Frew 19.4 COM Interface – guidelines

# Introduction

The COM (Component Object Model) interface allows creation of a data object that allows external programs to pass information and instructions to and from Frew.

COM objects can be used by many other programs, including Excel, MATLAB and Python. The interface in Frew 19.4 has been revised from that in Frew 19.3 so that it will now work directly from MATLAB and Python. Some of the functions and the way they are called have been changed and the sample spreadsheet has been updated to show the new format.

# How to Use the COM Interface

The following describes how to use the COM interface using Excel as an example. All 4 sections should be read by those wishing to write/amend macros using the COM interface. Section 2.1 should be read by those wishing to use spreadsheets with macros that use the COM interface. Section 2.4 has instructions for the sample spreadsheet included with the program.

## Install Frew 19.4 and link to Excel

First install Frew 19.4 on the computer. Once this is installed then Excel needs to be linked to the Frew template library which contains the COM object definitions. For a 32 bit version of Excel this needs to be done manually, as follows:

1. Open Excel
2. Go to the developer tab
3. Click on the ‘Visual Basic’ button to open the VBA window
4. In the VBA window click on ‘Tools’ in the menu bar and select ‘References’
5. Check the list for an entry ‘frewLib’ and tick – if this is not here then go to browse to add a new library
6. Within the browser navigate to the installation folder for Frew which should default to C:\Program Files\Oasys\Frew 19.4
7. Select the Frew template library (Frew.tlb) and click on the ‘Open’ button
8. This will have added frewLib to the references list, and should have checked it automatically – if not then check before exiting the references dialog.
9. The various windows can now be closed and macros prepared as required using the Frew Com object and related functions.

## Create FrewCom object

### IN VBA

Within a VBA macro or subroutine, create the COM interface object as follows:

Dim FrewObject

Set FrewObject = CreateObject(“frewLib.FrewComAuto”)

The first line declares a named variable (any name can be used) and the second line defines what this entry is and creates the object. The name “frewLib.FrewComAuto” provides the address in the registry for the Frew COM interface – defining what the object is, and linking it to the relevant DLLs.

### From Python

In a Python script, or the interactive window, create the COM object as follows:

Import win32com.client

frw = win32com.client.Dispatch(“frewLib.FrewComAuto”)

The first line imports the Python library which supports COM. The second line creates the COM object and associates it with the published COM interface with name FrewComAuto contained in the type library frewLib.

## Execute Object Functions

To execute a function using the COM interface the following structure should be used:

Result = FrewObject.FunctionName(Variable1, Variable2. Etc.).

For the above FrewObject is the name of the object, FunctionName the name of the function (see available functions section) and Variable1, Variable2, etc. the names of the variables required by the function. Note that it is only the type of the variable and not the name that needs to match the variables listed in the available functions.

As an example a function to open a file and extract the number of nodes and stages could be written as follows (using VBA).

Dim FrewObj ‘declare FrewCom object

Dim iRet As Integer ‘declare integers for return variables

Dim iNumNodes As Integer

Dim iNumStages As Integer

Dim sFilePath As String ‘declare a string for the file path

Set FrewObj = CreateObject(“frewLib.FrewComAuto”) ‘Set FrewObj as FrewCom interface and link

sFilePath = “C:\Files\Frewdata.fwd” ‘Set the relevant file path

FrewObj.Open(sFilePath) ‘Call the function to open the file

iNumNodes = FrewObj.GetNumNodes ‘Call the function to get number of nodes

iNumStage = FrewObj.GetNumStages ‘Call the function to get number of stages

The returned variables can then be used in loop counters or for assigning to cell values of the spreadsheet.

## Sample spreadsheet and Python script

To illustrate some of the functions available in the COM interface, a sample spreadsheet and a Python script are distributed with the program.

To use the spreadsheet, open FrewCOMTemplate2.xlsm in the Sample folder beneath the program installation folder (usually C:\Program Files\Oasys\Frew 19.4).

First enter an existing Frew filename in the yellow cell D3. Then click the button labelled “Get basic data”. This will open the file and retrieve some data (number of nodes and stages) and add some node data to a new page on the spreadsheet called “Data check page”. This page becomes the active page and you can review the data.

Return to Sheet1 and type a stage to which to analyse. Typing “7” for example will stop the analysis at Stage 7. Then click the Analyse button. When complete (or if analysis fails for any reason), a message will be shown on Sheet1.

