OpenSTAAD 2.0 and higher, STAAD.Pro is equipped with a functional VBA editor capable of writing macros only in VBA.

OpenSTAAD supports Visual Basic for Applications, which is an implementation of Microsoft's Visual Basic language and an associated **IDE**[1].

Documentation for VBA is beyond the scope of this manual and Bentley does not provide direct support on how to write VBA macros. There are, however, several useful and free sites on the Web to assist a beginner on writing macros in STAAD, Excel, AutoCAD, or any other VBA compliant software. It should be noted that the VBA language is the same from software to software. However, the functions, objects, and core libraries will obviously vary.

The sites recommended by Bentley are as follows:

[1]Integrated Development Environment

- **http://www.excel-vba.com**
- **http://www.beyondtechnology.com/vba.shtml**
- **http://www.afralisp.com**
- **http://www.wrox.com**

Also, it is worth noting that many third-party programs such as Microsoft Office Excel or Autodesk AutoCADhas a nice *Record Macro* feature. You can run the recorder and then select any commands from Excel's menus and toolbars. The corresponding VBA syntax will automatically be generated. Look under 'Recording Macros' in the Excel help for additional information.

For additional information on using macros and VBA in Excel or other Microsoft Office programs, Microsoft's MSDN is a recommended starting point:

- **http://msdn.microsoft.com/en-us/library/ee814735.aspx#Y2242**

2.2 OpenSTAAD User Forum and Code Resource

Bentley hosts an OpenSTAAD community on the company's Be Communities website. Here you can exchange macros with peers using the file upload, ask questions in the community forum, or share other information in the community blog and image gallery.

Join the Bentley **OpenSTAAD** group at Be Communities today.

Warning: Code or files uploaded to Be Communities are *not* subject to a screening process in advance. Any files or code which are found to result in malicious behavior will be removed. Bentley Systems accepts no responsibility for any mistake, error or misrepresentation in or as a result of the usage of software code obtained from the Be Communities site.

2.3 Application Program Interface (API)

The *OpenSTAAD* library of functions is classified under the following general categories:

- STAAD File Input and Output (I/O)
- Structure Geometry
- Member Specifications
- Properties
- Loads
- Output Results
  - Nodes
  - Beams
  - Plates
  - Solids
- STAAD Pre-Processor

- STAAD Post-Processor
- Creating Dialog Boxes and Menu Items

2.4 Instantiating the OpenSTAAD Library for Use

The first thing necessary to access STAAD project data from within another application is to instantiate, or create an instance of OpenSTAAD within the other application. In Visual Basic for Applications (VBA), this may be done by creating an object variable and then assigning to it the OpenSTAAD object. The VBA *GetObject* function may be used for this.

The object which controls the STAAD.Pro environment is referred to as *StaadPro.OpenSTAAD*. This object must be created in order to get access to any of the internal graphical functions within STAAD.Pro (including the creating of menu items and dialog boxes) as well as access to STAAD's viewing, geometry modeling, results grid, and post-processing functions. The following VBA function can be used to instantiate or create this object:

```
Set MyObject = GetObject("filepath", "ojbectclass")
```

Where:

MyObject
: the Object name declared in a previous *Dim* statement.

filepath
: the optional string providing the full file path and name of the file containing objects to retrieve. In the case of OpenSTAAD, this can be omitted, along with the trailing comma.

objectclass
: the string representing the class of the object. In the case of OpenSTAAD, this is always **Staadpro.OpenSTAAD**.

Thus:

```
Set objName = GetObject("StaadPro.OpenSTAAD")
```

At the conclusion of your OpenSTAAD application, the OpenSTAAD object(s) should be terminated, to unlock the handles within the application to the OpenSTAAD functions, and to free system resources.

2.4.1 Example

```
Sub How2Begin()
        'Create a variable to hold your OpenSTAAD object(s).
        Dim objOpenSTAAD As Object
        'Launch OpenSTAAD Object
        Set objOpenSTAAD = GetObject(, "StaadPro.OpenSTAAD")
        'At the end of your application, remember to terminate the OpenSTAAD
     objects.
        Set objOpenSTAAD = Nothing
End Sub
```

2.5 Function Return Value

Most OpenSTAAD functions return a value to either:

- indicate the success or failure of the function (a Boolean result), or
- results value of the function (a numeric value result).

If a function returns a Boolean result and that return value for an OpenSTAAD function is equal to `0` (zero), it means that OpenSTAAD was unable to execute the function. If you see this result, check that you have passed all the required parameters to the function. Make sure that all parameters being passed are valid. A return value of `1` (one) indicates that OpenSTAAD successfully executed the function.

VB Example

```
objOpenSTAAD.BooleanReturn param1, param2, … ,paramn
'or, if you would like to capture the Boolean result for flagging:
functionFlag = objOpenSTAAD.BooleanReturn param1, param2, … ,paramn
```

Unless specified otherwise, results returned by a function are stored in variable names passed to it for the purpose.

A few of the OpenSTAAD Application functions return the results as the return value of the function. In those cases, the above comments regarding the function return value do not apply.

If a function returns the value of the function,

```
Dim returnValue
returnValue = objOpenSTAAD.BooleanReturn (param1, param2, … ,paramn)
```

Hint: See "Functions and Subroutines" on page 12 for information on parenthesis required in VB Syntax.

2.6 STAAD Nomenclature

In STAAD documentation and in the program menus and dialog boxes, the terms “member” and “beam” are used interchangeably. Use of the term “beam” should not be taken to imply that the member cannot take an axial load.

Similarly, the terms “joint” and “node” are used interchangeably in STAAD. Both terms refer to the connections between elements.

Connections are also referred to as incidences. The terms “member incidences,” “plate incidences,” and “solid incidences” refer to the nodes that connect these elements to other elements and supports.

2.7 OpenSTAAD Compatibility with STAAD.Pro

For optimum performance, *OpenSTAAD* should be run in STAAD.Pro 2002 or higher. The

object which controls the STAAD.Pro graphical interface (StaadPro.OpenSTAAD - dialogs, modeling tools, etc.) is only available in STAAD.Pro 2003 or higher.

As software development proceeds, it is sometimes necessary to modify STAAD's database to accommodate new features or to maximize efficiency and increase processing speed. STAAD.Pro 2002 is the first STAAD release to include *OpenSTAAD*. All subsequent releases of *OpenSTAAD* will be backward-compatible with STAAD.Pro 2002.

It may be possible to run at least some of *OpenSTAAD*'s functions in earlier releases of STAAD.Pro or STAAD-III, but the results may be unpredictable, since *OpenSTAAD* did not exist at that time, and therefore compatibility with *OpenSTAAD* was not considered during the development of these earlier STAAD versions. Bentley Systems, Inc. does not provide any technical support for *OpenSTAAD* running in STAAD versions prior to STAAD.Pro 2002.

In terms of STAAD's input file, STAAD.Pro is backward-compatible with all previous STAAD releases. To use *OpenSTAAD* on a project that was created using an earlier version of STAAD.Pro or STAAD-III, you can open your old input file in STAAD.Pro 2002 and run the analysis. This action will create a new database in a format fully compatible with *OpenSTAAD*.

OpenSTAAD is compatible with any version of Microsoft Office® Excel® that supports VBA macros. *OpenSTAAD* is also compatible with Autodesk AutoCAD® 2000 (or higher) VBA.

With OpenSTAAD 2.0 and higher (available with STAAD.Pro 2003 and higher), a VBA editor is embedded within the STAAD.Pro environment. The version of VBA which the editor supports is not 100% VBA compatible. There are a few functions which can be supported in the Microsoft version of VBA which are not supported in the VBA editor/compiler which comes with STAAD.Pro. Every attempt is being made to support these functions. Currently, the functions which are not supported are not yet documented in this manual.

2.8 Visual Basic Conventions

2.8.1 Comments

In Visual Basic or Visual Basic for Applications, an apostrophe (') is used to denote a comment. Anything to the right of the apostrophe will be ignored by the program.

Hint: Throughout this documentation, these are displayed in the color green to make comments visually distinct from the remaining code.

2.8.2 Declaring Arrays

VB/VBA is flexible in the way it allows arrays to be declared. Most examples involving arrays in this reference manual will conform to the C++ zero indexing convention. In an array of 6 values, the positions in the array are referred to as 0-5. Therefore an array of 6 values would be declared as follows:

```
Dim pdArray(5) As Double
```

or

```
Dim pdArray(0 To 5) As Double
```

In VB, a six value array can also be declared as:

```
Dim pdArray(1 To 6) As Double
```

In doing so, however, we might find that our loops and other statements used to access the various positions in the array might not work correctly in C++.

2.8.3 Line Continuation Character

A long coding statement can be written on more than one line, to make the code easier to read. The VB line continuation character consists of a space followed by an underscore at the end of the line. The line of code beneath the continuation character will be handled as though it was written on the same line as the line continuation character.

2.8.4 Functions and Subroutines

Though functions and subroutines perform similar actions in any language, it is important to understand a key difference in order to make writing OpenSTAAD macros easier. A function can return a value back to the function or routine that called it. A subroutine, on the other hand, does not directly return a value. Instead, a subroutine must make use of the ByRef methodology internally. OpenSTAAD has been written so that you can use many of its functions to return more than one value via a single function. While this is a very powerful feature, it can lead to some confusion given how VB syntax uses parenthesis. In VB syntax, a subroutine does not use parenthesis to enclose any values which are passed to it. VB reserves their use for functions only.

Thus, in OpenSTAAD the subroutines which are used to return multiple values from the model input, analysis results, or design results are required to use the subroutine syntax.

Note: As this is a merely a result of VB syntax and not that these OpenSTAAD functions behave differently, the documentation simply refers them all as "functions."

Hint: Often, though, the data passed in such a function is in the form of an array, which *do* use parenthesis (without a space) to indicate their dimensions.

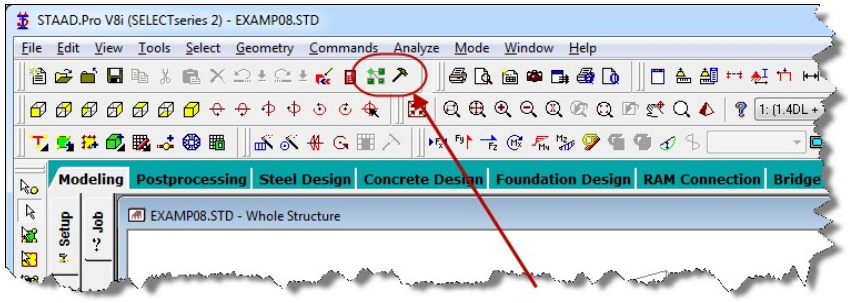
2.8.5 Example

```
objOpenSTAAD.GetNodeDisplacements  nNodeNo, nLC, pdDisps(0)
'The line of code above may also be written as the line shown below:
objOpenSTAAD.GetNodeDisplacements _
nNodeNo, nLC, pdDisps(0)
```

Section 3
Using Macros in STAAD.Pro

This section describes how to start the VBA editor and create a new macro or load an existing one within the STAAD.Pro environment. It also describes how to run the macros you have created in the STAAD.Pro macro editor by adding a simple menu item.

Macro related tools found in the File toolbox



3.1 Creating a new macro

1. Select either

   Edit > Create New VB Macro...

   or

the **Run VB Macro** tool  and then click **Create New** in the ***Macro* dialog**.

The ***Select New Macro File Name* dialog** opens.

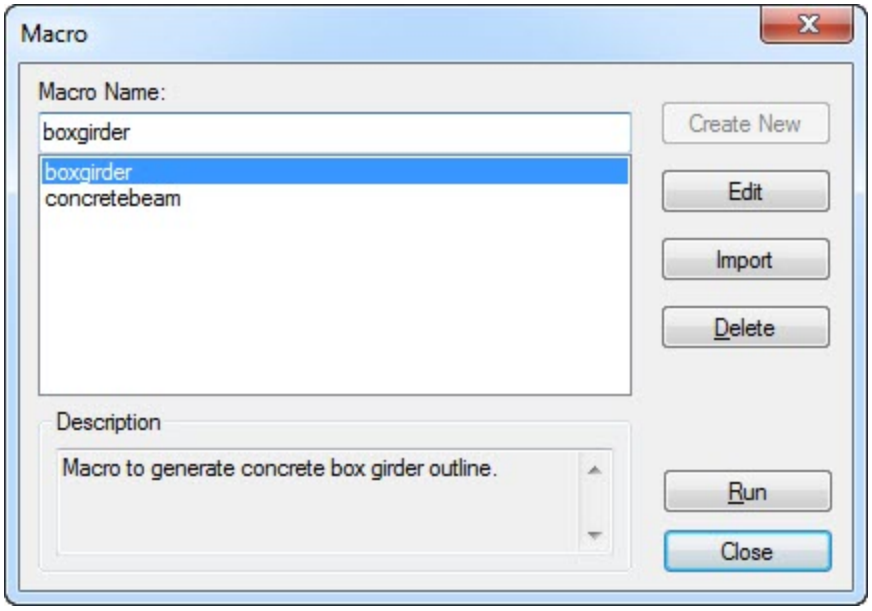
2. Navigate to the folder where you want to save the new macro file.
3. Specify a **File name**.
4. (Optional) In the **Save as type** drop-down menu, select to use a plain text macro (file extension `.vbs`) or a protected macro (file extension `.vbz`).
5. (Optional) Provide a brief description in the **Description of Macro** text box.
6. Click **New** to create the macro file.

   The macro design window opens with the empty VB macro open.

3.2 Macro dialog

Used to manage VB macros in STAAD.Pro.

Opens when the Run VB Macros tool is selected.



Macro name and list

The first box displays the name of the currently selected macro file, if one is selected. The list box displays a list of all linked macro files.

Description

Displays a brief description of the macro, if available.

Create New

Opens the Select New Macro File Name dialog, which is used to specify a file name, file

type, and description for an empty VB macro file. The Macro editor window then opens, which used to record or edit macro commands.

Active only when no macro is selected.

Edit

Opens the Macro editor window, which is used to record or edit macro commands.

Import

Opens the Add an existing Macro dialog, which is used to select macro for linking to STAAD.Pro.

Delete

Removes the macro from the linked macros list.

Note: The macro file is not deleted from the disk, only the link association in STAAD.Pro.

Run

Runs the selected macro.

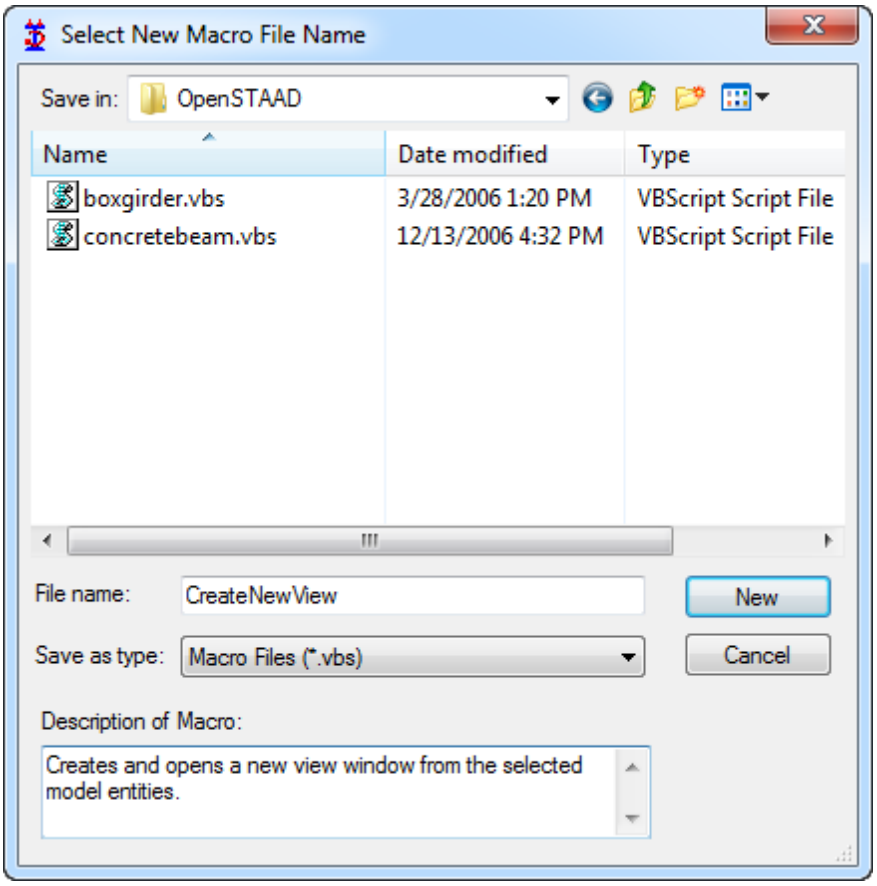
Close

Closes the dialog.

3.3 Select New Macro File Name dialog

Used to create a new VB macro file for use in STAAD.Pro.

Opens when either **Edit** > **Create New VB Macro...** is selected or **Create New** is clicked in the **Macro dialog**.



Save in

Select the drive and folder where the new macro file will be saved. Several common Windows folder navigation tools are provided.

File list

A list of existing macro files is displayed here.

File name

Specify the file name in this text box.

Save as type

VB macros can be saved in one of two file types, depending on the level of file protection you want.

- A VBS macro file is a standard macro file, the contents (code) of which can be viewed by other users in any standard text editor.
- A VBZ macro file is a protected macro file, the contents of which cannot be viewed even in an external editor like *Notepad*. This is useful when you want to sell the macro or simply protect its contents from unintentional editing as part of a quality control program.

Description

Used to add a brief description of the macros function. This is helpful in identifying similarly named macros

Create New

Creates the file with the specified information and opens the file in the STAAD.Pro Macro Editor window.

Cancel

Closes the dialog without creating a new macro file

3.4 Importing an existing macro

1. Select the **Run VB Macro** tool 

   The **Macro dialog** opens.

2. Click **Import**.

   The Add an Existing Macro dialog opens. The controls in this dialog are analogous to those in the Select New Macro File Name dialog, though some are inactive.

3. Navigate to where the existing macro file is located.

4. Select the macro file.

5. Click **Open**.

   The macro name is added to the list of names in the Macro dialog..

3.5 Running a linked macro

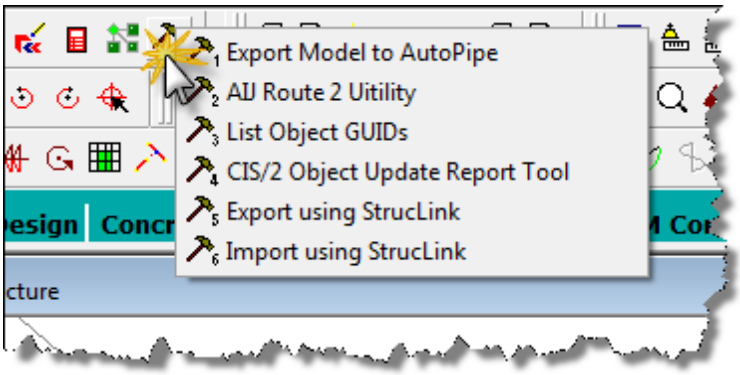
1. Select either

   the **User tools** tool in the File toolbar 

   or

   Tools > User Tools

   a drop-down menu displays all of the linked macros.



2. Select the macro you want to run from this list.

3.6 Displaying OpenSTAAD Functions in the Objects Browser

The objects browser is very useful to see an overview of functions while writing OpenSTAAD macros.

1. Open the SAX Basic editor.
2. Right click anywhere in the window and select Edit > References from the pop-up mnenu.

   The References dialog opens.
3. Set the option for OpenSTAADUI (1.0) in the Available References list and click OK.

Now, when the Browse Object tool is selected, you will see OpenSTAAD functions included in the Library list.

Section 4

OpenSTAAD Functions

This section contains an exhaustive reference for OpenSTAAD Application Object functions.

4.1 Root Functions

Analyze

This function analyzes the current .STD file.

VB Syntax

```
Analyze ()
```

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
Dim objOpenSTAAD As Object
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Run Analysis
objOpenSTAAD.Analyze ()
```

CloseSTAADFile

This function closes the currently open .STD file.

VB Syntax

```
CloseSTAADFile
```

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
Dim objOpenSTAAD As Object
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Close current STAAD file
objOpenSTAAD.CloseSTAADFile
```

CreateNamedView

This function creates a view with the specified name.

VB Syntax

```
CreateNamedView ViewName, FlagVal, ErrorVal
```

Where:

ViewName

A string variable that will hold the name of the view to be created.

FlagVal

A long variable that will hold the flag value depending upon which the view will be created.

ErrorVal

A long variable that will hold the error number if the view cannot be created.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
Dim objOpenSTAAD As Object
Dim strViewName As String
Dim lFlagVal as Long
Dim lErrorVal as Long
'Set the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
objOpenSTAAD.CreateNamedView strViewName, lFlagVal, lErrorVal
```

GetBaseUnit

Returns the base unit for the currently open .STD file.

VB Syntax

```
GetBaseUnit ()
```

Return Value

(Numeric value) Value will return 1 for English and 2 for Metric system of units.

VB Example

```
Dim objOpenSTAAD As Object
'Set the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
objOpenSTAAD.GetBaseUnit ()
```

GetInputUnitForForce

Retrieves the input unit of force of the currently open .STD file.

VB Syntax

GetInputUnitForForce *InputUnitForForce*

Where:

InputUnitForForce

A string variable that holds the input unit for force of the currently open .STD file. Later the value is internally converted to an integer ranging from 0 to 7

0- Kilopound

1- Pound

2- Kilogram

3- Metric Ton

4- Newton

5- KiloNewton

6- MegaNewton

7- DecaNewton

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
Dim objOpenSTAAD As Object
Dim strInputUnitForForce As String
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Force Unit
objOpenSTAAD.GetInputUnitForForce strInputUnitForForce
```

GetInputUnitForLength

Retrieves the input unit of length of the currently open .STD file.

VB Syntax

GetInputUnitForLength *InputUnitForLength*

Where:

InputUnitForLength

A string variable that will hold the input unit for length of the currently open .STD file. Later the value will be internally converted to an integer ranging from 0 to 7 (0-

Inch, 1- Feet, 2- Feet, 3- CentiMeter, 4- Meter, 5- MilliMeter, 6 - DeciMeter, 7 – KiloMeter).

VB Example

```
Dim objOpenSTAAD As Object
Dim strInputUnitForLength As String
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Length Unit
objOpenSTAAD.GetInputUnitForLength strInputUnitForLength
```

GetMainWindowHandle

This function retrives the main STAAD.Pro window handle.

VB Syntax

```
GetMainWindowHandle ()
```

VB Example

```
Dim objOpenSTAAD As Object
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Main Window Handle
objOpenSTAAD.GetMainWindowHandle
```

GetProcessHandle

This function retrives the current STAAD.Pro process handle.

VB Syntax

```
GetProcessHandle ()
```

VB Example

```
Dim objOpenSTAAD As Object
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Process Handle
objOpenSTAAD.GetProcessHandle
```

GetProcessId

This function retrives the current STAAD.Pro process ID.

VB Syntax

```
GetProcessId ()
```

VB Example

```
Dim objOpenSTAAD As Object
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Process ID
objOpenSTAAD.GetProcessId
```

GetShortJobInfo

Retrieves the short job information of the currently open .STD file.

VB Syntax

```
GetShortJobInfoJobName, JobClient, EngName
```

Where:

JobName

A string variable that will hold the Job Name for the current .STD file.

JobClient

A string variable that will hold the Job Client for the current .STD file.

EngName

A string variable that will hold the Engineer's Name for the current .STD file.

VB Example

```
Dim objOpenSTAAD As Object
Dim strJobName as String
Dim strJobClient as String
Dim strEnggName as String
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Job Info
objOpenSTAAD.GetShortJobInfo strJobName, strJobClient, strEngName
```

GetSTAADFile

Retrieves the name of the current .STD file.

VB Syntax

```
GetSTAADFile FileName, IncludePath
```

Where:

FileName

A string variable that will hold the name of the currently open .STD file (without the path name).

IncludePath

A Boolean variable which if true, will write the entire path name of the .STD file in the variable FileName.

VB Example

```
Dim objOpenSTAAD As Object
Dim strFileName As String
Dim bIncludePath As Boolean
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Retrieve the entire path
bIncludePath = true
objOpenSTAAD.GetSTAADFile strFileName, bIncludePath
```

GetSTAADFileFolder

Retrieves only the path of the current .STD file.

VB Syntax

GetSTAADFileFolder *FileFolder*

Where:

FileFolder

A string variable that will hold the path name of folder where the currently open .STD file resides. It will not write the name of the .STD file into the variable.

VB Example

```
Dim objOpenSTAAD As Object
Dim strFileFolder As String
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get the file folder
objOpenSTAAD.GetSTAADFileFolder strFileFolder
```

ModifyNamedView

This function modifies the named views of the currently open .STD file.

VB Syntax

```
ModifyNamedView ViewName, Entities, Array EntityArray, ArrayQualifier,
      ModifyFlag, ErrorVal
```

Where:

ViewName

A string variable that will hold the name of the view to be modified.

Entities

An long variable that will hold number of entities.

- EntityArray — An long that will hold entity number.
- ArrayQualifier — A integer variable that will hold entity qualifier value. Value may vary from 0 to 4(0 - Node, 1 - Beam, 2 - Plate, 3 - Solid, 4 – Surface)

ModifyFlag

A long variable that will hold the flag value depending upon which the view will be modified.

ErrorVal

A long variable that will hold the error number if the view cannot be modified.

VB Example

```
Dim objOpenSTAAD As Object
Dim strViewName As String
Dim intEntities As Integer
Dim lEntityNo as Long
Dim lEntityQualifier as Long
Dim lFlagVal as Long
Dim lErrorVal as Long
'Set the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
objOpenSTAAD. ModifyNamedView strViewName, intEntities, lEntityNo,
     lEntityQualifier, _
lFlagVal, lErrorVal
```

NewSTAADFile

This function creates a .STD file with specified length and force units.

VB Syntax

```
NewSTAADFile FileName, InputUnitForLength, InputUnitForForce
```

Where:

FileName

A string variable that will hold the name of the .STD file, which needs to be created.

InputUnitForLength

An integer variable that will hold the input unit to be assigned for length of the new .STD file. Value may vary from 0 to 7 (0- Inch, 1- Feet, 2- Feet, 3- CentiMeter, 4- Meter, 5- MilliMeter, 6 - DeciMeter, 7 – KiloMeter).

InputUnitForForce

An integer variable that will hold the input unit to be assigned for force of the new .STD file. Value may vary from 0 to 7 (0- Kilopound, 1- Pound, 2- Kilogram, 3-Metric Ton, 4- Newton, 5-Kilo Newton, 6- Mega Newton, 7- DecaNewton).

VB Example

```
Dim objOpenSTAAD As Object
Dim strFileName As String
Dim intInputUnitForLength As Integer
Dim intInputUnitForForce As Integer
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Create New File
objOpenSTAAD.NewSTAADFile strFileName, intInputUnitForLength,
      intInputUnitForForce
```

OpenSTAADFile

This function will open the specified .STD file.

VB Syntax

OpenSTAADFile *FileName*

Where:

FileName

A string variable that will hold the name of the .STD file, which needs to be open.

VB Example

```
Dim objOpenSTAAD As Object
Dim strFileName As String
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Open the file
objOpenSTAAD.OpenSTAADFile strFileName
```

Quit

This function quits STAAD.Pro application environment.

