

FEMAP BASIC Scripting Language

API Reference

Copyright 1996-1999 by Enterprise Software Products, Inc.

Overview

The FEMAP BASIC Script Language provides direct access to the FEMAP Database Engine through the BASIC Interpreter built in to FEMAP.

The Mechanism

The FEMAP BASIC Script Language is based on Cypress Enable BASIC Scripting for Applications. Cypress Enable provides a the complete array BASIC programming functionality. Wherever possible, Cypress Enable's implementation of BASIC follows the Microsoft Visual Basic syntax and semantics. The Cypress Enable portion of the BASIC Scripting Language handles all flow of control, subroutines, and functions created in your script. We have added a FEMAP specific interface (the Application Programming Interface, or API) that allows your BASIC program to customize FEMAP. The FEMAP menu has also been extended to include the capability to launch and run BASIC scripts from your own user-defined menus.

Details regarding the general elements of the Cypress Enable BASIC Scripting Language are included in the Language Reference Manual. This document covers variables, constants, control structures, subroutines and functions, file input/output, arrays, dialog support, etc. that will help you craft your BASIC Scripts.

Creating and Executing BASIC Scripts

The BASIC Script files themselves are simple ASCII Text files. Every program that you write must contain a main subroutine that acts as your entire program, or calls other functions and subroutines.

```
Sub Main ( )  
...  
'Your Program  
...  
End Sub
```

Dialog boxes can be created using the dlgdsn.exe program. To launch this program from the script editor use the Edit-Dialog Editor command. The dlgdsn.exe program must be run once by itself so it can register itself in the registry.

BASIC Script files can be executed in five different ways.

1. From the FEMAP Main Menu, select File Program Run Script. You will then be presented with the File Open Common Dialog box from

which you can select the script file to execute

2. From the FEMAP Main Menu, select File Program Edit Script. The Script Editor is a standard Microsoft Windows Single Document Interface ASCII Text Editor. Here you can open up text files, copy and paste between them, or copy and paste between other Windows applications, to create or edit your BASIC Script.
3. Custom menus can be created and linked to FEMAP scripts or program files. A .esp file should be created and selected in the preferences, libraries, menu. The format of the file is similar to a window's menu resource. An example is below.

```
POPUP "&CustomCommands"
BEGIN
  MENUITEM "&Cut Element", "c:\Femap60\cut.bas"
  MENUITEM "&Radius", "c:\ Femap60\radius.bas"
  MENUITEM SEPARATOR
  MENUITEM "&Group by Elem", "c:\ Femap60\group.bas"
  MENUITEM "Group &Elem 3", "c:\ Femap60\group3.bas"
  MENUITEM "Group &Node", "c:\ Femap60\grpnd.bas"
  MENUITEM SEPARATOR
  MENUITEM "E&xtrude Plates", "c:\ Femap60\plex.bas"
END
```

4. Finally, FEMAP Program Files can themselves launch BASIC Scripts. A new Program File command, #RUN has been added that will run the BASIC Script File specified.
5. As a command line argument when launching FEMAP, use -P and the name of the script. The script must end with the .bas extension, otherwise FEMAP will think it is a Program File.

Variables

A short note on variables. Although the BASIC engine included with FEMAP can handle many variable types, to avoid problems in passing data into and out of FEMAP you must declare all variables as the same type specified in the arguments for the functions.

Overview of Functionality

The FEMAP specific functions that have been added to the BASIC Scripting Language are described in this section, broken down by their general area of application.

Output Data Manipulation

The functions concerned with the manipulation of FEMAP output data all begin with the `esp_Outp` prefix. Using these functions, you can query output that has been loaded into the FEMAP database by any of the supported FEA programs and use that output in your own calculations. You can also put output data into the FEMAP database for further manipulation with your own program, for graphical post-processing within FEMAP, or for text based reporting using the FEMAP listing commands and formatted output.

Manipulation of output data is broken down into three categories, getting, putting, and manipulating.

FEMAP Output Data

Before you begin crafting your own program to manipulate FEMAP output data, it is important that you understand some of the underlying organizational issues.

Output Sets

Every piece of output data in a FEMAP model is linked to an Output Sets. Output Sets are analogous to distinct FEA loads and/or boundary condition sets. For example, a FEA model that is subjected to three distinct loading conditions will have three distinct output sets. You can manipulate the output data of existing output sets, create new data in existing output sets, or create your own new output set to store and partition your calculated from the actual data that was returned from your finite element analysis.

Output Set Creation

Every Output Set in FEMAP contains the following information that you must provide in order to create a new output set.

Item	Description	Possible Values	Function Used to Set
Set ID	ID Number of the Set. Must be unique with regard to other existing output sets.	1 to 99,999,999	<code>esp_OutpCreateSet</code>
Title	Descriptive Title of the Output Set	Maximum of 25 characters	<code>esp_OutpCreateSet</code>
Program	Analysis Program where output came from.	0 - Unknown 1 - FEMAP 2 - PAL 3 - PAL2 4 - MSC/NASTRAN 5 - ANSYS	<code>esp_OutpCreateSet</code>

		6 - STARDYNE 7 - COSMOS 8 - PATRAN 9 - FEMAP Neutral 10 - ALGOR 11 - SSS/NASTRAN 12 - Comma Separated 13 - UAI/NASTRAN 14 – COSMIC/NASTRAN 15 - STAAD 16 - ABAQUS 17 - WECAN 18 - MTAB 19 - CDA/Sprint 20 - CAEFEM	
Analysis Type	Type of Analysis	0 - Unknown 1 - Static 2 - Modes 3 - Transient 4 - Frequency Response 5 - Response Spectrum 6 - Random 7 - Linear Buckling 10 - Nonlinear Static 11 - Nonlinear Buckling 20 - Steady State Heat 21 - Transient Heat	esp_OutpCreateSet
value	Numerical Value associated with this Output Set. Typical users are the time value for a transient analysis, or the frequency value for a modal run.	Real	esp_OutpCreateSet

Example:

This example creates Output Set 1, makes the title “BASIC Script Set”, sets the from_program flag to “FEMAP”, the analysis type to Nonlinear Static, and sets the Set Value to 1.455. Notice the variable ‘j’, this scripts uses j to determine whether or not a new output set has been created. If Output Set 1 already existed, esp_OutpCreateSet would have returned FALSE (-1).

Example outp_1.bas:

```
Sub Main
Dim ExistFlag as Long
Dim j as Long
Dim setID as Long
Dim Title as String * 25
Dim from_program as Long
Dim anal_type as Long
Dim setValue as Double
Dim Msg

setID = 1
Title = "BASIC Script Set"
from_program = FEMAP
anal_type = NONLINSTATIC
setValue = 1.455

j = esp_OutpCreateSet( setID, from_program, anal_type,
    setValue, Title )

If j = TRUE THEN
    Msg = "Created Output Set " + Str(setID)
Else
    Msg = "Could Not Create Output Set " + Str(setID)
End If

Print Msg

End Sub
```

Output Data Vector Numbering

FEMAP uses output vector numbers between 1 and 99,999 for standard output data that is read in from the supported FEA programs. User defined output data, such as that created using the output calculation menu commands in FEMAP, is stored in vector numbers 300,000 and greater. Output created with your own programs should be placed in this region. If you are creating or calculating output that matches similar output that is normally read in directly by FEMAP, it is strongly recommended that you follow the standard FEMAP numbering convention for consistency. Appendix A of this manual contains a list of some of the standard vector ID numbers used by FEMAP for your reference.

Nodal and Elemental Data

All output data contained in the FEMAP database is either nodal or elemental. Nodal output data is just that, output data attached to nodes in the FEMAP model. Elemental data is attached to individual elements. In addition to basic data, where there is one value for each node for a nodal data vector or one value for each element in an elemental data vector, there are special cases that you should be aware of.

Output Vector Description

In order for FEMAP to know how to handle your output data, there is

some information that must be provided for every output vector that you create.

Item	Description	Possible Values	Function Used to Set
Set ID Number	The ID of the Output Set to which this vector belongs.	Any Existing Output Set	esp_OutpCreateVector
Vector ID Number	The ID number of this output vector.	1 to 99,999,999	esp_OutpCreateVector
Title	Description of this Output Vector	Any Text	esp_OutpCreateVector
Out_Type	Type of Output Data	0 - Any 1 - Disp. 2 - Accel. 3 - Force 4 - Stress 5 - Strain 6 - Temp. 7 - Other	esp_OutpCreateVector
Data_type	Nodal or Elemental Output Data	7 - Nodal 8 - Elem.	esp_OutpCreateVector
Comp_dir	<p>Component Direction Flag</p> <p>0 - Default value. Set during vector creation, indicates that this data stands alone and does not have connected component information.</p> <p>1 - vector data, comp[0..2] of this vector contain the vector ID's of the X, Y, Z components of this vector.</p> <p>2 - comp[0..1] contain the vector ID's of the EndA and EndB for the beam data corresponding to this vector's centroidal data.</p> <p>3 - comp[0..1] contain the vector ID's of the EndA and EndB for the beam data</p>	<p>0 - No</p> <p>1 - Yes</p> <p>2 - Beam</p> <p>3 - Beam Reversed</p>	esp_OutpSetVectorComponentFlag

	corresponding to this vector's centroidal data, where EndB is reversed in sign convention from standard beam data.		
min_val	Minimum value in vector	Real	esp_OutpVectorSetMaxMin
max_val	Maximum value in vector	Real	esp_OutpVectorSetMaxMin
abs_max	Maximum absolute value in vector	Real	esp_OutpVectorSetMaxMin
id_min	ID of entity where minimum value occurs.	Long	esp_OutpVectorSetMaxMin
id_max	ID of entity where maximum value occurs.	Long	esp_OutpVectorSetMaxMin
calc_flag	Flag to whether or not this output data can be linearly combined.	0 - Yes 1 - No	Set to a default value of 1 - No when the vector is created with esp_OutpCreateVector, can be overridden with esp_OutpSetVectorCalcFlag
centroidal flag	Flag indicating that this data vector contains elemental or nodal centroidal data. Usually 1 for standard nodal and elemental data, only set to 0 if this vector contains element corner data.	0 - Corner 1 - Yes	Set to a default value of 1 - Yes when the vector is created with esp_OutpCreateVector, can be overridden with esp_OutpSetVectorCentroidalFlag

Plain Nodal or Elemental Data

The simplest and most common type of data that will need to be loaded into and out of FEMAP is plain nodal or elemental data. By plain, we mean a single output value for each node or element of your model. The following example demonstrates opening a file on disk and reading in nodal values into a new FEMAP output vector.

Example - outp_2.bas, requires outp_2.dat and a 10 x 10 plate model with nodes numbered 1 through 100.

```

Sub Main

' Output Set Variables
Dim setID as Long
Dim Title as String * 25
Dim from_program as Long
Dim anal_type as Long
Dim setValue as Double

' Output Vector Variables
Dim vectorID as Long

```



```

Dim vectorTitle as String * 25

' "Global" Variables
Dim Msg
Dim j as Long
Dim k as Long
Dim l as Long
Dim nodecount as Long
Dim nodevalue as Double
Dim nodeID as Long
Dim st as String

' Initialize Output Set Values
setID = 1
Title = "BASIC Script Set"
from_program = FEMAP
anal_type = NONLINSTATIC
setValue = 1.455

' Initialize Output Vector Values
vectorID = 300000
vectorTitle = "Nodal Temperature Data"

' First Create an Empty Output Set

j = esp_OutpCreateSet( setID, from_program, anal_type,
    setValue, Title )

If j = TRUE Then
    Msg = "Created Output Set " + Str(setID)
Else
    Msg = "Could Not Create Output Set " + Str(setID)
End If

Print Msg

' Now Create an Empty Output Vector

j = esp_OutpCreateVector( setID, vectorID, Temp, Node,
    VectorTitle )

If j = TRUE Then
    Msg = "Created Output Vector " + Str(vectorID)
Else
    Msg = "Could Not Create Output Vector " + Str(vectorID)
End If

Print Msg

If j = TRUE Then
    ' Open the data file
    Open "outp_2.dat" for Input as #1
    Line Input #1, st
    ret_val = esp_MiscParseInit( st )
    ret_val = esp_MiscParseInt( 1, nodecount )
    If ret_val = TRUE Then
        For k = 1 to nodecount
            Line Input #1, st
            ret_val = esp_MiscParseInit( st )
            ret_val = esp_MiscParseInt( 1, nodeID )
            ret_val = esp_MiscParseDouble( 2, nodevalue )
            l = esp_OutpPutData( setID, vectorID, nodeID, nodevalue )
        Next k
    End If
    ret_val = esp_OutpVectorFinish( setID, vectorID )

```

```

End If

Close #1

End Sub

```

Nodal Vector Data

FEMAP can store and post-process vector data that is made up of three global components. With vector data, there are typically four output vectors total, the first, the vector sum of the other three components. FEMAP stores the vector containing the vector sum with pointers to the constituent individual component vectors. By doing this, it is possible in graphical post-processing to select a single vector, and be able to display on screen the direction of this vector data since FEMAP knows where the component values come from. In your programmatic access to FEMAP, you are responsible for setting up only to components vectors, and then calling a built in routine that creates the vector sum vector, and connects everything up for later graphical post-processing.

The following example creates three nodal vectors, and then calls the `esp_OutpCreateVectorVectorSum` function to create the vector total vector.

Example - `outp_3.bas` - requires a two node model connected by one element.

```

Sub Main

' Output Set Variables
Dim setID as Long
Dim Title as String * 25
Dim from_program as Long
Dim anal_type as Long
Dim setValue as Double

' "Global" Variables
Dim Msg
Dim j as Long
Dim k as Long
Dim l as Long
Dim nodecount as Long
Dim nodevalue as Double
Dim nodeID as Long
Dim st as String

' Initialize Output Set Values
setID = 1
Title = "BASIC Script Set"
from_program = FEMAP
anal_type = STAT
setValue = 0.0

' First Create an Empty Output Set
j = esp_OutpCreateSet( setID, from_program, anal_type,
    setValue, Title )

```

```

If j = TRUE Then
    Msg = "Created Output Set " + Str(setID)
Else
    Msg = "Could Not Create Output Set " + Str(setID)
End If

Print Msg

' Now Create the Component Vectors
' Bad programming practice, but we will assume that the
' creation
' will happen and not fail.
j = esp_OutpCreateVector( setID, 300001, Disp, Node, "X-Value"
)
j = esp_OutpCreateVector( setID, 300002, Disp, Node, "Y-Value"
)
j = esp_OutpCreateVector( setID, 300003, Disp, Node, "Z-Value"
)

l = esp_OutpPutData( setID, 300001, 1, .125 )
l = esp_OutpPutData( setID, 300002, 1, .25 )
l = esp_OutpPutData( setID, 300003, 1, .375 )
l = esp_OutpPutData( setID, 300001, 2, .5 )
l = esp_OutpPutData( setID, 300002, 2, .25 )
l = esp_OutpPutData( setID, 300003, 2, .375 )

' Start FEMAP Internal Cleanup of Vectors

ret_val = esp_OutpVectorFinish( setID, 300001 )
ret_val = esp_OutpVectorFinish( setID, 300002 )
ret_val = esp_OutpVectorFinish( setID, 300003 )
ret_val = esp_OutpCreateVectorVectorSum( setID, 300000,
    300001, 300002, 300003, Disp, Node, "Vector Sum" )

End Sub

```

Elemental Output Data

Elemental output data can be broken down into two distinct categories. Straight elemental data contains a single output value for each element in your model. The most important thing to remember about elemental output data in FEMAP, is how it is used when drawing color contour plots. FEMAP must, in order to draw a color contour plot, resolve the output data to the nodes. With pure elemental data, FEMAP averages the output value reported for all elements connected to a node to determine what contour level will be drawn at that node. This process is repeated for each node of every element.