If the analysis has succeeded, clicking the Plot button will extract and plot the displacement, bending moment and shear for each analysed stage. The results and graphs are saved to new pages in the spreadsheet.

The VBA macros are viewable by selecting the Developer tab in Excel and choosing “View code”. They can be freely amended for your own purposes. Section 3 describes all the functions available in the COM interface.

To use the Python script, open the file Frew\_ComTest.py using your interface for Python 2.7. The supplied script opens a data file, prints the number of nodes and stages found in the file, then carries out two analyses, revising one of the material parameters for the second analysis. You will need to replace the path name to the data file with a path valid on your system.

# Interface definition

The following functions are available in the COM interface. Note that the first function that should be used is ‘Open’. Functions return either a number value (e.g. node level or displacement) or an integer return code so that success or failure can be tested if required.

The Analyse function needs to be run before using any of the functions which retrieve results, to ensure that there are results available.

The units of the various parameters are as defined in the current data file. The default unit system is kN for force, metres for length and millimetres for displacement.

**Show**

This function has no input/output variables. It opens Frew and shows the main program window.

**Open**(String sPathName) - returns an integer

This function opens a saved file. sPathName is a string giving the file path. If the file is opened successfully the function will return 0, or if it was unsuccessful it will return -1.

**Analyse(**Integer iStage) – returns an integer

This is a function instructing Frew to analyse the currently open file, up to the stage number iStage.

**Save** – returns an integer

Saves the currently open file.

**SaveAs(**String sPathName) – returns an integer

Saves the currently open file with a different name.

**Close** – returns an integer

Closes the current file.

**GetNumNodes** – returns an integer

Get the number of nodes.

**GetNumStages** – returns an integer

Get the number of stages.

**GetNodeLevel** (Integer index) – returns a double precision number

Get the level of node ‘index’.

**NOTE:** When there are multiple analyses results in a file, the result functions in COM return only the values corresponding to the first result set. If unfactored analysis is selected, then the first result set is this unfactored analysis set i.e. SLS set.

**GetNodeDisp (**Integer index, Integer istage)

Get the displacement at node ‘index’ at stage ‘istage’.

**GetNodeBending (**Integer index, Integer istage)

Get the bending moment at node ‘index’ at stage ‘istage’.

**GetStrutBendingAtNode (**Integer index, Integer istage)

Get the strut bending moment at node ‘index’ at stage ‘istage’.

**GetNodeShear** (Integer index, Integer istage)

Get the shear force at node ‘index’ at stage ‘istage’.

**GetStrutShearAtNode** (Integer index, Integer istage)

Get the strut shear force at node ‘index’ at stage ‘istage’.

**GetNumMat**

Get the number of materials defined in the current file.

**GetUnitWeight (**Integer index)

Get the unit weight of material ‘index’ (zero-based counter, so the first material is 0).

**GetEref** (Integer index)

Get the Young’s Modulus of material ‘index’ (zero-based counter, so the first material is 0).

**GetKref (**Integer index)

Get the k0 value of material ‘index’ (zero-based counter, so the first material is 0

**GetPhi (**Integer index)

Get the angle of friction in **degrees** of material ‘index’ (zero-based counter, so the first material is 0).

**GetCohesion** (Integer index)

Get the cohesion of material ‘index’ (zero-based counter, so the first material is 0).

**GetRefLevel** (Integer index)

Get the reference level for E and cohesion of material ‘index’ (zero-based counter, so the first material is 0).

**GetCGrad** (Integer index)

Get the gradient of cohesion of material ‘index’ (zero-based counter, so the first material is 0).

**GetEGrad** (Integer index)

Get the gradient of Young’s Modulus of material ‘index’ (zero-based counter, so the first material is 0).

**SetUnitWeight** (Integer index, Double newVal)

Set the unit weight of material ‘index’ to the value ‘newVal’.

**SetEref** (Integer index, Double newVal)

Set the Young’s Modulus of material ‘index’ to the value ‘newVal’.

**SetKref** (Integer index, Double newVal)

Set the K0 value of material ‘index’ to the value ‘newVal’.

**SetPhi** (Integer index, Double newVal)

Set the angle of friction in **degrees** of material ‘index’ to the value ‘newVal’.

**SetCohesion** (Integer index, Double newVal)

Set the cohesion of material ‘index’ to the value ‘newVal’.