VB Syntax

```
Quit ()
```

VB Example

```
Dim objOpenSTAAD As Object
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Quit Application environment
objOpenSTAAD.Quit
```

RemoveNamedView

This function removes the current view with the specified name.

VB Syntax

```
RemoveNamedView ViewName, ErrorVal
```

Where:

ViewName

A string variable that will hold the name of the view to be removed.

ErrorVal

A long variable that will hold the error number if the view cannot be removed.

VB Example

```
Dim objOpenSTAAD As Object
Dim strViewName As String
Dim nErrorVal as Long
'Application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
objOpenSTAAD.RemoveNamedView strViewName, nErrorVal
```

SaveNamedView

This function saves the current view with the specified name.

VB Syntax

```
SaveNamedView ViewName, ErrorVal
```

Where:

ViewName

A string variable that will hold the name of the view to be saved.

ErrorVal

A long variable that will hold the error number if the view cannot be saved.

VB Example

```
Dim strViewName As String
Dim lErrorVal as Long
'Set the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
objOpenSTAAD.SaveNamedView strViewName, lErrorVal
```

SetInputUnitForForce

This function sets the input unit of force of the currently open .STD file.

VB Syntax

SetInputUnitForForce *InputUnitForForce*

Where:

InputUnitForForce

An integer variable that will hold the input unit to be assigned for force of the currently open .STD file. Value may vary from 0 to 7 (0- Kilopound, 1- Pound, 2- Kilogram, 3-Metric Ton, 4- Newton, 5-Kilo Newton, 6- Mega Newton, 7- DecaNewton).

VB Example

```
Dim objOpenSTAAD As Object
Dim intInputUnitForForce As Integer
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Set Force Unit
objOpenSTAAD.SetInputUnitForForce intInputUnitForForce
```

SetInputUnitForLength

This function sets the input unit of length of the currently open .STD file.

VB Syntax

SetInputUnitForLength *InputUnitForLength*

Where:

InputUnitForLength

An integer variable that will hold the input unit to be assigned for length of the currently open .STD file. Value may vary from 0 to 7 (0- Inch, 1- Feet, 2- Feet, 3- CentiMeter, 4- Meter, 5- MilliMeter, 6 - DeciMeter, 7 – KiloMeter).

VB Example

```
Dim objOpenSTAAD As Object
Dim intInputUnitForLength As Integer
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Set Length Unit
objOpenSTAAD.SetInputUnitForLength intInputUnitForLength
```

SetInputUnits

This function sets the input units of length and force of the currently open .STD file.

VB Syntax

SetInputUnits *InputUnitForLength, InputUnitForForce*

Where:

InputUnitForLength

An integer variable that will hold the input unit to be assigned for length of the currently open .STD file. Value may vary from 0 to 7 (0- Inch, 1- Feet, 2- Feet, 3- CentiMeter, 4- Meter, 5- MilliMeter, 6 - DeciMeter, 7 – KiloMeter).

InputUnitForForce

An integer variable that will hold the input unit to be assigned for force of the currently open .STD file. Value may vary from 0 to 7 (0- Kilopound, 1- Pound, 2- Kilogram, 3-Metric Ton, 4- Newton, 5-Kilo Newton, 6- Mega Newton, 7- DecaNewton).

VB Example

```
Dim objOpenSTAAD As Object
Dim intInputUnitForLength As Integer
Dim intInputUnitForForce As Integer
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Set Input Units
objOpenSTAAD.SetInputUnits intInputUnitForLength, intInputUnitForForce
```

SetShortJobInfo

This function sets the short job information of the currently open .STD file.

VB Syntax

SetShortJobInfo*JobName, JobClient, EngName*

Where:

JobName

A string variable that will hold the Job Name for the current .STD file.

JobClient

A string variable that will hold the Job Client for the current .STD file.

EngName

A string variable that will hold the Engineer's Name for the current .STD file.

VB Example

```
Dim objOpenSTAAD As Object
Dim strJobName as String
Dim strJobClient as String
Dim strEnggName as String
'Set the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
objOpenSTAAD.SetShortJobInfo strJobName, strJobClient, strEngName
```

ShowApplication

This function makes the STAAD.Pro application window active.

VB Syntax

ShowApplication ()

VB Example

```
Dim objOpenSTAAD As Object
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Show Application Window
objOpenSTAAD.ShowApplication
```

UpdateStructure

This function updates the current structure.

VB Syntax

UpdateStructure

VB Example

```
Dim objOpenSTAAD As Object
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Update structure
objOpenSTAAD.UpdateStructure
```

4.2 Geometry Functions

Geometry.AddBeam

This function adds a beam between two specified existing nodes.

VB Syntax

Geometry.AddBeam*NodeA, NodeB*

Where:

NodeA, NodeB

Long variables, provide member connectivity.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
Dim objOpenSTAAD As Object
Dim NodeA As Long
Dim NodeB As Long
'Get the application object --
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Add a beam connected between nodes 2 and 4
NodeA = 2
NodeB = 4
objOpenSTAAD.Geometry.AddBeam NodeA, NodeB
```

Geometry.AddMultipleBeams

This function adds multiple beams.

VB Syntax

Geometry.AddMultipleBeams (*naIncidences*)

Where:

naIncidences

Long array of m x 2 dimension containing member connectivity.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim Incidences(6, 1) As Long
'Get the application object --
 For I = 0 To 6
  Incidences (I, 0) = … 'ith node
Incidences (I, 1) = … 'jth node
 Next I
'Add multiple beams
 objOpenSTAAD.Geometry.AddMultipleBeams Incidences
```

Geometry.AddMultipleNodes

This function adds multiple nodes in the structure.

VB Syntax

Geometry.AddMultipleNodes *faCoordinates*

Where:

faCoordinates

Double array of m x 3 dimension containing X, Y and Z coordinates of nodes.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim Coordinates(6,2) As Double
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
 For I = 0 To 6
  Coordinates(I, 0) = … 'X coordinate
Coordinates(I, 1) = … 'Y coordinate
Coordinates(I, 2) = … 'Z coordinate
 Next I
'Add multiple nodes
 objOpenSTAAD.Geometry.AddMultipleNodes Coordinates
```

Geometry.AddMultiplePlates

This function adds multiple plate elements.

VB Syntax

Geometry.AddMultiplePlates *naIncidences*

Where:

naIncidences

Long array of m x 4 dimension containing plate element connectivity.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim Incidences(6, 3) As Long
'Get the application object --
 For I = 0 To 6
  Incidences (I, 0) = … 'ith node
Incidences (I, 1) = … 'jth node
Incidences (I, 2) = … 'kth node
Incidences (I, 3) = … 'lth node
 Next I
'Add multiple plates
 objOpenSTAAD.Geometry.AddMultiplePlates Incidences
```

Geometry.AddMultipleSolids

This function adds multiple solid elements.

VB Syntax

Geometry.AddMultipleSolids *naIncidences*

Where:

naIncidences

Long array of m x 8 dimension containing solid element connectivity.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim Incidences(6, 7) As Long
'Get the application object --
 For I = 0 To 6
 'Front face
  Incidences (I, 0) = … 'ith node
Incidences (I, 1) = … 'jth node
Incidences (I, 2) = … 'kth node
Incidences (I, 3) = … 'lth node
 'Back face
  Incidences (I, 4) = … 'mth node
  Incidences (I, 5) = … 'nth node
  Incidences (I, 6) = … 'oth node
  Incidences (I, 7) = … 'pth node
 Next I
'Add multiple solids
 objOpenSTAAD.Geometry.AddMultipleSolids Incidences
```

Geometry.AddNode

This function adds a node in the structure.

VB Syntax

```
Geometry.AddNodeCoordX, CoordY, CoordZ
```

Where:

CoordX, CoordY, CoordZ

Double variables providing the nodal coordinates X, Y and Z of the NodeNo.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
Dim objOpenSTAAD As Object
Dim CoordX As Double
Dim CoordY As Double
Dim CoordZ As Double
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
CoordX = 3.0
CoordY = 2.0
CoordZ = 3.0
'Add a node (3.0, 2.0, 3.0)
objOpenSTAAD.Geometry.AddNode CoordX, CoordY, CoordZ
```

Geometry.AddPlate

This function adds a plate element between existing nodes.

VB Syntax

```
Geometry.AddPlateNodeA, NodeB, NodeC, NodeD
Geometry.AddPlateNodeA, NodeB, NodeC
```

Where:

NodeA, NodeB, NodeC, NodeD

Long variables, provide element connectivity.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
Dim objOpenSTAAD As Object
Dim NodeA As Long
Dim NodeB As Long
Dim NodeC As Long
Dim NodeD As Long
'Get the application object --
'Add a plate connected between nodes 2, 4, 5 and 6
NodeA = 2
NodeB = 4
NodeC = 5
NodeD = 6
objOpenSTAAD.Geometry.AddPlate NodeA, NodeB, NodeC, NodeD
```

Geometry.AddSolid

This function adds a solid element between existing nodes.

VB Syntax

Geometry.AddSolid*NodeA, NodeB, NodeC, NodeD, NodeE, NodeF, NodeG, NodeH*
Geometry.AddSolid*NodeA, NodeB, NodeC, NodeD, NodeE, NodeF*

Where:

NodeA, NodeB, NodeC, NodeD, NodeE, NodeF, NodeG, NodeH

Long variables, provide solid element connectivity.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
Dim objOpenSTAAD As Object
Dim NodeA As Long
Dim NodeB As Long
Dim NodeC As Long
Dim NodeD As Long
Dim NodeE As Long
Dim NodeF As Long
Dim NodeG As Long
Dim NodeH As Long
'Get the application object --
'Add a solid connected between nodes 2, 4, 5, 6 and 9, 10, 11, 12
NodeA = 2
NodeB = 4
NodeC = 5
NodeD = 6
```

```
NodeE = 9
NodeF = 10
NodeG = 11
NodeH = 12
objOpenSTAAD.Geometry.AddSolid NodeA, NodeB, NodeC, NodeD _
NodeE, NodeF, NodeG, NodeH
```

Geometry.CreateBeam

Adds a beam in the structure between nodes *NodeA* and *NodeB* with the number specified in *nBeamNo*. The difference between **CreateBeam** and **AddBeam** is the former has an option to label the beam with any user-defined number.

VB Syntax

Geometry.CreateBeam*nBeamNo, NodeA, NodeB*

Where:

nBeamNo

A long variable containing the number to assign the newly created beam.

NodeA, NodeB

Long variables, provide member connectivity.

VB Example

```
Dim objOpenSTAAD As Object
Dim NodeA As Long
Dim NodeB As Long
'Get the application object --
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Add a beam connected between nodes 2 and 4. Call it beam # 77
NodeA = 2
NodeB = 4
objOpenSTAAD.Geometry.CreateBeam 77, NodeA, NodeB
```

Geometry.CreateGroup

Creates a group with specified name for the specified type.

VB Syntax

Geometry.CreateGroup *GroupType, GroupName*

Where:

GroupType

A long variable providing the group type.

1 = Nodes

2 = Beams/Members

3 = Plates

4 = Solids

5 = Geometry (Beams, Plates and Solids)

6 = Floor (Floor beams)

GroupName

A string variable providing the group name.

VB Example

```
Dim objOpenSTAAD As Object
Dim lGroupType As Long
Dim strGruopName As String
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Create a Group
objOpenSTAAD.Geometry.CreateGroup lGruopType, strGruopName
```

Geometry.CreateNode

This function adds a node in the structure with the number specified in nNodeNo. The difference between CreateNode and AddNode is the former has an option to label the node with any user-defined number.

VB Syntax

Geometry.CreateNode *nNodeNo, CoordX, CoordY, CoordZ*

Where:

nNodeNo

A long variable containing the number to assign the newly created node.

CoordX, CoordY, CoordZ

Double variables providing the nodal coordinates X, Y and Z of the nNodeNo.

VB Example

```
Dim objOpenSTAAD As Object
Dim CoordX As Double
Dim CoordY As Double
Dim CoordZ As Double
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
CoordX = 3.0
```

```
CoordY = 2.0
CoordZ = 3.0
'Add a node (3.0, 2.0, 3.0) and call it Node # 45
objOpenSTAAD.Geometry.CreateNode 45, CoordX, CoordY, CoordZ
```

Geometry.CreatePlate

This function adds a plate in the structure between existing nodes. The difference between **CreatePlate** and **AddPlate** is the former has an option to label the plate with any user-defined number.

VB Syntax

```
Geometry.CreatePlate nPlateNo, NodeA, NodeB, NodeC, NodeD
Geometry.CreatePlate nPlateNo, NodeA, NodeB, NodeC
```

Where:

nPlateNo

A long variable containing the number to assign the newly created plate.

NodeA, NodeB, NodeC, NodeD

Long variables, provide element connectivity. NodeD is not used for triangular (3-noded) elements.

VB Example

```
Dim objOpenSTAAD As Object
Dim NodeA As Long
Dim NodeB As Long
Dim NodeC As Long
Dim NodeD As Long
'Get the application object --
'Add a plate connected between nodes 2, 4, 5 and 6, Call it Plate # 22
NodeA = 2
NodeB = 4
NodeC = 5
NodeD = 6
objOpenSTAAD.Geometry.CreatePlate 22, NodeA, NodeB, NodeC, NodeD
```

Geometry.CreateSolid

This function adds a plate in the structure between existing nodes. The difference between CreateSolid and AddSolid is the former has an option to label the solid with any user-defined number.

VB Syntax

```
Geometry.CreateSolid nSolidNo, NodeA, NodeB, NodeC, NodeD, NodeE, NodeF, NodeG,
        NodeH
```

Geometry.CreateSolid *nSolidNo, NodeA, NodeB, NodeC, NodeD, NodeE, NodeF*

Where:

nSolidNo

A long variable containing the number to assign the newly created solid.

NodeA, NodeB, NodeC, NodeD, NodeE, NodeF, NodeG, NodeH

Long variables, provide solid element connectivity.

VB Example

```
Dim objOpenSTAAD As Object
Dim NodeA As Long
Dim NodeB As Long
Dim NodeC As Long
Dim NodeD As Long
Dim NodeE As Long
Dim NodeF As Long
Dim NodeG As Long
Dim NodeH As Long
'Get the application object --
'Add a solid connected between nodes 2, 4, 5, 6 and 9, 10, 11, 12. Call
     it Solid '# 99
NodeA = 2
NodeB = 4
NodeC = 5
NodeD = 6
NodeE = 9
NodeF = 10
NodeG = 11
NodeH = 12
objOpenSTAAD.Geometry.CreateSolid 99, NodeA, NodeB, NodeC, NodeD _
NodeE, NodeF, NodeG, NodeH
```

Geometry.DeleteBeam

Deletes a specified beam.

VB Syntax

Geometry.DeleteBeam *BeamNo*

Where:

BeamNo

A Long variable containing the beam number to be deleted from the structure.

VB Example

```
Dim objOpenSTAAD As Object
Dim BeamNo As Long
'Get the application object --
'Delete beam number 25
BeamNo = 25
objOpenSTAAD.Geometry.DeleteBeam BeamNo
```

Geometry.DeleteNode

Deletes a node specified by NodeNo.

VB Syntax

Geometry.DeleteNode *NodeNo*

Where:

NodeNo

A Long variable containing the node number to be deleted from the structure.

VB Example

```
Dim objOpenSTAAD As Object
Dim NodeNo As Long
'Get the application object --
'Delete node number 25
NodeNo = 25
objOpenSTAAD.Geometry.DeleteNode NodeNo
```

Geometry.DeletePlate

Deletes a specified plate element.

VB Syntax

Geometry.DeletePlate *PlateNo*

Where:

PlateNo

A long variable containing the plate number to be deleted from the structure.

VB Example

```
Dim objOpenSTAAD As Object
Dim PlateNo As Long
'Get the application object --
```

```
'Delete plate number 25
 PlateNo = 25
 objOpenSTAAD.Geometry.DeletePlate PlateNo
```

Geometry.DeleteSolid

Deletes a specified solid element.

VB Syntax

Geometry.DeleteSolid *SolidNo*

Where:

SolidNo

A long variable containing the solid number to be deleted from the structure.

Remarks

VB Example

```
 Dim objOpenSTAAD As Object
 Dim SolidNo As Long
'Get the application object --
'Delete solid number 25
 SolidNo = 25
 objOpenSTAAD.Geometry.DeleteSolid SolidNo
```

Geometry.GetBeamLength

Returns the length of the beam

VB Syntax

Geometry.GetBeamLength *BeamNo*

Where:

BeamNo

A long variable providing the beam member number for which the length is to be retrieved.

Return Value

A double variable containing the length of the beam.

VB Example

```
Dim objOpenSTAAD As Object
```

```
Dim BeamNo As Long
Dim BeamLen As Long
'Get the application object --
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get length of the beam 10
BeamNo = 10
BeamLen = objOpenSTAAD.Geometry.GetBeamLength BeamNo
```

Geometry.GetBeamList

This function returns the member list of the current STAAD file.

VB Syntax

Geometry.GetBeamList *BeamNumberArray*

Where:

BeamNumberArray

Long Array variable in which the beam numbers are returned.

VB Example

```
Dim objOpenSTAAD As Object
Dim lBeamCnt as Long
Dim BeamNumberArray() As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Beam Numbers
lBeamCnt = objOpenSTAAD.GetMemberCount
ReDim BeamNumberArray(0 to (lBeamCnt-1)) As Long
'Get Beam list
objOpenSTAAD.Geometry.GetBeamList BeamNumberArray
```

Geometry.GetLastBeamNo

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

VB Syntax

Geometry.GetLastBeamNo ()

Return Value

(Numeric value) The number of the highest beam number in the model as a long variable.

VB Example

```
Dim objOpenSTAAD As Object
Dim LastBeamNo As Long
```

```
'Get the application object --
'Get last beam no.
 LastBeamNo = objOpenSTAAD.Geometry.GetLastBeamNo ()
```

Geometry.GetLastNodeNo

Returns the node number of the last node in the model.

VB Syntax

```
Geometry.GetLastNodeNo ()
```

Return Value

(Numeric value) The number of the highest node number in the model as a long variable.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim LastNodeNo As Long
'Get the application object --
'Get last node no.
 LastNodeNo = objOpenSTAAD.Geometry.GetLastNodeNo ()
```

Geometry.GetLastPlateNo

Returns the member number of the last plate in the model.

VB Syntax

```
Geometry.GetLastPlateNo ()
```

Return Value

(Numeric value) The number of the highest plate number in the model as a long variable.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim LastPlateNo As Long
'Get the application object --
'Get last plate no.
 LastPlateNo = objOpenSTAAD.Geometry.GetLastPlateNo ()
```

Geometry.GetLastSolidNo

Returns the member number of the last solid in the model.

VB Syntax

```
Geometry.GetLastSolidNo ()
```

Return Value

(Numeric value) The number of the highest solid number in the model as a long variable.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim LastSolidNo As Long
'Get the application object --
'Get last solid no.
 LastSolidNo = objOpenSTAAD.Geometry.GetLastSolidNo ()
```

Geometry.GetMemberCount

Returns the number of members in the currently open STAAD file.

VB Syntax

```
Geometry.GetMemberCount ()
```

Return Value

(Numeric value) The total number of members in the model as a long variable.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim lMemberCount As Long
'Get the application object --
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Member Numbers
 lMemberCount = objOpenSTAAD.Geometry.GetMemberCount ()
```

Geometry.GetMemberIncidence

Returns the connecting joints of the specified member.

VB Syntax

```
Geometry.GetMemberIncidence MemberNo, NodeA, NodeB
```

Where:

MemberNo

A long variable providing the member number.

NodeA

Long variable in which the starting node number of the member is returned.

NodeB

Long variable in which the ending node number of the member is returned.

VB Example

```
Dim objOpenSTAAD As Object
Dim MemberNo As Long
Dim NodeA As Long
Dim NodeB As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get a Member
MemberNo = 5
objOpenSTAAD.Geometry.GetMemberIncidence MemberNo, NodeA, NodeB
```

Geometry.GetNodeCoordinates

Returns the coordinates of the specified node.

VB Syntax

Geometry.GetNodeCoordinates*NodeNo, CoordX, CoordY, CoordZ*

Where:

NodeNo

A long variable providing the node number.

CoordX, CoordY, CoordZ

Double variables in which the nodal coordinates X, Y, and Z of the NodeNo are returned.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
Dim objOpenSTAAD As Object
Dim NodeNo As Long
Dim CoordX As Double
Dim CoordY As Double
Dim CoordZ As Double
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
```

```
'Get a node
 NodeNo = 10
 objOpenSTAAD.Geometry.GetNodeCoordinates NodeNo, CoordX, CoordY, CoordZ
```

Geometry.GetNodeCount

This function returns the number of nodes in the currently open STAAD file.

VB Syntax

```
Geometry.GetNodeCount ()
```

Return Value

(Numeric value) The total number of nodes in the model as a long variable.

VB Example

```
Dim objOpenSTAAD As Object
Dim lNodeCount as Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Nodes Count
lNodeCount = objOpenSTAAD.Geometry.GetNodeCount ()
```

Geometry.GetNodeDistance

Returns the distance between the nodes as a double variable.

VB Syntax

```
Geometry.GetNodeDistance   NodeNoA, NodeNoB
```

Where:

NodeNoA, NodeNoB

Long variables providing the node numbers.

Return Value

(Numeric value) A double variable containing distance between the nodes.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim NodeNoA As Long
 Dim NodeNoB As Long
 Dim NodeDistance As Double
'Get the application object
'Get the distance between node 10 and 11
```

```
NodeNoA = 10
NodeNoB = 11
NodeDistance = objOpenSTAAD.Geometry.GetNodeDistance NodeNoA, NodeNoB
```

Geometry.GetNodeIncidence

Returns the coordinates of the specified node.

VB Syntax

Geometry.GetNodeIncidence *NodeNo, CoordX, CoordY, CoordZ*

Where:

NodeNo

A long variable providing the node number.

CoordX, CoordY, CoordZ

Double variables in which the nodal coordinates X, Y and Z of the NodeNo are returned.

VB Example

```
Dim objOpenSTAAD As Object
Dim NodeNo As Long
Dim CoordX As Double
Dim CoordY As Double
Dim CoordZ As Double
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get a node
NodeNo = 10
objOpenSTAAD.Geometry.GetNodeIncidence NodeNo, CoordX, CoordY, CoordZ
```

Geometry.GetNodeList

This function returns the node list of the current STAAD file.

VB Syntax

Geometry.GetNodeList *NodeNumberArray*

Where:

NodeNumberArray

Long Array variable in which the node numbers are returned.

VB Example

```
Dim objOpenSTAAD As Object
```

```
 Dim lNodeCnt as Long
 Dim NodeNumberArray() As Long
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Node Numbers
 lNodeCnt = objOpenSTAAD.Geometry.GetNodeCount
 ReDim NodeNumberArray(0 to (lNodeCnt-1)) As Long
'Get node list
 objOpenSTAAD.Geometry.GetNodeList (NodeNumberArray)
```

Geometry.GetNodeNumber

Returns the node number at specified coordinates

VB Syntax

Geometry.GetNodeNumber *CoordX, CoordY, CoordZ*

Where:

CoordX, CoordY, CoordZ

Double variables of the X, Y, and Z nodal coordinates for which the node number is returned.

Return Value

A long variable containing the number of the node.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim NodeNo As Long
 Dim CoordX As Double
 Dim CoordY As Double
 Dim CoordZ As Double
'Get the application object
'Get a node
 NodeNo = objOpenSTAAD.Geometry.GetNodeNumber CoordX, CoordY, CoordZ
```

Geometry.GetNoOfSelectedBeams

Returns the number of selected beams.

VB Syntax

Geometry.GetNoOfSelectedBeams ()

Return Value

A Long variable containing the number of beams selected.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim SelBeamsNo As Long
'Get the application object --
'Get no. of selected beams
 SelBeamsNo = objOpenSTAAD.Geometry.GetNoOfSelectedBeams
```

Geometry.GetNoOfSelectedNodes

Returns the number of selected nodes.

VB Syntax

```
Geometry.GetNoOfSelectedNodes ()
```

Return Value

A Long variable containing the number of nodes selected.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim SelNodesNo As Long
'Get the application object --
'Get no. of selected nodes
 SelNodesNo = objOpenSTAAD.Geometry.GetNoOfSelectedNodes
```

Geometry.GetNoOfSelectedPlates

Returns the number of selected plates.

VB Syntax

```
Geometry.GetNoOfSelectedPlates ()
```

Return Value

A Long variable containing the number of plates selected.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim SelPlatesNo As Long
'Get the application object --
'Get no. of selected plates
 SelPlatesNo = objOpenSTAAD.Geometry.GetNoOfSelectedPlates
```

Geometry.GetNoOfSelectedSolids

Returns the number of selected solids.

VB Syntax

```
Geometry.GetNoOfSelectedSolids ()
```

Return Value

A Long variable containing the number of solids selected.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim SelSolidsNo As Long
'Get the application object --
'Get no. of selected solids
 SelSolidsNo = objOpenSTAAD.Geometry.GetNoOfSelectedSolids
```

Geometry.GetPlateCount

This function returns the number of plates in the currently open STAAD file.

VB Syntax

```
Geometry.GetPlateCount ()
```

Return Value

(Numeric value) The total number of plates in the model as a long variable.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim lPlateCount as Long
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Plates Count
 lPlateCount = objOpenSTAAD.Geometry.GetPlateCount ()
```

Geometry.GetPlateIncidence

Returns the connecting joints of the specified plate.

VB Syntax

```
Geometry.GetPlateIncidence PlateNo, NodeA, NodeB, NodeC, NodeD
```

Where:

PlateNo

A long variable providing the plate number.

NodeA, NodeB, NodeC, NodeD

Long variables in which all the node numbers for the plate element are returned.

VB Example

```
Dim objOpenSTAAD As Object
Dim PlateNo As Long
Dim NodeA As Long
Dim NodeB As Long
Dim NodeC As Long
Dim NodeD As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get a Plate
PlateNo = 5
objOpenSTAAD.Geometry.GetPlateIncidence PlateNo, NodeA, NodeB, NodeC,
     NodeD
```

Geometry.GetPlateList

This function returns the plate list of the current STAAD file to an array.

VB Syntax

Geometry.GetPlateList *PlateNumberArray*

Where:

PlateNumberArray

Long Array variable in which the plate numbers are returned.

VB Example

```
Dim objOpenSTAAD As Object
Dim lPlateCnt as Long
Dim PlateNumberArray() As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
‘Get Plate Numbers
lPlateCnt = objOpenSTAAD.Geometry.GetPlateCount
ReDim PlateNumberArray(0 to (lPlateCnt-1)) As Long
'Get Plate list
objOpenSTAAD.Geometry.GetPlateList (PlateNumberArray)
```

Geometry.GetSelectedBeams

Returns the beam numbers selected in *SelBeamArray* variable. Call this function after GetNoOfSelectedBeams.

VB Syntax

Geometry.GetSelectedBeams *SelBeamArray, Sorted*

Parameter

SelBeamArray

A Long array variable to receive the selected beam numbers.