Elemental Output Data with Corner Data

In line with the actual output data from several of the supported FEA programs, FEMAP can also store elemental output data that contains references to the actual corner data for each element. In this case, there is a centroidal value for each element, as well as references to other data vectors that contain actual corner values for each of the nodes connected to that element. For example, 4-node plate output data with corners is actually made up of five output data vectors in FEMAP. The first,

represents the value at the centroid of the element, and the other four represent values for that elements, for each of the four nodes. With corner data, FEMAP can create a color contour plot that is more accurate, since instead of averaging adjacent element centroidal data to determine the contour color at a node, we can use the actual corner values.

The following example - outp_4.bas, creates element centroidal output data and corresponding corner data for a two element plate model.

Example - outp_4.bas - requires a two element plate model, elements number 1 and 2, with nodes numbered 1 through 6.

```
Sub Main

' Output Set Variables
Dim setID as Long
Dim Title as String * 25
Dim from_program as Long
Dim anal_type as Long
Dim setValue as Double

' Output Vector Variables
Dim vectorID as Long
Dim vectorTitle as String * 25

' "Global" Variables
Dim Msg
Dim j as Long
Dim k as Long
Dim l as Long
Dim m as Long
Dim nodecount as Long
Dim nodevalue as Double
Dim nodeID as Long
Dim st as String

' Initialize Output Set Values
setID = 1
Title = "BASIC Script Set"
from_program = FEMAP
anal_type = STAT
setValue = 0.0

' First Create an Empty Output Set

j = esp_OutpCreateSet( setID, from_program, anal_type,
    setValue, Title )

If j = TRUE Then
    Msg = "Created Output Set " + Str(setID)
    Print Msg
Else
    Msg = "Could Not Create Output Set " + Str(setID)
    Print Msg
    GoTo Failed
End If

' First, we will create the element centroidal vector

j = esp_OutpCreateVector( setID, 7033, Stress, Elem, "Plt. Top
    VonMises Stress" )
```

```

If j = TRUE Then
    Msg = "Created Centroidal Vector"
    Print Msg
Else
    Msg = "Error Creating Centroidal Vector"
    Print Msg
    GoTo Failed
End If

' now create the corner vectors

j = esp_OutpCreateVector( setID, 20133, Stress, Elem, "Plt.
Top VonMises Str C1" )
k = esp_OutpCreateVector( setID, 30133, Stress, Elem, "Plt.
Top VonMises Str C2" )
l = esp_OutpCreateVector( setID, 40133, Stress, Elem, "Plt.
Top VonMises Str C3" )
m = esp_OutpCreateVector( setID, 50133, Stress, Elem, "Plt.
Top VonMises Str C4" )

If j = TRUE Then
    Msg = "Created Corner 1 Vector"
    Print Msg
Else
    Msg = "Error Creating Corner 1 Vector"
    Print Msg
    GoTo Failed
End If

If k = TRUE Then
    Msg = "Created Corner 2 Vector"
    Print Msg
Else
    Msg = "Error Creating Corner 2 Vector"
    Print Msg
    GoTo Failed
End If

If l = TRUE Then
    Msg = "Created Corner 3 Vector"
    Print Msg
Else
    Msg = "Error Creating Corner 3 Vector"
    Print Msg
    GoTo Failed
End If

If m = TRUE Then
    Msg = "Created Corner 4 Vector"
    Print Msg
Else
    Msg = "Error Creating Corner 4 Vector"
    Print Msg
    GoTo Failed
End If

' Set up Corner Pointers on Centroidal Vector

j = esp_OutpSetVectorComponent( setID, 7033, 0, 20133 )
If j = FALSE Then
    Msg = "Error Setting Corner Reference"
    Print Msg
    GoTo Failed
End If

j = esp_OutpSetVectorComponent( setID, 7033, 1, 30133 )
If j = FALSE Then

```

```

    Msg = "Error Setting Corner Reference"
    Print Msg
    GoTo Failed
End If

j = esp_OutpSetVectorComponent( setID, 7033, 2, 40133 )
If j = FALSE Then
    Msg = "Error Setting Corner Reference"
    Print Msg
    GoTo Failed
End If

j = esp_OutpSetVectorComponent( setID, 7033, 3, 50133 )
If j = FALSE Then
    Msg = "Error Setting Corner Reference"
    Print Msg
    GoTo Failed
End If

' Also need to set the Centroidal Flag to indicate that the
' corner
' vectors contain corner data.

j = esp_OutpSetVectorCentroidalFlag( setID, 20133, 0 )
If j = FALSE Then
    Msg = "Error Setting Centroidal Flag"
    Print Msg
    GoTo Failed
End If

j = esp_OutpSetVectorCentroidalFlag( setID, 30133, 0 )
If j = FALSE Then
    Msg = "Error Setting Centroidal Flag"
    Print Msg
    GoTo Failed
End If

j = esp_OutpSetVectorCentroidalFlag( setID, 40133, 0 )
If j = FALSE Then
    Msg = "Error Setting Centroidal Flag"
    Print Msg
    GoTo Failed
End If

j = esp_OutpSetVectorCentroidalFlag( setID, 50133, 0 )
If j = FALSE Then
    Msg = "Error Setting Centroidal Flag"
    Print Msg
    GoTo Failed
End If

' Now pump in the data,
' this script assumes there are two elements, 1 and 2
' and 6 nodes, 1 through 6

l = esp_OutpPutData( setID, 7033, 1, 600 ) ' Center, Elem 1
l = esp_OutpPutData( setID, 7033, 2, 800 ) ' Center, Elem 2

l = esp_OutpPutData( setID, 20133, 1, 100 ) ' C1 E1
l = esp_OutpPutData( setID, 30133, 1, 200 ) ' C2 E1
l = esp_OutpPutData( setID, 40133, 1, 500 ) ' C3 E1
l = esp_OutpPutData( setID, 50133, 1, 400 ) ' C4 E1

l = esp_OutpPutData( setID, 20133, 2, 200 ) ' C1 E2
l = esp_OutpPutData( setID, 30133, 2, 300 ) ' C2 E2
l = esp_OutpPutData( setID, 40133, 2, 600 ) ' C3 E2
l = esp_OutpPutData( setID, 50133, 2, 500 ) ' C4 E2

```

```

ret_val = esp_OutpVectorFinish( setID, 20133 )
ret_val = esp_OutpVectorFinish( setID, 30133 )
ret_val = esp_OutpVectorFinish( setID, 40133 )
ret_val = esp_OutpVectorFinish( setID, 50133 )
ret_val = esp_OutpVectorFinish( setID, 7033 )
GoTo Success:

Failed:

Msg = "Error Executing Script File"
Print Msg

Success:

End Sub

```

Getting Output Data

Getting output data out of FEMAP is much easier to describe since it does not have the setup requirement that creating data does. The following example demonstrates getting output set data, in this case natural frequency values. It also demonstrates the OLE Automation capabilities of the FEMAP BASIC Scripting Language. In this example, the frequency values associated with the output sets of a natural frequency analysis are transferred to Microsoft Word.

Example - outp_5.bas - assumes you have a model containing the results from a modal analysis.

```

Sub Main ( )

' Word Variables
Dim MSWord As object
Dim Doc As object

' Output Set Variables
Dim setID as Long
Dim Title as String * 25
Dim from_program as Long
Dim anal_type as Long
Dim setValue as Double

' Global Variables
Dim j as Long
Dim k as Long
Dim Msg

j = esp_DBNextEntity( Existing, Out_Case, After, 0 )

If j > MAX_LABEL Then
    Msg = "No Output Sets Exist"
    Print Msg
    GoTo Failed
End If

' Connect to Word
Set MSWord = CreateObject("Word.Application")
MSWord.Application.Visible = True

```

```

MSWord.Documents.Add

' Insert into the Document
Set Doc = MSWord.ActiveDocument
Doc.Content.InsertAfter "Natural Frequencies"
Doc.Content.InsertParagraphAfter
Doc.Content.InsertParagraphAfter

While j < MAX_LABEL
' Walk through all the Output Sets

k = esp_OutpGetSet( j, from_program, anal_type, setValue,
Title )
If k = TRUE Then
    If anal_type = MODES Then
        Msg = "        Output Set " + Str(j) + ":        Frequency
= " + Str(setValue) + " Hz."
        Doc.Content.InsertAfter Msg
        Doc.Content.InsertParagraphAfter
    End If
End If
j = esp_DBNextEntity( Existing, Out_Case, After, j )

Wend

Doc.Content.InsertParagraphAfter
Doc.Content.InsertParagraphAfter

' Format the Title
Doc.Paragraphs(1).Range.Bold = True
Doc.Paragraphs(1).Range.Font.Name = "Arial"
Doc.Paragraphs(1).Range.Font.Size = 24

GoTo Success

Failed:

Success:

End Sub

```

Function Definitions

All functions return TRUE if successful, FALSE if the action requested is not possible, unless otherwise noted.

Declare Function esp_OutpGetSet App (ByVal setID as Long, ByRef program as Long, ByRef anal_type as Long, ByRef value as Double, ByVal Title as String) as Long

Gets information about the Output Set defined by setID. Fills program, anal_type, value, and Title with the corresponding information from inside of FEMAP.

Declare Function esp_OutpGetVector App (ByVal setID as Long, ByVal vectorID as Long, ByRef out_type as Long, ByRef data_type as Long, ByVal Title as String) as Long

Gets information about the Output Vector defined by vectorID in setID. Fills out_type, data_type, and Title with the corresponding information from inside of FEMAP.

Declare Function esp_OutpGetVectorComponentFlag App (ByVal setID as Long, ByVal vectorID as Long, ByRef flag as Long) as Long

Gets the Vector Component Flag from Output Vector vectorID in Output Set setID, and fills flag with its value, TRUE or FALSE.

Declare Function esp_OutpGetVectorCalcFlag App (ByVal setID as Long, ByVal vectorID as Long, ByRef flag as Long) as Long

Gets the Calculation Flag information for Output Vector vectorID in Output Set setID, and returns this flag in flag, TRUE or FALSE.

Declare Function esp_OutpGetVectorCentroidalFlag App (ByVal setID as Long, ByVal vectorID as Long, ByRef flag as Long) as Long

Gets the Centroidal Flag information for Output Vector vectorID in Output Set setID, and returns this flag in flag, TRUE or FALSE.

Declare Function esp_OutpGetVectorMaxMinData App (ByVal setID as Long, ByVal vectorID as Long, ByRef absmax as Double, ByRef max as Double, ByRef min as Double, ByRef maxID as Long, ByRef minID as Long) as Long

Retrieves the max/min data for Output Vector vectorID in Output Set setID, and fills in the appropriate absmax, max, min, maxID, and minID values.

Declare Function esp_OutpGetVectorComponent App (ByVal setID as Long, ByVal vectorID as Long, ByVal index as Long, ByRef comp as Long) as Long

Retrieves the component ID number for the Output Vector vectorID in Output Set setID, at the Index value index, and fills this value into comp.

Declare Function esp_OutpGetData App (ByVal setID as Long, ByVal vectorID as Long, ByVal ID as Long, ByRef value as Double) as Long

Retrieves the output data value for entity ID from the Output Vector vectorID in the

Output Set setID, places this value in value.

Declare Function esp_OutpCreateSet App (ByVal setID as Long, ByRef program as Long, ByRef anal_type as Long, ByRef value as Double, ByVal title as String) as Long

Creates a new empty Output Set. Values passed in make it possible to control the ID number, program, anal_type, value, and title for the Output Set. Returns FALSE if this Output Set already exists.

Declare Function esp_OutpCreateVector App (ByVal setID as Long, ByVal vectorID as Long, ByVal out_type as Long, ByVal data_type as Long, ByVal title as String) as Long

Creates a new empty Output Vector, vectorID, in Output Set setID. Values passed in make it possible to control the out_type, data_type, and title associated with the new Output Vector.

Declare Function esp_OutpSetVectorComponentFlag App (ByVal setID as Long, ByVal vectorID as Long, ByVal flag as Long) as Long

Sets the component flag to the value (TRUE or FALSE) specified by flag, for Output Vector vectorID in Output Set setID.

Declare Function esp_OutpSetVectorCalcFlag App (ByVal setID as Long, ByVal vectorID as Long, ByVal flag as Long) as Long

Sets the Calc flag to the value (TRUE or FALSE) specified by flag, for Output Vector vectorID in Output Set setID.

Declare Function esp_OutpSetVectorCentroidalFlag App (ByVal setID as Long, ByVal vectorID as Long, ByVal flag as Long) as Long

Sets the Centroidal flag to the value (TRUE or FALSE) specified by flag, for Output Vector vectorID in Output Set setID.

Declare Function esp_OutpSetVectorComponent App (ByVal setID as Long, ByVal vectorID as Long, ByVal index as Long, ByVal comp as Long) as Long

Sets the component ID value specified in comp at Index index, for the Output Vector vectorID in Output Set setID.

Declare Function esp_OutpSetVectorMaxMinData App(ByVal setID as Long, ByVal vectorID as Long, ByVal max_val as Double, ByVal max_id as Long,

ByVal min_val as Double, ByVal min_id as Long, ByVal absmax_val as Double) as Long

Sets the max/min information for the Output Vector vectorID in Output Set setID.

Declare Function esp_OutPutData App (ByVal setID as Long, ByVal vectorID as Long, ByVal ID as Long, ByVal value as Double) as Long

Puts the output data value specified in value, for entity ID ID, into Output Vector vectorID in Output Set setID.

Declare Function esp_OutpConvElemToNodal App (ByVal setID as Long, ByVal vectorID as Long, ByVal groupID as Long, ByVal calc_mode as Long, ByVal newVectorID as Long) as Long

Converts an elemental output data vector to an equivalent nodal output data vector.

Requires the following:

setID - ID of the Output Set containing both the old and the new vectors.

VectorID - ID of the Elemental Output Data Vector to be converted.

groupID - ID of the FEMAP Group containing the element IDs where equivalent nodal data will be calculated.

calc_mode - Controls how the nodal values are computed. Options are:

0. If corner data is available, all corner data for each node is averaged to come up with a nodal value.
1. If corner data is available, the maximum value reported for each node is used.
2. Uses the average of the centroidal elemental values for elements connected to each node.
3. Uses the maximum of the centroidal elemental values for elements connected to this node.

newVectorID - Is the ID number of the new Output Vector where the nodal results will be stored.

Declare Function esp_OutpConvNodalToElem App (ByVal setID as Long, ByVal vectorID as Long, ByVal groupID as Long, ByVal calc_mode as Long, ByVal newVectorID as Long) as Long

Same as esp_OutpConvElemToNodal, with the following exception:

calc_mode - Controls how the elemental data values are computed, options are:

- 0, and 2 - Averages all nodes connected to an element to determine the elemental value.
- 1, and 3 - Uses the maximum value for any node

Declare Function esp_OutpVectorFinish App (ByVal setID as Long, ByVal

vectorID as Long) as Long

Used after you have filled an output vector with data. This function performs all the appropriate book keeping operations inside of FEMAP, most importantly, finding and filling in for you the max/min information for the entire vector.