**SetRefLevel** (Integer index, Double newVal)

Set the reference level for cohesion and Young’s modulus of material ‘index’ to the value ‘newVal’.

**SetCGrad** (Integer index, Double newVal)

Set the gradient of cohesion of material ‘index’ to the value ‘newVal’.

**SetEGrad** (Integer index, Double newVal)

Set the gradient of Young’s modulus of material ‘index’ to the value ‘newVal’.

**GetSoilZoneLeft/GetSoilZoneRight** (Integer index, Integer istage )

Get the left or right soil zones of node ‘index’ at stage ‘istage’.

**SetSoilZoneLeft/SetSoilZoneRight** (Integer index, Integer istage, Integer iVal)

Set the left or right soil zones of node ‘index’ at stage ‘istage’.

**GetWallEI** (Integer index, Integer istage)

Get the wall EI value at node ‘index’ in stage ‘istage’. Zero will be returned if the node is below the toe of the wall (or for all nodes in Stage 0).

**SetWallEI** (Integer index, Integer istage)

Set the wall EI value at node ‘index’ in stage ‘istage’ to the value ‘newVal’.

**DeleteResults**

Deletes the results. **Required** before re-running the analysis.

Added from 19.3 build 13 onwards:

**IsEGenerated** (Integer istage)

Returns 1 if the E profile for stage ‘istage’ is generated, 0 if it is user-specified.

**GetEBL** (Integer istage)

Get the (generated or specified) E value at the left rigid boundary for stage ‘istage’.

**GetEBR** (Integer istage)

Get the (generated or specified) E value at the right rigid boundary for stage ‘istage’.

**GetEGradL** (Integer istage)

Get the specified E gradient on the left side of the wall for stage ‘istage’.

**GetEGradR** (Integer istage)

Get the specified E gradient on the right side of the wall for stage ‘istage’.

**GetEGroundL** (Integer istage)

Get the generated E value at the left ground level for stage ‘istage’.

**GetEGroundR** (Integer istage)

Get the generated E value at the right ground level for stage ‘istage’.

**SetEBL** (Integer istage, Double newVal)

Set the specified E value at the left rigid boundary for stage ‘istage’ to ‘newVal’.

**SetEBR** (Integer istage, Double newVal)

Set the specified E value at the right rigid boundary for stage ‘istage’ to ‘newVal’.

**SetEGradL** (Integer istage, Double newVal)

Set the specified E gradient on the left side of the wall for stage ‘istage’ to ‘newVal’.

**SetEGradR** (Integer istage, Double newVal)

Set the specified E gradient on the right side of the wall for stage ‘istage’ to ‘newVal’.

**IsCOMServer**

Returns 1 if Frew is running as a COM server and 0 if not.

**GetNumStruts**

Returns the number of struts.

**GetStrutPrestress/GetStrutStiffness/GetStrutAngle** (Integer istrut)

Returns the prestress/stiffness/inclination angle of strut number ‘istrut’.

**GetStrutLeverArm** (Integer istrut)

Returns the lever arm of strut number ‘istrut’.

**GetStrutStageIn/GetStrutStageOut** (Integer istrut)

Returns the stage the strut is inserted or removed.

**SetStrutPrestress/SetStrutStiffness/SetStrutAngle** (Integer istrut, Double newVal)

Sets the prestress/stiffness/inclination angle of strut number ‘istrut’ to the value ‘newVal’.

**SetStrutLeverArm** (Integer istrut, Double newVal)

Sets the lever arm of strut number ‘istrut’ to the value ‘newVal’.

**SetStrutStageIn/SetStrutStageOut** (Integer istrut, Integer newVal)

Sets the stage the strut is inserted or removed to the value ‘newVal’.

**GetStrutForce/GetStrutHorizForce/GetStrutMoment/GetStrutMaxForce** (Integer istage,

(Integer istrut)

Returns the force/horizontal force/bending moment/maximum force in strut number ‘istrut’ at stage ‘istage’.

**GetNodeVeLeft/** **GetNodeVeRight** (Integer iNodeIndex, Integer istage)

Gets the vertical effective stress on the left side and right side respectively at the node having index iNodeIndex at stage ‘istage’.

**GetNodePeLeft/** **GetNodePeRight** (Integer iNodeIndex, Integer istage)

Gets the horizontal effective stress on the left side and right side respectively at the node having index iNodeIndex at stage ‘istage’.