Sorted

Integer variable: 1 = return the selections in sorted order, 0 = to return in the order of selection.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim SelBeamsNo As Long
 Dim SelBeams() As Long
'Get the application object --
'Get no. of selected beams
 SelBeamsNo = objOpenSTAAD.Geometry.GetNoOfSelectedBeams
'Reallocate
 ReDim SelBeams(SelBeamsNo-1) As Long
 objOpenSTAAD.Geometry.GetSelectedBeams SelBeams 1
```

Geometry.GetSelectedNodes

Returns the node numbers selected in *SelNodeArray* variable. Call this function after GetNoOfSelectedNodes.

VB Syntax

Geometry.GetSelectedNodes *SelNodeArray, Sorted*

Where:

SelNodeArray

An Long array variable to receive the selected node numbers.

Sorted

Integer variable: 1 = return the selections in sorted order, 0 = to return in the order of selection.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim SelNodesNo As Long
 Dim SelNodes() As Long
'Get the application object --
'Get no. of selected nodes
 SelNodesNo = objOpenSTAAD.Geometry.GetNoOfSelectedNodes
'Reallocate
 ReDim SelNodes (SelNodesNo - 1) As Long
 objOpenSTAAD.Geometry.GetSelectedNodes SelNodes 1
```

Geometry.GetSelectedPlates

Returns the plate numbers selected in *SelPlateArray* variable. Call this function after GetNoOfSelectedPlates.

VB Syntax

Geometry.GetSelectedPlates *SelPlateArray, Sorted*

Parameter

SelPlateArray

A Long array variable to receive the selected plate numbers.

Sorted

Integer variable: 1 = return the selections in sorted order, 0 = to return in the order of selection.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim SelPlatesNo As Long
 Dim SelPlates() As Long
'Get the application object --
'Get no. of selected plates
 SelPlatesNo = objOpenSTAAD.Geometry.GetNoOfSelectedPlates
'Reallocate
 ReDim SelPlates(SelPlatesNo-1) As Long
 objOpenSTAAD.Geometry.GetSelectedPlates SelPlates 1
```

Geometry.GetSelectedSolids

Returns the solid numbers selected in *SelSolidArray* variable. Call this function after GetNoOfSelectedSolids.

VB Syntax

```
Geometry.GetSelectedSolids SelSolidArray, Sorted
```

Parameter

SelSolidArray

A Long array variable to receive the selected solid numbers.

Sorted

Integer variable: 1 = return the selections in sorted order, 0 = to return in the order of selection.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim SelSolidsNo As Long
 Dim SelSolids() As Long
'Get the application object --
'Get no. of selected solids
 SelSolidsNo = objOpenSTAAD.Geometry.GetNoOfSelectedSolids
'Reallocate
 ReDim SelSolids(SelSolidsNo-1) As Long
 objOpenSTAAD.Geometry.GetSelectedSolids SelSolids 1
```

Geometry.GetSolidCount

This function returns the number of solids in the currently open STAAD file.

VB Syntax

```
Geometry.GetSolidCount ()
```

Return Value

(Numeric value) The total number of solids in the model as a long variable.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim lSolidCount as Long
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Solids Count
 lSolidCount = objOpenSTAAD.Geometry.GetSolidCount ()
```

Geometry.GetSolidIncidence

Returns the connecting joints of the specified solid.

VB Syntax

Geometry.GetSolidIncidence*SolidNo, NodeA, NodeB, NodeC, NodeD, NodeE, NodeF, NodeG, NodeH*

Where:

SolidNo

A long variable providing the solid number.

NodeA, NodeB, NodeC, NodeD, NodeE, NodeF, NodeG, NodeH

Long variables in which all the node numbers for the solid element are returned.

VB Example

```
Dim objOpenSTAAD As Object
Dim SolidNo As Long
Dim NodeA As Long
Dim NodeB As Long
Dim NodeC As Long
Dim NodeD As Long
Dim NodeE As Long
Dim NodeF As Long
Dim NodeG As Long
Dim NodeH As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get a Solid
SolidNo = 5
objOpenSTAAD.Geometry.GetSolidIncidence SolidNo, NodeA, NodeB, NodeC,
     NodeD, NodeE, NodeF, NodeG, NodeH
```

Geometry.GetSolidList

This function returns the solid list of the current STAAD file.

VB Syntax

Geometry.GetSolidList *SolidNumberArray*

Where:

SolidNumberArray

Long Array variable in which the solid numbers are returned.

VB Example

```
Dim objOpenSTAAD As Object
Dim lSolidCnt as Long
```

```
 Dim SolidNumberArray() As Long
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Solid Numbers
 lSolidCnt = objOpenSTAAD.Geometry.GetSolidCount
 ReDim SolidNumberArray(0 to (lSolidCnt-1)) As Long
'Get Solid list
 objOpenSTAAD.Geometry.GetSolidList (SolidNumberArray)
```

Geometry.GetSurfaceCount

This function returns the number of surfaces in the currently open STAAD file.

VB Syntax

```
Geometry.GetSurfaceCount ()
```

VB Example

```
 Dim objOpenSTAAD As Object
 Dim lSurfaceCount as Long
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Surfaces Count
 lSurfaceCount = objOpenSTAAD.GetSurfaceCount
```

Geometry.GetSurfaceList

This function returns the surface list of the current STAAD file.

VB Syntax

```
Geometry.GetSurfaceList SurfaceNumberArray
```

Where:

SurfaceNumberArray

Long Array variable in which the surface numbers are returned.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim lSurfaceCnt as Long
 Dim SurfaceNumberArray() As Long
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Get Surface Numbers
 lSurfaceCnt = objOpenSTAAD.GetSurfaceCount
 ReDim SurfaceNumberArray(0 to (lSurfaceCnt-1)) As Long
'Get Surface list
```

```
objOpenSTAAD.Geometry.GetSurfaceList (SurfaceNumberArray)
```

Geometry.SelectBeam

This function selects a beam in the structure.

VB Syntax

Geometry.SelectBeam *BeamNo*

Where:

BeamNo

Long variable providing the beam number to be selected.

VB Example

```
Dim objOpenSTAAD As Object
Dim BeamNo As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Select beam no 2
BeamNo = 2
objOpenSTAAD.Geometry.SelectBeam(BeamNo)
```

Geometry.SelectMultipleBeams

This function selects multiple beams in the structure.

VB Syntax

Geometry.SelectMultipleBeams *aBeamNos*

Where:

aBeamNos

Long array variable providing the beam numbers to be selected.

VB Example

```
Dim objOpenSTAAD As Object
Dim BeamNos(6) As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Select beams no 1 to 7
For I = 0 To 6
 BeamNos(I) = I+1
Next I
objOpenSTAAD.Geometry.SelectMultipleBeams(BeamNos)
```

Geometry.SelectMultipleNodes

This function selects multiple nodes in the structure.

VB Syntax

Geometry.SelectMultipleNodes *aNodeNos*

Where:

aNodeNos

Long array variable providing the node numbers to be selected.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim NodeNos(6) As Long
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Select nodes no 1 to 7
 For I = 0 To 6
  NodeNos(I) = I+1
 Next I
 objOpenSTAAD.Geometry.SelectMultipleNodes(NodeNos)
```

Geometry.SelectMultiplePlates

This function selects multiple plates in the structure.

VB Syntax

Geometry.SelectMultiplePlates *aPlateNos*

Where:

aPlateNos

Long array variable providing the plate numbers to be selected.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim PlateNos(6) As Long
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Select Plates no 1 to 7
 For I = 0 To 6
  PlateNos(I) = I+1
 Next I
 objOpenSTAAD.Geometry.SelectMultiplePlates(PlateNos)
```

Geometry.SelectMultipleSolids

This function selects multiple solids in the structure.

VB Syntax

Geometry.SelectMultipleSolids *aSolidNos*

Where:

aSolidNos

Long array variable providing the solid numbers to be selected.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim SolidNos(6) As Long
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Select Solids no 1 to 7
 For I = 0 To 6
  SolidNos(I) = I+1
 Next I
 objOpenSTAAD.Geometry.SelectMultipleSolids(SolidNos)
```

Geometry.SelectNode

This function selects a node in the structure.

VB Syntax

Geometry.SelectNode *NodeNo*

Where:

NodeNo

Long variable providing the node number to be selected.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim NodeNo As Long
'Get the application object
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Select node no 2
 NodeNo = 2
 objOpenSTAAD.Geometry.SelectNode(NodeNo)
```

Geometry.SelectPlate

This function selects a plate in the structure.

VB Syntax

```
Geometry.SelectPlate PlateNo
```

Where:

PlateNo

Long variable providing the plate number to be selected.

VB Example

```
Dim objOpenSTAAD As Object
Dim PlateNo As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Select plate no 2
PlateNo = 2
objOpenSTAAD.Geometry.SelectPlate(PlateNo)
```

Geometry.SelectSolid

This function selects a solid in the structure.

VB Syntax

```
Geometry.SelectSolid SolidNo
```

Where:

SolidNo

Long variable providing the solid number to be selected.

VB Example

```
Dim objOpenSTAAD As Object
Dim SolidNo As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Select solid no 2
SolidNo = 2
objOpenSTAAD.Geometry.SelectSolid(SolidNo)
```

Geometry.SplitBeam

This function splits a beam.

VB Syntax

```
Geometry.SplitBeam BeamNo, Nodes, DistToNodeArray
```

Where:

BeamNo

A long variable providing the beam member number to split.

Nodes

A long variable providing the number of nodes to be inserted in the beam.

DistToNodeArray

Double array variable containing the distance in length to the nodes.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim BeamNo As Long
 Dim Nodes As Long
 Dim DistToNode(4) As Double
'Get the application object --
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Split Beam no 10 (length 5m say) into three unequal parts
 BeamNo = 10
 Nodes = 2
 DistToNode(0) = 1.0
 DistToNode(1) = 4.0
 objOpenSTAAD.Geometry.SplitBeam BeamNo, Nodes, DistToNode
```

Geometry.SplitBeamInEqlParts

This function splits a beam into equal parts.

VB Syntax

Geometry.SplitBeamInEqlParts *BeamNo, Parts*

Where:

BeamNo

A long variable providing the beam member number to split.

Parts

Long variable providing the number of parts into which the beam is to be split.

VB Example

```
 Dim objOpenSTAAD As Object
 Dim BeamNo As Long
 Dim Parts As Long
'Get the application object --
 Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Split Beam no 10 (length 5m say) into three equal parts
```

```
BeamNo = 10
Parts = 3
objOpenSTAAD.Geometry.SplitBeamInEqlParts BeamNo, Parts
```

4.3 View Functions

View.CreateNewViewForSelections

Display a new view for the selected objects.

VB Syntax

```
View.CreateNewViewForSelections
```

VB Example

```
'Get the application object --
'Display a new view for selected objects
objOpenSTAAD.View.CreateNewViewForSelections
```

View.GetInterfaceMode

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

VB Syntax

```
View.GetInterfaceMode
```

Return

Returns an integer value as per the following:

- 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

VB Example

```
Dim bMode As Integer
'Get the application object --
bMode = staad.View.GetInterfaceMode()
```

View.HideAllMembers

Hides all the members in the current structure.

VB Syntax

```
View.HideAllMembers ()
```

VB Example

```
Dim objOpenSTAAD As Object
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Hide Member
objOpenSTAAD.View.HideAllMembers
```

View.HideEntity

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

VB Syntax

View.HideEntity*EntityNo*

Where:

EntityNo

A long variable that holds an entity (i.e., Member, Plates etc.) number to be hidden.

VB Example

```
Dim objOpenSTAAD As Object
Dim lEntityNo as Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Variables
lEntityNo = 12
'Hide Member
objOpenSTAAD.View.HideEntity (lEntityNo)
```

View.HideMember

Hide the specified member.

VB Syntax

View.HideMember *MemberNo*

Where:

MemberNo

A long variable that holds member number to be hidden.

VB Example

```
Dim objOpenSTAAD As Object
Dim lMemberNo as Long
```

```
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Variables
lMemberNo = 5

'Hide Member

objOpenSTAAD.View.HideMember (lMemberNo)
```

View.HideMembers

Hide the specified members.

VB Syntax

View.HideMembers *MemberNumber, MemberNoArray*

Where:

MemberNumber

A long variable that holds number of members to hide.

MemberNoArray

Long array variable holds the member nos, which need to be hidden.

VB Example

```
Dim objOpenSTAAD As Object
Dim lMemberNumber as Long
Dim MemberNoArray() As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Variables
lMemberNumber = 5
ReDim MemberNoArray(0 to 4) As Long
MemberNoArray(0) = 1
MemberNoArray(1) = 2
MemberNoArray(2) = 5
MemberNoArray(3) = 7
MemberNoArray(4) = 10
'Hide Member
objOpenSTAAD.View.HideMembers (lMemberNumber, MemberNoArray)
```

View.HidePlate

Hide the specified plate.

VB Syntax

View.HidePlate *PlateNo*

Where:

PlateNo

A long variable that holds plate number to be hidden.

VB Example

```
Dim objOpenSTAAD As Object
Dim lPlateNo as Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Variables
lPlateNo = 102

'Hide Plate

objOpenSTAAD.View.HidePlate (lPlateNo)
```

View.HideSolid

Hide the specified solid.

VB Syntax

View.HideSolid *SolidNo*

Where:

SolidNo

A long variable that holds solid number to be hidden.

VB Example

```
Dim objOpenSTAAD As Object
Dim lSolidNo as Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Variables
lSolidNo = 35

'Hide Solid

objOpenSTAAD.View.HideSolid (lSolidNo)
```

View.HideSurface

Hide the specified surface.

VB Syntax

View.HideSurface *SurfaceNo*

Where:

SurfaceNo

A long variable that holds surface number to be hidden.

VB Example

```
Dim objOpenSTAAD As Object
Dim lSurfaceNo as Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Variables
lSurfaceNo = 1005

'Hide Surface

objOpenSTAAD.View.HideSurface (lSurfaceNo)
```

View.RefreshView

Refreshes the viewing window.

VB Syntax

```
View.RefreshView ()
```

VB Example

```
'Get the application object --
'Refresh viewing area of STAAD.Pro Window
objOpenSTAAD.View.RefreshView()
```

View.RotateDown

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

VB Syntax

```
View.RotateDown Degrees
```

Where:

Degrees

Double variable providing the degree of rotation.

VB Example

```
'Get the application object --
'Rotate the structure about X Axis through 30 degrees
```

```
objOpenSTAAD.View.RotateDown 30.0
```

View.RotateLeft

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

VB Syntax

```
View.RotateLeft Degrees
```

Where:

Degrees

Double variable providing the degree of rotation.

VB Example

```
'Get the application object --
'Rotate the structure about Y Axis through 30 degrees
objOpenSTAAD.View.RotateLeft 30.0
```

View.RotateRight

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

VB Syntax

```
View.RotateRight Degrees
```

Where:

Degrees

Double variable providing the degree of rotation.

VB Example

```
'Get the application object --
'Rotate the structure about Y Axis through 30 degrees
objOpenSTAAD.View.RotateRight 30.0
```

View.RotateUp

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

VB Syntax

```
View.RotateUp Degrees
```

Where:

Degrees

Double variable providing the degree of rotation.

VB Example

```
'Get the application object --
'Rotate the structure about X Axis through 30 degrees
objOpenSTAAD.View.RotateUp 30.0
```

View.SelectByItemList

Select entities as specified.

VB Syntax

View.SelectByItemList *EntityType, ItemNumber, ItemListArray*

Where:

EntityType

An integer variable that holds entity type. Values may be as follows:

1 = Node

2 = Beam

3 = Plate

4 = Solid

5 = Surface

ItemNumber

A long variable that holds total number of entities needs to be selected.

ItemListArray

Long array variable holds the entity nos, which need to be selected.

VB Example

```
Dim objOpenSTAAD As Object
Dim intEntityType as Integer
Dim lItemNumber as Long
Dim lItemListArray() as Array
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Variables
lItemNumber = 3
ReDim lItemListArray(0 to 2) As Long
lItemListArray(0) = 1
lItemListArray(1) = 2
lItemListArray(2) = 5
```

```
'Select by list
objOpenSTAAD.View.SelectByItemList (intEntityType, lItemNumber,
      lItemListArray)
```

View.SelectByMissingAttribute

Select entity list for which specified entity is missing.

VB Syntax

View.SelectByMissingAttribute *AttributeCode*

Where:

AttributeCode

An integer variable that holds attribute type. Values may be as follows:

1 = Missing Property

2 = Missing Modulus of Elasticity

3 = Missing Density of Material

4 = Missing Alpha (Coefficient of Thermal Expansion)

5 = Missing Poisson Ratio

VB Example

```
Dim objOpenSTAAD As Object
Dim intAttributeCode as Integer
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Select by Missing Attribute
objOpenSTAAD.View.SelectByMissingAttribute (intAttributeCode)
```

View.SelectEntitiesConnectedToMember

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

VB Syntax

View.SelectEntitiesConnectedToMember *EntityType, MemberNo*

Where:

EntityType

An integer variable that holds entity type. Values may be as follows:

1 = Member

2 = Beam

3 = Plate

4 = Solid

5 = Surface

MemberNo

A long variable that holds Member numbers with which connected entities needs to be selected.

VB Example

```
Dim objOpenSTAAD As Object
Dim intEntityType as Integer
Dim lMemberNo as Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Entities Connected to the Member
objOpenSTAAD.View.SelectEntitiesConnectedToMember (intEntityType,
      lMemberNo)
```

View.SelectEntitiesConnectedToNode

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

VB Syntax

View.SelectEntitiesConnectedToNode *EntityType, NodeNo*

Where:

EntityType

An integer variable that holds entity type. Values may be as follows:

1 = Node

2 = Beam

3 = Plate

4 = Solid

5 = Surface

NodeNo

A long variable that holds node numbers with which connected entities needs to be selected.

VB Example

```
Dim objOpenSTAAD As Object
```

```
Dim intEntityType as Integer
Dim lNodeNo as Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Entities Connected to the Node
objOpenSTAAD.View.SelectEntitiesConnectedToNode (intEntityType, lNodeNo)
```

View.SelectEntitiesConnectedToPlate

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

VB Syntax

View.SelectEntitiesConnectedToPlate *EntityType, PlateNo*

Where:

EntityType

An integer variable that holds entity type. Values may be as follows:

1 = Plate

2 = Beam

3 = Plate

4 = Solid

5 = Surface

PlateNo

A long variable that holds Plate numbers with which connected entities needs to be selected.

VB Example

```
Dim objOpenSTAAD As Object
Dim intEntityType as Integer
Dim lPlateNo as Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Entities Connected to the Plate
objOpenSTAAD.View.SelectEntitiesConnectedToPlate (intEntityType,
      lPlateNo)
```

View.SelectEntitiesConnectedToSolid

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

VB Syntax

View.SelectEntitiesConnectedToSolid *EntityType, SolidNo*

Where:

EntityType

An integer variable that holds entity type. Values may be as follows:

1 = Solid

2 = Beam

3 = Plate

4 = Solid

5 = Surface

SolidNo

A long variable that holds Solid numbers with which connected entities needs to be selected.

VB Example

```
Dim objOpenSTAAD As Object
Dim intEntityType as Integer
Dim lSolidNo as Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Entities Connected to the Solid
objOpenSTAAD.View.SelectEntitiesConnectedToSolid (intEntityType,
      lSolidNo)
```

View.SelectGroup

Select the relevant entities of the specified group.

VB Syntax

View.SelectGroup *GroupName*

Where:

GroupName

A string variable that holds the group name.

VB Example

```
Dim objOpenSTAAD As Object
Dim strGroupName as String
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Select Group
```

```
objOpenSTAAD.View.SelectGroup (strGroupName)
```

View.SelectInverse

Inverse geometry selection for the specified entity.

VB Syntax

View.SelectInverse *EntityType*

Where:

EntityType

An integer variable that holds entity type. Values may be as follows:

1 = Node

2 = Beam

3 = Plate

4 = Solid

5 = Surface

VB Example

```
Dim objOpenSTAAD As Object
Dim intEntityType as Integer
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Inverse Selection
objOpenSTAAD.View.SelectInverse (intEntityType)
```

View.SelectMembersParallelTo

Select members parallel to the specified axis.

VB Syntax

View.SelectMembersParallelTo *AxisID*

Where:

AxisID

A string variable that holds the Axis ID. It may have three values:

X = X-Axis

Y = Y-Axis

Z = Z-Axis

VB Example

```
Dim objOpenSTAAD As Object
Dim strAxis as String
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
strAxis = "X"
'Select Members
objOpenSTAAD.View.SelectMembersParallelTo (strAxis)
```

View.SetBeamAnnotationMode

Annotates the requested values for beam results in the appropriate view. This function works only in the post-processing mode of STAAD.Pro.

VB Syntax

View.SetNodeAnnotationMode *annotationType, position, bRefresh*

Where:

annotationType

Integer variable controlling the annotation type. It may be one of the following values:

0: Axial Diagram

1: Torsion Diagram

2: Moment Diagram

3: Shear Diagram

4: Stress Diagram

5: Displacement Diagram

position

Integer variable controlling what values are to be shown for the *annotationType*. It may be one of the following values:

1: End Values

2: Max Absolute Values

4: Mid-span Values

bRefresh

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

VB Example

```
Dim bRefresh As Boolean
Dim dLabel As Long
Dim dValueLoc As Long
Dim bRet As Boolean
'Get the application object --
'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
bRet = staad.View.SetInterfaceMode(spNATypeDX)>
If bRet Then
  bRefresh = True
  dLabel = 2 ' Moment diagram
  dValueLoc = 1 ' End values
  staad.View.SetBeamAnotationMode(dLabel, dValueLoc , bRefresh)
End If
```

View.SetDiagramMode

Sets the label on the structure diagram on or off.

VB Syntax

View.SetDiagramMode *diagramID, ShowFlag, refreshFlag*

Where:

diagramID

Integer variable identifying the diagram type. It may be one of the following values:

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

ShowFlag

Boolean variable to set label mode on (True) or off (False).

refreshFlag

Boolean variable to .

VB Example

```
'Get the application object --
'Turn on the MZ diagram
objOpenSTAAD.View.SetDiagramMode 1, True
```

View.SetInterfaceMode

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

VB Syntax

View.SetInterfaceMode *mode*

Where:

mode

An integer variable to set the current visual mode in STAAD.Pro environment. Followings are the valid values for 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

Return

Returns true or false (boolean) value, signifying the success or failure of the call.

VB Example

```
Dim bRefresh As Boolean
Dim dLabel As Integer
Dim bRet As Boolean
'Get the application object --
'Make sure that Staad.Pro is in Postprocessing mode
bRet = staad.View.SetInterfaceMode 1
'Annotate the view with X displacement labels
If bRet Then
  bRefresh = True
  dLabel = 2 ' Y Reaction
  staad.View.SetReactionAnotationMode dLabel bRefresh
End If
```

View.SetLabel

Sets the label on the structure diagram on or off.

VB Syntax

View.SetLabel *LabelID, ShowFlag*

Where:

LabelID

Integer variable identifying the label type. It may be one of the following values:

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

ShowFlag

Boolean variable to set label mode on (True) or off (False).

VB Example

```
'Get the application object --
'Label the member numbers
objOpenSTAAD.View.SetLabel(1,True)
```

View.SetModeSectionPage

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

VB Syntax

View.SetModeSectionPage *modeSection, modeMainPage, modeSubPage*

Where:

modeSection

An integer variable to set the current visual mode in the STAAD.Pro environment. Valid values for *modeSection* are the following:

0. Pre-processor or modeling mode
1. Post-processing mode
2. Interactive design mode for STAAD.etc interoperability
4. Piping mode
5. Beava mode

modeMainPage

An integer 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 *modeMainPage*:

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

modeSubPage

An integer 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*:

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

Return Value

Returns true or false (boolean) value, signifying the success or failure of the call.

VB Example

```
Sub Main()
Dim bRet As Boolean
Dim bRefresh As Boolean
Dim staad As Object
  Set staad = GetObject(, "StaadPro.OpenSTAAD")
'Test pre-processing modes
  bRet = staad.View.SetInterfaceMode(1)
  If bRet Then
  staad.View.SetModeSectionPage(1,6,20)
 staad.View.SetBeamAnnotationMode(2,1,False)
End If
staad.View.RefreshView()
End Sub
```

View.SetNodeAnnotationMode

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

VB Syntax

View.SetNodeAnnotationMode *annotationMode, bRefresh*

Where:

annotationMode

Integer variable controlling the annotation type. It may be one of the following values:

1 = X Displacement

2 = Y Displacement

3 = Z Displacement

4 = Resultant Displacement

bRefresh

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

VB Example

```
Dim bRefresh As Boolean
Dim dLabel As Long
Dim bRet As Boolean
'Get the application object --
'Annotate the view with X displacement labels
'Make sure that Staad.Pro is in Postprocessing mode
bRet = staad.View.SetInterfaceMode 1
If bRet Then
  bRefresh = True
  dLabel = 1 ' disp x
  staad.View.SetNodeAnotationMode(dLabel, bRefresh)
End If
```

View.SetReactionAnnotationMode

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

VB Syntax

View.SetReactionAnnotationMode *annotationMode, bRefresh*

Where:

annotationMode

Integer variable controlling the annotation type. It may be one of the following values:

1 = X Reaction

2 = Y Reaction

3 = Z Reaction

4 = X Rotation

5 = Y Rotation

6 = Z Rotation

7 = Reaction Value Only

bRefresh

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

VB Example

```
Dim bRefresh As Boolean
Dim dLabel As Long
Dim bRet As Boolean
'Get the application object --
'Annotate the view with X displacement labels
'Make sure that Staad.Pro is in Postprocessing mode
bRet = staad.View.SetInterfaceMode (1)
If bRet Then
  bRefresh = True
  dLabel = 1 ' X Reaction
  staad.View.SetReactionAnotationMode(dLabel, bRefresh)
End If
```

View.SetSectionView

Creates a section view of the structure.

VB Syntax

View.SetSectionView*Plane, minVal, maxVal*

Where:

Plane

Integer variable identifying the section plane. It may be one of the following values:

0: XY Plane

1: YZ Plane

2: XZ Plane

minVal

Minimum range of the cutting plane.

maxVal

Maximum range of the cutting plane.

VB Example

```
'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:
Dim fmin As Double
Dim fmax As Double
Dim Plane As Integer
```

```
'Get the application object --
'Label the member numbers
Plane = 1 'YZ Plane
fmin = 0.4
fmax = 0.6
objOpenSTAAD.View.SetLabel(plane,fmin,fmax)
```

View.SetUnits

Set viewing unit for the active view.

VB Syntax

View.SetUnits *UnitType, UnitForIt*

Where:

UnitType

An integer variable that holds unit type. Values are as follows:

0 = Dimension

1 = Displacement (Translational Degrees of Freedom)

2 = Sectional Dimension

3 = Sectional Area

4 = Moment of Inertia

5 = Force

6 = Moment

7 = Distributed Force

8 = Distributed Moment

9 = Material Density

10 = Acceleration

11 = Spring Constants (Translational Degrees of Freedom)

12 = Spring Constants (Rotational Degrees of Freedom)

13 = Material Modulus

14 = Stress

15 = Alpha (Coefficient of Thermal Expansion)

16 = Temperature

17 = Mass

18 = Sectional Modulus

19 = Displacement (Rotational Degrees of Freedom)

20 = Soil Subgrade Modulus

-1 = No Unit

UnitForIt

A string variable that holds the unit for the specified type. Like “cm”, “kns”, “feet”, “kn/cm” etc.

VB Example