Declare Function esp_OutpCreateVectorVectorSum App (ByVal setID as Long, ByVal vectorID as Long, ByVal vec1 as Long, ByVal vec2 as Long, ByVal vec3 as Long, ByVal out_type as Long, ByVal data_type as Long, ByVal title as String) as Long

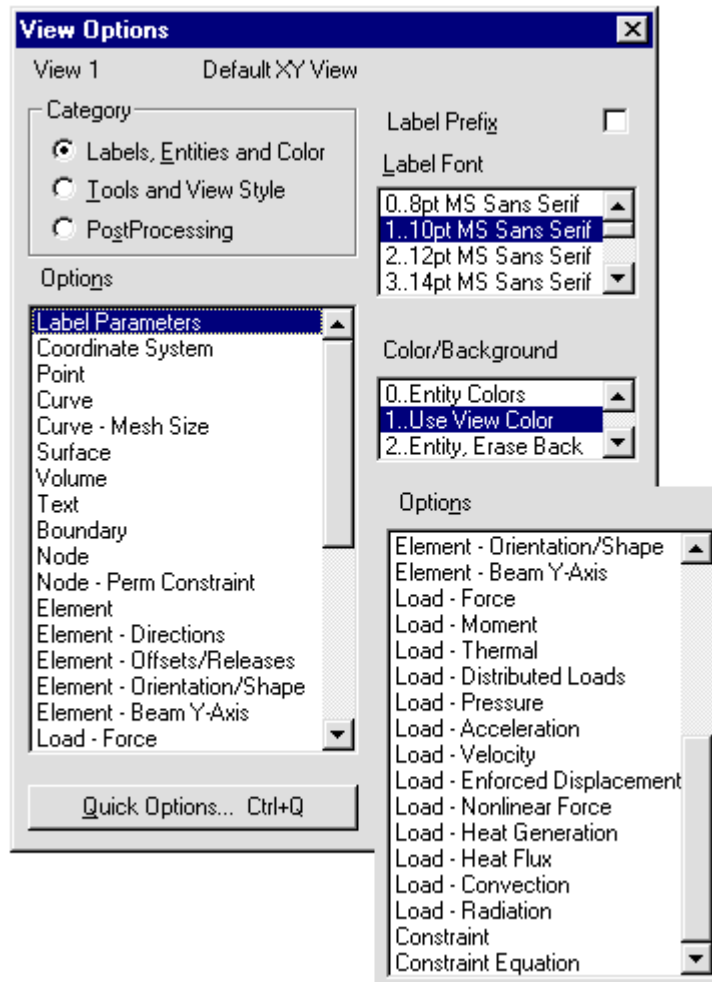
Once you have created three component vectors of vector data in the Global X, Global Y, and Global Z directions, this function will create a new output vector vectorID in setID that for every node in your model is the vector sum of the data in vec1, vec2, and vec3. This function also requires that you pass in the out_type, and data_type, and title for the new vector being created.

View Manipulation

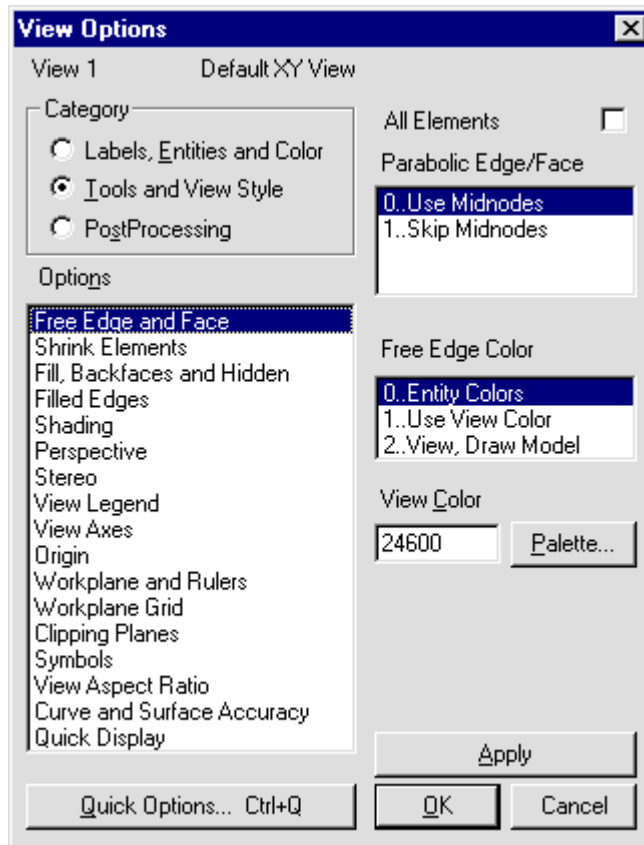
The functions beginning with esp_View.... have been designed to allow you to manipulate the display of FEMAP model through your own BASIC Scripts. Commands are included that make it possible to set all of the View Options to your liking.

The View Manipulation functions are broken down into three categories, Low-Level, High-Level, and Utility. Low level functions provide access to every View Option in FEMAP. Anyone experienced in using FEMAP knows that there is a large number of options available for controlling how your model is displayed on screen. The FEMAP View Options Dialog Box is itself broken down into three categories.

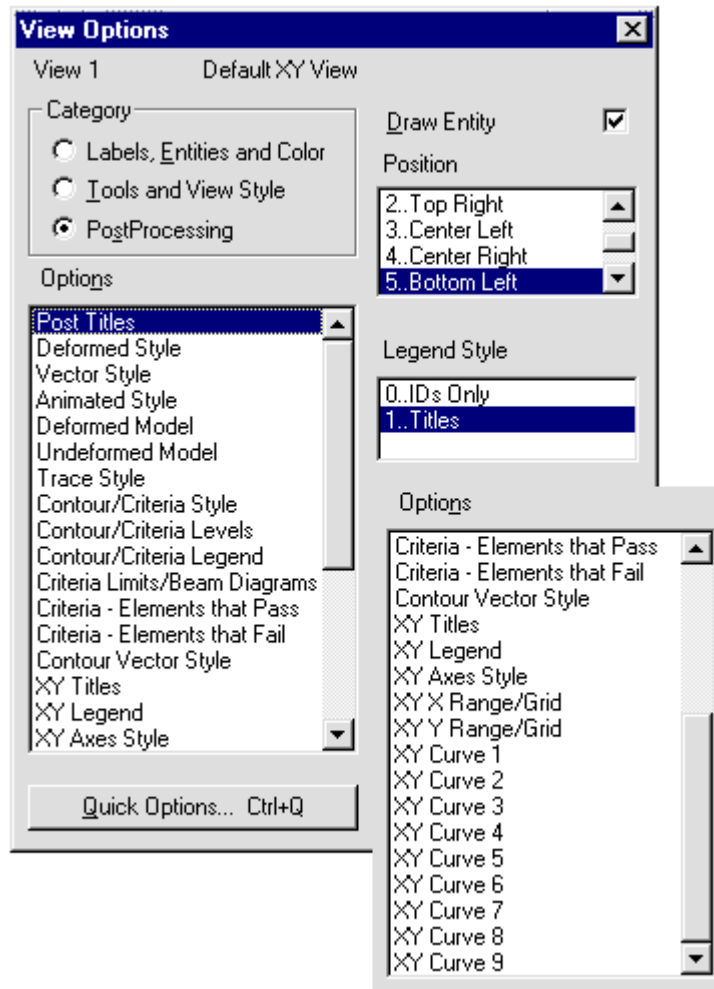
Labels, Entities and Color:



Tools and View Style:



and PostProcessing:

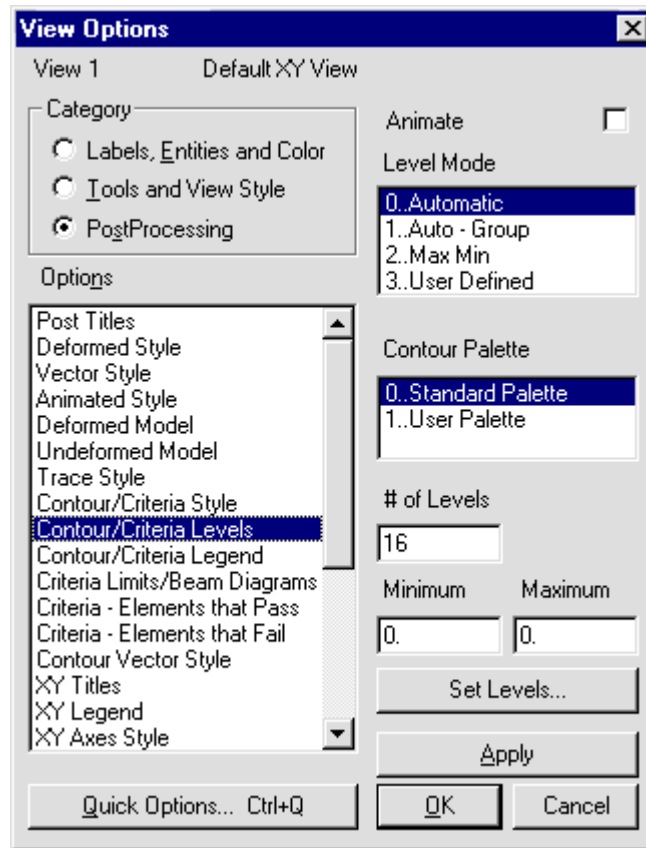


Almost all of the view options in FEMAP share three characteristics:

- Draw Entity Toggle
- Label Mode
- Color Mode

As you can see when you move up and down through the View Options Dialog Box, these three characteristics sometimes change to provide view control over features more appropriate for the option in question, but they exist for almost every option. Internally, FEMAP stores these three items in the same place for each view option. Using this fact, and by looking at the View Options Dialog Box, you can control almost every aspect of how your model is displayed. For example, the Contour/Criteria Levels option in the PostProcessing Category uses the three standard view option features to control three items related more directly to displaying color stress contours on screen. In this case, the draw entity toggle controls whether or not the color stress contour levels animate during a displacement animation, the Label Mode has been mapped to Level Mode

to control how the stress legend is calculated, and the Color Mode is used to control whether the Standard Palette or a User Defined Palette is used for the colors themselves.



The three low level functions that control setting these three options in the active view are:

```
esp_ViewActiveToggleDraw( INT4 opt, INT4 val )
```

```
esp_ViewActiveSetLabelMode( INT4 opt, INT4 val )
```

```
esp_ViewActiveSetColorMode( INT4 opt, INT4 val )
```

If you wish to manipulate views other than the active view, the following three commands provide the same functionality, plus the capability to specify a view number to update.

```
esp_ViewToggleDraw( INT4 vu_number, INT4 opt, INT4 val )
```

```
esp_ViewSetLabelMode( INT4 vu_number, INT4 opt, INT4 val )
```

```
esp_ViewSetColorMode( INT4 vu_number, INT4 opt, INT4 val )
```

Each of these function takes as an argument the option that you wish to manipulate. These options are defined in the BASICHDR.ESP file that gets automatically loaded with your BASIC Script File and are as follows:

Labels, Entities and Color	
Option	Constant
Label Parameters	PL_LABEL
Coordinate System	PL_CSYS
Point	PL_POINT
Curve	PL_CURVE
Curve - Mesh Size	PL_CURVE_MESH
Surface	PL_SURFACE
Volume	PL_VOLUME
Text	PL_TEXT
Boundary	PL_BOUNDARY
Node	PL_NODE
Node - Perm Constraint	PL_PERM_BC
Element	PL_ELEMENT
Element Directions	PL_ELEM_DIR
Element Offsets/Releases	PL_OFFSETS
Element Orientation/Shape	PL_BEAM_ORIENT
Element Beam Y-Axis	PL_BEAM_Y_AXIS
Load Vectors	PL_LOAD_VECTORS
Load Force	PL_LD_FORCE
Load Moment	PL_LD_MOMENT
Load Thermal	PL_LD_THERMAL
Load Distributed	PL_LD_LINELOAD
Load Pressure	PL_LD_PRESSURE
Load Acceleration	PL_LD_ACCEL
Load Velocity	PL_LD_VELOCITY
Load Enforced Displacement	PL_LD_DISP
Load Nonlinear Force	PL_LD_TRANSIENT
Load Heat Generation	PL_LD_HEATGEN
Load Heat Flux	PL_LD_HEATFLUX

Load Convection	PL_LD_CONVECTION
Load Radiation	PL_LD_RADIATION
Load Fluid Tracking	PL_LD_TRACKING
Load Unknown Condition	PL_LD_UNKNOWN
Load Slip Wall Condition	PL_LD_SLIP
Load Fan Curve	PL_LD_FANCURVE
Load Periodic Condition	PL_LD_PERIODIC
Constraint	PL_BC
Constraint Equation	PL_BC_EQUATION
Contact Segment	PL_CONTACT
Tools and View Style	
Free Edge and Face	PL_FREE_EDGE
Shrink Elements	PL_ELEM_SHRINK
Fill, Backfaces and Hidden	PL_ELEM_FILL
Filled Edges	PL_FILLED_EDGES
Render Options	PL_RENDER
Shading	PL_ELEM_SHADE
Perspective	PL_PERSPECTIVE
Stereo	PL_STEREO
View Legend	PL_LEGEND
View Axes	PL_AXES
Origin	PL_ORIGIN
Workplane and Rulers	PL_WORKPLANE
Workplane Grid	PL_GRID
Clipping Planes	PL_CUTPLANE
Symbols	PL_SYMBOL
View Aspect Ratio	PL_ASPECT_RATIO
Curve and Surface Accuracy	PL_CURVE_ERR
PostProcessing	
Post Titles	PL_POST_TITLES

Deformed Style	PL_DEFORM_STYLE
Vector Style	PL_ARROW_STYLE
Animated Style	PL_ANIMATE_STYLE
Deformed Model	PL_DEFORMED
Undeformed Model	PL_UNDEFORMED
Trace Style	PL_TRACE
Contour/Criteria Style	PL_CONTOUR_STYLE
Contour/Criteria Levels	PL_CONTOUR_LEVEL
Contour/Criteria Legend	PL_CONTOUR_LEGEND
Criteria Limits/Beam Diagrams	PL_CRITERIA_LIMITS
Criteria - Elements that Pass	PL_PASS_CRITERIA
Criteria - Elements that Fail	PL_FAIL_CRITERIA
Isosurface	PL_ISOSURFACE
Contour Vector Style	PL_CONTOUR_VECTORS
XY Titles	PL_XY_TITLES
XY Legend	PL_XY_LEGEND
XY Axes Style	PL_XY_AXIS_STYLE
XY X Range/Grid	PL_XY_XAXIS
XY Y Range/Grid	PL_XY_YAXIS
XY Curve 1	PL_XY_CURVE1
XY Curve 2	PL_XY_CURVE2
XY Curve 3	PL_XY_CURVE3
XY Curve 4	PL_XY_CURVE4
XY Curve 5	PL_XY_CURVE5
XY Curve 6	PL_XY_CURVE6
XY Curve 7	PL_XY_CURVE7
XY Curve 8	PL_XY_CURVE8
XY Curve 9	PL_XY_CURVE9

Examples:

To turn off the display of nodes:

```
esp_ViewActiveToggleDraw( PL_NODE, FALSE)
```

To change the color mode of elements to their material color:

```
esp_ViewActiveSetColorMode( PL_ELEMENT, 4 )
```

Function Definitions

All functions return TRUE if successful, FALSE if the action requested is not possible, unless otherwise noted.

Declare Function esp_ViewGetCurrent App () as Long

Returns the current active view as a Long Integer.

Declare Function esp_ViewLabelsAllOff App (ByVal vuID as Long) as Long

Turns all labels off in the View defined by vuID.

Declare Function esp_ViewLabelsAllOn App (ByVal vuID as Long) as Long
--

Turns all labels on in the View defined by vuID.

Declare Function esp_ViewRotateByAngle App (ByVal vuID as Long, ByVal dx as Double, ByVal dy as Double, ByVal dz as Double) as Long
--

Rotates the View defined by vuID by the angle specified in dx, dy, dz, about the view center.

Declare Function esp_ViewActiveLabelsAllOff App () as Long

Turns all labels off in the current active view.

Declare Function esp_ViewActiveLabelsAllOn App () as Long
--

Turns all labels on in the current active view.