**GetNodePPLeft/** **GetNodePPRight** (Integer iNodeIndex, Integer istage)

Gets the pore pressure on the left side and right side respectively at the node having index iNodeIndex at stage ‘istage’.

**SetAnalysisMethod**(Integer istage, Integer iMethod)

Sets the analysis method at stage ‘istage’ to iMethod.

iMethod = 0 for Safe method of analysis

= 1 for Mindlin method of analysis

= 2 for spring method of analysis

**SetSpringLengthLeft/** **SetSpringLengthRight** (Integer iStage, Integer iNodeIndex, Double dLen)

Sets the spring length on the left side and right side respectively to dLen at the node having index iNodeIndex at stage ‘istage’.

**GetSpringLengthLeft/** **GetSpringLengthRight** (Integer iStage, Integer iNodeIndex, Double dLen)

Gets the spring length on the left side and right side respectively at the node having index iNodeIndex at stage ‘istage’.

**GetMatName** (Integer iMat)

Gets the material name of the material with index iMat.

**GetDeltaPhi** (Integer iMat)

Gets the ratio of soil-wall friction angle to internal angle of friction for material with index iMat.

**SetDeltaPhi** (Integer iMat, Double dVal)

Sets the ratio of soil-wall friction angle to internal angle of friction for material with index iMat to dVal.

**GetKratio** (Integer iMat)

Gets the ratio of horizontal effective stress change to vertical effective stress change for material with index iMat.

**SetKratio** (Integer iMat)

Sets the ratio of horizontal effective stress change to vertical effective stress change for material with index iMat to dVal.

**GetWaterLevelLeft/** **GetWaterLevelRight** (Integer iStage)

Gets the water level on the left side and right side respectively at stage ‘istage’.

**SetWaterLevelLeft/** **SetWaterLevelRight** (Integer iStage, Double dLevel)

Gets the water level on the left side and right side respectively to dLevel at stage ‘istage’.

**GetPorePressGradLeft/** **GetPorePressGradRight** (Integer iStage)

Gets the pore pressure gradient on the left side and right side respectively at stage ‘istage’.

**SetPorePressGradLeft/** **SetPorePressGradRight** (Integer iStage, Double dGrad)

Gets the the pore pressure gradient on the left side and right side respectively to dGrad at stage ‘istage’.

**GetWallBaseNode** (Integer istage)

Gets the index of the node at the base of the wall at stage ‘istage’.

**GetTopNodeLeft/** **GetTopNodeRight** (Integer istage)

Gets the index of the top node at the ground level on the left side and right side respectively at stage ‘istage’.

**AddStage** (Integer iPrevStage, String stageName)

Adds a new stage with name ‘stageName’ after previous stage with index iPrevStage.

**DeleteStage** (Integer istage)

Deletes the stage index istage.

**DeleteLastStage** ()

Deletes the last stage.

**NumStageResults** ()

Returns the number of stages having results in the first result set.

**PrintTabular** (BSTR sPath, BOOL bNotes, [in] BOOL bGeom, BOOL bSoils, BOOL bLoads, BOOL bWallLoads, BOOL bStruts, BOOL bWater, BOOL bMeth, BOOL bConv, BOOL bMefp, BOOL bMaterialLayers, BOOL bWallEI, BOOL bResults)

Exports the tabular output for all stages with information on modules such as Notes, geometry data, soil data, results etc. The user can specify whether or not a particular module can be exported by setting its flag to True or False appropriately. The supported formats are csv, txt, rtf and htm.

**PrintGraphical** (BSTR sPath, Short istage, BOOL bDisp, BOOL bPressure, BOOL bBending, BOOL bShear)

Exports to WMF file the graphical output for stage with index ‘istage’ with information displacement, pressure, bending moment, and shear. The user can specify whether or not a particular module can be exported by setting its flag to True or False appropriately.

**PrintTabularStage** (BSTR sPath, BOOL bFull, Short istage)

Exports the tabular output for stage with index ‘istage’ either full results or summary of results – depending on whether the flag bFull is set to True or False respectively. The supported formats are csv, txt, rtf and htm.

**GetNodeStrut** (Integer istrut)

Gets the index of the node associated with the strut with index ‘istrut’.

**SetNodeStrut** (Integer istrut, Integer inode)

Sets the index of the node associated with the strut with index ‘istrut’ to ‘inode’.