```
Dim intUnitType as Integer
Dim strUnitForIt as String
'Get the application object --
'Set View Unit
objOpenSTAAD.View.SetUnits (intUnitType, strUnitForIt)
```

View.ShowAllMembers

Display all members of the current structure.

VB Syntax

```
View.ShowAllMembers ()
```

VB Example

```
'Get the application object --
'Display the whole structure
objOpenSTAAD.View.ShowAllMembers
```

View.ShowBack

This function displays the back view of the structure.

VB Syntax

```
View.ShowBack ()
```

VB Example

```
'Get the application object --
objOpenSTAAD.View.ShowBack
```

View.ShowBottom

This function displays the bottom view of the structure.

VB Syntax

```
View.ShowBottom ()
```

VB Example

```
'Get the application object --
objOpenSTAAD.View.ShowBottom
```

View.ShowFront

This function displays the front view of the structure.

VB Syntax

```
View.ShowFront ()
```

VB Example

```
'Get the application object --
objOpenSTAAD.View.ShowFront
```

View.ShowIsometric

This function displays the isometric view of the structure.

VB Syntax

```
View.ShowIsometric ()
```

VB Example

```
'Get the application object --
objOpenSTAAD.View.ShowIsometric
```

View.ShowLeft

This function displays the left view of the structure.

VB Syntax

```
View.ShowLeft ()
```

VB Example

```
'Get the application object --
objOpenSTAAD.View.ShowLeft
```

View.ShowMember

Show the specified member.

VB Syntax

View.ShowMember *MemberNo*

Where:

MemberNo

A long variable that holds member number to be shown.

VB Example

```
Dim objOpenSTAAD As Object
Dim lMemberNo as Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
'Variables
lMemberNo = 5

'Show Member

objOpenSTAAD.View.ShowMember (lMemberNo)
```

View.ShowMembers

Show the specified members.

VB Syntax

View.ShowMembers *MemberNumber, MemberNoArray*

Where:

MemberNumber

A long variable that holds number of members to show.

MemberNoArray

Long array variable holds the member nos, which need to be shown.

VB Example

```
Dim objOpenSTAAD As Object
Dim lMemberNumber as Long
Dim MemberNoArray() As Long
'Get the application object
Set objOpenSTAAD = GetObject( , "StaadPro.OpenSTAAD")
```

```
'Variables
lMemberNumber = 5
ReDim MemberNoArray(0 to 4) As Long
MemberNoArray(0) = 1
MemberNoArray(1) = 2
MemberNoArray(2) = 5
MemberNoArray(3) = 7
MemberNoArray(4) = 10
'Show Member
objOpenSTAAD.View.ShowMembers (lMemberNumber, MemberNoArray)
```

View.ShowPlan

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

VB Syntax

```
View.ShowPlan ()
```

VB Example

```
'Get the application object --
objOpenSTAAD.View.ShowPlan
```

View.ShowRight

This function displays the right view of the structure.

VB Syntax

```
View.ShowRight ()
```

VB Example

```
'Get the application object --
objOpenSTAAD.View.ShowRight
```

View.SpinLeft

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

VB Syntax

```
View.SpinLeft Degrees
```

Where:

Degrees

Double variable providing the degree of rotation.

VB Example

```
'Get the application object --
'Rotate the structure about Z Axis through 30 degrees
objOpenSTAAD.View.SpinLeft 30.0
```

View.SpinRight

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

VB Syntax

View.SpinRight *Degrees*

Where:

Degrees

Double variable providing the degree of rotation.

VB Example

```
'Get the application object --
'Rotate the structure about Z Axis through 30 degrees
objOpenSTAAD.View.SpinRight 30.0
```

View.ZoomAll

Display the whole structure.

VB Syntax

View.ZoomAll ()

VB Example

```
'Get the application object --
'Display the whole structure
objOpenSTAAD.View.ZoomAll
```

View.ZoomExtentsMainView

Adjust viewing scale to zoom main view to the extents.

VB Syntax

View.ZoomExtentsMainView ()

VB Example

```
'Get the application object --
'Display the whole structure
objOpenSTAAD.View.ZoomExtentsMainView
```

4.4 Properties Functions

Country Codes

Property Table Country Codes

Value	Country
1	American
2	Australian
3	British
4	Canadian
5	Chinese
6	Dutch
7	European
8	French
9	German
10	Indian
11	Japanese
12	Russian
13	South African
14	Spanish
15	Venezuelan
16	Korean
17	Aluminum
18	USColdformed
19	ISColdformed

Type Specification

Section Type Codes

Value	Type
0	ST

Value	Type
1	RA
2	D
3	LD
4	SD
5	T
6	CM
7	TC
8	BC
9	TB
10	BA
11	FR
-1	User Defined

Additional Specifications

For this type of spec-ification...	use these additional specifications		
	Add-Spec_1	AddSpec_2	Add-Spec_3
TC, BC and TB	WP	TH	
CM	CT	FC	
D, BA and FR	SP		
LD and SD	SP		
Tube Define	TH	WT	DT
Pipe Define	OD	ID	

Additional specifications should be given in current units.

Property.AssignBeamProperty

Assigns property to beam or beams.

VB Syntax

```
Property.AssignBeamProperty BeamNo, PropertyNo
Property.AssignBeamProperty BeamNoArray, PropertyNo
```

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

PropertyNo

Integer variable providing the property reference number.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Assign property no 1 to beam no 3
bResult = objOpenSTAAD.Property.AssignBeamProperty 3, 1
```

Property.AssignBetaAngle

Assigns Beta Angle to beam or beams.

VB Syntax

```
Property.AssignBetaAngle BeamNo, BetaAngle
Property.AssignBetaAngle BeamNoArray, BetaAngle
```

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

BetaAngle

Double variable providing the beta angle in degrees.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Assign beta angle 90.0 to beam no 3
bResult = objOpenSTAAD.Property.AssignBetaAngle 3, 90.0
```

Property.AssignElementSpecToPlate

Assigns specifications to plate or plates.

VB Syntax

```
Property.AssignElementSpecToPlate PlateNo, SpecNo
Property.AssignElementSpecToPlate PlateNoArray, SpecNo
```

Where:

PlateNo

Integer variable providing the plate number.

PlateNoArray

Integer variable array providing the plate numbers.

SpecNo

Integer variable providing the element specification reference number.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Assign specification no 1 to Plate no 3
bResult = objOpenSTAAD.Property.AssignElementSpecToPlate (3, 1)
```

Property.AssignMemberSpecToBeam

Assigns specifications to beam or beams.

VB Syntax

```
Property.AssignMemberSpecToBeam BeamNo, SpecNo
Property.AssignMemberSpecToBeam BeamNoArray, SpecNo
```

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

SpecNo

Integer variable providing the member specification reference number.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Assign specification no 1 to beam no 3
bResult = objOpenSTAAD.Property.AssignMemberSpecToBeam 3, 1
```

Property.AssignPlateThickness

Assigns thickness to plate.

VB Syntax

Property.AssignPlateThickness *PlateNo, PropertyNo*

Where:

PlateNo

Integer variable providing the plate number.

PropertyNo

Integer variable providing the property reference number.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Assign property no 1 to plate no 3
bResult = objOpenSTAAD.Property.AssignPlateThickness 3, 1
```

Property.CreateAnglePropertyFromTable

Creates angle property from database.

VB Syntax

Property.CreateAnglePropertyFromTable *Country, SectionName, TypeSpec, AddSpec_1*

Where:

Country

A Long variable providing the country code.

See "Country Codes " on page 95 at the beginning of this section.

SectionName

String variable containing the name of the section.

TypeSpec

A Long variable providing the type specification.

See "Type Specification " on page 95 at the beginning of this section.

AddSpec_1

Double variable providing additional information.

See "Additional Specifications " on page 96 at the beginning of this section.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create property ISA100X100X10 from Indian database
PropertyNo = objOpenSTAAD.Property.CreateAnglePropertyFromTable _
10, "ISA100X100X10", 0, 0.0
'0 = ST, no additional specification required
```

Property.CreateAssignProfileProperty

Creates "Assign Profile" property.

VB Syntax

Property.CreateAssignProfileProperty *TypeOfProfile*

Where:

TypeOfProfile

Integer variable providing the profile type as follows:

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

Return Value

A long value containing the reference number of the property created to be used to access the same.

For additional information, please refer to Section 5.20.5 of the Technical Reference manual.

VB Example

```
'Get the application object --
'Create Assign Profile property
PropertyNo = objOpenSTAAD.Property.CreateAssignProfileProperty 2
```

Property.CreateBeamPropertyFromTable

Creates beam property from database.

VB Syntax

Property.CreateBeamPropertyFromTable *Country, SectionName, TypeSpec, AddSpec_1, AddSpec_2*

Where:

Country

A Long variable providing the country code.

See "Country Codes " on page 95 at the beginning of this section.

SectionName

String variable containing the name of the section.

TypeSpec

A Long variable providing the type specification.

See "Type Specification " on page 95 at the beginning of this section.

AddSpec_1, AddSpec_2

Double variables providing additional information.

See "Additional Specifications " on page 96 at the beginning of this section.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create property ISMB600 from Indian database
PropertyNo = objOpenSTAAD.Property.CreateBeamPropertyFromTable _
10, "ISMB600", 0, 0.0, 0.0
'0 = ST, no additional specification required
```

Property.CreateChannelPropertyFromTable

Creates channel property from database.

VB Syntax

Property.CreateChannelPropertyFromTable *Country, SectionName, TypeSpec, AddSpec_1*

Where:

Country

A Long variable providing the country code.

See "Country Codes " on page 95 at the beginning of this section.

SectionName

String variable containing the name of the section.

TypeSpec

A Long variable providing the type specification.

See "Type Specification " on page 95 at the beginning of this section.

AddSpec_1

Double variable providing additional information.

See "Additional Specifications " on page 96 at the beginning of this section.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create property ISMC600 from Indian database
PropertyNo = objOpenSTAAD.Property.CreateChannelPropertyFromTable _
10, "ISMC200", 0, 0.0
'0 = ST, no additional specification required
```

Property.CreateElementIgnoreInplaneRotnSpec

Creates ELEMENT IGNORE INPLANE ROTATION specification.

VB Syntax

```
Property.CreateElementIgnoreInplaneRotnSpec
```

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
'Get the application object --
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateElementIgnoreInplaneRotnSpec
```

Property.CreateElementNodeReleaseSpec

Creates ELEMENT RELEASE specification.

VB Syntax

```
Property.CreateElementNodeReleaseSpec Node, DOFReleaseArray
```

Where:

Node

Integer variable specifying node (1, 2, 3 or 4) of the element to be released.

DOFReleaseArray

Integer variable array of size 6 providing releases in different degrees of freedom (0 = No Release, 1 = Release) FX, FY, FZ, MX, MY and MZ.

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
Dim DOFRelease(0 To 5) As Integer
'Get the application object --
'Set the flags for releases
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateElementNodeReleaseSpec 1,
      DOFRelease
```

Property.CreateElementPlaneStressSpec

Creates ELEMENT PLANE STRESS specification.

VB Syntax

Property.CreateElementPlaneStressSpec ()

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
'Get the application object --
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateElementPlaneStressSpec
```

Property.CreateMemberCableSpec

Creates MEMBER CABLE specification.

VB Syntax

Property.CreateMemberCableSpec *AddCableInfo, Value*

Where:

AddCableInfo

Integer variable specifying additional information about the cable:

0 = Initial TENSION of *Value* in the cable to be considered

1 = Unstressed LENGTH of *Value* to be considered

Value

Double variable providing Initial TENSION or Unstressed LENGTH.

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
'Get the application object --
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateMemberCableSpec 0, 4.5
```

Property.CreateMemberCompressionSpec

Creates MEMBER COMPRESSION specification.

VB Syntax

```
Property.CreateMemberCompressionSpec ()
```

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
'Get the application object --
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateMemberCompressionSpec
```

Property.CreateMemberIgnoreStiffSpec

Creates MEMBER IGNORE specification.

VB Syntax

```
Property.CreateMemberIgnoreStiffSpec ()
```

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
'Get the application object --
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateMemberIgnoreStiffSpec
```

Property.CreateMemberInactiveSpec

Creates MEMBER INACTIVE specification.

VB Syntax

Property.CreateMemberInactiveSpec

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
'Get the application object --
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateMemberInactiveSpec
```

Property.CreateMemberOffsetSpec

Creates MEMBER OFFSET specification.

VB Syntax

Property.CreateMemberOffsetSpec *StartOrEnd, LocalOrGlobal, OffsetX, OffsetY, OffsetZ*

Where:

StartOrEnd

Integer variable specifying whether offsets are at START (= 0) or END (= 1) of the member.

LocalOrGlobal

Integer variable specifying whether the offsets are given with respect to Local (= 1) axis or Global axis (= 0).

OffsetX, OffsetY, OffsetZ

Double variables providing the X, Y and Z offsets in current units.

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
'Get the application object --
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateMemberOffsetSpec 0, 0, 0.5, 0.0,
     0.0
```

Property.CreateMemberPartialReleaseSpec

Creates MEMBER RELEASE specification.

VB Syntax

```
Property.CreateMemberPartialReleaseSpec StartOrEnd, PartialRelease,
      ReleaseFactor
Property.CreateMemberPartialReleaseSpec StartOrEnd, PartialReleaseArray,
      ReleaseFactorArray
```

Where:

StartOrEnd

Integer variable specifying whether offsets are at START (= 0) or END (= 1) of the member.

PartialRelease

Integer variable to set the flag for MP (0 = No Release, 1 = Release).

PartialReleaseArray

Integer variable array of size 3 providing releases in different degrees of freedom (0 = No Release, 1 = Release) MPX, MPY and MPZ.

ReleaseFactor

Double variable providing the partial moment release factor to be applied uniformly to all three DOFs.

ReleaseFactorArray

Double variable array of size 3 providing the partial moment release factors in respective DOFs.

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
Dim DOFRelease(0 To 2) As Integer
Dim MPFactor (0 To 2) As Double
'Get the application object --
'Set the flags for releases
'Set moment release factors if any
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateMemberPartialReleaseSpec _
0, DOFRelease, MPFactor
```

Property.CreateMemberReleaseSpec

Creates MEMBER RELEASE specification.

VB Syntax

Property.CreateMemberReleaseSpec *StartOrEnd, DOFReleaseArray,*
SpringConstantsArray

Where:

StartOrEnd

Integer variable specifying whether offsets are at START (= 0) or END (= 1) of the member.

DOFReleaseArray

Integer variable array of size 6 providing releases in different degrees of freedom (0 = No Release, 1 = Release) FX, FY, FZ, MX, MY and MZ.

SpringConstantsArray

Double variable array of size 6 providing the spring constants KFX, KFY, KFZ, KMX, KMY and KMZ.

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
Dim DOFRelease(0 To 5) As Integer
Dim SpringConst (0 To 5) As Double
```

```
'Get the application object --
'Set the flags for releases
'Set spring constants if any
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateMemberReleaseSpec _
0, DOFRelease, SpringConst
```

Property.CreateMemberTensionSpec

Creates MEMBER TENSION specification.

VB Syntax

```
Property.CreateMemberTensionSpec ()
```

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
'Get the application object --
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateMemberTensionSpec
```

Property.CreateMemberTrussSpec

Creates MEMBER TRUSS specification.

VB Syntax

```
Property.CreateMemberTrussSpec ()
```

Return Value

A long value containing the reference number of the specification created to be used to access the same.

VB Example

```
'Get the application object --
'Create specification
lSpecNo = objOpenSTAAD.Property.CreateMemberTrussSpec
```

Property.CreatePipePropertyFromTable

Creates pipe property from database.

VB Syntax

```
Property.CreatePipePropertyFromTable Country, SectionName, TypeSpec, AddSpec_1,
      AddSpec_2
```

Where:

Country

A Long variable providing the country code.

See "Country Codes " on page 95 at the beginning of this section.

SectionName

String variable containing the name of the section.

TypeSpec

A Long variable providing the type specification.

See "Type Specification " on page 95 at the beginning of this section.

AddSpec_1, AddSpec_2

Double variable providing additional information.

See "Additional Specifications " on page 96 at the beginning of this section.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create property PIP1270.0M from Indian database
PropertyNo = objOpenSTAAD.Property.CreatePipePropertyFromTable _
10, " PIP1270.0M", 0, 0.0, 0.0
'0 = ST, no additional specification required
'Create user defined tube of OD 300mm and ID 280mm
PropertyNo = objOpenSTAAD.Property.CreatePipePropertyFromTable _
10, " ", -1, 0.3, 0.28
'-1 = User defined
```

Property.CreatePlateThicknessProperty

Creates plate thickness property.

VB Syntax

```
Property.CreatePlateThicknessProperty ThicknessArray
Property.CreatePlateThicknessProperty Thickness
```

Where:

ThicknessArray

Double array variable providing different plate thickness at all nodes.

Thickness

Double variable providing single plate thickness for all nodes.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create plate thickness property
PropertyNo = objOpenSTAAD.Property.CreatePlateThicknessProperty 0.2
```

Property.CreatePrismaticCircleProperty

Creates prismatic circle property.

VB Syntax

```
Property.CreatePrismaticCircleProperty YD
```

Where:

YD

A Double variable providing the circle diameter.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create circle property of 250mm diameter
PropertyNo = objOpenSTAAD.Property.CreatePrismaticCircleProperty 0.25
```

Property.CreatePrismaticGeneralProperty

Creates prismatic general section property

VB Syntax

Property.CreatePrismaticGeneralProperty *PropertyArray*

Where:

PropertyArray

A Double array variable providing the property values as follows:

Array Index	Property Type
0	AX
1	AY
2	AZ
3	IX
4	IY
5	IZ
6	YD
7	ZD
8	YB
9	ZB

Return Value

A long value containing the reference number of the property created to be used to access the same.

For additional information, please refer to Section 5.20.2 of the Technical Reference manual.

VB Example

```
Dim PropArray(0 To 9) As Double
'Get the application object --
'fill PropArray here
PropArray(0) =
PropArray(1) =
PropArray(2) =
PropArray(3) =
PropArray(4) =
PropArray(5) =
PropArray(6) =
PropArray(7) =
```

```
PropArray(8) =
PropArray(9) =
'Create general section property
PropertyNo = objOpenSTAAD.Property.CreatePrismaticGeneralProperty
      PropArray
```

Property.CreatePrismaticRectangleProperty

Creates prismatic rectangle property.

VB Syntax

Property.CreatePrismaticRectangleProperty *YD, ZD*

Where:

YD

A Double variable providing the depth along the local Y-axis.

ZD

A Double variable providing the width along the local Z-axis.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create rectangle property of 250x500mm
PropertyNo = objOpenSTAAD.Property.CreatePrismaticRectangleProperty 0.5,
      0.25
```

Property.CreatePrismaticTeeProperty

Creates prismatic tee property.

VB Syntax

Property.CreatePrismaticTeeProperty *YD, ZD, YB, ZB*

Where:

YD

A Double variable providing the overall depth along the local Y-axis.

ZD

A Double variable providing the overall width along the local Z-axis.

YB

A Double variable providing the depth of the web along the local Y-axis.

ZB

A Double variable providing the width of the web along the local Z-axis.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create tee property of 250x500mm
PropertyNo = objOpenSTAAD.Property.CreatePrismaticTeeProperty _
0.5, 0.25, 0.4, 0.1
```

Property.CreatePrismaticTrapezoidalProperty

Creates prismatic trapezoidal property.

VB Syntax

Property.CreatePrismaticTrapezoidalProperty *YD, ZD, ZB*

Where:

YD

A Double variable providing the depth along the local Y-axis.

ZD

A Double variable providing the top width along the local Z-axis.

ZB

A Double variable providing the bottom width along the local Z-axis.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create trapezoidal property
```

```
PropertyNo = objOpenSTAAD.Property.CreatePrismaticTeeProperty 0.5, 0.25,
      0.2
```

Property.CreateTaperedIProperty

Creates tapered I property

VB Syntax

Property.CreateTaperedIProperty *PropertyArray*

Where:

PropertyArray

A Double array variable providing the property values as follows:

Array Index	Property Type
0	F1
1	F2
2	F3
3	F4
4	F5
5	F6
6	F7

Return Value

A long value containing the reference number of the property created to be used to access the same.

For additional information, please refer to Section 5.20.3 of the Technical Reference manual.

VB Example

```
Dim PropArray(0 To 6) As Double
'Get the application object --
'Create tapered I section property
'fill PropArray here
PropArray(0) =
PropArray(1) =
PropArray(2) =
PropArray(3) =
PropArray(4) =
PropArray(5) =
```

```
PropArray(6) =
PropertyNo = objOpenSTAAD.Property.CreateTaperedIProperty PropArray
```

Property.CreateTaperedTubeProperty

Creates tapered tube property.

VB Syntax

Property.CreateTaperedTubeProperty *TypeOfTube, d1, d2, Th*

Where:

TypeOfTube

Integer variable providing the tube type as follows:

Type of Tube	Value
Round	0
HexDecagonal	1
Dodecagonal	2
Octagonal	3
Hexagonal	4
Square	5

For additional information, please refer to Section 5.20.2.1 of the Technical Reference manual.

d1, d2

Double variables providing the depth of tube at start and end nodes.

Th

Double variable providing the thickness of tube.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create tapered tube section property
```

```
PropertyNo = objOpenSTAAD.Property.CreateTaperedTubeProperty 0.5, 0.4,
     0.01
```

Property.CreateTubePropertyFromTable

Creates tube property from database.

VB Syntax

```
Property.CreateTubePropertyFromTable Country, SectionName, TypeSpec, AddSpec_1,
      AddSpec_2, AddSpec_3
```

Where:

Country

A Long variable providing the country code.

See "Country Codes " on page 95 at the beginning of this section.

SectionName

String variable containing the name of the section.

TypeSpec

A Long variable providing the type specification.

See "Type Specification " on page 95 at the beginning of this section.

AddSpec_1, AddSpec_2, AddSpec_3

Double variable providing additional information.

See "Additional Specifications " on page 96 at the beginning of this section.

Return Value

A long value containing the reference number of the property created to be used to access the same.

VB Example

```
'Get the application object --
'Create property TUB30302.6 from Indian database
PropertyNo = objOpenSTAAD.Property.CreateTubePropertyFromTable _
10, " TUB30302.6", 0, 0.0, 0.0, 0.0
'0 = ST, no additional specification required
'Create user defined tube of 300mm x 200mm x 8mm
PropertyNo = objOpenSTAAD.Property.CreateTubePropertyFromTable _
10, " ", -1, 0.008, 0.3, 0.2
'-1 = User defined
```

Property.GetBeamConstants

Retrives material properties of the specified beam member.

VB Syntax

Property.GetBeamConstants *BeamNo, Elasticity, Poisson, Density, Alpha, Damp*

Where:

BeamNo

Long variable holds the beam number.

Elasticity, Poisson, Density, Alpha, Damp

Double variables return material properties.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetBeamConstants (lBeamNo, dElasticity, dPoisson,
      dDensity, dAlpha, dDamp)
```

Property.GetBeamMaterialName

Returns beam material name for the specified member.

VB Syntax

Property.GetBeamMaterialName *BeamNo*

Where:

BeamNo

Long variable holds the beam no.

VB Example

```
Dim lBeamMaterialName As Long
Dim lBeamNo as Long
'Get the application object
'Get Property
lBeamMaterialName = objOpenSTAAD.Property.GetBeamMaterialName (lBeamNo)
```

Property.GetBeamProperty

Retrives short member properties of the specified beam member.

VB Syntax

```
Property.GetBeamProperty BeamNo, Width, Depth, Ax, Ay, Az, Ix, Iy, Iz
```

Where:

BeamNo

Long variable holds the beam number.

Width, Depth, Ax, Ay, Az, Ix, Iy, Iz

Double variables return only sectional member properties.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetBeamProperty (lBeamNo, dWidth, dDepth, dAx,
      dAy, dAz, dIx, dIy, dIz)
```

Property.GetBeamPropertyAll

Retrives long member properties of the specified beam member.

VB Syntax

```
Property.GetBeamPropertyAll BeamNo, Width, Depth, Ax, Ay, Az, Ix, Iy, Iz, Tf,
      Tw
```

Where:

BeamNo

Long variable holds the beam number.

Width, Depth, Ax, Ay, Az, Ix, Iy, Iz, Tf, Tw

Double variables return all member properties.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetBeamPropertyAll (lBeamNo, dWidth, dDepth, dAx,
      dAy, dAz, dIx, dIy, dIz, dTf, _
dTw)
```

Property.GetBeamSectionName

Returns beam section name for the specified member.

VB Syntax

Property.GetBeamSectionName *BeamNo*

Where:

BeamNo

Long variable holds the beam no.

VB Example

```
Dim lBeamSectionName As Long
Dim lBeamNo as Long
'Get the application object
'Get Property
lBeamSectionName = objOpenSTAAD.Property.GetBeamSectionName (lBeamNo)
```

Property.GetBeamSectionPropertyTypeNo

Returns beam section property type no for the specified member.

VB Syntax

Property.GetBeamSectionPropertyTypeNo *BeamNo*

Where:

BeamNo

Long variable holds the beam no.

VB Example

```
Dim lBSPropertyTypeNo As Long
Dim lBeamNo as Long
'Get the application object
'Get Property
lBSPropertyTypeNo = objOpenSTAAD.Property.GetBeamSectionPropertyTypeNo
      (lBeamNo)
```

Property.GetBetaAngle

Retrives beta angle of the specified beam member.

VB Syntax

Property.GetBetaAngle *BeamNo*

Where:

BeamNo

Long variable holds the beam number.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetBetaAngle (lBeamNo)
```

Property.GetCountryTableNo

Returns country table no for the specified member.

VB Syntax

Property.GetCountryTableNo *BeamNo*

Where:

BeamNo

Long variable holds the beam no.

VB Example

```
Dim lCountryTableNo As Long
Dim lBeamNo as Long
'Get the application object
'Get Property
lCountryTableNo = objOpenSTAAD.Property.GetCountryTableNo (lBeamNo)
```

Property.GetIsotropicMaterialCount

Returns number of isotropic material present in the current structure.

VB Syntax

Property.GetIsotropicMaterialCount ()

VB Example

```
Dim lIsotropicMaterialCount as Long
'Get the application object --
'Get Property
lIsotropicMaterialCount =
      objOpenSTAAD.Property.GetIsotropicMaterialCount
```

Property.GetIsotropicMaterialProperties

Returns the properties for the specified isotropic material number.

VB Syntax

```
Property.GetIsotropicMaterialProperties MatNo, E, Poisson, G, Density, Alpha,
      CrDamp
```

Where:

MatNo

Long variable holds the material reference number.

E, Poisson, G, Density, Alpha, CrDamp

Double variables return isotropic material properties.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetIsotropicMaterialProperties (lMatNo, dE,
     dPoisson, dG, dDensity, dAlpha,  _
dCrDamp)
```

Property.GetMaterialProperty

Retrives material properties of the specified material.

VB Syntax

```
Property.GetMaterialProperty MaterialName, Elasticity, Poisson, Density, Alpha,
      Damp
```

Where:

MaterialName

String holds the material name.