Declare Function esp_ViewActiveToggleDraw App (ByVal vu_option as Long, ByVal draw_option as Long) as Long

Toggles the Draw Entity Box in the FEMAP View Options Dialog Box for the current active View.

vu_option is any of the constants defined in the table above that correspond to options in the FEMAP View Options Dialog box.

draw_option is TRUE to toggle the draw entity box on, FALSE to toggle the draw entity box off.

Declare Function esp_ViewDrawNow App (ByVal vu_number as Long) as Long

Force FEMAP to actually draw the View specific by vu_number. While making changes to views in FEMAP with the esp_View commands, the view is not updated on screen until your BASIC Script has completed. With esp_ViewDrawNow, you can force the view to redraw. This can be used to create a slide show of your model, making the view rotate, redraw, change display options, redraw, etc.

Declare Function esp_ViewActiveSetLabelMode App (ByVal opt as Long, ByVal opt_value as Long) as Long

For each of the display options in the FEMAP View Options Dialog Box, this function controls the label mode. This function acts on the current active view.

Declare Function esp_ViewActiveSetColorMode App (ByVal opt as Long, ByVal opt_value as Long) as Long

For each of the display options in the FEMAP View Options Dialog Box, this function controls the color mode. This function acts on the current active view.

Declare Function esp_ViewToggleDraw App (ByVal vu_number as Long, ByVal vu_option as Long, ByVal draw_option as Long) as Long

Toggles the Draw Entity option for any FEMAP view option in the View specified by vu_number.

Declare Function esp_ViewSetLabelMode App (ByVal vu_number as Long, ByVal opt as Long, ByVal opt_value as Long) as Long

Sets the label mode for any FEMAP view option in the View specified by vu_number.

Declare Function esp_ViewSetColorMode App (ByVal vu_number as Long, ByVal opt as Long, ByVal opt_value as Long) as Long

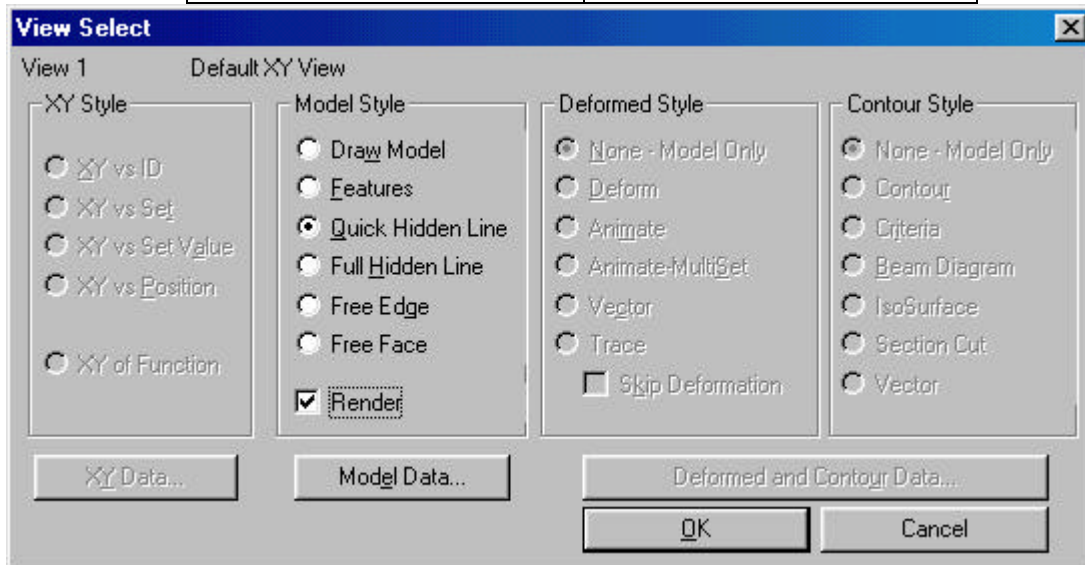
Sets the color mode for any FEMAP view options in the View specified by vu_number.

Declare Function esp_ViewSetDisplayMode App (ByVal vu_number as Long,

ByVal opt_value as Long) as Long

Sets the display mode for the View specified by vu_number. Possible option values are:

Constant	Meaning
PLT_DRAW	Normal Wireframe Display
PLT_FEATURE	Feature Plot
PLT_SORT	Quick Hidden Line
PLT_HIDE	Full Hidden Line
PLT_FREE	Free Edge Plot
PLT_FREE_FACE	Free Face Plot
PLT_XY_VS_ID	XY vs. ID
PLT_XY_VS_CASE	XY vs. Output Set
PLT_XY_VS_VALUE	XY vs. Set Value
PLT_XY_VS_POSITION	XY vs. Position
PLT_XY_OF_FUNCTION	XY of a FEMAP Function



The Display Mode options correspond direction with the Model Style and XY Style options of the FEMAP View Select Dialog Box.

Declare Function esp_ViewSetDeformMode App (ByVal vu_number as Long, ByVal opt_value as Long) as Long

Sets the Deformed Style for the View specified by vu_number. Valid opt_value's are:

Constant	Meaning
Model_Undef	Normal NonDeformed Display
Deformed	Deformed

Animate	Animate
Animate_MultiCase	Animate MultiSet
Arrow	Vector
Trace	Trace

These options correspond directly with the Deform Options in the FEMAP View Select Dialog Box.

Declare Function esp_ViewSetContourMode App (ByVal vu_number as Long, ByVal opt_value as Long) as Long

Sets the contour mode in the View specified by vu_number. Again, these mimic the contour options found in the FEMAP View Select Dialog Box. Valid options are:

Constant	Meaning
Model_Color	Normal
Contour	Contour
Criteria	Criteria
Beam_Diagram	Beam Diagram
IsoSurface	IsoSurface
Section_Cut	Section Cut
Contour_Vector	Vector

Declare Function esp_ViewSetRenderMode App (ByVal vu_number as Long, ByVal opt_value as Long) as Long

Set render mode on (1) or off (0)

Declare Function esp_ViewSetShrinkFactor App (ByVal vu_number as Long, ByVal shrink as Double) as Long

Sets the shrink factor for the view defined by vu_number.

Declare Function esp_ViewSetAmbientLight App (ByVal vu_number as Long, ByVal light as Double) as Long

Sets the Ambient Light factor for the view defined by vu_number.

Declare Function esp_ViewSetPerspectiveDistance App (ByVal vu_number as Long, ByVal distance as Double) as Long

Sets the Perspective Distance for the view defined by vu_number.

Declare Function esp_ViewSetLoadVectorLength App (ByVal vu_number as Long,

ByVal value as Double) as Long

Sets the Load Vector Length factor for the view defined by vu_number.

Declare Function esp_ViewSetOtherVectorLength App (ByVal vu_number as Long, ByVal value as Double) as Long

Sets the Other Vector Length factor for the view defined by vu_number.

Declare Function esp_ViewSetAspectRatio App (ByVal vu_number as Long, ByVal value as Double) as Long

Sets the Aspect Ratio factor for the view defined by vu_number.

Declare Function esp_ViewSetCurveError App (ByVal vu_number as Long, ByVal value as Double) as Long

Sets the Curve Error factor for the view defined by vu_number.

Declare Function esp_ViewSetSurfaceDivisions App (ByVal vu_number as Long, ByVal value as Long) as Long

Sets the Surface Divisions number for the view defined by vu_number.

Declare Function esp_ViewSetDeformedRelativeScale App (ByVal vu_number as Long, ByVal value as Double) as Long

Sets the Deformed Relative Scale factor for the view defined by vu_number. Controls the exaggeration of the deformation of your model based on a percentage of overall model size. Used in any of the deformed or animated plots created by FEMAP.

Declare Function esp_ViewSetDeformedAbsoluteScale App (ByVal vu_number as Long, ByVal value as Double) as Long

Sets the Absolute Scale factor for the view defined by vu_number.

Declare Function esp_ViewSetVectorLabelTopPercent App (ByVal vu_number as Long, ByVal value as Double) as Long

Sets the Top Percentage Factor for labeling of vector plots.

Declare Function esp_ViewSetAnimationFrames App (ByVal vu_number as Long, ByVal value as Long) as Long

Sets the number of frames for an animation in the View defined by vu_number.

Declare Function esp_ViewSetAnimationDelay App (ByVal vu_number as Long, ByVal value as Long) as Long

Sets the animation delay for the view defined by vu_number.

Declare Function esp_ViewSetContourLabelFreq App (ByVal vu_number as Long, ByVal value as Long) as Long

Sets the contour label frequency parameter for the View defined by vu_number.

Declare Function esp_ViewSetFunctionDisplay App (ByVal vu_number as Long, ByVal value as Long) as Long

Sets the ID of the function to be displayed using the XY vs. Function view mode for the view defined by vu_number. This is equivalent to the FEMAP View Select, Model Data, Function parameter.

Declare Function esp_ViewSetOutputSetID App (ByVal vu_number as Long, ByVal value as Long) as Long

Sets the current output set for the view defined in vu_number. Used as the Output Set in all post-processing features.

Declare Function esp_ViewSetDeformVectorID App (ByVal vu_number as Long, ByVal value as Long) as Long

Sets the deformation output vector ID used for all post-processing options for the view defined in vu_number.

Declare Function esp_ViewSetContourVectorID App (ByVal vu_number as Long, ByVal value as Long) as Long

Sets the contour output vector ID used for all post-processing options for the view defined in vu_number.

Declare Function esp_ViewSetXYSetID App (ByVal vu_number as Long, ByVal curve_number as Long, ByVal value as Long) as Long

Sets the xy plot output set ID used for all post-processing options for the view defined in

vu_number.

**Declare Function esp_ViewSetXYSetRange App (ByVal vu_number as Long,
ByVal curve_number as Long, ByVal start_set as Long, ByVal end_set as
Long) as Long**

Sets the xy plot output set range IDs used for all post-processing options for the view defined in vu_number.

**Declare Function esp_ViewSetXYVectorID App (ByVal vu_number as Long, ByVal
curve_number as Long, ByVal value as Long) as Long**

Sets the xy plot output vector ID used for all post-processing options for the view defined in vu_number.

**Declare Function esp_ViewSetXYEntityID App (ByVal vu_number as Long, ByVal
curve_number as Long, ByVal value as Long) as Long**

Sets the xy plot entity ID (node or element depending on type of output used) used for all post-processing options for the view defined in vu_number.

Declare Function esp_ViewAutoScale App (ByVal vu_number as Long)

AutoScales the view specified by vu_number.

Declare Function esp_ViewSave App () as Long

Save active view into library.

Declare Function esp_ViewLoad App (ByVal vu_number as Long) as Long

Load view vu_number from library.

**Declare Function esp_ViewSetXYSetID App (ByVal vu_number as Long, ByVal
curve_number as Long, ByVal value as Long) as Long**

Sets the xy plot output set ID to value for view of vu_number and curve 1-9 in curve_number.

**Declare Function esp_ViewSetXYVectorID App (ByVal vu_number as Long, ByVal
curve_number as Long, ByVal value as Long) as Long**

Sets the xy plot output vector ID to value for view of vu_number and curve 1-9 in

curve_number.

Declare Function esp_ViewSetXYEntityID App (ByVal vu_number as Long, ByVal curve_number as Long, ByVal value as Long) as Long

Sets the xy plot entity (node/element) ID to value for view of vu_number and curve 1-9 in curve_number.

Declare Function esp_ViewSetXYSetRange App (ByVal vu_number as Long, ByVal curve_number as Long, ByVal start_set as Long, ByVal end_set as Long) as Long

Sets the xy plot output set ID range from start_set to end_set for view of vu_number and curve 1-9 in curve_number.

Declare Function esp_ViewNew App (ByVal copy as Long, ByVal vu_number as Long, ByVal vu_title as String) as Long

Creates a new view with title as vu_title. If copy is 1 it copies the view in vu_number.

Declare Function esp_ViewRegenerate App (ByVal vu_number as Long) as Long

Forces a regeneration of view in vu_number.

Declare Function esp_ViewActivate App (ByVal vu_number as Long) as Long

Activates view in vu_number.

Declare Function esp_ViewGroup App (ByVal vu_number as Long, ByVal active_group as Long, ByVal group_number as Long) as Long

Sets the view in view number to show group in group_number or if active_group is 1 sets the view to show the active group. If both active_group and group_number are zero it sets the view to no groups(show all).

Declare Function esp_ViewCloseAll App () as Long

Closes all views

Declare Function esp_ViewTile App () as Long

Tiles the view windows.

Declare Function esp_ViewLoadSet App (ByVal vu_number as Long, ByVal

active_set as Long,ByVal set_number as Long) as Long

Sets the view in view number to show load set in set_number or if active_set is 1 sets the view to show the active load set. If both active_set and set_number are zero it sets the view to no sets(show all).

Declare Function esp_ViewBCSet App (ByVal vu_number as Long,ByVal active_set as Long,ByVal set_number as Long) as Long

Sets the view in view number to show constraint set in set_number or if active_set is 1 sets the view to show the active constraint set. If both active_set and set_number are zero it sets the view to no sets(show all).

Declare Function esp_ViewLayerShow App (ByVal vu_number as Long, ByVal showall as Long, ByVal layerID as Long) as Long

Shows layer of layerID in view of vu_number. If show all is 1 it shows all, if zero it shows only the visible layers.

Declare Function esp_ViewLayerHide App (ByVal vu_number as Long, ByVal layerID as Long) as Long

Hides layer of layer ID in view of vu_number.

List Processing

The esp_List.... functions make it possible to query the user for a selection in FEMAP. The selection can be any of the supported entity types, nodes, elements, etc. Once selected, additional esp_List.... functions make it possible for your basic script to work with list of entities. In the following example, the user is asked to select a set of nodes. Once selected, the BASIC Script goes through this list and using BASIC Script functions that query nodal output, finds the maximum total translation associated with any of the nodes originally selected.

```
Sub main ( )

Dim listID as Long
Dim ret_val as Long
Dim nodeID as Long
Dim max_node as Long
Dim Msg as String
Dim j as Long
Dim max_tran as Double
Dim test_val as Double

max_tran = 0.0

listID = esp_ListNextAvailableID

j = esp_ListSelectAll( listID, Node )

nodeID = 0
```

```

nodeID = esp_ListGetNextItem( listID, nodeID )

While nodeID < MAX_LABEL

    ret_val = esp_OutpGetData( 1, 1, nodeID, test_val )

    If test_val > max_tran Then
        max_tran = test_val
        max_node = nodeID
    End If

    nodeID = esp_ListGetNextItem( listID, nodeID )

Wend

Msg = "Maximum Value is: " + Str(max_tran)
Print Msg
Msg = "at Node: " + Str(max_node)
Print Msg

esp_ListClear( listID )

End Sub

```

Functions:

Declare Function esp_ListNextAvailableID App () as Long

Returns the next available List ID. Call this when creating a new list to find an empty one. This ID number is then used in all subsequent calls that involve this list.

Declare Sub esp_ListClearAll App ()

Clears and deletes all lists.

Declare Sub esp_ListClear App (ByVal listID as Long)

Clears all the entries in the List specified by listID.

Declare Sub esp_ListAdd App (ByVal listID as Long, ByVal ID as Long)

Adds ID to the List defined by listID.

Declare Sub esp_ListDelete App (ByVal listID as Long, ByVal ID as Long)

Deletes ID from the List defined by listID.

Declare Function esp_ListItemExist App (ByVal listID as Long, ByVal ID as Long)

as Long

Returns TRUE if ID exists in the List defined by listID, FALSE if it does not.

Declare Function esp_ListGetNextItem App(ByVal listID as Long, ByVal ID as Long) as Long

Returns the next ID in the list defined by listID after ID.

Declare Function esp_ListNumber App(ByVal listID as Long) as Long

Returns the number of entities in the list defined by listID.

Declare Function esp_ListSelect App(ByVal listID as Long, ByVal ent_type as Long) as Long

Select entities of ent_type into a list of listID using the standard FEMAP entity selection dialog box.