Elasticity, Poisson, Density, Alpha, Damp

Double variables return material properties.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetMaterialProperty (strMaterialName, dElasticity,
     dPoisson, dDensity, _
dAlpha, dDamp)
```

Property.GetMemberGlobalOffSet

Returns beam end offsets in all three global directions.

VB Syntax

Property.GetMemberGlobalOffSet*BeamNo, End, xOffSet, yOffSet, zOffSet*

Where:

BeamNo

Long variable holds the member reference number.

End

Integer value specifies the end of the member.

0 = Start

1 = End

xOffSet, yOffSet, zOffSet

Double variables return offset value of the specified member in all direction at the specified end.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetMemberGlobalOffSet (lBeamNo, iEnd, dxOffSet,
      dyOffSet, dzOffSet)
```

Property.GetMemberLocalOffSet

Returns beam end offsets in all three local directions.

VB Syntax

Property.GetMemberLocalOffSet *BeamNo, End, xOffSet, yOffSet, zOffSet*

Where:

BeamNo

Long variable holds the member reference number.

End

Integer value specifies the end of the member.

0 = Start

1 = End

xOffSet, yOffSet, zOffSet

Double variables return offset value of the specified member in all three local

directions at the specified end.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetMemberLocalOffSet (lBeamNo, iEnd, dxOffSet,
      dyOffSet, dzOffSet)
```

Property.GetMemberReleaseSpec

Returns releases for the specified member at the specified end.

VB Syntax

Property.GetMemberReleaseSpec *BeamNo, End, ReleaseArray, SpringConstArray*

Where:

BeamNo

Long variable holds the member reference number.

End

Integer value specifies the end of the member.

0 = Start

1 = End

ReleaseArray, SpringConstArray

Long array returns translational and rotational releases.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetMemberReleaseSpec (lBeamNo, iEnd,
      lReleaseArray, lSpringConstArray
```

Property.GetOrthotropic2DMaterialCount

Returns number of 2D orthotropic material present in the current structure.

VB Syntax

Property.GetOrthotropic2DMaterialCount ()

VB Example

```
Dim lOrthotropic2DMaterialCount as Long
```

```
'Get the application object --
'Get Property
lOrthotropic2DMaterialCount =
      objOpenSTAAD.Property.GetOrthotropic2DMaterialCount
```

Property.GetOrthotropic2DMaterialProperties

Returns the properties for the specified isotropic material number.

VB Syntax

```
Property.GetOrthotropic2DMaterialProperties MatNo, EArray, PoissonArray,
      GArray, DensityArray, AlphaArray, CrDampArray
```

Where:

MatNo

Long variable holds the material reference number.

E, Poisson, G, Density, Alpha, CrDamp

Double array (dimension 2) variables return 2D orthotropic material properties.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetOrthotropic2DMaterialProperties (lMatNo,
      dEArray, dPoissonArray, dGArray, _
dDensityArray, dAlphaArray, dCrDampArray)
```

Property.GetOrthotropic3DMaterialCount

Returns number of 3D orthotropic material present in the current structure.

VB Syntax

```
Property.GetOrthotropic3DMaterialCount ()
```

VB Example

```
Dim lOrthotropic3DMaterialCount as Long
'Get the application object --
'Get Property
lOrthotropic3DMaterialCount =
      objOpenSTAAD.Property.GetOrthotropic3DMaterialCount
```

Property.GetOrthotropic3DMaterialProperties

Returns the properties for the specified isotropic material number.

VB Syntax

```
Property.GetOrthotropic3DMaterialProperties MatNo, EArray, PoissonArray,
       GArray, DensityArray, AlphaArray, CrDampArray
```

Where:

MatNo

Long variable holds the material reference number.

E, Poisson, G, Density, Alpha, CrDamp

Double array (dimension 3) variables return 3D orthotropic material properties.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetOrthotropic3DMaterialProperties (lMatNo,
     dEArray, dPoissonArray, dGArray, _
dDensityArray, dAlphaArray, dCrDampArray)
```

Property.GetPlateThickness

Returns plate thickness at corners for the specified plate.

VB Syntax

```
Property.GetPlateThickness  PlateNo, PlateThicknessArray
```

Where:

PlateNo

Long variable holds the plate reference number.

PlateThicknessArray

Long array returns plate thickness at the corners of the plate element.

VB Example

```
'Get the application object --
'Get Property
Dim oSTD As Object
Dim plateNo as Long
Dim plateThkArray(0 to 3)
plateNo = 2
oSTD.Property.GetPlateThickness(plateNo, plateThkArray)
```

Property.GetSectionPropertyCount

Returns total number of different sectional properties exist in the current STAAD file.

VB Syntax

```
Property.GetSectionPropertyCount ()
```

VB Example

```
Dim lSectionPropertyCount as Long
'Get the application object --
'Get Property
lSectionPropertyCount = objOpenSTAAD.Property.GetSectionPropertyCount
```

Property.GetSectionPropertyCountry

Returns the country reference number for the section property reference number specified.

VB Syntax

```
Property.GetSectionPropertyCountry SecRefNo
```

Where:

SecRefNo

Long variable holds the section property reference number.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetSectionPropertyCountry (lSecRefNo)
```

Property.GetSectionPropertyName

Returns the property name for the specified section property reference number.

VB Syntax

```
Property.GetSectionPropertyName SecRefNo, PropertyName
```

Where:

SecRefNo

Long variable holds the section property reference number.

PropertyName

String variable returns the property name.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetSectionPropertyName (lSecRefNo,
      strPropertyName)
```

Property.GetSectionPropertyType

Returns the section property type for the specified section property reference number.

VB Syntax

Property.GetSectionPropertyType *SecRefNo*

Where:

SecRefNo

Long variable holds the section property reference number.

VB Example

```
'Get the application object --
'Get Property
objOpenSTAAD.Property.GetSectionPropertyType (lSecRefNo)
```

Property.GetSectionTableNo

Returns section table no for the specified member.

VB Syntax

Property.GetSectionTableNo *BeamNo*

Where:

BeamNo

Long variable holds the beam no.

VB Example

```
Dim lSectionTableNo As Long
Dim lBeamNo as Long
'Get the application object
'Get Property
lSectionTableNo = objOpenSTAAD.Property.GetSectionTableNo (lBeamNo)
```

Property.SetMaterialID

Sets the material for the member property.

VB Syntax

Property.SetMaterialID *MaterialID*

Where:

MaterialID

A Long variable providing the material ID (0 = Steel, 1 = Concrete, and 2 = Aluminum).

VB Example

```
'Get the application object --
'Set material
objOpenSTAAD.Property.SetMaterialID 0
```

4.5 Loads Functions

Load.AddElementPressure

Adds pressure load to plate elements.

VB Syntax

```
Load.AddElementPressure PlateNo, Direction, Pressure, X1, Y1, X2, Y2
Load.AddElementPressure PlateNoArray, Direction, Pressure, X1, Y1, X2, Y2
```

Where:

PlateNo

Integer variable providing the plate number.

PlateNoArray

Integer variable array providing the plate numbers.

Direction

Integer variable giving the direction of pressure:

direction	integer value
0	Local z
1	Global X
2	Global Y
3	Global Z

Pressure

Double variable providing the magnitude of the pressure over the element.

X1, Y1, X2, Y2

Double variable providing the top left coordinate and bottom right coordinate of concentrated load.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add element pressure
objOpenSTAAD.Load.AddElementPressure 2, 1, 2.0
```

Load.AddElementTrapPressure

Adds trapezoidal pressure loading to plate elements.

VB Syntax

```
Load.AddElementTrapPressure PlateNo, Direction, StartPressure, EndPressure
Load.AddElementTrapPressure PlateNoArray, Direction, StartPressure, EndPressure
```

Where:

PlateNo

Integer variable providing the plate number.

PlateNoArray

Integer variable array providing the plate numbers.

Direction

Integer variable giving the direction of pressure:

1 = Local X direction

2 = Local Y direction

StartPressure, EndPressure

Double variable providing the pressure at start and end.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add element pressure
objOpenSTAAD.Load.AddElementTrapPressure 2, 1, 2.0, 3.0
```

Load.AddLoadAndFactorToCombination

Adds a primary load case with specified multiplication factor to an existing load combination.

VB Syntax

```
Load.AddLoadAndFactorToCombination LoadCombNo, LoadNo, LoadFactor
```

Where:

LoadCombNo

An integer variable holds the load combination number.

LoadNo

A long variable holds the primary load case number.

LoadFactor

Multiplication factor for the specified primary load case.

VB Example

```
'Get the application object --
Dim intLoadCombNo as Integer
Dim lLoadNo as Long
Dim fFactor as Float
'Add Load to Load Combination
objOpenSTAAD.Load.AddLoadAndFactorToCombination intLoadCombNo, lLoadNo,
      fFactor
```

Load.AddMemberAreaLoad

Adds AREA LOAD to beam or beams.

VB Syntax

```
Load.AddMemberAreaLoad BeamNo, AreaLoad
Load.AddMemberAreaLoad BeamNoArray, AreaLoad
```

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

AreaLoad

Double variable providing the magnitude of the load value.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add area load to member 2
objOpenSTAAD.Load.AddMemberAreaLoad2, 2.0
```

Load.AddMemberConcForce

Adds CONCENTRATED FORCE to beam or beams.

VB Syntax

```
Load.AddMemberConcForce BeamNo, Direction, ForceValue, D1Value, D2Value
Load.AddMemberConcForce BeamNoArray, Direction, ForceValue, D1Value, D2Value
```

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

Direction

Integer variable giving the direction of load: X direction = 1, Y direction = 2, Z direction = 3, GX direction = 4, GY direction = 5, GZ direction = 6, PX direction = 7, PY direction = 7, PZ direction = 8.

ForceValue

Double variable providing the magnitude of the concentrated force in current units.

D1Value, D2Value

Double variables providing values of d1, d2 in current units. For additional information, refer to Section 5.32.2 of the Technical Reference manual.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add member concentrated load of -6.43 units to member 57 in GY
     direction
objOpenSTAAD.Load.AddMemberConcForce 57, 5, -6.43, 4.7, 0.92
```

Load.AddMemberConcMoment

Adds CONCENTRATED MOMENT to beam or beams.

VB Syntax

```
Load.AddMemberConcMoment BeamNo, Direction, MomentValue, D1Value, D2Value
Load.AddMemberConcMoment BeamNoArray, Direction, MomentValue, D1Value, D2Value
```

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

Direction

Integer variable giving the direction of load: X direction = 1, Y direction = 2, Z direction = 3, GX direction = 4, GY direction = 5, GZ direction = 6, PX direction = 7, PY direction = 7, PZ direction = 8.

MomentValue

Double variable providing the magnitude of the concentrated moment in current units.

D1Value, D2Value

Double variable providing value of d1, d2 in current units. For additional information, refer to Section 5.32.2 of the Technical Reference manual.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add member concentrated moment of 2 units to member 2 in GY direction
objOpenSTAAD.Load.AddMemberConcMoment 2, 5, 2.0, 0.0, 0.0, 0.0
```

Load.AddMemberFixedEnd

Adds FIXED END LOAD to beam or beams.

VB Syntax

```
Load.AddMemberFixedEnd BeamNo, LoadAtStartArray, LoadAtEndArray
Load.AddMemberFixedEnd BeamNoArray, LoadAtStartArray, LoadAtEndArray
```

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

LoadAtStartArray, LoadAtEndArray

Double variable arrays of dimension 6 providing the fixed end load values at member start and end. The indices are as follows: 0 = Fx_1, 1 = Fy_1, 2 = Fz_1, 3 = Mx_1, 4 = My_1, 5 = Mz_1 for start and 0 = Fx_2, 1 = Fy_2, 2 = Fz_2, 3 = Mx_2, 4 = My_2, 5 = Mz_2 for end.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
Dim start(0 To 5) As Double
Dim end(0 To 5) As Double
'Get the application object --
'Fill up the array accordingly
'Add fixed end load to member 2
objOpenSTAAD.Load.AddMemberFixedEnd 2, start, end
```

Load.AddMemberFloorLoad

Automatically finds enclosed panels in the given boundary and adds a FLOOR LOAD command in the format:

VB Syntax

Load.AddMemberFloorLoad*Pressure, YMIN, YMAX, ZMIN, ZMAX, XMIN, XMAX*

Where:

Pressure

A double value to indicate the pressure intensity to be applied as a floor load.

YMIN, YMAX, ZMIN, ZMAX, XMIN, XMAX

Double values to indicate what the boundary of the floor is. The XMIN, XMAX and ZMIN and ZMAX values must lie on the same XZ plane.

VB Example

```
'Get the application object --
'Add a floor load with intensity of -5
objOpenSTAAD.Load.AddMemberFloorLoad -5, 2.9, 3.1, 0, 80, 0, 200
```

Load.AddMemberLinearVari

Adds LINEARLY VARYING load to beam or beams.

VB Syntax

Load.AddMemberLinearVari *BeamNo, Direction, StartLoad, EndLoad, TriLoad*
Load.AddMemberLinearVari *BeamNoArray, Direction, StartLoad, EndLoad, TriLoad*

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

Direction

Integer variable giving the direction of load: X direction = 1, Y direction = 2, Z direction = 3.

StartLoad, EndLoad

Double variable providing the load value at the start and end of the member respectively.

TriLoad

Double variable providing the magnitude of the load value for triangular load. If it has a value other than 0, the load is regarded as a triangular load irrespective of values in StartLoad and EndLoad.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add member linearly varying to member 2 in GY direction
objOpenSTAAD.Load.AddMemberLinearVari 2, 2, 2.0, 0.0, 0.0
```

Load.AddMemberTrapezoidal

Adds trapezoidal linearly varying load to beam or beams.

VB Syntax

```
Load.AddMemberTrapezoidal BeamNo, Direction, StartLoad, EndLoad, StartDistance,
      LoadLength
Load.AddMemberTrapezoidal BeamNoArray, Direction, StartLoad, EndLoad,
      StartDistance, EndDistance
```

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

Direction

Integer variable giving the direction of load:

Direction	Integer Value
local x	1
local y	2
local z	3
global X	4
global Y	5
global Z	6
projected x	7
projected y	8
projected z	9

For additional information, refer to Section 5.32.2 of the Technical Reference manual.

StartLoad, EndLoad

Double variable providing the load value at the start and end of the member respectively.

StartDistance, EndDistance

Double variable providing the start and end distance of the load.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add member linearly varying to member 2 in GY direction
objOpenSTAAD.Load.AddMemberTrapezoidal 2, 2, 2.0, 0.0, 0.0
```

Load.AddMemberUniformForce

Adds UNIFORM FORCE to beam or beams.

VB Syntax

```
Load.AddMemberUniformForce BeamNo, Direction, ForceValue, D1Value, D2Value,
      D3Value
Load.AddMemberUniformForce BeamNoArray, Direction, ForceValue, D1Value,
      D2Value, D3Value
```

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

Direction

Integer variable giving the direction of load: X direction = 1, Y direction = 2, Z direction = 3, GX direction = 4, GY direction = 5, GZ direction = 6, PX direction = 7, PY direction = 7, PZ direction = 8.

ForceValue

Double variable providing the magnitude of the uniform force in current units.

D1Value, D2Value, D3Value

Double variable providing the value of d1, d2, d3 in current units. For additional information, refer to Section 5.32.2 of the Technical Reference manual.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add member uniform load of 2 units to member 2 in GY direction
objOpenSTAAD.Load.AddMemberUniformForce 2, 5, 2.0, 0.0, 0.0, 0.0
```

Load.AddMemberUniformMoment

Adds UNIFORM MOMENT to beam or beams.

VB Syntax

```
Load.AddMemberUniformMoment BeamNo, Direction, MomentValue, D1Value, D2Value,
      D3Value
Load.AddMemberUniformMoment BeamNoArray, Direction, MomentValue, D1Value,
      D2Value, D3Value
```

Where:

BeamNo

Integer variable providing the beam number.

BeamNoArray

Integer variable array providing the beam numbers.

Direction

Integer variable giving the direction of load: X direction = 1, Y direction = 2, Z direction = 3, GX direction = 4, GY direction = 5, GZ direction = 6, PX direction = 7, PY direction = 7, PZ direction = 8.

MomentValue

Double variable providing the magnitude of the uniform moment in current units.

D1Value, D2Value, D3Value

Double variable providing value of d1, d2, d3 in current units. For additional information, refer to Section 5.32.2 of the Technical Reference manual.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add member uniform moment of 2 units to member 2 in GY direction
objOpenSTAAD.Load.AddMemberUniformMoment 2, 5, 2.0, 0.0, 0.0, 0.0
```

Load.AddNodalLoad

Adds JOINT LOAD to the specified node number or numbers.

VB Syntax

```
Load.AddNodalLoad NodeNo, ForceX,  double ForceY, ForceZ, MomentX, MomentY,
       MomentZ
Load.AddNodalLoad NodeNoArray, ForceX, ForceY, ForceZ, MomentX, MomentY,
       MomentZ
```

Where:

NodeNo

Integer variable providing the node number.

NodeNoArray

Integer variable array providing the node numbers.

ForceX, ForceY, ForceZ, MomentX, MomentY, MomentZ

Double variables providing the load values in individual directions.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add joint load of 2units in X direction at node number 2
objOpenSTAAD.Load.AddNodalLoad 2, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0
```

Load.AddSelfWeightInXYZ

Adds SELFWEIGHT of the structure in X or Y or Z direction to the active load case.

VB Syntax

Load.AddSelfWeightInXYZ *Direction, LoadFactor*

Where:

Direction

Integer variable giving the direction of selfweight:

1 = X direction

2 = Y direction

3 = Z direction

LoadFactor

Double variable providing the multiplying factor for selfweight.

For additional information, refer to Section 5.32.9 of the Technical Reference Manual.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add selfweight in negative Y direction
objOpenSTAAD.Load.AddSelfWeightInXYZ 2, -1.0
```

Load.AddStrainLoad

Adds STRAIN LOAD to beam or plate elements.

VB Syntax

Load.AddStrainLoad *ElementNo, AxialStrain*
Load.AddStrainLoad *ElementNoArray, AxialStrain*

Where:

ElementNo

Integer variable providing the member/element number.

ElementNoArray

Integer variable array providing the member/element numbers.

AxialStrain

Double variable providing the strain value due to misfit.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add strain load
objOpenSTAAD.Load.AddStrainLoad 2, 0.01
```

Load.AddSupportDisplacement

Adds SUPPORT DISPLACEMENT to node or nodes.

VB Syntax

```
Load.AddSupportDisplacement NodeNo, Direction, Displacement
Load.AddSupportDisplacement NodeNoArray, Direction, Displacement
```

Where:

NodeNo

Integer variable providing the node number.

NodeNoArray

Integer variable array providing the node numbers.

Direction

Integer variable giving the direction of displacement: X direction = 1, Y direction = 2, Z direction = 3.

Displacement

Double variable providing the magnitude of the support displacement in current units.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add joint load of 2mm displacement to node 2 in global X
objOpenSTAAD.Load.AddSupportDisplacement 2, 1, 2.0
```

Load.AddTemperatureLoad

Adds TEMPERATURE LOAD to beam or plate elements.

VB Syntax

```
Load.AddTemperatureLoad ElementNo, AxialElongation, DiffTempTopAndBtm,
      DiffTempSideToSide
Load.AddTemperatureLoad ElementNoArray, AxialElongation, DiffTempTopAndBtm,
      DiffTempSideToSide
```

Where:

ElementNo

Integer variable providing the member/element number.

ElementNoArray

Integer variable array providing the member/element numbers.

AxialElongation

Double variable providing the temperature causing axial elongation.

DiffTempTopAndBtm

Double variable providing the differential temperature between top and bottom surface.

DiffTempSideToSide

Double variable providing the differential temperature from side to side.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Add temperature load
objOpenSTAAD.Load.AddTemperatureLoad 2, 10.0, 20.0, 30.0
```

Load.CreateNewLoadCombination

Creates new load combination with the number and title defined.

VB Syntax

Load.CreateNewLoadCombination *LoadCombTitle, LoadCombNo*

Where:

LoadCombTitle

A string variable, which defines title for new load combination.

LoadCombNo

A long variable, which defines number for new load combination.

VB Example

```
Dim strLoadCombTitle as String
Dim lLoadCombNo as Long
'Get the application object --
'Create New Load Combination
objOpenSTAAD.Load.CreateNewLoadCombination strLoadCombTitle, lLoadCombNo
```

Load.CreateNewPrimaryLoad

Creates new PRIMARY load case.

VB Syntax

Load.CreateNewPrimaryLoad *NewPrimaryLoadTitle*

Where:

NewPrimaryLoadTitle

String variable providing the title of the load case.

Return Value

Long integer containing the load number created.

VB Example

```
'Get the application object --
'Create new load
objOpenSTAAD.Load.CreateNewPrimaryLoad "My Load"
```

Load.GetActiveLoad

Returns the current load case number.

VB Syntax

```
Load.GetActiveLoad
```

VB Example

```
Dim lActiveLoad as Long
'Get the application object --
'Get active load
lActiveLoad = objOpenSTAAD.GetActiveLoad
```

Load.GetBeamCountAtFloor

Get total no of beams within the specified boundary.

VB Syntax

Load.GetBeamCountAtFloor *MinX, MaxX, MinY, MaxY, MinZ, MaxZ, Direction*

Where:

MinX, MaxX, MinY, MaxY, MinZ, MaxZ

Float variables indicate what the boundary of the floor pane.

Direction

Integer variable provide the direction.

1 = X-axis

2 = Y-axis

3 = Z-axis

VB Example

```
Dim fMinX as Float
Dim fMaxX as Float
Dim fMinY as Float
Dim fMaxY as Float
Dim fMinZ as Float
Dim fMaxZ as Float
Dim intDirection as Integer
'Get the application object --
'Get beam count at floor
objOpenSTAAD.Load.GetBeamCountAtFloor fMinX, fMaxX, fMinY, fMaxY, fMinZ,
      fMaxZ, intDirection
```

Load.GetConcForceCount

Get number of concentrated loads present for the specified beam.

VB Syntax

Load.GetConcForceCount *BeamNo*

Where:

BeamNo

Long variable provides the beam number.

VB Example

```
'Get the application object --
'Get Concentrated Load count
objOpenSTAAD.Load.GetConcForceCount (lBeamNo)
```

Load.GetConcForces

Returns the concentrated forces with all the parameters for the specified member.

VB Syntax

Load.GetConcForces*BeamNo, DirectionArray, ForceArray, D1Array, D2Array*

Where:

BeamNo

Long variable providing the member number.

DirectionArray, ForceArray, D1Array, D2Array

Returns the force value, direction along with d1 & d2 parameters for beam trapezoidal load(s).

VB Example

```
'Get the application object --
'Get Concentrated Load values
objOpenSTAAD.Load.GetConcForces lBeamNo, lDirectionArray, dForceArray,
     dD1Array, dD2Array
```

Load.GetConcMomentCount

Get number of concentrated moment present for the specified beam.

VB Syntax

Load.GetConcMomentCount *BeamNo*

Where:

BeamNo

Long variable provides the beam number.

VB Example

```
'Get the application object --
'Get Conc Moment Count
objOpenSTAAD.Load.GetConcMomentCount (lBeamNo)
```

Load.GetConcMoments

Returns the beam concentrated moment(s) with all the parameters for the specified member.

VB Syntax

Load.GetConcMoments *BeamNo, DirectionArray, MomentArray, D1Array, D2Array*

Where:

BeamNo

Long variable providing the member number.

DirectionArray, MomentArray, D1Array, D2Array

Returns the moment value, direction along with d1 & d2 parameters for beam concentrated moment(s).

VB Example

```
'Get the application object --
'Get Conc Moment Value
objOpenSTAAD.Load.GetConcMoments (lBeamNo, lDirectionArray,
      dMomentArray, dD1Array, dD2Array)
```

Load.GetInfluenceArea

Returns the beam numbers and corresponding influence areas within the specified boundary.

VB Syntax

Load.GetInfluenceArea*MinX, MaxX, MinY, MaxY, MinZ, MaxZ, Direction,*
BeamNosArray, AreasArray

Where:

MinX, MaxX, MinY, MaxY, MinZ, MaxZ

Float variables indicate what the boundary of the floor pane.

Direction

Integer variable provide the direction.

1 = X-axis

2 = Y-axis

3 = Z-axis

BeamNosArray

Long array returns all the beam numbers under consideration.

AreasArray

Double array returns all the influence areas for the beams returned in the last parameter.

VB Example

```
'Get the application object --
'Get Influence Area
objOpenSTAAD.Load.GetInfluenceArea fMinX, fMaxX, fMinY, fMaxY, fMinZ,
      fMaxZ, iDirection, _
lBeamNosArray, dAreasArray
```

Load.GetLoadCaseTitle

Returns title of the specified load case as a text string.

VB Syntax

Load.GetLoadCaseTitle *LongPrimaryLoadCaseNumber*

Where:

PrimaryLoadCaseNumber

A long variable that holds the primary load case number.

VB Example

```
Dim lLoadCase as Long
Dim strLoadCaseName as String
For lLoadCase 1 to 3
        strLoadCaseName = oStd.Load.GetLoadCaseTitle(lLoadCase)
Next lLoadCase
```

Load.GetLoadCombinationCaseCount

Gets total number of combination load case(s) present in the current structure.

VB Syntax

Load.GetLoadCombinationCaseCount ()

VB Example

```
Dim lGetLoadCombinationCaseCount as Long>
'Get the application object --
'Get Combination Load Case Count
lGetLoadCombinationCaseCount =
      objOpenSTAAD.Load.GetLoadCombinationCaseCount
```

Load.GetLoadCombinationCaseNumbers

Gets all primary load case number(s).

VB Syntax

Load.GetLoadCombinationCaseNumbers *CombineLoadCaseNumbersArray*

Where:

CombineLoadCaseNumbersArray

A long array which stores the load case numbers for all the primary load cases present in the current structure.

VB Example

```
Dim lGetLoadCombinationCaseCount as Long
Dim lLoadCombinationCaseNumbersArray() as Long
'Get the application object --
'Get Primary Load Case Numbers
lGetLoadCombinationCaseCount =
      objOpenSTAAD.Load.GetLoadCombinationCaseCount
ReDim lLoadCombinationCaseNumbersArray (lGetLoadCombinationCaseCount)
objOpenSTAAD.Load. GetLoadCombinationCaseNumbers
      lLoadCombinationCaseNumbersArray ()
```

Load.GetLoadType

Returns primary load case category(s) as an integer value.

VB Syntax

Load.GetLoadType *PrimaryLoadCaseNumber*

Where:

PrimaryLoadCaseNumber

A variant (can be integer or long) variable that holds the primary load case number.

Return Value

Returns primary load case category(s) as one of the following integers:

0. Dead
1. Live
2. Roof Live
3. Wind
4. Seismic
5. Snow
6. Fluids
7. Soil
8. Rain
9. Ponding
10. Dust
11. Traffic
12. Temp
13. Imperfection
14. Accidental
15. Flood
16. Ice
17. Wind Ice
18. Crane Hook
19. Mass
20. Gravity
21. Push
22. None (i.e., no load assigned in the input file)

VB Example

```
Dim lPrimaryLoadCaseCount as Long
Dim lLoadType() as Long
ReDim lLoadType(lPrimaryLoadCaseCount)
'Get the application object --
'Get Primary Load Case Types
For i = 0 to (lPrimaryLoadCaseCount - 1)
   lLoadType(i) = objOpenSTAAD.Load.GetLoadType(LoadCaseNo)
next i
```

Load.GetNodalLoadCount

Get number of nodal loads present for the specified node.

VB Syntax

Load.GetNodalLoadCount *NodeNo*

Where:

NodeNo

Long variable provides the node number.

VB Example

```
Dim iNodalLoadCount as Integer
Dim lNodeNo as Long
'Get the application object --
'Get Nodal Load Count
iNodalLoadCount = objOpenSTAAD.Load.GetNodalLoadCount ( lNodeNo )
```

Load.GetNodalLoads

Returns the array of load values for the specified node. Array will be formed and dimensioned as per defined load counts.

VB Syntax

Load.GetNodalLoads *NodeNo, FXArray, FYArray, FZArray, MXArray, MYArray, MZArray*

Where:

NodeNo

Long variable holds the node number for which nodal loads needs to be retrieved.

FXArray, FYArray, FZArray, MXArray, MYArray, MZArray

Double variables return the load array for all the load cases and all directions.

VB Example

```
'Get the application object --
'Get Nodal Loads
objOpenSTAAD.Load.GetNodalLoads (lNodeNo, dFXArray, dFYArray, dFZArray,
      dMXArray, dMYArray, dMZArray)
```

Load.GetPrimaryLoadCaseCount

Returns the total number of primary load cases present in the current structure.

VB Syntax

Load.GetPrimaryLoadCaseCount ()

VB Example

See "Load.GetPrimaryLoadCaseNumbers " on page 151 for an example.

Load.GetPrimaryLoadCaseNumbers

Gets all primary load case numbers.

VB Syntax

Load.GetPrimaryLoadCaseNumbers *PrimaryLoadCaseNumbersArray*

Where:

PrimaryLoadCaseNumbersArray

A long array which stores the load case numbers for all the primary load cases present in the current structure.

VB Example

```
Dim lPrimaryLoadCaseCount as Long
Dim lPrimaryLoadCaseNumbersArray() as Long
'Get the application object --
'Get Primary Load Case Count
lPrimaryLoadCaseCount = objOpenSTAAD.Load.GetPrimaryLoadCaseCount
ReDim lPrimaryLoadCaseNumbersArray (lPrimaryLoadCaseCount)
'Get Primary Load Case Numbers
objOpenSTAAD.Load.GetPrimaryLoadCaseNumbers
      lPrimaryLoadCaseNumbersArray
```

Load.GetTrapLoadCount

Get number of trapezoidal loads present for the specified beam.

VB Syntax

Load.GetTrapLoadCount *BeamNo*

Where:

BeamNo

Long variable provides the beam number.

VB Example

```
'Get the application object --
'Get Trapezoidal Load count
objOpenSTAAD.Load.GetTrapLoadCount (lBeamNo)
```

Load.GetTrapLoads

Returns the trapezoidal load(s) with all the parameters for the specified member.

VB Syntax

Load.GetTrapLoads *BeamNo, DirectionArray, W1Array, W2Array, D1Array, D2Array*

Where:

BeamNo

Long variable providing the member number.

DirectionArray, W1Array, W2Array, D1Array, D2Array

Returns the force value, direction along with w1, w2, d1 & d2 parameters for beam trapezoidal load(s).

VB Example

```
'Get the application object --
'Get Trapezoidal Load values
objOpenSTAAD.Load.GetTrapLoads (lBeamNo, lDirectionArray, dW1Array,
      dW2Array, dD1Array, dD2Array)
```

Load.GetUDLLoadCount

Get number of uniformly distributed loads present for the specified beam.

VB Syntax

Load.GetUDLLoadCount *BeamNo*

Where:

BeamNo

Long variable provides the beam number.

VB Example

```
'Get the application object --
'Get UDL count
objOpenSTAAD.Load.GetUDLLoadCount (lBeamNo)
```

Load.GetUDLLoads

Returns the UDL with all the parameters for the specified member.

VB Syntax

Load.GetUDLLoads *BeamNo, DirectionArray, ForceArray, D1Array, D2Array, D3Array*

Where:

BeamNo

Long variable providing the member number.

DirectionArray, ForceArray, D1Array, D2Array, D3Array

Returns the force value, direction alongwith d1, d2 & d3 parameter for beam UDL.

VB Example

```
'Get the application object --
'Get UDL value
objOpenSTAAD.Load.GetUDLLoads (lBeamNo, lDirectionArray, dForceArray,
      dD1Array, dD2Array, dD3Array)
```

Load.GetUNIMomentCount

Gets the count of uniformly distributed moment for the specified member.

VB Syntax

Load.GetUNIMomentCount *BeamNo*

Where:

BeamNo

Long variable providing the member number.

VB Example

```
'Get the application object --
'Get UMOM count
objOpenSTAAD.Load.GetUNIMomentCount (lBeamNo)
```

Load.GetUNIMoments

Returns the uni moments with all the parameters for the specified member.

VB Syntax

Load.GetUNIMoments *BeamNo, DirectionArray, ForceArray, D1Array, D2Array, D3Array*

Where:

BeamNo

Long variable providing the member number.

DirectionArray, ForceArray, D1Array, D2Array, D3Array

Returns the value, direction alongwith d1, d2 & d3 parameter for beam uniformly distributed moments.

VB Example

```
'Get the application object --
'Get UMOM value
```

```
objOpenSTAAD.Load.GetUNIMoments (lBeamNo, lDirectionArray, dForceArray,
      dD1Array, dD2Array, dD3Array)
```

Load.SetLoadActive

Makes the specified load number active.

VB Syntax

Load.SetLoadActive *LoadNo*

Where:

LoadNo

Integer variable providing the load number which will be made active.

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Make load case 1 active
objOpenSTAAD.Load.SetLoadActive 1
```

4.6 Supports Applications

Support.AssignSupportToNode

Assigns the specified support to node or nodes.

VB Syntax

```
Support.AssignSupportToNode NodeNo,  SupportNo
Support.AssignSupportToNode NodeNoArray,  SupportNo
```

Where:

NodeNo

Long variable providing the node number.

NodeNoArray

Long variable array providing the node numbers.

SupportNo

Long variable providing the reference number of the support

Return Value

(Boolean) True (1) if the function is successful, false (0) otherwise.

VB Example

```
'Get the application object --
'Assign support 2 to node 1
bResult = objOpenSTAAD.Support.AssignSupportToNode 1, 2
```

Support.CreateSupportFixed

Creates a fully fixed support.

VB Syntax

```
Support.CreateSupportFixed ()
```

Return Value

Long integer containing the reference number to the support created.

VB Example

```
'Get the application object --
'Create fixed support
```

```
objOpenSTAAD.Support.CreateSupportFixed
```

Support.CreateSupportFixedBut

Creates fixed support with releases in specified directions or a spring support with spring constants in specified directions.

VB Syntax

Support.CreateSupportFixedBut *ReleaseCodeArray, SpringConstantArray*

Where:

ReleaseCodeArray

Double array of six elements specifying releases in FX, FY, FZ, MX, MY, and MZ directions. Each element should have a value equal to 0.0 (Fixed) or 1.0 (Released).

SpringConstantArray

Double array of six elements specifying spring constants in FX, FY, FZ, MX, MY, and MZ directions.

Note: Values should be in the current unit system.

Return Value

Long integer containing the reference number to the support created.

VB Example

```
Sub Main()
    Dim objOpenSTAAD As Object
    Dim stdFile As String
'Launch OpenSTAAD Object
    Set objOpenSTAAD = GetObject(, "StaadPro.OpenSTAAD")
'Load your STAAD file - make sure you have successfully run the file
    objOpenSTAAD.GetSTAADFile stdFile, "TRUE"
    If stdFile = "" Then
        MsgBox "This macro can only be run with a valid STAAD file
      loaded.", vbOkOnly
        Set objOpenSTAAD = Nothing
        Exit Sub
    End If
Dim SupNo As Long
Dim release(5) As Double
Dim spring(5) As Double
release (0) = 0.0 'FX
release (1) = 1.0 'FY
release (2) = 0.0 'FZ
release (3) = 0.0 'MX
```

```
release (4) = 0.0 'MY
release (5) = 0.0 'MZ
spring(0) = 0.0 'FX
spring(1) = 0.0 'FY
spring(2) = 0.0 'FZ
spring(3) = 0.0 'MX
spring(4) = 0.0 'MY
spring(5) = 0.0 'MZ
SupNo = objOpenSTAAD.Support.CreateSupportFixedBut (release, spring)
MsgBox Str$(SupNo)
Set objOpenSTAAD = Nothing
End Sub
```

Support.CreateSupportPinned

Creates a pinned support (i.e., free to rotate about local y and z axis, fixed in all other degrees of freedom).

VB Syntax

```
Support.CreateSupportPinned ()
```

Return Value

Long integer containing the reference number to the support created.

VB Example

```
'Get the application object --
'Create pinned support
objOpenSTAAD.Support.CreateSupportPinned
```

Support.GetSupportCount

Returns total number of supported nodes exist in the current structure.

VB Syntax

```
Support.GetSupportCount ()
```

VB Example

```
'Get the application object --
'Get Support Count
objOpenSTAAD.Support.GetSupportCount
```

Support.GetSupportInformation

Returns support information for the specified node.

VB Syntax

Support.GetSupportInformation*NodeNo, ReleaseSpecArray, SpringSpecArray*

Where:

NodeNo

Long variable providing the node number.

ReleaseSpecArray

Integer array of dimension six retrieved values.

0 = Released

1 = Not Released

SpringSpecArray

Integer array of dimension six retrieved values for the spring stiffness of each degree of freedom.

VB Example

```
'Get the application object --
'Get Support Information
objOpenSTAAD.Support.GetSupportInformation lNodeNo, iReleaseSpecArray,
     dSpringSpecArray
```

Support.GetSupportNodes

Returns all supported nodes in an array.

VB Syntax

Support.GetSupportNodes *SupportNodesArray*

Where:

SupportNodesArray

A long Array stores the numbers of the nodes, which are supported in the current structure.

VB Example

```
Dim lSupportNodesArray() as Long
Dim iSupportCount as Integer
'Get the application object --
iSupportCount = Support.GetSupportCount
ReDim lSupportNodesArray(0 to (iSupportCount-1)) as Long
'Get Support Nodes
```

```
objOpenSTAAD.Support.GetSupportNodes (lSupportNodesArray)
```

Support.GetSupportType

Returns back the support type for the specified node.

VB Syntax

Support.GetSupportType *SupportNode*

Where:

SupportNode

Long variable providing the supported node number.

Return Value

A long value designating the support type:

1. Pinned
2. Fixed
3. Fixed But

VB Example

```
'Get the application object --
'Get Support Type
objOpenSTAAD.Support.GetSupportType (lSupportNode)
```

4.7 Command Functions

Command.CreateSteelDesignCommand

Create steel design command.

VB Syntax

```
Command.CreateSteelDesignCommandDesignCode, CommandNo, IntValuesArray,
        FloatValuesArray, StringValuesArray, AssignList
```

Where:

DesignCode

An string variable containing the STAAD.Pro design code name.

```
STEELDESIGNAASHTO = 1001,
STEELDESIGNAISC = 1002,
STEELDESIGNAUSTRALIAN = 1003,
STEELDESIGNBS5950 = 1004,
STEELDESIGNBS5400 = 1005,
STEELDESIGNCANADIAN = 1006,
STEELDESIGNFRENCH = 1007,
STEELDESIGNDIN18800 = 1008,
STEELDESIGNIS800 = 1009,
STEELDESIGNJAPANESE = 1010,
STEELDESIGNLRFD = 1011,
STEELDESIGNNS3472 = 1012,
STEELDESIGNSPANISH = 1013,
STEELDESIGNNPD = 1014,
STEELDESIGNBSK90 = 1015,
STEELDESIGNAPI = 1016,
STEELDESIGNEC3 = 1017,
STEELDESIGNCHINESE = 1018,
STEELDESIGNDS412 = 1019,
STEELDESIGNASCE = 1020,
STEELDESIGNB7 = 1021,
STEELDESIGNBSK99 = 1022,
STEELDESIGNDUTCH = 1023,
STEELDESIGNCYPRUS = 1024,
STEELDESIGNRUSSIAN = 1025,
STEELDESIGNAISI = 1026,
STEELDESIGNS136 = 1027,
STEELDESIGNIS801 = 1028,
STEELDESIGNIS802 = 1029,
STEELDESIGNMEXICANLRFD = 1030,
STEELDESIGNEGYPTIAN = 1031,
STEELDESIGNIS800LSD = 1032,
STEELDESIGNBS5950_COLD = 1033,
STEELDESIGNSOUTHAFRICAN = 1034,
STEELDESIGNAISC_RECO = 1035,
```

```
STEELDESIGNCANADIANRECO_1994 = 1036,
STEELDESIGNCANADIANRECO_2001 = 1037,
STEELDESIGNAISI_2001RCECO = 1038,
STEELDESIGNAISI_1996RCECO = 1040,
STEELDESIGNS136_2001RCECO = 1041,
STEELDESIGNS136_1996RCECO = 1043,
STEELDESIGNAISC_CAST = 1102,
STEELDESIGNBS5950_1990 = 1104,
STEELDESIGNLRFD_CAST = 1111,
STEELDESIGNAISC_N690 = 1202,
```

CommandNo

This argument is for specifying the STEEL DESIGN Parameters. One has to call this method for each PARAMETER type. Valid values are:

```
scSteelParameterFyld = 9100,
scSteelParameterSteelType = 9101,
scSteelParameterWstr = 9110,
scSteelParameterPy = 9120,
scSteelParameterPyRussian = 9121,
scSteelParameterLx = 9125,
scSteelParameterLy = 9130,
scSteelParameterLz = 9140,
scSteelParameterUnl = 9150,
scSteelParameterUnlRussian = 9151,
scSteelParameterDmax = 9160,
scSteelParameterDmin = 9170,
scSteelParameterWmin = 9180,
scSteelParameterLvv = 9190,
scSteelParameterStiff = 9200,
scSteelParameterStiffBs5400 = 9201,
scSteelParameterStiffEC3 = 9202,
scSteelParameterDff = 9210,
scSteelParameterDffRussian = 9211,
scSteelParameterMax = 9220,
scSteelParameterMin = 9230,
scSteelParameterKx = 9235,
scSteelParameterKy = 9240,
scSteelParameterKz = 9250,
scSteelParameterNsf = 9260,
scSteelParameterUnf = 9270,
scSteelParameterCb = 9280,
scSteelParameterCbRussianOrBS5950 = 9281,
scSteelParameterSsy = 9290,
scSteelParameterSsyNs = 9291,
scSteelParameterSsz = 9300,
scSteelParameterSszNs = 9301,
scSteelParameterCmy = 9310,
scSteelParameterCmz = 9320,
scSteelParameterMain = 9330,
scSteelParameterMainBs5950 = 9331,
```

```
scSteelParameterMainBs5400 = 9332,
scSteelParameterMainRussian = 9333,
scSteelParameterMainIs800 = 9334,
scSteelParameterTmainIs800 = 9335,
scSteelParameterMainAIJ = 9336,
scSteelParameterPunch = 9340,
scSteelParameterTrack = 9350,
scSteelParameterTrackNorway = 9351,
scSteelParameterTrackBs5400 = 9352,
scSteelParameterTrackEc3 = 9353,
scSteelParameterTrackDin18800 = 9354,
scSteelParameterTrackFrench = 9355,
scSteelParameterTrackRussian = 9356,
scSteelParameterTrackBs5950 = 9357,
scSteelParameterTrackJapanese = 9358,
scSteelParameterRatio = 9360,
scSteelParameterWeld = 9370,
scSteelParameterWeldBs5950 = 9371,
scSteelParameterBeam = 9380,
scSteelParameterBeamBs5950 = 9381,
scSteelParameterBeamEc3 = 9382,
scSteelParameterBeamDin18800 = 9383,
scSteelParameterBeamNorway = 9384,
scSteelParameterBeamFrench = 9385,
scSteelParameterDj1 = 9390,
scSteelParameterDj2 = 9400,
scSteelParameterCy = 9410,
scSteelParameterCz = 9420,
scSteelParameterBy = 9430,
scSteelParameterBz = 9440,
scSteelParameterMf = 9450,
scSteelParameterSgr = 9460,
scSteelParameterSgrEc3 = 9461,
scSteelParameterSgrDin18800 = 9462,
scSteelParameterSgrRussian = 9463,
scSteelParameterSgrBS5950 = 9464,
scSteelParameterEN1993NA = 9465,
scSteelParameterSgrAS4100 = 9466,
scSteelParameterSblt = 9470,
scSteelParameterSbltRussian = 9471,
scSteelParameterSbltBs5950 = 9472,
scSteelParameterCmm = 9480,
scSteelParameterCmmBs5950 = 9481,
scSteelParameterCmmEc3 = 9482,
scSteelParameterCmmDin18800 = 9483,
scSteelParameterCmmRussian = 9484,
scSteelParameterCmn = 9490,
scSteelParameterCmnBs5950 = 9491,
scSteelParameterCmnEc3 = 9492,
scSteelParameterCmnDin18800 = 9493,
scSteelParameterCmnRussian = 9494,
scSteelParameterLeg = 9500,
```

```
scSteelParameterLegBs5950OrEC3 = 9501,
scSteelParameterLegRussian = 9502,
scSteelParameterFSJApi = 9503,
scSteelParameterGm0EC3 = 9504,
scSteelParameterGm1EC3 = 9505,
scSteelParameterGm2EC3 = 9506,
scSteelParameterZGEC3 = 9507,
scSteelParameterFabEC3 = 9508,
scSteelParameterWet = 9510,
scSteelParameterProfile = 9520,
scSteelParameterTb = 9530,
scSteelParameterEst = 9540,
scSteelParameterC1 = 9550,
scSteelParameterC2 = 9560,
scSteelParameterC3 = 9565,
scSteelParameterEta = 9570,
scSteelParameterGrade = 9580,
scSteelParameterCompression = 9590,
scSteelParameterCompressionSpanish = 9591,
scSteelParameterTension = 9600,
scSteelParameterTensionAlpha = 9601,
scSteelParameterPfy = 9610,
scSteelParameterPfz = 9620,
scSteelParameterSfy = 9630,
scSteelParameterSfz = 9640,
scSteelParameterSby = 9641,
scSteelParameterSbz = 9642,
scSteelParameterUnt = 9650,
scSteelParameterUnb = 9660,
scSteelParameterTorsion = 9670,
scSteelParameterCMT = 9671,
scSteelParameterTOM = 9672,
scSteelParameterEFT = 9673,
scSteelParameterALH = 9674,
scSteelParameterBET = 9675,
scSteelParameterGST = 9676,
scSteelParameterMthEC3 = 9677,
scSteelParameterTmp = 9680,
scSteelParameterTorsionEC3 = 9688,
scSteelParameterEstiff = 9690,
scSteelParameterMU = 9695,
scSteelParameterKc = 9696,
scSteelParameterElbEC3  = 9697,
scSteelParameterPnl = 9700,
scSteelParameterFu = 9705,
scSteelParameterCmp = 9710,
scSteelParameterDia = 9715,
scSteelParameterHgt = 9720,
scSteelParameterCyc = 9725,
scSteelParameterDr1 = 9730,
scSteelParameterDr2 = 9735,
scSteelParameterWid = 9740,
```

```
scSteelParameterFpc = 9745,
scSteelParameterImp = 9750,
scSteelParameterPlt = 9755,
scSteelParameterPlw = 9760,
scSteelParameterRbh = 9765,
scSteelParameterRbw = 9770,
scSteelParameterShr = 9775,
scSteelParameterThk = 9780,
scSteelParameterFlx = 9781,
scSteelParameterTsa = 9782,
scSteelParameterCwy = 9783,
scSteelParameterCbCanadian = 9785,
scSteelParameterCmyCanadian = 9790,
scSteelParameterCmzCanadian = 9795,
scSteelParameterIst = 9800,
scSteelParameterPhi = 9801,
scSteelParameterNsc = 9802,
scSteelParameterAlm = 9803,
scSteelParameterAlb = 9804,
scSteelParameterKt = 9805,
scSteelParameterLt = 9806,
scSteelParameterSKt = 9807,
scSteelParameterSKl = 9808,
scSteelParameterSKr = 9809,
scSteelParameterGammaC1 = 9810,
scSteelParameterGammaC2 = 9815,
scSteelParameterGammaM = 9816,
scSteelParameterENMain = 9817,
scSteelParameterENSgr = 9818,
scSteelParameterMises = 9819,
scSteelParameterMLT = 9820,
scSteelParameterMLT_JL = 9821,
scSteelParameterMYX = 9830,
scSteelParameterMYX_JL = 9831,
scSteelParameterMX = 9840,
scSteelParameterMX_JL = 9841,
scSteelParameterMY = 9850,
scSteelParameterMY_JL = 9851,
scSteelParameterSway = 9860,
scSteelParameterEla  = 9861,
scSteelParameterElb  = 9862,
scSteelParameterDbl  = 9863,
scSteelParameterFyb  = 9864,
scSteelParameterFvb  = 9865,
scSteelParameterNhl  = 9866,
scSteelParameterMainAsce = 9867,
scSteelParameterTaper = 9868,
scSteelParameterSame = 9870,
scSteelParameterCnsf = 9871,
scSteelParameterDangle = 9872,
scSteelParameterGusset = 9873,
scSteelParameterLdr = 9874,
```

```
scSteelParameterIrr = 9875,
scSteelParameterIno = 9876,
scSteelParameterImm = 9877,
scSteelParameterCmb = 9878,
scSteelParameterDsd = 9879,
scSteelParameterOvr = 9880,
scSteelParameterWMax = 9881,
scSteelParameterFss = 9882,
scSteelParameterCan = 9883,
scSteelParameterSopen = 9884,
scSteelParameterEopen = 9885,
scSteelParameterCty = 9886,
scSteelParameterThe = 9887,
scSteelParameterEdi = 9888,
scSteelParameterDcf = 9889,
scSteelParameterCog = 9890,
scSteelParameterSpa = 9891,
scSteelParameterAxis = 9892,
scSteelParameterShear = 9893,
scSteelParameterStp = 9894,
scSteelParameterRHole = 9895,
scSteelParameterRDim = 9896,
scSteelParameterCHole = 9897,
scSteelParameterCDia = 9898,
scSteelParameterHEcc = 9899,
scSteelParameterElectrode = 9900,
scSteelParameterNT = 9901,
scSteelParameterAD = 9902,
scSteelParameterDinc = 9903,
scSteelParameterFinc = 9904,
scSteelParameterFtin = 9905,
scSteelParameterFbin = 9906,
scSteelParameterBmax = 9907,
scSteelParameterIncludeSeismicProvisions = 9908,
scSteelParameterBracedFrameCondition = 9909,
scSteelParameterHoleDiameter = 9910,
scSteelParameterBracedLocation_Major = 9911,
scSteelParameterBracedLocation_Minor_OuterFlange = 9912,
scSteelParameterBracedLocation_Minor_InnerFlange = 9913,
scSteelParameterBearingWidthStart = 9914,
scSteelParameterInsulationThickness = 9915,
scSteelParameterCladding = 9916,
scSteelParameterByPassCondition = 9917,
scSteelParameterSimpleSpanCondition = 9918,
scSteelParameterGalva = 9919,
scSteelParameterBeamRCeco = 9920,
scSteelParameterBearingWidthEnd = 9921,
scSteelParameterSlf = 9922,
scSteelParameterMethod = 9923,
scSteelParameterCT = 9924,
```

Additional parameters used by IS800 LSD:

```
scSteelParameterCmx = 9925,
scSteelParameterAlpha = 9926,
scSteelParameterDBS = 9927,
scSteelParameterPSI = 9928,
scSteelParameterLAT = 9929,
scSteelParameterPLG = 9930,
scSteelParameterTST = 9931,
scSteelParameterTSP = 9932,
scSteelParameterLST = 9933,
scSteelParameterAVG = 9934,
scSteelParameterAVN = 9935,
scSteelParameterATG = 9936,
scSteelParameterATN = 9937,
```

Additional parameters used by IS800 LSD code:

```
scSteelParameterLHT = 9938,
scSteelParameterPBrace = 9939,
```

Additional parameters used by NORSOK code:

```
scSteelParameterHYD = 9940,
scSteelParameterPSD = 9941,
```

Additional parameters used by NORSOK and AISC N690 1984/1994 codes:

```
scSteelParameterSFC = 9942,
scSteelParameterSFT = 9943,
scSteelParameterSMZ = 9944,
scSteelParameterSMY = 9945,
```

Additional parameters used by AISC N690 1984 and NORSOK codes:

```
scSteelParameterEqn = 9946,
```

Additional parameters used by NORSOK and NF3000 codes:

```
scSteelParameterSRL = 9947,
scSteelParameterKS = 9948,
scSteelParameterKV = 9949,
scSteelParameterKBK = 9950,
```

Additional parameters used by NF3000 code:

```
scSteelCheckCode = 9981,
scSteelSelect = 9982,
scSteelSelectOptimized = 9983,
scSteelSelectWeld = 9984,
scSteelSelectWeldTruss = 9985,
scSteelFixedGroup = 9986,
scSteelGroup = 9987,
```

```
scSteelTakeOff = 9988,
scSteelMemberTakeOff = 9989,
```

IntValuesArray

An integer variable array, which specify the values for the parameter where required. Refer to the Technical Reference Manual and the International Design Codes Manual for appropriate values.

Note: Do not pass a null array as this may result in errors or unexpected output.

FloatValuesArray

An floating variable array, which specify the values for the parameter where required. Refer to the Technical Reference Manual and the International Design Codes Manual for appropriate values.

Note: Do not pass a null array as this may result in errors or unexpected output.

StringValuesArray

An string variable array, which specify the values for the parameter where required. Refer to the Technical Reference Manual and the International Design Codes Manual for appropriate values.

Note: Do not pass a null array as this may result in errors or unexpected output.

AssignList

A long array containing the member numbers to which the steel design command is assigned.

VB Example

```
Dim DesignCode As Long
' STEELDESIGNBS5950 = 1004,
DesignCode = 1004
Dim CommandNo As Long
'scSteelParameterTrackBs5950 = 9357,
CommandNo = 9357
Dim IntValuesArray(1) As Long
'Track 2
IntValuesArray(0) = 2
Dim FloatValuesArray(1) As Double
'No floats
FloatValuesArray(0) = 0.0
Dim StringValuesArray(1) As String
'No strings
StringValuesArray(0)="4.0"
```

```
Dim AssignList(1) As Long
'Member 10
AssignList(0) = 10
nReturn = objOpenSTAAD.Command.CreateSteelDesignCommand  (DesignCode,
      CommandNo,  IntValuesArray,  FloatValuesArray,  StringValuesArray,
       AssignList)
```

Command.PerformAnalysis

Perform a first order elastic analysis of the current structure and produce output with specified print option.

VB Syntax

Command.PerformAnalysis *PrintOption*

Where:

PrintOption

An integer variable, which specifies your choice of print command used with Performa Analysis command. The values may vary as:

0 = No Print

1 = Print Load Data

2 = Print Statics Check

3 = Print Statics Load

4 = Print Mode Shapes

5 = Print Both

6 = Print All

VB Example

```
Dim intPrintOption as Integer
intPrintOption = 2
'Get the application object --
'Perform Analysis
objOpenSTAAD.Command.PerformAnalysis intPrintOption
```

Command.PerformPDeltaAnalysisConverge

Perform second order P-Delta analysis of the current structure and produce output with specified print option. Here the program checks whether the solution is converging as successive iterations are performed. If the successive iterations result in divergent values of displacements for a particular load case, then the iterations are discontinued for the specific load case.