Declare Function esp_ListSelectAll App(ByVal listID as Long, ByVal ent_type as Long) as Long

Select all entities of ent_type into the list at listID.

Declare Function esp_ListSelectGroup App(ByVal listID as Long, ByVal ent_type as Long, ByVal groupID as Long) as Long

Select all the entities of ent_type in the group of groupID into a list at listID.

Declare Function esp_ListFillByEntity App (ByVal listID as Long, ByVal entity_type as Long, ByVal ID2 as Long, ByVal entity_type2 as Long) as Long

Put entities of entity_type into listID based on their association to entity_type2. E.g. select all points(entity_type) on curve(entity_type2) of ID2 into a list of listID.

Declare Function esp_ListFillEntByEntList App (ByVal listID as Long, ByVal entity_type as Long, ByVal listID2 as Long, ByVal entity_type2 as Long) as Long

Put entities of entity_type into listID based on their association to entity_type2. E.g. select all points(entity_type) on curves(entity_type2) in listID2 into a list of listID.

Declare Function esp_ListPointbyNode App (ByVal nodeID as Long, ByRef pointID as Long) as Long

If the node of nodeID is attached to a point the return value will be TRUE, and the point's ID will be in pointID. If the return value is FALSE the node is not attached.

Declare Function esp_ListCurvebyNode App (ByVal nodeID as Long, ByRef

curveID as Long) as Long

If the node of nodeID is attached to a curve the return value will be TRUE, and the curve's ID will be in curveID. If the return value is FALSE the node is not attached.

Declare Function esp_ListSurfbyNode App (ByVal nodeID as Long, ByRef surfID as Long) as Long

If the node of nodeID is attached to a surface the return value will be TRUE, and the surface's ID will be in surfID. If the return value is FALSE the node is not attached.

Parsing Text

The Line Input command in the FEMAP BASIC Scripting Language brings in a complete line of text from the file being read. In order to parse the individual data entries on the line of text, the following esp_Misc functions are provided.

Every time that you read a line of text, and then want to extract individual pieces of data from that line, the esp_MiscParseInit must be called first to initialize the parsing on the FEMAP side. Once a string has been initialized in the FEMAP parser, you can retrieve Integer and Real values from that string by calling esp_MiscParseInt and esp_MiscParseDouble.

Example:

This example opens the following data file:

```
1 2 3 4 5 6
12, 1.88834, 14, 1.93939, -12.2E5
```

The codes is as follows:

```
Sub Main ( )

Dim st as String
Dim ret_val as Long
Dim value as Long
Dim int_value as Long
Dim real_value as Double

Open "parse.dat" for Input as #1

Line Input #1, st

ret_val = esp_miscParseInit( st )

Print ret_val

ret_val = esp_miscParseInt( 1, value )
Msg = "Read Value " + Str(ret_value) + " ," + Str(value)
Print Msg
```

```

ret_val = esp_miscParseInt( 2, value )
Msg = "Read Value " + Str(ret_val) + " ," + Str(value)
Print Msg

ret_val = esp_miscParseInt( 3, value )
Msg = "Read Value " + Str(ret_val) + " ," + Str(value)
Print Msg

ret_val = esp_miscParseInt( 4, value )
Msg = "Read Value " + Str(ret_val) + " ," + Str(value)
Print Msg

'Read in second line
Line Input #1, st

ret_val = esp_miscParseInit( st )

ret_val = esp_MiscParseInt( 1, int_value )
Msg = "Line 2, Value 1 is " + Str(int_value)
Print Msg

ret_val = esp_MiscParseDouble( 2, real_value )
Msg = "Line 2, Value 2 is " + Str(real_value)
Print Msg

ret_val = esp_MiscParseInt( 3, int_value )
Msg = "Line 2, Value 3 is " + Str(int_value)
Print Msg

ret_val = esp_MiscParseDouble( 4, real_value )
Msg = "Line 2, Value 4 is " + Str(real_value)
Print Msg

ret_val = esp_MiscParseDouble( 5, real_value )
Msg = "Line 2, Value 5 is " + Str(real_value)
Print Msg

Close #1

End Sub

```

As you can see the parser makes it easy to read data separated by spaces or commas.

Functions:

Declare Function esp_MiscParseInit App(ByVal st as String) as Long

Initializes the parsing of a string specified by st.

Declare Function esp_MiscParseInt App(ByVal Index as Long, ByVal value as Long) as Long
--

Extracts the integer number located in the field specified by Index and stores it in value. Index starts at 1.

Declare Function esp_MiscParseDouble App(ByVal Index as Long, ByRef value as Double) as Long

Extracts the real number located in the field specified by Index and stores it in value. Index starts at 1.

Declare Function esp_MiscSerialNumber App (ByRef sn as String) as Long

Retrieves the serial number of the license and puts it in sn.

Declare Function esp_MiscOEMCode App () as Long

Returns the value of the OEM code of the program.

Declare Function esp_MiscSaveNotes App (ByVal notes as String, ByVal newline as Long) as Long

Puts notes into FEMAP translation text notes and turns on include during write. If newline is TRUE a newline of text is created in the existing notes. If FALSE any previous notes are deleted.

Declare Function esp_MiscGetNotes App (ByRef notes as String) as Long

Gets FEMAP translation text notes. String must then be parsed for newlines.

Printing

Once the view and output manipulation functions were complete, we realized that users would probably want to automate the printing of FEMAP graphics. The following function makes it possible to invoke a print of the current graphics window. It is up to you to make sure that the Page Setup and Printer Setup are to your liking before using the BASIC scripting engine to invoke a series of prints.

The command to print the current view is:

Declare Function esp_PrinCurrentView App () as Long

No input parameters, just call this function and the active view is printed using the current printer and page setup parameters.

Database

Database functions that are often used in the actual coding of FEMAP that will certainly come in useful for your own BASIC scripts are as follows:

Declare Function esp_DBNextEntity App(ByVal exist_flag as Long, ByVal

entity_type as Long, ByVal location as Long, ByVal startID as Long) as Long

Used to query the FEMAP database and determined the next entity. The next entity ID value is returned based on your input to this function. Options are:

- exist_flag - Existing - causes this function to return the ID of an existing entity.
NonExisting - causes this function to return the next available ID that is not used.
- entity_type - Any valid FEMAP entity type as defined in basichdr.esp. Common entity types are as follows:

Point
Curve
Surface
Volume
Node
Elem
CSys
Matl
Prop
Load_Dir
Surf_Load
nTherm_Load
eTherm_Load
BC_Dir
BCo
BEq
esp_Text
View
Group
Var
Out_Case
Out_Dir
Out_Data
Report
Boundary
Layer
Matl_Table
Function_Dir

location - Determines in which direction the search for the next entity will happen, options are:

Equal Really the same as the upcoming function DBExist, causes this

	function to only check the existence or nonexistence of the ID specified in startID.
After	Looks at ID numbers greater than startID.
Before	Looks at ID numbers less than startID.
After_Equal	Looks at ID numbers greater than or equal to startID
Before_Equal	Looks at ID numbers less than or equal to startID

startID - ID of the ID to start lo for the next entity.

Declare Function esp_DBExist App(ByVal entity_type as Long, ByVal ID as Long) as Long

Returns TRUE if the entity specified by entity_type and ID exists, returns FALSE if it does not.

The database gets and puts use FEMAP defined types to hold the appropriate data for the entity. Refer to the list of FEMAP defined types to determine the structure of the data for these functions.

Declare Function esp_DBGetPoint App(ByVal ID as Long, ByRef ent as esp_Point) as Long

Retrieves the data of the point at ID and puts it in ent

Declare Function esp_DBGetCurve App(ByVal ID as Long, ByRef ent as esp_Curve) as Long

Retrieves the data of the curve at ID and puts it in ent

Declare Function esp_DBGetSurface App(ByVal ID as Long, ByRef ent as esp_Surface) as Long

Retrieves the data of the surface at ID and puts it in ent

Declare Function esp_DBGetNode App(ByVal ID as Long, ByRef ent as esp_Node) as Long

Retrieves the data of the node at ID and puts it in ent

Declare Function esp_DBGetElement App(ByVal ID as Long, ByRef ent as esp_Element) as Long

Retrieves the data of the element at ID and puts it in ent

Declare Function esp_DBGetBC App(ByVal ent_type as Long, ByVal ID as Long, ByVal setID as Long, ByRef ent as esp_BC) as Long

Retrieves the data of the constraint at ID and puts it in ent

Declare Function esp_DBGetBoundary App(ByVal ID as Long, ByRef ent as esp_Boundary) as Long

Retrieves the data of the boundary at ID and puts it in ent

Declare Function esp_DBGetLoad App(ByVal ent_type as Long, ByVal load_type as Long, ByVal ID as Long, ByVal setID as Long, ByRef L as esp_Load_Value, ByVal l_dir as Long, ByRef d as esp_Load_Dir) as Long

Gets a load based on ent_type and load_type, with ID referring to the entity the load is on. SetID is the load set ID. L, l_dir and d are filled in appropriately.

Declare Function esp_DBGetTitle App(ByVal ID as Long, ByVal ent_type as Long, ByRef title as String) as Long

Gets the title of the entity of ent_type at ID. Use for entity sets only, not individual entities.

DBPut should only be used on entities that have been filled in by DBGet. Otherwise information may not be valid and FEMAP could crash.

Declare Function esp_DBPutPoint App(ByVal ID as Long, ByRef ent as esp_Point) as Long

Puts the data in ent into an existing point record at ID.

Declare Function esp_DBPutCurve App(ByVal ID as Long, ByRef ent as esp_Curve) as Long

Puts the data in ent into an existing curve record at ID.

Declare Function esp_DBPutSurface App(ByVal ID as Long, ByRef ent as esp_Surface) as Long

Puts the data in ent into an existing surface record at ID.

Declare Function esp_DBPutNode App(ByVal ID as Long, ByRef ent as esp_Node) as Long

Puts the data in ent into an existing node record at ID.

Declare Function esp_DBPutElement App(ByVal ID as Long, ByRef ent as esp_Element) as Long

Puts the data in ent into an existing element record at ID.

File Functions

The FEMAP BASIC Scripting Language contains three functions for processing standard FEMAP files.

Declare Function esp_FileSave App () as Long

Calling this function saves the current FEMAP model to the current model file name. If his model has not been saved before, FEMAP will prompt you for a file name and location with the standard file dialog box. This function returns TRUE if the save was successful, otherwise FALSE.

Declare Function esp_FileSaveAs App (ByVal fn as String) as Long

Calling this function saves the current FEMAP model to the file name in fn a path may be included in the filename. This function returns TRUE if the save was successful, otherwise FALSE.

Declare Function esp_FileNeutralRead App (ByVal fn as String) as Long

Read the FEMAP Neutral File specified by fn. Command will search the current directory and your path for the file. This function returns FALSE if the file referenced by fn cannot be found, otherwise it returns TRUE.

Declare Function esp_FileNeutralWrite App (ByVal fn as String) as Long

Writes out a FEMAP Neutral File of your current model to the file specified by fn. If file name contains the complete directory information for a file, it will be stored there. If instead in contains just a filename and extension, it will be stored in the current directory. This function always returns TRUE.

Declare Function esp_FileNastranRead App (ByVal fn as String, ByVal type as Long) as Long

Read a NASTRAN input deck of filename fn. Type is one of the following: MSC_NASTRAN, CSA_NASTRAN, UAI_NASTRAN, ME_NASTRAN, SSS_NASTRAN, or COSMIC_NASTRAN. This function returns FALSE if the file referenced by fn cannot be found, otherwise it returns TRUE.

Declare Function esp_FileNastranPost App (ByVal fn as String, ByVal type as Long) as Long

Read a NASTRAN post file (.f06 or .op2) of filename fn. Type is one of the following: MSC_NASTRAN, CSA_NASTRAN, UAI_NASTRAN, ME_NASTRAN, SSS_NASTRAN, or COSMIC_NASTRAN. This function returns FALSE if the file referenced by fn cannot be found, otherwise it returns TRUE.

Declare Function esp_FileAbaqusRead App (ByVal fn as String) as Long

Reads nodes and elements from an Abaqus .fil or .fin file. This function returns FALSE if the file referenced by fn cannot be found, otherwise it returns TRUE.

Declare Function esp_FileAbaqusPost App (ByVal fn as String) as Long

Reads output data from an Abaqus .fil or .fin file. This function returns FALSE if the file referenced by fn cannot be found, otherwise it returns TRUE.

Declare Function esp_FileDynaRead App (ByVal fn as String) as Long

Reads nodes and elements from an LS Dyna d3* file. This function returns FALSE if the file referenced by fn cannot be found, otherwise it returns TRUE.

Declare Function esp_FileDynaPost App (ByVal fn as String) as Long

Reads output data from an LS Dyna d3* file. This function returns FALSE if the file referenced by fn cannot be found, otherwise it returns TRUE.

Declare Function esp_FileAnsysRead App (ByVal fn as String) as Long

Reads an Ansys .ans or .cdb file This function returns FALSE if the file referenced by fn cannot be found, otherwise it returns TRUE.

Declare Function esp_FileAnsysPost App (ByVal fn as String) as Long

Reads an Ansys .rst or .rth file. This function returns FALSE if the file referenced by fn cannot be found, otherwise it returns TRUE.

Declare Function esp_FileNastranWrite App (ByVal anal_prog as Long, ByVal sol_type as Long) as Long

Runs the file export analysis model command with MSC_NASTRAN, CSA_NASTRAN, UAI_NASTRAN, ME_NASTRAN, SSS_NASTRAN, or COSMIC_NASTRAN as the anal_prog and the sol_type. This function always returns TRUE.

Declare Function esp_FileAnsysWrite App (ByVal sol_type as Long) as Long

Runs the file export analysis model command with Ansys as the solver and sol_type. This function always returns TRUE.

Declare Function esp_FileDynaWrite App (ByVal sol_type as Long) as Long

Runs the file export analysis model command with LS Dyna as the solver and sol_type. This function always returns TRUE.

Declare Function esp_FileProgramRun App (ByVal fn as String) as Long

Run the program file specified by fn. This function returns TRUE if the program runs, otherwise FALSE.

Declare Function esp_FileScriptRun App (ByVal fn as String) as Long

Run the script file specified by fn. This function returns TRUE if the script runs, otherwise FALSE.

Declare Function esp_FilePictSave App (ByVal fn as String, ByVal type as Long) as Long

Save the current view in fn as type 1 for bitmap, 2 for metafile, 3 for placeable metafile or 4 for jpeg. This function returns TRUE if the picture file was created, otherwise FALSE.

Declare Function esp_FileExecWait App (ByVal fn as String, ByVal cl as String) as Long

Launch an executable(.exe) file of filename fn with command line arguments of cl. Function will return when program has finished running.

Declare Function esp_FileFindFile App (ByRef fn as String) as Long

Uses the standard windows file location dialog box and returns the path and filename chosen in fn.

Declare Function esp_FileNew App () as Long

Starts a new file. Will not save the existing model.

Declare Function esp_FileExit App () as Long

Exits FEMAP. Will not save the existing model

Declare Function esp_FilePrefMenu App (ByVal fn as String) as Long

Puts the filename and path in fn into the library preferences for the menu

Declare Function esp_FilePrefMatlDef App (ByVal fn as String) as Long

Puts the filename and path in fn into the library preferences for the material type definition.

Coordinate Functions

These functions mimic the methods in the standard FEMAP coordinate definition dialog box.

Declare Function esp_CoordInWorkplane App (ByVal x as Double, ByVal y as Double, ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates of the location at x,y in the workplane.

Declare Function esp_CoordIntersectCurves App (ByVal cuID1 as Long, ByVal cuID2 as Long, ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates of the intersection of two curves, cuID1 and cuID2. If the curves do not intersect the return value is FALSE.

Declare Function esp_CoordOntoCurve App (ByVal cuID as Long, ByRef loc as esp_Coord, ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates of loc projected onto curve cuID.

Declare Function esp_CoordAlongCurve App (ByVal cuID as Long, ByVal dist as Single, ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates of a percentage distance along the curve cuID

Declare Function esp_CoordCenter App (ByVal cuID as Long, ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates of the center of the arc/circle of cuID.

Declare Function esp_CoordMidpoint App (ByVal cuID as Long, ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates fo the midpoint of the curve cuID.

Declare Function esp_CoordOntoSurface App (ByVal suID as Long, ByRef loc as esp_Coord, ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates of the loc projected onto the surface suID.

Declare Function esp_CoordInSurface App (ByVal suID as Long, ByVal u as Double, ByVal v as Double, ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates of the u,v parametric coordinates of the surface suID.

Declare Function esp_CoordSurfaceIntersect App (ByVal cuID as Long, ByVal suID as Long, ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates of the intersection between the curve cuID and the surface suID. Return is FALSE if they do not intersect.

Declare Function esp_CoordOnPoint App (ByVal pID as Long, ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates of the point pID.

Declare Function esp_CoordOnNode App (ByVal nID as Long, ByRef coord as

esp_Coord) as Long

Get the x,y,z coordinates of the node nID.

Declare Function esp_CoordPick App (ByRef coord as esp_Coord) as Long

Get the x,y,z coordinates using the standard FEMAP coordinate location dialog box.

Vector Functions

These functions mimic the methods in the standard FEMAP vector definition dialog box.

Declare Function esp_VecLocate App (ByRef vec as esp_Vector) as Long

Creates a vector between locations input in Base and Comp.

Declare Function esp_VecLocateLength App (ByRef vec as esp_Vector) as Long

Creates a vector between locations input in Base and Comp of Length.

Declare Function esp_VecDirection App (ByRef vec as esp_Vector) as Long

Creates a vector using base, directional components and length.

Declare Function esp_VecComponents App (ByRef vec as esp_Vector) as Long

Creates a vector using Base and Comp to get length.

Declare Function esp_VecNormal App (ByRef base as esp_Coord, ByRef p1 as esp_Coord, ByRef p2 as esp_Coord, ByVal length as Double, ByRef vec as esp_Vector) as Long

Gets vector using cross product of base, p1,p2 and of length.

Declare Function esp_VecBisect App (ByRef base as esp_Coord, ByRef p1 as esp_Coord, ByRef p2 as esp_Coord, ByVal length as Double, ByRef vec as esp_Vector) as Long

Gets vector of length using bisector of lines between base and p1 and base and p2

Declare Function esp_VecAxis App (ByRef base as esp_Coord, ByVal csys as Long, ByVal axis as Long, ByVal pos as Long, ByVal length as Double, ByRef vec as esp_Vector) as Long

Gets vector using axis of coordinate system csys in direction pos with length.

Declare Function esp_VecTangent App (ByVal cuID as Long, ByRef base as esp_Coord, ByVal pos as Long, ByVal length as Double, ByRef vec as esp_Vector) as Long

Declare Function esp_VecNormSurf App (ByVal pID as Long, ByVal suID as Long, ByVal pos as Long, ByVal length as Double, ByRef vec as esp_Vector) as Long

Get a vector normal to surface suID at point pID using pos for direction and length.

Declare Function esp_VecNormView App (ByRef base as esp_Coord, ByVal pos as Long, ByVal length as Double, ByRef vec as esp_Vector) as Long

Get vector normal to the view at base location using pos for direction and using length

Declare Function esp_VecPick App (ByRef vec as esp_Vector) as Long

Allows you to pick a vector using the standard FEMAP vector definition dialog box.

Duplication Functions

These functions are used to duplicate standard FEMAP entities.

Declare Function esp_DupCopy App (ByVal listID as Long, ByVal ent_type as Long, ByRef vec as esp_Vector, ByVal length as Double) as Long

Copies all entities of ent_type in list along the vector specified by vec and at length. Works for nodes, elements and geometry only.

Declare Function esp_DupRadialCopy App (ByVal listID as Long, ByVal ent_type as Long, ByRef loc as esp_Coord, ByVal length as Double) as Long

Copies all entities of ent_type in list radially along the vector from loc to the entity and at length. Works for nodes, elements and geometry only.

Declare Function esp_DupRotate App (ByVal listID as Long, ByVal ent_type as Long, ByRef vec as esp_Vector, ByVal angle as Double, ByVal dist as Double) as Long

Rotates all entities of ent_type in list around the vector specified by vec and at angle using dist as an offset. Works for nodes, elements and geometry only.

Declare Function esp_DupScale App (ByVal listID as Long, ByVal ent_type as Long, ByVal coord_sys as Long, ByRef loc as esp_Coord, ByVal sx as Double, ByVal sy as Double, ByVal sz as Double) as Long

Scales all entities of ent_type in list about the point loc with scale factors sx,sy,sz. Coord_sys can be used to specify a local coordinate system to scale about. Works for nodes, elements and geometry only.

Declare Function esp_DupReflect App (ByVal listID as Long, ByVal ent_type as

Long, ByVal x as Double, ByVal y as Double, ByVal z as Double, ByVal dx as Double, ByVal dy as Double, ByVal dz as Double) as Long

Reflects all entities of ent_type in list across the plane specified by a normal vector from x,y,z and dx,dy,dz. Works for nodes, elements and geometry only.

Geometry Functions

These functions are used to create FEMAP geometry. They mimic the commands on the FEMAP Geometry menu.

Declare Function esp_LineEndpoints App (ByVal color as Long, ByVal layer as Long, ByVal loc1 as esp_Coord, ByVal loc2 as esp_Coord) as Long

Creates a line of color, on layer between endpoints loc1 and loc2. Returns the ID of the created curve.

Declare Function esp_LinePoints App (ByVal color as Long, ByVal layer as Long, ByVal p1ID as Long, ByVal p2ID as Long) as Long

Creates a line of color, on layer between points of ID p1ID, p2ID. Returns the ID of the created curve.

Declare Function esp_LinePerp App (ByVal color as Long, ByVal layer as Long, ByVal loc as esp_Coord, ByVal cuID as Long) as Long

Creates a line of color, on layer through loc, perpendicular to curve cuID. Returns the ID of the created curve.

Declare Function esp_LineParallel App (ByVal color as Long, ByVal layer as Long, ByVal offset as Double, ByVal cuID as Long) as Long

Creates a line of color, on layer parallel to curve cuID at offset. Returns the ID of the created curve.

Declare Function esp_LineAngletoCurve App (ByVal color as Long, ByVal layer as Long, ByVal angle as Double, ByVal loc as esp_Coord, ByVal cuID as Long) as Long

Creates a line of color, on layer through loc at angle to curve cuID. Returns the ID of the created curve.

Declare Function esp_LineOffset App (ByVal color as Long, ByVal layer as Long, ByVal vec as esp_Vector, ByVal length as Double, ByVal cuID as Long) as Long

Creates a line of color, on layer offset from curve cuID along vec at length. Returns the ID of the created curve.

Declare Function esp_LinePointTangent App (ByVal color as Long, ByVal layer as Long, ByVal loc as esp_Coord, ByVal cuID as Long, ByVal first_half as Integer) as Long

Creates a line of color, on layer through loc tangent to arc/circle cuID. Set first half TRUE to use first segment of arc/circle, false for end segment. Returns the ID of the created curve.

Declare Function esp_LineTangentTangent App (ByVal color as Long, ByVal layer as Long, ByVal cuID1 as Long, ByVal cuID2 as Long, ByVal loc as esp_Coord) as Long

Creates a line of color, on layer tangent to both arcs/circles cuID1, cuID2 near loc. Returns the ID of the created curve.

Declare Function esp_LineAtAngle App (ByVal color as Long, ByVal layer as Long, ByVal angle as Double, ByVal loc as esp_Coord) as Long

Creates a line of color, on layer through loc at angle to x-axis of the workplane. Returns the ID of the created curve.

Declare Function esp_CurveFSIntersect App (ByVal color as Long, ByVal layer as Long, ByVal suID1 as Long, ByVal suID2 as Long, ByVal update as Long) as Long

Creates a curve of color, on layer along the intersection of surfaces suID1 and suID2. If update is true the surfaces are split at the curve. Returns TRUE if the intersection was successful, FALSE otherwise.

Declare Function esp_CurveFSProject App (ByVal color as Long, ByVal layer as Long, ByVal cuID as Long, ByVal suID as Long, ByVal update as Long) as Long

Creates a curve of color, on layer using the normal projection of curve cuID onto solid suID. If update is true the surfaces are split at the curve. Returns TRUE if the projection was successful, FALSE otherwise.

Declare Function esp_CurveFSProjectVector App (ByVal color as Long, ByVal layer as Long, ByVal cuID as Long, ByVal suID as Long, ByVal vec as esp_Vector, ByVal update as Long) as Long

Creates a curve of color, on layer using the vector (vec) projection of curve cuID onto surface suID. If update is true the surfaces are split at the curve. Returns the ID of the created curve.

Declare Function esp_CurveFSParametric App (ByVal color as Long, ByVal layer as Long, ByVal suID as Long, ByVal loc as esp_Coord, ByVal u_dir as Long, ByVal update as Long) as Long

Creates a curve of color, on layer through loc along a constant parametric line. If u_dir is

true the u direction of the surface is used, otherwise the v direction is used. If update is true the surfaces are split at the curve. Returns TRUE if the parametric curve call was successful, FALSE otherwise.

Declare Function esp_CurveLength App (ByVal curveID as Long) as Double

Returns the length of the curve specified by curveID.

Declare Function esp_CircleRadius App (ByVal color as Long, ByVal layer as Long, ByVal center as esp_Coord, ByVal loc as esp_Coord) as Long

Creates a circle of color, on layer with center and through loc.

Declare Function esp_CircleCenter App (ByVal color as Long, ByVal layer as Long, ByVal radius as Double, ByVal center as esp_Coord) as Long

Creates a circle of color, on layer with center and radius. Returns the ID of the created curve.

Declare Function esp_CircleTwoPoints App (ByVal color as Long, ByVal layer as Long, ByVal radius as Double, ByVal loc1 as esp_Coord, ByVal loc2 as esp_Coord) as Long

Creates a circle of color, on layer with radius and through loc1 and loc2. Returns the ID of the created curve.

Declare Function esp_CirclePointsOnArc App (ByVal color as Long, ByVal layer as Long, ByVal loc1 as esp_Coord, ByVal loc2 as esp_Coord, ByVal loc3 as esp_Coord) as Long

Creates a circle of color, on layer through the three locations. Returns the ID of the created curve.

Declare Function esp_CirclePointTangent App (ByVal color as Long, ByVal layer as Long, ByVal cuID as Long, ByVal center as esp_Coord) as Long

Creates a circle of color, on layer tangent to curve cuID with center. Returns the ID of the created curve.

Declare Function esp_CircleTangentTangent App (ByVal color as Long, ByVal layer as Long, ByVal cuID1 as Long, ByVal cuID2 as Long, ByVal radius as Double, ByVal loc as esp_Coord) as Long

Creates a circle of color, on layer tangent to both curve cuID1 and cuID2 with radius and a center near loc. Returns the ID of the created curve.

Declare Function esp_CircleConcentric App (ByVal color as Long, ByVal layer as Long, ByVal cuID as Long, ByVal radius as Double) as Long

Creates a circle of color, on layer concentric to circle cuID with radius. Returns the ID of the created curve.

Declare Function esp_ArcRadiusSE App (ByVal color as Long, ByVal layer as

Long, ByVal radius as Double, ByRef start as esp_Coord, ByRef end as esp_Coord) as Long

Create an arc of color, on layer, with radius and start and end coordinates. Returns the ID of the arc.

Declare Function esp_ArcCenterSE App (ByVal color as Long, ByVal layer as Long, ByRef center as esp_Coord, ByRef start as esp_Coord, ByRef end as esp_Coord) as Long

Create an arc of color, on layer, with center, start and end coordinates. Returns the ID of the arc.

Declare Function esp_ArcAngleSE App (ByVal color as Long, ByVal layer as Long, ByVal angle as Double, ByRef start as esp_Coord, ByRef end as esp_Coord) as Long

Create an arc of color, on layer, of angle and start and end coordinates. Returns the ID of the arc.

Declare Function esp_ArcAngleCS App (ByVal color as Long, ByVal layer as Long, ByVal angle as Double, ByRef center as esp_Coord, ByRef start as esp_Coord) as Long

Create an arc of color, on layer, of angle and center and start coordinates. Returns the ID of the arc.

Declare Function esp_ArcChordCS App (ByVal color as Long, ByVal layer as Long, ByVal length as Double, ByRef center as esp_Coord, ByRef start as esp_Coord) as Long

Create an arc of color, on layer, with radius and start and end coordinates. Returns the ID of the arc.

Declare Function esp_ArcPoints App (ByVal color as Long, ByVal layer as Long, ByRef start as esp_Coord, ByRef mid as esp_Coord, ByRef end as esp_Coord) as Long

Create an arc of color, on layer, with radius and start and end coordinates. Returns the ID of the arc.

Declare Function esp_ArcSEDirection App (ByVal color as Long, ByVal layer as Long, ByRef start as esp_Coord, ByRef end as esp_Coord, ByRef vec as esp_Vector) as Long

Create an arc of color, on layer, with radius and start and end coordinates. Returns the ID of the arc.

Declare Function esp_SplineAddPoint App (ByVal point as esp_Coord, ByVal project as Long) as Long

Add point to an internal list that is used by the esp_SplineCreate function to make a spline of up to 110 points. If project is 1 points are projected onto workplane, if 0 they are not.

Declare Function esp_SplineCreate App (ByVal color as Long, ByVal layer as Long, ByVal control_points as Long) as Long

Creates a spline of color, on layer using the internal list of points generated by esp_SplineAddPoint. The function returns the ID of the spline and clears the internal list. If control_points is true the points are used as control points.

Declare Function esp_SplineParabola App (ByVal color as Long, ByVal layer as Long, ByVal vertex as esp_Coord, ByVal focus as esp_Coord, ByVal end as esp_Coord) as Long

Creates a spline of color, on layer in the shape of a parabola with vertex, focus and stopping near end. Returns the ID of the created spline.

Declare Function esp_SplineEllipse App (ByVal color as Long, ByVal layer as Long, ByVal center as esp_Coord, ByVal vec as esp_Vector, ByVal maj_rad as Double, ByVal min_rad as Double) as Long

Creates a spline of color, on layer in the shape of an ellipse with center, vector of the major radius, the maj_rad and min_rad lengths. Returns the ID of the created spline.

Declare Function esp_SplineHyperbola App (ByVal color as Long, ByVal layer as Long, ByVal vertex as esp_Coord, ByVal vec as esp_Vector, ByVal end as esp_Coord, ByVal height as Double, ByVal angle as Double) as Long

Creates a spline of color, on layer in the shape of a hyperbola with vertex, vec towards focus, asymptote angle and height, and stopping near end. Returns the ID of the created spline.

Declare Function esp_SplineBlend App (ByVal color as Long, ByVal layer as Long, ByVal cuID1 as Long, ByVal end1 as Long, ByVal cuID2 as Long, ByVal end2 as Long, ByVal factor as Double) as Long

Creates a spline of color, on layer by blending between two curves. End1 and end2 control whether the starting point or endpoint of the respective curves is used, 0 for start 1 for end. Returns the ID of the created spline.

Declare Function esp_SplineMidspline App (ByVal color as Long, ByVal layer as Long, ByVal cuID1 as Long, ByVal cuID2 as Long) as Long

Creates a spline of color, on layer between two curves, cuID1 and cuID2. Returns the ID of the created spline.