VB Syntax

Command.PerformPDeltaAnalysisConverge *NumberOfIteration, PrintOption*

Where:

NumberOfIteration

An integer variable, which specify the no of iteration to be performed.

PrintOption

An integer variable, which specify the user's choice of print command with P-Delta Analysis command. The values may vary as:

0 = No Print

1 = Print Load Data

2 = Print Statics Check

3 = Print Statics Load

4 = Print Mode Shapes

5 = Print Both

6 = Print All

VB Example

```
Dim intNumberOfIteration as Integer
Dim intPrintOption as Integer
'Get the application object --
'P-Delta Analysis with Converge
objOpenSTAAD.Command.PerformPDeltaAnalysisNoConverge
      intNumberOfIteration,  intPrintOption
```

Command.PerformPDeltaAnalysisNoConverge

Performs a second order, P-Delta analysis of the current structure and produce output with specified print option. Here, the program will perform the specified number of iterations whether or not the solution converges.

VB Syntax

Command.PerformPDeltaAnalysisNoConverge*NumberOfIteration, PrintOption*

Where:

NumberOfIteration

An integer variable, which specify the no of iteration to be performed.

PrintOption

An integer variable, which specifies your choice of print command with P-Delta analysis command. The values may vary as:

0 = No Print

1 = Print Load Data

2 = Print Statics Check

3 = Print Statics Load

4 = Print Mode Shapes

5 = Print Both

6 = Print All

VB Example

```
Dim intNumberOfIteration as Integer
Dim intPrintOption as Integer
intNumberOfIteration = 25
intPrintOption = 0
'Get the application object --
'P-Delta Analysis no Converge
objOpenSTAAD.Command.PerformPDeltaAnalysisNoConverge
      intNumberOfIteration,  intPrintOption
```

4.8 Output Results Functions

Output.GetAllPlateCenterForces

Returns plate center forces for the specified plate and load case.

VB Syntax

Output.GetAllPlateCenterForces *PlateNo, LoadCase, CenterForcesArray*

Where:

PlateNo

A long variable contains plate element no.

LoadCase

A long variable contains load case no.

CenterForcesArray

A five dimensional array which returns plate center forces.

VB Example

```
'Get the application object --
'GetPlateCenter Forces
objOpenSTAAD.Output.GetAllPlateCenterForces (lPlateNo, lLoadCase,
      dCenterForcesArray)
```

Output.GetAllPlateCenterMoments

Returns plate center moments for the specified plate and load case.

VB Syntax

Output.GetAllPlateCenterMoments*PlateNo, LoadCase, CenterMomentsArray*

Where:

PlateNo

A long variable contains plate element no.

LoadCase

A long variable contains load case no.

CenterMomentsArray

A long array of dimension 3, which returns plate center moments.

VB Example

```
'Get the application object --
'GetPlateCenter Moments
objOpenSTAAD.Output.GetAllPlateCenterMoments (lPlateNo, lLoadCase,
      lCenterMomentsArray)
```

Output.GetAllPlateCenterPrincipalStressesAndAngles

Returns plate center stresses and angles for the specified plate and load case.

VB Syntax

Output.GetAllPlateCenterPrincipalStressesAndAngles *PlateNo, LoadCase, StressesAndAnglesArray*

Where:

PlateNo

A long variable contains plate element no.

LoadCase

A long variable contains load case no.

StressesAndAnglesArray

A long array of dimension 8, which returns plate center stresses and angles.

VB Example

```
'Get the application object --
'GetPlateCenter Principal Stresses and Angles
objOpenSTAAD.Output.GetAllPlateCenterPrincipalStressesAndAngles
      (lPlateNo, lLoadCase, _
lStressesAndAnglesArray)
```

Output.GetAllPlateCenterStressesAndMoments

Returns plate center stresses and moments for the specified plate for specified load case.

VB Syntax

Output.GetAllPlateCenterStressesAndMoments *PlateNo, LoadCase, CenterStressesArray*

Where:

PlateNo

A long variable contains plate element no.

LoadCase

A long variable contains load case no.

CenterStressesArray

An eight dimensional array which returns plate center stresses and moments.

VB Example

```
'Get the application object --
'GetAllPlateCenterStresses And Moments
objOpenSTAAD.Output.GetAllPlateCenterStressesAndMoments (lPlateNo,
      lLoadCase, dCenterStressesArray)
```

Output.GetAllSolidNormalStresses

Returns normal stresses for the specified solid element at the specified corner.

VB Syntax

Output.GetAllSolidNormalStresses *SolidNo, Corner, LoadCase, StressesArray*

Where:

SolidNo

A long variable contains solid element no.

Corner

A long variable specifies the corner of that solid element.

LoadCase

A long variable contains load case no.

StressesArray

A long array of dimension 3, which returns solid corner stresses.

VB Example

```
'Get the application object --
'Get Solid Corner Normal Stresses
objOpenSTAAD.Output.GetAllSolidNormalStresses (lSolidNo, lCorner,
      lLoadCase, lStressesArray)
```

Output.GetAllSolidPrincipalStresses

Returns principal stresses for the specified solid element at the specified corner.

VB Syntax

```
Output.GetAllSolidPrincipalStresses SolidNo, Corner, LoadCase,
      PrincipalStressesArray
```

Where:

SolidNo

A long variable contains solid element no.

Corner

A long variable specifies the corner of that solid element. (-1 = Center, 1 to 8 all corner nodes).

LoadCase

A long variable contains load case no.

PrincipalStressesArray

A long array of dimension 3, which returns solid corner principal stresses.

VB Example

```
'Get the application object --
'Get Solid Corner Principal Stresses
objOpenSTAAD.Output.GetAllSolidPrincipalStresses (lSolidNo, lCorner,
     lLoadCase, lPrincipalStressesArray)
```

Output.GetAllSolidShearStresses

Returns shear stresses for the specified solid element at the specified corner.

VB Syntax

```
Output.GetAllSolidShearStresses SolidNo, Corner, LoadCase, ShearStressesArray
```

Where:

SolidNo

A long variable contains solid element no.

Corner

A long variable specifies the corner of that solid element. (-1 = Center, 1 to 8 all corner nodes).

LoadCase

A long variable contains load case no.

ShearStressesArray

A long array of dimension 3, which returns solid corner shear stresses.

VB Example

```
'Get the application object --
'Get Solid Corner Shear Stresses
objOpenSTAAD.Output.GetAllSolidShearStresses (lSolidNo, lCorner,
     lLoadCase, lShearStressesArray)
```

Output.GetAllSolidVonMisesStresses

Returns normal stresses for the specified solid element at the specified corner.

VB Syntax

Output.GetAllSolidVonMisesStresses *SolidNo, Corner, LoadCase, VonMisesStressesArray*

Where:

SolidNo

A long variable contains solid element no.

Corner

A long variable specifies the corner of that solid element. (-1 = Center, 1 to 8 all corner nodes).

LoadCase

A long variable contains load case no.

VonMisesStressesArray

A long array of dimension 3, which returns solid corner Von-Mises stresses.

VB Example

```
'Get the application object --
'Get Solid Corner Von Stresses
objOpenSTAAD.Output.GetAllSolidVonMisesStresses (lSolidNo, lCorner,
     lLoadCase, lVonMisesStressesArray)
```

Output.GetIntermediateMemberForcesAtDistance

Returns sectional forces and moments for specified member number, distance, and load case.

VB Syntax

Output.GetIntermediateMemberForcesAtDistance *MemberNo, Distance, LoadCase, DisplacementArray*

Where:

nMemberNo

Long variable contains the member number.

Distance

Double variable contains distance from the start of the memberd in length units.

nLoadCase

Long variable contains the load case number.

pdForcesArray

A six dimensional long array, which returns member sectional forces and moments.

VB Example

```
'Get the application object --
'Get Member Intermediate Forces
objOpenSTAAD.Output.GetIntermediateMemberForcesAtDistance lMemberNo,
     dDistance, lLoadCase, _
lForceArray
```

Output.GetIntermediateMemberTransDisplacements

Returns sectional displacements for specified member number, distance, and load case.

VB Syntax

```
Output.GetIntermediateMemberTransDisplacements MemberNo, Distance, LoadCase,
      DisplacementArray
```

Where:

MemberNo

Long variable contains the member number.

Distance

Double variable contains distance from starting end in terms of member length.

LoadCase

Long variable contains the load case number.

DisplacementArray

A six dimensional long array, which returns member sectional displacements.

VB Example

```
'Get the application object --
'Get Member Intermediate Displacements
```

```
objOpenSTAAD.Output.GetIntermediateMemberTransDisplacements lMemberNo,
      dDistance, lLoadCase, _
lDisplacementArray
```

Output.GetMemberEndDisplacements

Returns member end displacements for specified member number, member end and load case.

VB Syntax

Output.GetMemberEndDisplacements *MemberNo, End, LoadCase, DisplacementArray*

Where:

MemberNo

Long variable contains the member number.

End

Long variable contains member end.

0 = Start

1 = End

LoadCase

Long variable contains the load case number.

DisplacementArray

A long array of dimension 6, which returns member end displacements.

VB Example

```
'Get the application object --
'Get Member End Displacements
objOpenSTAAD.Output.GetMemberEndDisplacements (lMemberNo, lEnd,
      lLoadCase, lDisplacementArray)
```

Output.GetMemberEndForces

Returns member end forces for specified member number, member end and load case.

VB Syntax

Output.GetMemberEndForces *MemberNo, End, LoadCase, ForceArray*

Where:

MemberNo

Long variable contains the member number.

End

Long variable contains member end.

0. = Start
1. = End

LoadCase

Long variable contains the load case number.

ForceArray

A six dimensional array which returns member end forces.

VB Example

```
'Get the application object --
'Get Member End Forces
objOpenSTAAD.Output.GetMemberEndForces (lMemberNo, lEnd, lLoadCase,
      dForceArray)
```

Output.GetNodeDisplacements

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

VB Syntax

Output.GetNodeDisplacements*NodeNo, LoadCase, DisplacementArray*

Where:

NodeNo

Long variable contains the node number.

LoadCase

Long variable contains the load case number.

DisplacementArray

A double array of dimension six, which returns nodal displacements.

VB Example

```
'Get the application object --
'Get Nodal Displacement
objOpenSTAAD.Output.GetNodeDisplacements (lNodeNo, lLoadCase,
      dDisplacementArray)
```

Output.GetOutputUnitForDensity

Returns output unit for material density used by the GUI as a string and is set in the Options dialog box.

VB Syntax

```
Output.GetOutputUnitForDensity Unit
```

Where:

Unit

A string containing the output unit for material density.

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForDensity as String
'Get the application object --
'Get Output Unit For Material Density
strOutputUnitForDensity = objOpenSTAAD.Output.GetOutputUnitForDensity
```

Output.GetOutputUnitForDimension

Returns output unit for dimension used by the GUI as a string and is set in the Options dialog box.

VB Syntax

```
Output.GetOutputUnitForDimension Unit
```

Where:

Unit

A string containing the output unit for dimension.

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForDimension as String
```

```
'Get the application object --
'Get Output Unit For Dimension
strOutputUnitForDimension =
      objOpenSTAAD.Output.GetOutputUnitForDimension
```

Output.GetOutputUnitForDisplacement

Returns output unit for translational displacements used by the GUI as a string and is set in the Options dialog box.

VB Syntax

Output.GetOutputUnitForDisplacement *Unit*

Where:

Unit

A string containing the output unit for translational displacements.

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForDisplacement as String
'Get the application object --
'Get Output Unit For Displacement
strOutputUnitForDisplacement =
      objOpenSTAAD.Output.GetOutputUnitForDisplacement
```

Output.GetOutputUnitForDistForce

Returns output unit for distributed forces used by the GUI as a string and is set in the Options dialog box.

VB Syntax

Output.GetOutputUnitForDistForce *Unit*

Where:

Unit

A string containing the output unit for distributed forces.

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForDistForce as String
'Get the application object --
'Get Output Unit For Distributed Force
strOutputUnitForDistForce =
      objOpenSTAAD.Output.GetOutputUnitForDistForce
```

Output.GetOutputUnitForDistMoment

Returns output unit for distributed moments used by the GUI as a string and is set in the Options dialog box.

VB Syntax

Output.GetOutputUnitForDistMoment *Unit*

Where:

Unit

A string containing the output unit for distributed moments.

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForDistMoment as String
'Get the application object --
'Get Output Unit For Distributed Moment
strOutputUnitForDistMoment =
      objOpenSTAAD.Output.GetOutputUnitForDistMoment
```

Output.GetOutputUnitForForce

Returns output unit for forces (FX, FY, FZ) used by the GUI as a string and is set in the Options dialog box.

VB Syntax

Output.GetOutputUnitForForce *Unit*

Where:

Unit

A string containing the output unit for forces (FX, FY, FZ).

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForForce as String
'Get the application object --
'Get Output Unit For Force
strOutputUnitForForce = objOpenSTAAD.Output.GetOutputUnitForForce
```

Output.GetOutputUnitForMoment

Returns output unit for moments (MX, MY, MZ) used by the GUI as a string and is set in the Options dialog box.

VB Syntax

Output.GetOutputUnitForMoment *Unit*

Where:

Unit

A string containing the output unit for moments (MX, MY, FZ).

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForMoment as String
'Get the application object --
'Get Output Unit For Moment
strOutputUnitForMoment = objOpenSTAAD.Output.GetOutputUnitForMoment
```

Output.GetOutputUnitForRotation

Returns output unit for rotational displacement used by the GUI as a string and is set in the Options dialog box.

VB Syntax

```
Output.GetOutputUnitForRotation Unit
```

Where:

Unit

A string containing the output unit for rotational displacement.

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForRotation as String
'Get the application object --
'Get Output Unit For Rotation
strOutputUnitForRotation = objOpenSTAAD.Output.GetOutputUnitForRotation
```

Output.GetOutputUnitForSectArea

Returns output unit for cross sectional area used by the GUI as a string and is set in the Options dialog box.

VB Syntax

```
Output.GetOutputUnitForSectArea Unit
```

Where:

Unit

A string containing the output unit for cross sectional area.

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForSectArea as String
'Get the application object --
'Get Output Unit For Sectional Area
strOutputUnitForSectArea = objOpenSTAAD.Output.GetOutputUnitForSectArea
```

Output.GetOutputUnitForSectDimension

Returns output unit for cross sectional dimension(s) used by the GUI as a string and is set in the Options dialog box.

VB Syntax

```
Output.GetOutputUnitForSectDimension Unit
```

Where:

Unit

A string containing the output unit for sectional dimension(s).

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForSectDimension as String
'Get the application object --
'Get Output Unit For Sectional Dimension
strOutputUnitForSectDimension =
      objOpenSTAAD.Output.GetOutputUnitForSectDimension
```

Output.GetOutputUnitForSectInertia

Returns output unit for sectional inertia used by the GUI as a string and is set in the Options dialog box.

VB Syntax

```
Output.GetOutputUnitForSectInertia Unit
```

Where:

Unit

A string containing the output unit for sectional inertia.

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForSectInertia as String
'Get the application object --
'Get Output Unit For Sectional Inertia
strOutputUnitForSectInertia =
      objOpenSTAAD.Output.GetOutputUnitForSectInertia
```

Output.GetOutputUnitForSectModulus

Returns output unit for sectional modulus used by the GUI as a string and is set in the Options dialog box.

VB Syntax

Output.GetOutputUnitForSectModulus *Unit*

Where:

Unit

A string containing the output unit for sectional modulus.

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForSectModulus as String
'Get the application object --
'Get Output Unit For Sectional Modulus
strOutputUnitForSectModulus =
      objOpenSTAAD.Output.GetOutputUnitForSectModulus
```

Output.GetOutputUnitForStress

Returns output unit for stress used by the GUI as a string and is set in the Options dialog box.

VB Syntax

Output.GetOutputUnitForStress *Unit*

Where:

Unit

A string containing the output unit for stress.

Note: This is not a conversion unit used in accessing data through OpenSTAAD. Values are always returned in the units of the base unit system, which can be obtained using the **GetBaseUnit function**.

VB Example

```
Dim strOutputUnitForStress as String
'Get the application object --
'Get Output Unit For Stress
strOutputUnitForStress = objOpenSTAAD.Output.GetOutputUnitForStress
```

Output.GetPlateCenterNormalPrincipalStresses

Returns Smax and Smin for Top and Bottom of the specified plate.

VB Syntax

Output.GetPlateCenterNormalPrincipalStresses *PlateNo, LoadCase, SMAXTop, SMINTop, SMAXBottom, SMINBottom*

Where:

PlateNo

A long variable contains plate element no.

LoadCase

A long variable contains load case no.

SMAXTop, SMINTop, SMAXBottom, SMINBottom

Double variables, which return the corresponding values for the center of the specified plate.

VB Example

```
'Get the application object --
'GetPlateCenter Normal Principal Stresses
objOpenSTAAD.Output.GetPlateCenterNormalPrincipalStresses (lPlateNo,
     lLoadCase, dSMAXTop, dSMINTop, _
dSMAXBottom, dSMINBottom)
```

Output.GetPlateCenterVonMisesStresses

Returns plate center Von-Mises Top and Bottom for the specified plate and load case.

VB Syntax

Output.GetPlateCenterVonMisesStresses *PlateNo, LoadCase, VONT, VONB*

Where:

PlateNo

A long variable contains plate element no.

LoadCase

A long variable contains load case no.

VONT, VONB

Double variables return the value of Von-Mises Top and Bottom for the specified plate center point.

VB Example

```
'Get the application object --
'GetPlateCenterCenterVon-Mises Stresses
objOpenSTAAD.Output.GetPlateCenterVonMisesStresses (lPlateNo, lLoadCase,
      dVONT, dVONB)
```

Output.GetSupportReactions

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

VB Syntax

Output.GetSupportReactions *NodeNo, LoadCase, ReactionArray*

Where:

NodeNo

Long variable contains the node number, which is supported.

LoadCase

Long variable contains the load case number.

ReactionArray

A long array of dimension 6, which returns support reactions.

VB Example

```
'Get the application object --
'Get Support Reaction
objOpenSTAAD.Output.GetSupportReactions (lNodeNo, lLoadCase,
      dReactionArray)
```

4.9 Results Tables Functions

Table.AddTable

Adds a table to the specified *ReportNo*.

VB Syntax

Table.AddTable *ReportNo, szTableName, NumRows, NumCols*

Where:

ReportNo

Long variable containing the report number to which this table will be added.

szTableName

A null terminated string containing the name of the table.

NumRows, NumCols

Long variables providing the number of rows and columns of the table.

Return Value

A long value containing the reference number for the table created to be used to access the table.

VB Example

```
'Get the application object --
'Add table to report no 1 with 10 rows and 5 columns
NumRows = 10
NumCols = 5
TableNo = objOpenSTAAD.Table.AddTable 1, "My Table", NumRows, NumCols
```

Table.CreateReport

Creates a report with the specified title.

VB Syntax

Table.CreateReport"*string szReportTitle*"

Where:

szReportTitle

A null terminated string containing the title of the report.

Return Value

A long value containing the reference number for the report created to be used to access the report.

VB Example

```
'Get the application object --
'Create report
ReportNo = objOpenSTAAD.Table.CreateReport "My Report"
```

Table.DeleteTable

Deletes a table specified by *TableNo* in a report specified by *ReportNo*.

VB Syntax

Table.DeleteTable *ReportNo, TableNo*

Where:

ReportNo

Long variable containing the report number from which a table is to be deleted.

TableNo

Long variable containing the table number to be deleted.

VB Example

```
'Get the application object --
'Delete Table
objOpenSTAAD.Table.DeleteTable ReportNo, TableNo
```

Table.GetCellValue

Gets a value in the cell of table at the specified row and column in a report.

VB Syntax

Table.GetCellValue *ReportNo, TableNo, RowNo, ColNo, szValue*

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

RowNo, ColNo

Long variables containing the row and column number of the cell.

szValue

String variable in which the value in the cell will be copied .

Return Value

Copies the data in the specified cell into the string provided.

VB Example

```
Dim szValue As String
'Get the application object --
'Set value to cell
RowNo = 2
ColNo = 5
objOpenSTAAD.Table.GetCellValue ReportNo, TableNo, Rowno, ColNo, szValue
```

Table.GetReportCount

Returns the number of reports created.

VB Syntax

```
Table.GetReportCount ()
```

Return Value

A long value containing the number of reports created.

VB Example

```
'Get the application object --
'Get number of reports
ReportNos = objOpenSTAAD.Table.GetReportCount
```

Table.GetTableCount

Returns the number of tables created.

VB Syntax

```
Table.GetTableCount ReportNo
```

Where:

ReportNo

Long variable containing the report number.

Return Value

A long value containing the number of tables in the report.

VB Example

```
'Get the application object --
'Get number of reports
TableNos = objOpenSTAAD.Table.GetTableCount ReportNo
```

Table.RenameTable

Renames a table in a report specified by *TableNo*.

VB Syntax

Table.RenameTable *ReportNo, TableNo, szNewTableName*

Where:

ReportNo

Long variable containing the report number whose table will be renamed.

TableNo

Long variable containing the table number to be renamed.

szNewTableName

A null terminated string containing the new name for the table.

VB Example

```
'Get the application object --
'Rename Table
objOpenSTAAD.Table.RenameTable ReportNo, TableNo, "My New Table"
```

Table.ResizeTable

Resizes a table specified by *TableNo* in a report specified by *ReportNo*.

VB Syntax

Table.ResizeTable *ReportNo, TableNo, NumRows, NumCols*

Where:

ReportNo

Long variable containing the report number whose table will be resized.

TableNo

Long variable containing the table number to be resized.

NumRows, NumCols

Long variables providing the new number of rows and columns of the table.

VB Example

```
'Get the application object --
'Resize Table
NumRows = 10
NumCols = 5
objOpenSTAAD.Table.ResizeTable ReportNo, TableNo, NumRows, NumCols
```

Table.SaveReport

Saves the specified report along with all its tables.

VB Syntax

Table.SaveReport *ReportNo*

Where:

ReportNo

A long variable providing the report number identifying the STAAD report which is to be saved.

VB Example

```
'Get the application object --
'Save report
objOpenSTAAD.Table.SaveReport ReportNo
```

Table.SaveReportAll

Saves all the reports created.

VB Syntax

Table.SaveReportAll ()

VB Example

```
'Get the application object --
'Save all reports
objOpenSTAAD.Table.SaveReportAll
```

Table.SaveTable

Saves a table in a report specified by *TableNo*.

VB Syntax

```
Table.SaveTable ReportNo, TableNo
```

Where:

ReportNo

Long variable containing the report number whose table will be saved.

TableNo

Long variable containing the table number to be saved.

VB Example

```
'Get the application object --
'Save Table
objOpenSTAAD.Table.SaveTable ReportNo, TableNo
```

Table.SetCellTextBold

Sets the text in a given row and column to bold.

VB Syntax

```
Table.SetCellTextBold ReportNo, TableNo, RowNo, ColNo
```

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

RowNo

Long variable containing the row number.

ColNo

Long variable containing the column number.

VB Example

```
'Get the application object --
‘Set the text in row 9 and column 4 to bold
objOpenSTAAD.Table.SetCellTextBold ReportNo, TableNo, 9, 4
```

Table.SetCellTextColor

Sets the color of the text to be displayed in a particular cell. By default, the color is always set to black.

VB Syntax

```
Table.SetCellTextColor ReportNo, TableNo, RowNo, ColNo, RedVal, GreenVal,
      BlueVal
```

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

RowNo

Long variable containing the row number.

ColNo

Long variable containing the column number.

RedVal, GreenVal, BlueVal

An integer between 0 and 255 that represents the intensity of red, green or blue in the color for the text. A value of 0 for a particular color indicates that no shade of that color will be mixed into the final color.

For example, to have the text written in yellow, the values would be RedVal = 255, GreenVal = 255 and BlueVal = 0.

VB Example

```
'Get the application object --
Dim RedVal As Integer
Dim GreenVal As Integer
Dim BlueVal As Integer
'Set the text to yellow
RedVal = 255
GreenVal = 255
BlueVal = 0
objOpenSTAAD.Table.SetCellTextColor ReportNo, TableNo, RowNo, ColNo,
     RedVal, GreenVal, BlueVal
```

Table.SetCellTextHorzAlignment

Sets the text in a particular row and column to a specified horizontal alignment. By default, all the text is right justified.

VB Syntax

Table.SetCellTextHorzAlignment *ReportNo, TableNo, RowNo, ColNo, Alignment*

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

RowNo

Long variable containing the row number.

ColNo

Long variable containing the column number.

Alignment

An integer variable containing the type of horizontal alignment to apply. The possible values are:

0 = left

1 = center

2 = right

VB Example

```
'Get the application object --
‘Set the horizontal alignment for the text in row 10 and column 6 to
     center-    ‘ justified.
objOpenSTAAD.Table.SetCellTextHorzAlignment ReportNo, TableNo, 10, 6, 1
```

Table.SetCellTextItalic

Italicizes the text in a given row and column.

VB Syntax

Table.SetCellTextItalic *ReportNo, TableNo, RowNo, ColNo*

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

RowNo

Long variable containing the row number.

ColNo

Long variable containing the column number.

VB Example

```
'Get the application object --
‘Italicize the text in row 9 and column 4
objOpenSTAAD.Table.SetCellTextItalic ReportNo, TableNo, 9, 4
```

Table.SetCellTextSize

Sets the text in a particular row and column to a certain font size. The font sizes are equivalent to the ones used in Microsoft Word.

VB Syntax

Table.SetCellTextSize *ReportNo, TableNo, RowNo, ColNo, FontSize*

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

RowNo

Long variable containing the row number.

ColNo

Long variable containing the column number.

FontSize

Double variable containing the size of the font to set the text to.

VB Example

```
'Get the application object --
‘Set the font size for the text in row 10 and column 6 to 18 pt
objOpenSTAAD.Table.SetCellTextSize ReportNo, TableNo, 10, 6, 16.0
```

Table.SetCellTextSizeAll

Sets the text in the entire table to FontSize. The font sizes are equivalent to the ones used in Microsoft Word.

VB Syntax

```
Table.SetCellTextSizeAll ReportNo, TableNo, FontSize
```

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

FontSize

Double variable containing the size of the font to set the entire table to.

VB Example

```
'Get the application object --
‘Set the font size for the text in the table to 16 pt
objOpenSTAAD.Table.SetCellTextSizeAll ReportNo, TableNo, 16.0
```

Table.SetCellTextUnderline

Underlines the text in a given row and column.

VB Syntax

```
Table.SetCellTextUnderline ReportNo, TableNo, RowNo, ColNo
```

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

RowNo

Long variable containing the row number.

ColNo

Long variable containing the column number.

VB Example

```
'Get the application object --
‘Underline the text in row 9 and column 4
objOpenSTAAD.Table.SetCellTextUnderline ReportNo, TableNo, 9, 4
```

Table.SetCellTextVertAlignment

Sets the text in a particular row and column to a specified vertical alignment. By default, all the text is top justified.

VB Syntax

Table.SetCellTextVertAlignment *ReportNo, TableNo, RowNo, ColNo, Alignment*

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

RowNo

Long variable containing the row number.

ColNo

Long variable containing the column number.

Alignment

An integer variable containing the type of vertical alignment to apply. The possible values are:

0 = top

4 = center

8 = bottom

VB Example

```
'Get the application object --
‘Set the horizontal alignment for the text in row 10 and column 6 to
     center-    ‘ justified.
```

Table.SetCellValue

Puts a value into the cell of table at the specified row and column in a report.