Declare Function esp_SplineOffset App (ByVal color as Long, ByVal layer as Long, ByVal cuID as Long, ByVal offset as Double, ByVal positive_side as Long)

as Long

Creates a spline of color, on layer offset from spline cuID. If positive_side is 1 spline is offset in global positive direction, if 0 it uses global negative direction. Returns the ID of the created spline.

Declare Function esp_SplineTangents App (ByVal color as Long, ByVal layer as Long, ByRef vec1 as esp_Vector, ByRef vec2 as esp_Vector) as Long

Creates a spline of color, on layer between start and end tangents vec1 and vec2. Returns the ID of the created spline.

Declare Function esp_SplineEquation App (ByVal color as Long, ByVal layer as Long, ByRef c3 as esp_Coord, ByRef c2 as esp_Coord, ByRef c1 as esp_Coord, ByRef c as esp_Coord) as Long

Creates a spline of color, on using parametric equations defined by constants c3,c2,c1,c. Returns the ID of the created spline.

Declare Function esp_SurfBoundary App (ByVal list as Long) as Long

Create a boundary surface from curves in list. Returns the id of the surface.

Declare Function esp_SurfExtrude App (ByVal list as Long, ByRef ex_vec as esp_Vector) as Long

Create a surface by extruding curves in list along ex_vec. Returns the id of the surface.

Declare Function esp_SurfRevolve App (ByVal list as Long, ByRef rot_vec as esp_Vector, ByVal angle as Double) as Long

Create a surface by rotating curves in list an angle around rot_vec. Returns the id of the surface.

Declare Function esp_SurfRuled App (ByVal cuID1 as Long, ByVal cuID2 as Long) as Long

Create a ruled surface between curves cuID1 and cuID2. Returns the id of the surface.

Declare Function esp_SurfSweep App (ByVal culist as Long, ByVal pathlist as Long) as Long

Create a boundary surface from curves in list. Returns the id of the surface.

Declare Function esp_SurfLoft App (ByVal culist as Long) as Long

Create a boundary surface from curves in list. Returns the id of the surface.

Declare Function esp_SurfEdges App (ByVal culist as Long) as Long

Create a surface from 3 or 4 curves in list. Returns the id of the surface.

Declare Function esp_SurfCorners App (ByVal four_corners as Long, ByRef c1 as

esp_Coord, ByRef c2 as esp_Coord, ByRef c3 as esp_Coord, ByRef c4 as esp_Coord) as Long

Create a surface from 3 or 4 corners, c1,c2,c3,c4. Returns the id of the surface.

Declare Function esp_SurfOffset App (ByVal suID as Long, ByVal offset as Double

Create a surface by offseting suID. Returns the id of the surface.

Declare Function esp_SurfArea App (ByVal surfID as Long) as Double

Returns the area of the surface specified by surfID.

Declare Function esp_SolidExtrude App (ByVal suID as Long, ByVal mode as Long, ByRef dir_vec as esp_Vector) as Long

Extrude surface suID into a solid along dir_vec. If mode is 0 it is a new solid, if 1 it is added to the active solid, if 2 it is subtracted from the active solid.

Declare Function esp_SolidRevolve App (ByVal suID as Long, ByVal angle as Double, ByVal mode as Long, ByRef axis as esp_Vector) as Long

Revolve surface suID into a solid using angle around axis. If mode is 0 it is a new solid, if 1 it is added to the active solid, if 2 it is subtracted from the active solid.

Declare Function esp_SolidAdd App (ByVal baseID as Long, ByVal IDlist as Long) as Long

Add solids in IDlist to baseID.

Declare Function esp_SolidSubtract App (ByVal baseID as Long, ByVal IDlist as Long) as Long

Sbtract solids in IDlist from baseID.

Declare Function esp_SolidIntersect App (ByVal baseID as Long, ByVal IDlist as Long) as Long

Intersect solids in IDlist with baseID.

Declare Function esp_SolidFillet App (ByVal IDlist as Long, ByVal radius as Double) as Long

Fillet solid edge curves in IDlist with radius.

Declare Function esp_SolidChamfer App (ByVal IDlist as Long, ByVal length as Double) as Long

Chamfer solid edge curves in IDlist with length.

Declare Function esp_SolidStitch App (ByVal IDlist as Long) as Long

Stitch surfaces in IDlist into a solid.

Declare Function esp_SolidExplode App (ByVal ID as Long) as Long

Runs the explode command on the solid of ID.

Declare Function esp_SolidCleanup App (ByVal ID as Long,ByVal redundant as Long,ByVal sliver as Long,ByVal check as Long) as Long

Runs the cleanup command on the solid of ID. Options are true/false to remove redundant geometry, cleanup slivers, and check geometry.

Meshing Functions

These functions are used to set mesh sizes on geometry, mesh attributes on geometry and to mesh the geometry.

Declare Function esp_MSizeDefault App (ByVal size as Double, ByVal number as Long) as Long

Set default mesh size and minimum number of elements on curves

Declare Function esp_MSizePoint App (ByVal ID as Long, ByVal size as Double) as Long

Set mesh size on point of ID

Declare Function esp_MSizeCurve App (ByVal ID as Long, ByVal size as Long, ByVal value as Double, ByVal bias as Single, ByVal bias_end as Long, ByVal replace as Long) as Long

Set mesh size on curve of ID. If size is true then value contains the size of elements. If it is false value contains the number of elements. Bias and bias_end contain mesh biasing info. If replace if true the command will replace any existing mesh size info.

Declare Function esp_MSizeSurf App (ByVal ID as Long, ByVal size as Double, ByVal replace as Short) as Long

Set mesh size on a surface. If replace if true the command will replace any existing mesh size info.

Declare Function esp_MSizeSolid App (ByVal ID as Long, ByVal size as Double, ByVal replace as Long) as Long

Set mesh size on a solid. If replace if true the command will replace any existing mesh size info.

Declare Function esp_MAttrPoint App (ByVal ID as Long, ByVal propID as Long) as Long

Sets property ID on point.

Declare Function esp_MAttrSurf App (ByVal ID as Long, ByVal propID as Long)

as Long

Sets property ID on surface

Declare Function esp_MAttrSolid App (ByVal ID as Long, ByVal propID as Long) as Long

Sets property ID on solid

Declare Function esp_MAttrCurveProp App (ByVal ID as Long, ByVal propID as Long, ByVal orient as Long, ByRef vec as esp_Vector) as Long

Sets property ID on curve. Use orient to set the orientation type(0 for vector, 1 for location) and fill vec with orientation data.

Declare Function esp_MAttrCurveOffset App (ByVal ID as Long, ByVal type as Long, ByRef End_A as esp_Vector, ByRef End_B as esp_Vector) as Long

Set offsets on curve ID. Use type(0 is vector, 2 is point) use vectors at End_A and End_B for data.

Declare Function esp_MAttrCurveRelease App (ByVal ID as Long, ByRef End_A as esp_BC, ByRef End_B as esp_BC) as Long

Set releases on curve at End_A and End_B.

Declare Function esp_MeshSolid App (ByVal IDlist as Long, ByVal prop as Long) as Long

Mesh solid using property. If meshing attributes are set they will be used

Declare Function esp_MeshSolidFromSurf App (ByVal IDlist as Long, ByVal prop as Long) as Long

Mesh solid from surfaces using property.

Declare Function esp_MeshSolidFromEl App (ByVal IDlist as Long, ByVal prop as Long) as Long

Mesh solid from elements using property.

Declare Function esp_MeshCurve App (ByVal IDlist as Long) as Long

Mesh curves using meshing attributes. Will not work if meshing attributes are not set.

Declare Function esp_MeshSurface App (ByVal IDlist as Long, ByVal use_quads as Long) as Long

Mesh surfaces using meshing attributes. Will not work if meshing attributes are not set.

Declare Function esp_MeshEdgeMembers App (ByVal el_list as Long,ByVal node_list as Long, ByVal propID as Long,ByRef orient_vec as esp_Vector) as Long

Runs the edge members command using el_list, node_list, the property in propID and the orient_vec if the property needs one. Refer to the documentation on the edge members command for more information.

Model Functions

These functions used to create nodes and elements.

Declare Function esp_NodeCreate App (ByVal ID as Long, ByVal color as Long, ByVal layer as Long, ByVal defcs as Long, ByVal outcs as Long, ByVal x as Double, ByVal y as Double, ByVal z as Double) as Long

Create a node. If ID is -1 the node will get the next ID, if ID is positive the node will be created at that ID. If a node already exists at that ID the function will fail.

Declare Function esp_NodeAttach App (ByVal ID as Long, ByVal geom_type as Long, ByVal geom_ID as Long) as Long

Attaches node of nodeID to geomtry of geom_type and geomID

Declare Function esp_NodeRemove App (ByVal ID as Long, ByVal geom_type as Long, ByVal geom_ID as Long) as Long

Removes node of nodeID from geomtry of geom_type and geomID

Declare Function esp_ElemCreate App (ByVal ID as Long, ByRef elem as esp_Element) as Long

Create an element. If ID is -1 the element will get the next ID, if ID is positive the element will be created at that ID. If a element already exists at that ID the function will fail.

Declare Function esp_ElemOrient App (ByVal ID as Long, ByRef vec as esp_Vector) as Long

Orients an element along a vector in vec. The normal direction for plate elements and the Y-axis direction for line elements

Declare Function esp_ElemGetArea App (ByVal ID as Long) as Double

Returns the area of the planar element specified by ID. Returns 0 if not a planar element.

Declare Function esp_ElemGetVolume App (ByVal ID as Long) as Double

Returns the volume of the solid element specified by ID. Returns 0 if not a solid element.

Declare Function esp_LayerCreate App (ByVal ID as Long, ByVal title as String, ByVal color as Long) as Long

Creates a layer of ID with title and color. If a layer at that ID already exists the function returns false.

Property Functions

These functions are used to create FEMAP properties

Declare Function esp_PropCreate App (ByVal ID as Long, ByVal Title as String, ByVal prop_type as Long, ByVal matID as Long, ByVal color as Long, ByVal layer as Long, ByRef flags as esp_Flags, ByRef property as esp_Property) as Long

Creates a property at ID with title, matID, color and layer. Property values are contained in property and are placed based on the prop_type. The table below give the cooresponding values for the various property types.

Property Values

	ROD	BAR	TUBE	LINK	BEAM	SPRING
Type	1	2	3	4	5	6
Flags						
0					tapered	
1						axial(1)/torsion(0)
Values						
0	Area	Area	Dout	Ku_A	Area_A	Stiffness
1		I1	Din	Kv_A	I1_A	Damping
2		I2		Kw_A	I2_A	
3		I12		Kthu_A	I12_A	
4	J	J		Kthv_A	J_A	
5	Ctors	K1,eff		Kthw_A	K1_A,eff	
6		K2,eff		Ku_B	K2_A,eff	
7	NSM	NSM	NSM	Kv_B	NSM_A	
8	Initial Tension	Yf_A1		Kw_B	Yf_A1	
9		Zf_A1		Kthu_B	Zf_A1	
10		Yf_A2		Kthv_B	Yf_A2	
11		Zf_A2		Kthw_B	Zf_A2	
12		Yf_A3			Yf_A3	
13		Zf_A3			Zf_A3	
14		Yf_A4			Yf_A4	
15		Zf_A4			Zf_A4	
16					Yoff_A	
17					Zoff_A	
18						
19						
20					Area_B	
21					I1_B	
22					I2_B	
23					I12_B	
24					J_B	
25					K1_B,eff	
26					K2_B,eff	
27					NSM_B	

28					Yf_B1	
29					Zf_B1	
30					Yf_B2	
31					Zf_B2	
32					Yf_B3	
33					Zf_B3	
34					Yf_B4	
35					Zf_B4	
36					Yoff_B	
37					Z_offB	
38 thru 41						

Property Values (continued)

	DOF SPRING	CURVE BEAM	GAP	SHEAR	MEM-BRANE	BEND-ING
Type	7	8	9	11/12	13/14	15/16
Flags						
0						
1						
2	DOF_A					
3	DOF_B					
Values						
0	Stiffness	Area	Gap,initial	T	T	T
1	Damping	I1	Stiff,tens			
2		I2	Stiff,comp			
3		I12	Stiff,trans			
4		J	Mu,y			
5		K1,eff	Mu,z			
6		K2,eff	PreloadT			
7		NSM	Plane X	NSM	NSM	NSM
8		Yf_A1	Plane Y	F1,eff.fact.		Top Fiber
9		Zf_A1	Plane Z	F2,eff.fact.		Bot Fiber
10		Yf_A2	Width/Area			12I/T3
11		Zf_A2	MaxPenRat			
12		Yf_A3	MaxAdjRat			
13		Zf_A3	MinPenRat			
14		Yf_A4				
15		Zf_A4				
16						
17		R, bend rad				
18 thru 41						

Property Values (continued)

	PLATE	PLANE STRAIN	LAMINATE	MASS	MASS MAT	STIFF MAT
Type	17/18	19/20	21/22	27	28	30
Flags						
0			failure			
1			symmetry			
Values						
0	Tavg,T1	T	Bottom Fiber		M11	K11
1	T2		NSM	Ixx	M12	K12
2	T3		(4.1+) Bond Shear	Ixy	M13	K13
3	T4			Iyy	M14	K14
4				Izx	M15	K15

5				Iyz	M16	K16
6				Izz		
7	NSM	NSM		M or Mx	M22	K22
8	Top Fiber	Top Fiber		Xoff,refCS	M23	K23
9	Bot Fiber	Bot Fiber		Yoff,refCS	M24	K24
10	I2I/T3			Zoff,refCS	M25	K25
11	Ts/T			My	M26	K26
12				Mz		
13						
14					M33	K33
15					M34	K34
16					M35	K35
17					M36	K36
18						
19						
20						
21					M44	K44
22					M45	K45
23					M46	K46
24						
25						
26						
27						
28					M55	K55
29					M56	K56
30						
31						
32						
33						
34						
35					M66	K66
36						

Declare Function esp_PropAddLamina App (ByVal ID as Long, ByVal ply_num as Long, ByVal matID as Long, ByVal thickness as Double, ByVal angle as Double) as Long

Adds a ply to a laminate property ID at ply_num with material matID, thickness and angle. Ply_num must be 1-90

Declare Function esp_PropGetLamina App (ByVal ID as Long, ByVal ply_num as Long, ByRef matID as Long, ByRef thickness as Double, ByRef angle as Double) as Long

Gets the ply material ID, thickness and angle for a laminate property of ID at ply number.

Declare Function esp_PropBeamXSection App (ByVal ID as Long, ByVal Title as String, ByVal matID as Long, ByVal color as Long, ByVal layer as Long, ByVal suID as Long, ByRef y_axis as esp_Vector, ByRef str1 as esp_Coord, ByRef str2 as esp_Coord, ByRef str3 as esp_Coord, ByRef str4 as esp_Coord) as Long

Used to create a general beam cross section property. SuID is ID of a surface in FEMAP that is to be used as cross section. Y_axis is a vector that denotes the y direction of the beam cross section. Str1,str2,str3,str4 are the x,y locations of the stress recovery locations on the cross section.

Declare Function esp_PropGet App (ByVal ID as Long, ByRef flags As esp_Flags,ByRef property as esp_Property) as Long

Fill flags and property with the data from property ID

Declare Function esp_PropGetType App (ByVal ID as Long, ByRef type as Long, ByRef matID as Long) as Long

Get the type and material ID of property at ID

Declare Function esp_PropPut App (ByVal ID as Long, ByRef flags As esp_Flags,ByRef property as esp_Property) as Long

Put the flags and property data into property ID. This command will overwrite all existing property data at that ID.