VB Syntax

Table.SetCellValue *ReportNo, TableNo, RowNo, ColNo, Data_Type Value*

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

RowNo, ColNo

Long variables containing the row and column number of the cell.

Value

A variable of Data_Type (Integer, Long, Double, String) containing the value to be inserted in the cell.

VB Example

```
'Get the application object --
'Set value to cell
RowNo = 2
ColNo = 5
Value = 5.25       'Declared as Double
objOpenSTAAD.Table.SetCellValue ReportNo, TableNo, Rowno, ColNo, Value
```

Table.SetColumnHeader

Sets the heading of a column.

VB Syntax

Table.SetColumnHeader *ReportNo, TableNo, ColNo, szHeader*

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

ColNo

Long variable containing the column number.

szHeader

String variable providing the column header.

VB Example

```
'Get the application object --
'Set header for the column
Header = "Column 5"
ColNo = 5
objOpenSTAAD.Table.SetColumnHeader ReportNo, TableNo, ColNo, szHeader
```

4.9.1 Table.SetColumnUnitString

Sets the unit string of a column.

VB Syntax

Table.SetColumnUnitString *ReportNo, TableNo, ColNo, szUnitString*

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

ColNo

Long variable containing the column number.

szUnitString

String variable providing the column unit string.

VB Example

```
'Get the application object --
'Set unit for the column
szUnit = "kN/m^2"
ColNo = 5
objOpenSTAAD.Table.SetColumnUnitString ReportNo, TableNo, ColNo, szUnit
```

Table.SetRowHeader

Sets the heading of a row.

VB Syntax

Table.SetRowHeader *ReportNo, TableNo, RowNo, szHeader*

Where:

ReportNo

Long variable containing the report number.

TableNo

Long variable containing the table number.

RowNo

Long variable containing the row number.

szHeader

String variable providing the row header.

VB Example

```
'Get the application object --
'Set header for the row
Header = "Row 5"
RowNo = 5
objOpenSTAAD.Table.SetRowHeader ReportNo, TableNo, RowNo, szHeader
```

Section 5
Troubleshooting

The following is a list of error messages or warnings which may be displayed and methods used to address them.

5.1 Method Object Failed

Error message:

"Method 'OpenSTAADFunctionName' of object IOutput failed.",

where OpenSTAADFunctionName is the name of an OpenSTAAD function. For example:

"Method 'GetMemberBetaAngle' of object IOutput failed."

Check to be sure that you are passing all parameters required by the function. If a parameter is missing, or if the parameter being passed is not valid, the 'Method ...of object IOutput failed' message may appear. For example, the message may appear if a member number is passed to the function, but that member number does not exist in the currently open STAAD file.

You may also receive similar messages if your version of STAAD.Pro is not compatible with OpenSTAAD. OpenSTAAD is compatible with STAAD.Pro 2001 Build 1006.US.REL and higher. You will need to reanalyze your model in that build. This is because we switched the way the data is recorded in Build 1006 to save more space. Build 1006 US.REL was released in November 2001.

UK Builds prior to STAAD.Pro 2002 are not compatible with OpenSTAAD.

5.2 Function is not retrieving correct values

Check to be sure that you have saved the STAAD file after making changes to the input before you run any OpenSTAAD functions that retrieve input information. If you are running STAAD functions that retrieve analysis results, check to be sure that you have saved the STAAD file and re-run the analysis after making any changes to the input.

5.3 Type Mismatch

Error message:

"Type mismatch"

Check to make sure that you have declared all variables using the DIM statement at the beginning of your program or macro, e.g.:

```
Dim pnIsReleased As Integer
Dim pdSpringStiffnesses(0 To 5) As Double
Dim pdPartialMomRelFactors(0 To 2) As Double
```

Confirm that when you pass array variables to the function, you have specified the starting position in the array for the function to use in filling up the array, e.g.:

```
objOpenSTAAD.GetFullMemberReleaseInfoAtStar 3, pnIsReleased, _

pdSpringStiffnesses(0), pdPartialMomRelFactors(0)
```

5.4 Property or Method Not Supported

Error message:

"Object doesn't support this property or method"

Check to make sure you have typed the function name correctly.

5.5 ActiveX Component in Microsoft Excel

Error message:

"ActiveX component can't create object"

when attempting to run an OpenSTAAD macro in Microsoft Excel.

1. Make sure that you have a directory called "OpenSTAAD" inside of your SProV8i folder (i.e., C:\SProV8i\ OpenSTAAD).
2. In that folder, make sure you have the files Openstaad.dll, Staadread.tbl, Staadread.dll, and Steeltable.dll. If not, please contact **Bentley Technical Support**.

3. When you open the Excel file, is it asking you to *Enable Macros*? If so, make sure you press *Yes*. If not, you need to change the security settings in Excel. Select Tools > Macro > Security... from the Excel menu. Make sure the *Security Level* is set to *Medium*. Close Excel down (completely, not just the file) and reopen the file again. This time it should ask you if you want your Macros enabled or disabled. Choose *Enabled*.

4. When you run your macro, if it still gives you the error ("ActiveX component can't create object"), it might be because the OpenSTAAD library was not registered properly when the program was installed. To register the OpenSTAAD dll, close down Excel, go to your Windows *Start* menu and select the *Run* command. Type in:

   ```
   regsvr32 c:\SProV8i\openstaad\openstaad.dll
   ```

   and click **OK** .

   Note: The path to the `Openstaad.dll` file may be different on your computer if you did not use the default installation path provided by the STAAD.Pro Setup program when you installed your software – if so, you should modify the above command to correspond to the actual location on your computer.

   A message dialog opens to display the message, "`DllRegisterServer in c:\SProV8i\openstaad\openstaad.dll succeeded`." If the registration did not succeed, please contact our technical support staff for further instructions.

5. Try opening and running the Excel beam example file provided with your STAAD.Pro software (`C:\SProV8i\OpenSTAAD\Rectangle-beam.xls`).

5.6 User Type Not Defined

Error message:

```
"User Defined type not defined"
```

This message "User Defined type not defined" may appear on the line which declares the OpenSTAAD object variable if the variable is declared "As Output", e.g., "Dim objOpenSTAAD As Output". The message may be appearing because the OpenSTAAD library references have not been included. The VBA compiler therefore does not know which functions are associated with the OpenSTAAD object.

There are two ways to eliminate this error message:

1. Declare the OpenSTAAD object As Object, instead of As Output, e.g.

   ```
   Dim objOpenSTAAD As Object
   ```

2. Include the OpenSTAAD library reference in your VBA editor. This second option has the added benefit that once you do it, the compiler will now recognize the OpenSTAAD object and will pop up a list of functions whenever you refer to the object in your VBA editor.

To include the OpenSTAAD library reference, select **Tools** > **References** in your VBA editor. A dialog box with title “References – Normal” will open. You will see a scroll box inside the dialog box labeled “Available References.” Scroll down through the list of references until you find one called “OpenSTAAD 1.0 Type Library”. Toggle on the corresponding check box, then click **OK**.

Now re-run your macro to see if the problem with the “User Defined type not defined” error message has been solved.

5.7 Files Not Compatible

Error message:

```
"One or more results files are not compatible with the current model and
cannot be loaded..."
```

You need to have a successful analysis before you can run any OpenSTAAD macro, including such simple macros as one that only asks for the number of nodes in the model. The program is not smart enough to know which commands will work and which will not when no results are present. It first checks to see if valid results are available. If they are not, it displays the error message and terminates.

You can use the OpenSTAAD command AreResultsAvailable in your macro to check to see if results are available. In some cases, this function also will result in the error message "One or more results files are not compatible with the current model and cannot be loaded...". That means that even though the STAAD results file with an ANL extension has been created, the file contains only error messages and no meaningful results.

You can also try inserting a FINISH command in your input file preceding the location in the input that is causing the analysis errors. Depending on whether the errors take place before any analysis is performed, you might get rid of the original error message, but in its place you may see the message, "No Results Available." However, the best way to eliminate this problem is to modify your input file so that you obtain a successful analysis before you try to run your OpenSTAAD macro.

Under certain conditions, you might get this “...results files are not compatible...” error message, even though the analysis runs successfully. OpenSTAAD currently will not work with the Moving Load Generator. The reason is that the results from the Moving Load Generator are not kept in the same database as the other STAAD results. Therefore, when OpenSTAAD tries to read the Output File, it is not able to find those missing load results, so it displays an error message and stops processing the macro. This same situation is in effect for any other loads that you are unable to see until you perform the analysis, like UBC loads, for example. If the loads are defined in the input file, OpenSTAAD will work fine, but not with loads that are only generated when the analysis is run, because these results are not kept in a location where OpenSTAAD can find them.

If you remove your moving load generation command from your input file, and run the analysis, you should then be able to run your OpenSTAAD macro.

Section 6
Examples

The following examples have been provided to assist in creating macros using OpenSTAAD in commonly used applications like Microsoft Excel, Autodesk AutoCAD, Microsoft Word.

6.1 STAAD.Pro Macro

This example demonstrates a small macro which can be used within STAAD.Pro.

Often, when learning to program, you begin with a program which simply outlines the basic structure of an application, module, or function; typically resulting with a screen message displaying the phrase "Hello World." This example expands upon this to include the foundation from a practical application as well.

Note: Additional examples in this section demonstrate how to poll STAAD data from external programs.

1. Open STAAD.Pro.
2. Select **Edit** > **Create New VB Macro** to create a new file. In the **Select New Macro Name dialog**, enter a title of **`CreateNewView.vbs`** with the following description:

   `Creates a new view from the selected beams.`

Note: You can save the macro file anywhere, so long as the directory has write permission for your user account.

The STAAD VBA Editor window opens with an subroutine title Main.

3. Just after the description, type the following just after the description comment line:

```
DimobjOpenSTAAD As Object
Dim SelBeamsNo As Long
Dim SelBeams() As Long
```

This is used to provide some declarations of the objects and variables used in this program.

4. Add the following lines to instantiate the OpenSTAAD object:

```
'Launch the OpenSTAAD Object
Set objOpenSTAAD = GetObject(,"StaadPro.OpenSTAAD")
```

Note: The first line beginning with the apostrophe (') is a comment. It isn't necessary, but it is good practice to add remarks such as this to make your code clear to others (as well as to yourself when you revisit the code at a later time).

5. Add the following lines to set up a logical check for if any beams are selected:

```
'Get no. of selected beams
SelBeamsNo = objOpenSTAAD.Geometry.GetNoOfSelectedBeams
If (SelBeamsNo > 0) Then
```

Here, the **GetNoOfSelectedBeams Geometry function** in OpenSTAAD is being used to aid our test. The test is a if... then... else... statement, which continues in the following steps.

6. The following lines will instruct the program what to do if our statement is true (i.e., there is at least one beam selected). That is to create a new view from the active selection using the **CreateNewViewForSelection View function** in OpenSTAAD.

```
  ReDim SelBeams(SelBeamsNo) As Long
  'Create a new view
  objOpenSTAAD.View.CreateNewViewForSelections
```

7. Since this macro might be run with no beams selected, a message can be provided to the user for some feedback in this instance with the following line:

```
  Else
  MsgBox "No beams are currently selected.", vbOkOnly
End If
```

Hint: You could add the message **`Hello World`**, if you prefer to stick with a more traditional introductory example of programming.

8. Add the following statement to close the instance of the OpenSTAAD object:

```
Set objOpenSTAAD = Nothing
```

This is all it requires to create a macro. Obviously, this particular example really only duplicates the functionality of selecting View > New View in STAAD.Pro. However, it easy to combine other OpenSTAAD functions to automate a series of commonly used features in order to create your own time saving tools.

In this example, for the sake of brevity, the only model entities checked for selection are beams (that is, a new view is only created if beam elements are selected). You could easily expand this to Nodes, Plates, Solids, etc.

6.1.1 Example

The full code for this macro is as follows:

```
Sub Main()
  'DESCRIPTION:Creates a new view from the selected beams.
  Dim objOpenSTAAD As Object
  Dim SelBeamsNo As Long
  Dim SelBeams() As Long
  'Launch OpenSTAAD Object
  Set objOpenSTAAD = GetObject(,"StaadPro.OpenSTAAD")
  'Get no. of selected beams
  SelBeamsNo = objOpenSTAAD.Geometry.GetNoOfSelectedBeams
  If (SelBeamsNo > 0) Then
    ReDim SelBeams(SelBeamsNo) As Long
    'Create a new view
    objOpenSTAAD.View.CreateNewViewForSelections 'SelBeams
  Else
    MsgBox "No beams are currently selected.", vbOkOnly
  End If
  Set objOpenSTAAD = Nothing
End Sub
```

6.2 Microsoft Excel Macro

A Microsoft Office Excel spreadsheet which can be used to check the capacity of a rectangular concrete beam with the reinforcement already laid out.

This Excel file checks the bending capacity of a rectangular concrete beam with the reinforcement already laid out. The capacity is checked against the maximum sagging moment produced from a series of load cases. The beam is analyzed in STAAD.Pro. The results are extracted and linked into Excel using OpenSTAAD and VBA. A macro name d `Examp8` has been created to retrieve the maximum sagging moment along the span of the beam as well as the

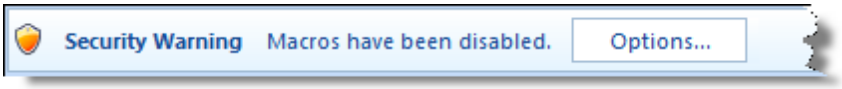
width and depth of the cross-section of the beam. The maximum sagging moment for each load case is extracted, and the governing moment is the largest compression moment of the maximum moments.

The calculations for checking the capacity are located on the sheet marked "Concrete" while the extraction of the values from STAAD occurs on the sheet marked "STAAD.Pro Output".

1. Open the file …\SProV8i\OpenSTAAD\Rectangle-Beam.xls.

   Note: Excel will warn of macros present in the file. Simply click the **Options...** button in the Security Warning message area to open the *Microsoft Office Security Options* dialog, where you will select the **Enable this Content** option and click **OK**.

   Warning: It is always recommended to do a virus scan on any macro before opening it - including this one!

2. Select the sheet named *STAAD.Pro Output* and change the values of **File Name** (cell B14) and **Member Number** (cell B15) to reflect the file and member number to be considered.

   Note: Make sure that there are valid results for the STAAD file selected and they exist in the same directory as the STD file.

3. To run the macro, either:

   select the Macros tool on the Developer tab (Excel 2007 and higher).

   Note: The Developer tab must first be enabled in Excel 2007 and higher. Refer to the Excel help for your version for instructions on how to do this.

   or

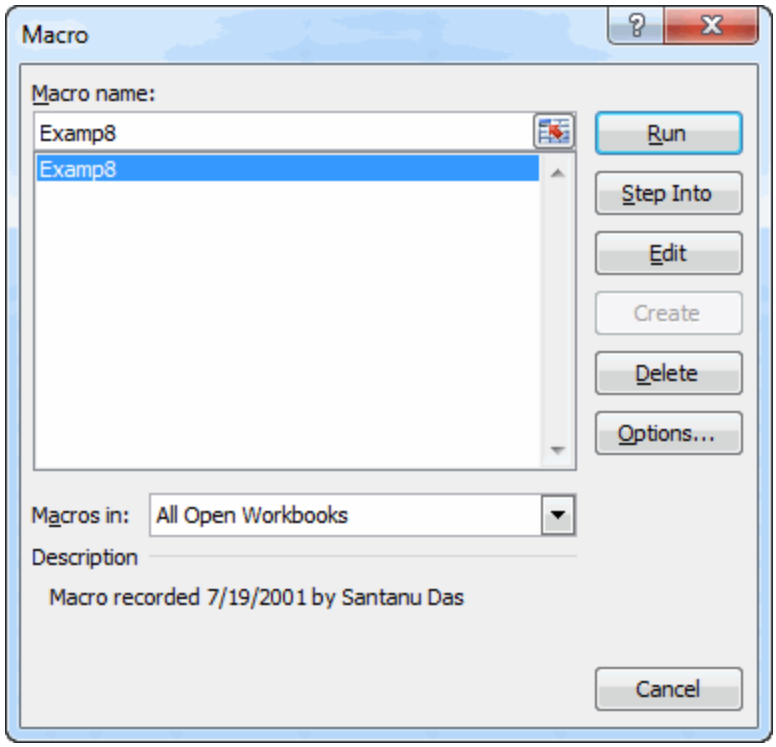
   select **Tools** > **Macro** > **Macros** (Excel 2005 and lower).

   or

   press **ALT+F8**.

   The Macro dialog opens.

Macro dialog in Microsoft Office Excel



4. Select the macro named **Examp8** and click **Run** to start.

   The values from the STAAD file will automatically be linked into the Excel sheet.

Hint: To see what is going on "under the hood", open the Examp8 macro by clicking **Edit** from the macro dialog box.

6.3 Autodesk AutoCAD® Macro

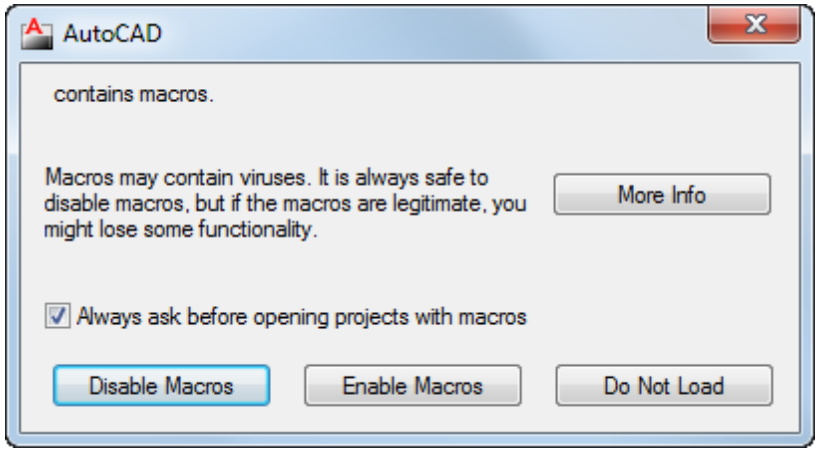
Writing the Member Section Forces at any Distance Along a Member in AutoCAD 2000 or higher.

OpenSTAAD can also be used to integrate pre or post-processing data with AutoCAD 2000 or higher. AutoCAD 2000 supports VBA macros with its own exposed set of functions. This example automatically calculates the section forces at any distance along a member from a STAAD file and plots the results in a table as a separate layer. A dialog box was created in AutoCAD using AutoCAD VBA forms. The input box will prompt for the STAAD file name, the member number, the load case and the point along the member (expressed as a fraction of the length) where the section forces are to be determined. The macro returns the section forces (by using OpenSTAAD to read STAAD's database) and plots them in a pre-defined table in the DXF file (`STDandAcad.dxf`).

1. Open the file …\SProV8i\OpenSTAAD\STDandACAD.dwg.

   AutoCAD will warn of macros present in the file.

   Warning: It is always recommended to do a virus scan on any macro before opening it - including this one!



2. Click **Enable Macros**.
3. To run the macro, either:

   select the RUN VBA Macro tool on the Manage tab (AutoCAD 2009 and higher).

   Note: Microsoft Visual Basic for Applications Module is not installed with recent versions of AutoCAD but can be downloaded from the Autodesk website. Refer to the help included with AutoCAD for more information.

   or

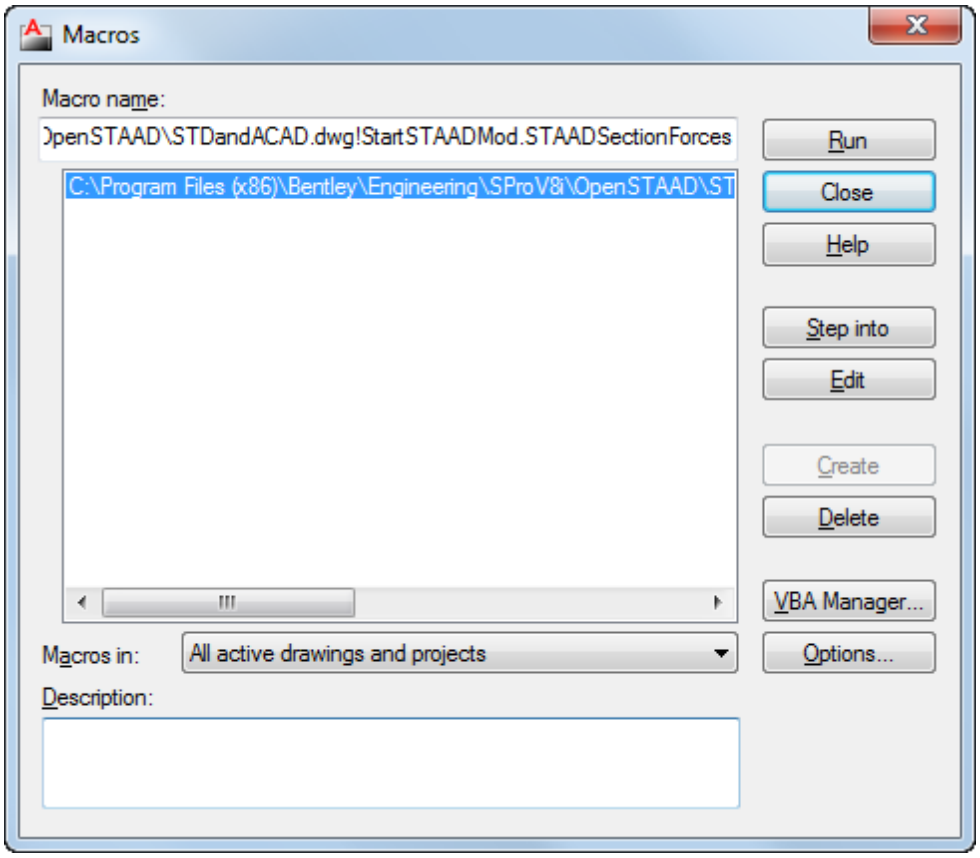
   select **Tools** > **Macro** > **Macros** (previous versions of AutoCAD or when AutoCAD 2009 is in AutoCAD Classic mode).

   or

   press **ALT+F8**.

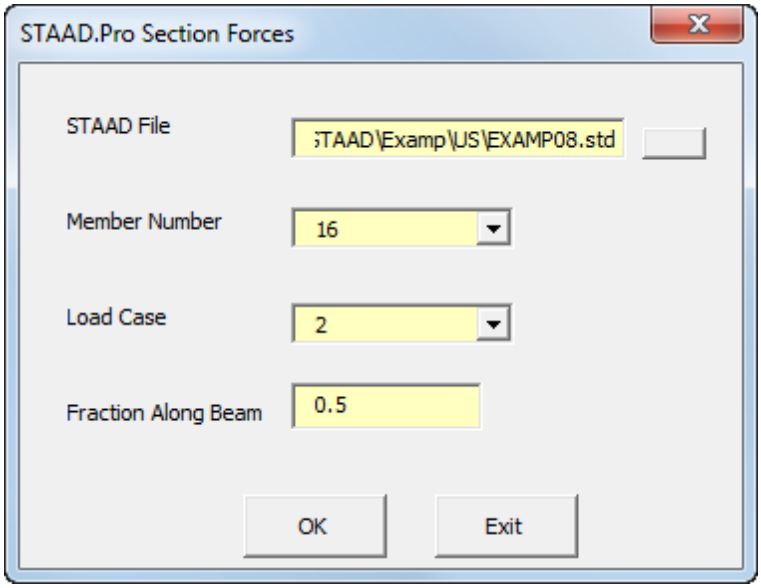
   The Macro dialog opens.

Macro dialog in Autodesk AutoCAD 2011



4. Select the macro named `STAADSectionForces` and click **Run**.

   The STAAD.Pro Section Forces dialog opens. A warning also opens informing you that the STAAD File name must be specified.



5. Enter a valid filename, including complete file path, for the current input file open in STAAD.Pro.

   The Member Number and Load Case drop-down lists are populated with data from the selected model.

6. Select a Member Number and Load Case and specify a Fraction Along Beam.

7. Click **OK**.

   The table is updated

8. (Optional) To populate the table with different results, first clear all contents on the **`STAAD Results`** layer in the AutoCAD drawing. Then repeat steps 4 through 7.

Note: For more information on how to call AutoCAD commands in VBA, please refer to the AutoCAD VBA Help provided with AutoCAD.

OpenSTAAD Example

STAAD File:	Member No:	Distance:	Load Case:
Shear Force / Moment	Value		Unit
FX			Kips
FY			Kips
FZ			Kips
MX			Kips-in
MY			Kips-in
MZ			Kips-in

OpenSTAAD Example

STAAD File: example1.std	Member No: 2	Distance: 0.5	Load Case: 1
Shear Force / Moment	Value		Unit
FX	58.3245983123779		Kips
FY	18.1449718475342		Kips
FZ	0		Kips
MX	0		Kips-in
MY	0		Kips-in
MZ	4183.53515625		Kips-in

6.4 Microsoft Word Macro

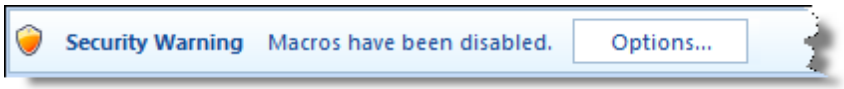
A Microsoft Office Word document is included which contains a partial report for analysis results which can be used as a basis for a customized report document.

The document file includes a macro named `Pro` which checks for the number of supported nodes as well as the number of load cases and load combinations and reports these back to the document. The document can then report support reactions for the selected node number and load case number.

1. Open an input file within STAAD.Pro.
2. Open the file …`\SProV8i\OpenSTAAD\STAADandWord.doc`.

   Note: Word will warn of macros present in the file. Simply click the **Options...** button in the Security Warning message area to open the *Microsoft Office Security Options* dialog, where you will select the **Enable this Content** option and click **OK**.

   Warning: It is always recommended to do a virus scan on any macro before opening it - including this one!

3. Click the **Update Nodes and Load Cases** button.

   The **Node Number** and **Load Case Number** selectors update with the data from the selected file.

4. Select the **Node Number** and **Load Case Number** numbers for which results will be retrieved.
5. Click **Get Results**.

   The support reactions and units update in the table below.

Technical Support

These resources are provided to help you answer support questions:

- Service Ticket Manager — **http://www.bentley.com/serviceticketmanager** — Create and track a service ticket using Bentley Systems' online site for reporting problems or suggesting new features. You do not need to be a Bentley SELECT member to use Service Ticket Manager, however you do need to register as a user.
- Knowledge Base — **http://appsnet.bentley.com/kbase/** — Search the Bentley Systems knowledge base for solutions for common problems.
- FAQs and TechNotes — **http://communities.bentley.com/Products/Structural/Structural_Analysis___Design/w/Structural_Analysis_and_Design__Wiki/structural-product-technotes-and-faqs.aspx** — Here you can find detailed resolutions and answers to the most common questions posted to us by you.
- Ask Your Peers — **http://communities.bentley.com/forums/5932/ShowForum.aspx** — Post questions in the Be Community forums to receive help and advice from fellow users.

Index

P

Bentley Systems, Incorporated

685 Stockton Drive, Exton, PA 19341 USA

+1 (800) 236-8539

www.bentley.com