Declare Function esp_PropPlateMats App (ByVal ID as Long, ByVal Bending_ID as Long, ByVal Shear_ID as Long, ByVal Coupling_ID as Long) as Long

Sets material ID's for Bending, Shear and Coupling on plate property. Supported in NASTRAN only.

Refer to the section above for property type numbers and value locations.

Material Functions

The FEMAP BASIC Scripting Language contains eleven functions for creating FEMAP materials.

Declare Function esp_MatlCreateIsotropic App (ByVal ID as Long, ByVal Title as String, ByVal color as Long, ByVal layer as Long, ByRef mat as esp_Matl_Iso) as Long

Creates an Isotropic material at ID, using the specified title, color, layer and values contained in mat as follows.


```
Type esp_MatI_Iso
  E as Double
  G as Double
  Nu as Double
  a as Double
  k as Double
  Cp as Double
  L_Tension as Double
  L_Comp as Double
  L_Shear as Double
  Density as Double
  Damping as Double
  Mat_Temp as Double
```

End Type

Declare Function esp_MatICreateOrtho2D App (ByVal ID as Long, ByVal Title as String, ByVal color as Long, ByVal layer as Long, ByRef mat as esp_MatI_Ortho_2D) as Long

Creates a 2D Orthotropic material at ID, using the specified title, color, layer and values contained in mat as follows.

```
Type esp_MatI_Ortho_2D
  E1 as Double
  E2 as Double
  G12 as Double
  G1z as Double
  G2z as Double
  Nu as Double
  a1 as Double
  a2 as Double
  k11 as Double
  k12 as Double
  k13 as Double
  k22 as Double
  k23 as Double
  k33 as Double
  Cp as Double
  Stress_Limits as Long
  L_Tension1 as Double
  L_Tension2 as Double
  L_Comp1 as Double
  L_Comp2 as Double
  L_Shear as Double
  Density as Double
  Damping as Double
  Mat_Temp as Double
  Tsia_Wu as Double
```

End Type

Declare Function esp_MatICreateOrtho3D App (ByVal ID as Long, ByVal Title as String, ByVal color as Long, ByVal layer as Long, ByRef mat as esp_MatI_Ortho_3D) as Long

Creates a 3D Orthotropic material at ID, using the specified title, color, layer and values contained in mat as follows.

```
Type esp_MatI_Ortho_3D
```

```
E1 as Double
```

```
E2 as Double
```

E3 as Double
G12 as Double
G23 as Double
G13 as Double
Nu12 as Double
Nu23 as Double
Nu13 as Double
a1 as Double
a2 as Double
a3 as Double
k11 as Double
k12 as Double

k13 as Double
k22 as Double
k23 as Double
k33 as Double
Cp as Double
L_Tension as Double
L_Comp as Double
L_Shear as Double
Density as Double
Damping as Double
Mat_Temp as Double

End Type

Declare Function esp_MatlCreateAnIso2D App (ByVal ID as Long, ByVal Title as String, ByVal color as Long, ByVal layer as Long, ByVal mat as esp_Matl_AnIso_2D) as Long

Creates a 2D Anisotropic material at ID, using the specified title, color, layer and values contained in mat as follows.

Type esp_Matl_AnIso_2D

G11 as Double

G12 as Double
G13 as Double
G22 as Double
G23 as Double
G33 as Double
a1 as Double
a2 as Double
a12 as Double
k11 as Double
k12 as Double

k13 as Double
k22 as Double
k23 as Double
k33 as Double
Cp as Double
L_Tension as Double
L_Comp as Double
L_Shear as Double
Density as Double
Damping as Double
Mat_Temp as Double

End Type

Declare Function esp_MatlCreateAnIso3D App (ByVal ID as Long, ByVal Title as String, ByVal color as Long, ByVal layer as Long, ByVal mat as esp_Matl_AnIso_3D) as Long

Creates a 3D Anisotropic material at ID, using the specified title, color, layer and values contained in mat as follows.

Type esp_Matl_AnIso_3D

G11 as Double
G12 as Double
G13 as Double
G14 as Double
G15 as Double

G16 as Double
G22 as Double
G23 as Double
G24 as Double
G25 as Double

G26 as Double
G33 as Double
G34 as Double
G35 as Double
G36 as Double
G44 as Double
G45 as Double
G46 as Double
G55 as Double
G56 as Double
G66 as Double
a1 as Double
a2 as Double
a3 as Double

a4 as Double
a5 as Double
a6 as Double
k1 as Double
k2 as Double
k3 as Double
k4 as Double
k5 as Double
k6 as Double
Cp as Double
Density as Double
Damping as Double
Mat_Temp as Double

End Type

Declare Function esp_MatlCreateHyper App (ByVal ID as Long, ByVal Title as String, ByVal color as Long, ByVal layer as Long, ByVal matl as esp_Matl_HyperElastic) as Long

Creates a Hyperelastic material at ID, using the specified title, color, layer and values contaned in mat as follows.

Type esp_Matl_HyperElastic

A01 as Double
A02 as Double
A03 as Double
A04 as Double
A05 as Double
A10 as Double
A11 as Double
A12 as Double
A13 as Double
A14 as Double
A20 as Double
A21 as Double
A22 as Double
A23 as Double
A30 as Double
A31 as Double
A32 as Double
A40 as Double
A41 as Double
A50 as Double
D1 as Double
D2 as Double

D3 as Double
D4 as Double
D5 as Double
D6 as Double
Dist_Order as Double
Vol_Order as Double
a as Double
Cp as Double
Density as Double
Damping as Double
Mat_Temp as Double

End Type

Declare Function esp_MatlCreateGeneral App (ByVal ID as Long, ByVal Title as String, ByVal color as Long, ByVal layer as Long, ByVal subtype as Long) as Long

Creates a general material at ID, using the specified title, color, layer and of the specified subtype. Subtype must be the number of a material contained in the Material Type Definition file. The material values are added using the following four functions.

Declare Function esp_MatlAddRValue App (ByVal ID as Long, ByVal index as Long, ByVal value as Double) as Long

Add a real value at the index of a general material of ID.

Declare Function esp_MatlAddIValue App (ByVal ID as Long, ByVal index as Long, ByVal value as Long) as Long

Add an integer value at the index of a general material of ID.

Declare Function esp_MatlAddBValue App (ByVal ID as Long, ByVal index as Long, ByVal value as Long) as Long

Add a boolean value at the index of a general material of ID.

Declare Function esp_MatlAddFunction App (ByVal ID as Long, ByVal index as Long, ByVal funcID as Long) as Long

Add a function of funcID at the index of a general material of ID.

Declare Function esp_MatlGetRValue App (ByVal ID as Long, ByVal index as Long, ByRef value as Double) as Long

Get the real value of material ID at index.

Declare Function esp_MatlGetIValue App (ByVal ID as Long, ByVal index as Long, ByRef value as Long) as Long

Get the integer value of material ID at index.

Declare Function esp_MatlGetBValue App (ByVal ID as Long, ByVal index as Long, ByRef value as Long) as Long

Get the boolean value of material ID at index.

Declare Function esp_MatlGetFunction App (ByVal ID as Long, ByVal index as Long, ByRef funcID as Long) as Long

Get the function ID of material ID at index.

Declare Function esp_MatlGetType App (ByVal ID as Long, ByRef type as Long) as Long

Get the type of material ID. The types are numbers 0-6 and coorespond to isotropic, orthotropic 2D and 3D, anisotropic 2D and 3D, hyperelastic and general.

Declare Function esp_MatlGetSubType App (ByVal ID as Long, ByRef subtype as Long) as Long

Get the subtype of material ID. Subtypes are returned for general materials only. They are returned as numbers, not the string names.

Refer to the neutral file documetation for what values are at what indices for regular FEMAP materials

Load Functions

These functions are used to create FEMAP loads

Declare Function esp_LoadCreateSet App (ByVal ID as Long, ByVal Title as String) as Long

Creates a load set at ID with title.

Declare Function esp_LoadNodal App (ByVal type as Long, ByVal ID_List as Long,ByVal c_sys as Long, ByRef L as esp_Load_Value, ByVal l_dir as Long, ByRef d as esp_Load_Dir) as Long

Create a nodal load of type on nodes in ID_list. C_sys is the coordinate system for the loads, values are in L, l_dir is the direction method with d containing the direction values.

Declare Function esp_LoadElemental App (ByVal type as Long, ByVal ID_List as Long,ByVal c_sys as Long, ByRef L as esp_Load_Value) as Long

Create an elemental load of type on elements in ID_List. C_sys is for local load coordinate system. The direction is always perpendicular to the element face normal specified by L.face_num.

Declare Function esp_LoadPoint App (ByVal type as Long, ByVal ID_List as Long,ByVal c_sys as Long, ByRef L as esp_Load_Value, ByVal l_dir as Long, ByRef d as esp_Load_Dir) as Long

Create a point load of type on points in ID_list. C_sys is the coordinate system for the loads, values are in L, l_dir is the direction method with d containing the direction values.

Declare Function esp_LoadCurve App (ByVal type as Long, ByVal ID_List as Long,ByVal c_sys as Long, ByRef L as esp_Load_Value, ByVal l_dir as

Long, ByRef d as esp_Load_Dir) as Long

Create a curve load of type on curves in ID_list. C_sys is the coordinate system for the loads, values are in L, l_dir is the direction method with d containing the direction values.

Declare Function esp_LoadSurface App (ByVal type as Long, ByVal ID_List as Long, ByVal c_sys as Long, ByRef L as esp_Load_Value, ByVal l_dir as Long, ByRef d as esp_Load_Dir) as Long

Create a surface load of type on surfaces in ID_list. C_sys is the coordinate system for the loads, values are in L, l_dir is the direction method with d containing the direction values.

Constraint Functions

Declare Function esp_BCCreateSet App (ByVal ID as Long, ByVal Title as String) as Long

Creates a constraint set at ID with title.

Declare Function esp_BCNode App (ByVal ID_List as Long, ByVal c_sys as Long, ByRef bcc as esp_BC) as Long

Create a nodal constraint on nodes in ID_list. C_sys is the coordinate system for the constraints, dof are in bcc.

Declare Function esp_BCPoint App (ByVal ID_List as Long, ByVal opt as Long) as Long

Create nodal constraint on points in ID_list. Option is 1 for fixed, 2 for translations, 3 for rotations.

Declare Function esp_BCCurve App (ByVal ID_List as Long, ByVal opt as Long) as Long

Create nodal constraint on curves in ID_list. Option is 1 for fixed, 2 for translations, 3 for rotations.

Declare Function esp_BCSurface App (ByVal ID_List as Long, ByVal opt as Long) as Long

Create nodal constraint on surfaces in ID_list. Option is 1 for fixed, 2 for translations, 3 for rotations.

Group Functions

These functions are used to create and manipulate FEMAP groups.

Declare Function esp_GrpNewGroup App (ByVal ID as Long, ByVal title as String) as Long

Create a new group at ID with title.

Declare Function esp_GrpActivateGroup App (ByVal ID as Long) as Long

Make group at ID the active group.

Declare Function esp_GrpEvaluate App (ByVal ID as Long, ByVal always as Long) as Long

Evaluate group at ID. If always is true evaluate always will be turned on.

Declare Function esp_GrpAutomaticAdd App (ByVal ID as Long) as Long

Turn on auto add for group at ID.

Declare Function esp_GrpRenumber App (ByVal ID as Long, ByVal yesno as Long) as Long

Renumber group at ID. If yesno is true automatic renumbering takes place if entities in group are renumbered

Declare Function esp_GrpSelectModel App () as Long

Creates rules in active group to select all entities in the model

Declare Function esp_GrpResetRules App () as Long

Resets all rules in active group

Declare Function esp_GrpCopy App (ByVal ID as Long, ByVal newID as Long, ByVal title as String) as Long

Copies group of ID into group of newID with title

Declare Function esp_GrpCondense App (ByVal ID as Long, ByVal newID as Long, ByVal title as String) as Long

Copies group of ID into group of newID and title using only ID rules

Declare Function esp_GrpAnd App (ByVal ID1 as Long, ByVal ID2 as Long, ByVal newID as Long, ByVal title as String) as Long

Use logical AND between groups ID1 and ID2 to make group newID.

Declare Function esp_GrpOr App (ByVal ID1 as Long, ByVal ID2 as Long, ByVal newID as Long, ByVal title as String) as Long

Use logical OR between groups ID1 and ID2 to make group newID.

Declare Function esp_GrpExclusiveOr App (ByVal ID1 as Long, ByVal ID2 as

Long, ByVal newID as Long, ByVal title as String) as Long

Use logical EXCLUSIVE OR between groups ID1 and ID2 to make group newID.

Declare Function esp_GrpNot App (ByVal ID as Long, ByVal newID as Long, ByVal title as String) as Long

Use logical NOT on groups ID to make group newID.

Declare Function esp_GrpGenerateProperty App (ByVal IDlist as Long) as Long

Generate groups using properties in IDlist.

Declare Function esp_GrpGenerateMaterial App (ByVal IDlist as Long) as Long

Generate groups using materials in IDlist.

Declare Function esp_GrpGenerateElemType App (ByVal IDlist as Long) as Long

Generate groups using types of elements in IDlist.

Declare Function esp_GrpPeel App (ByVal IDlist as Long, ByVal no_layers as Long, ByVal from_outer as Long, ByVal from_remain as Long) as Long

Declare Function esp_GrpEntID App (ByVal ent_type as Long, ByVal IDlist as Long, ByVal Grp_ID as Long) as Long

Add entities of ent_type contained in IDlist into group of Grp_ID.

Declare Function esp_GrpEntMethod App (ByVal ent_type as Long, ByVal method as Long, ByVal IDlist as Long, ByVal Grp_ID as Long) as Long

Add entities of ent_type based on method, using entities in IDlist.

Modify Functions

These functions are similar to those found under the FEMAP modify menu.

Declare Function esp_ModScaleLoad App (ByVal IDlist as Long, ByVal ent_type as Long, ByVal load_type as Long, ByVal scale_factor as Double, ByVal add_factor as Double) as Long

Scales loads of load_type that exist on entities of ent_type that are in IDlist. Loads are multiplied by scale_factor and then add_factor is added.

Declare Function esp_ModLoadFunc App (ByVal IDlist as Long, ByVal ent_type as Long, ByVal load_type as Long, ByVal funcID as Long) as Long

Change function on loads of load_type that exist on entities of ent_type that are in IDlist.

Declare Function esp_ModLoadPhase App (ByVal IDlist as Long, ByVal ent_type

as Long, ByVal phase as Double) as Long

Change phase of loads of load_type that exist on entities of ent_type that are in Idlist.

Declare Function esp_ModPermBC App (ByVal IDlist as Long, ByVal bc as esp_BC) as Long

Change permanent constraints of nodes in Idlist.

Declare Function esp_ModPropID App (ByVal IDlist as Long, ByVal new_prop as Long) as Long

Change the property ID of elements in Idlist. New property must be the same as type as the old one.

Declare Function esp_ModMatlID App (ByVal IDlist as Long, ByVal new_matl as Long) as Long

Change the Material ID of elements in Idlist.

Declare Function esp_ModElOffset App (ByVal IDlist as Long, ByVal type as Long, ByVal EndA as esp_Vector, ByVal EndB as esp_Vector) as Long

Change element offsets of elements in Idlist. Only line elements will be used. Type is the offset type(0 is vector, 2 is node).

Declare Function esp_ModElOrient App (ByVal IDlist as Long, ByVal orient_method as Long, ByVal node_ID as Long, ByVal Vec as esp_Vector) as Long

Change orientation of elements in Idlist. If orient_method is 0 use node_ID, if 1 use vec if 2 it converts nodal orientation to equivalent vector orientation, if 3 it makes them perpendicular to the elemental x-axis.

Declare Function esp_ModElType App (ByVal IDlist as Long, ByVal prop_ID as Long, ByVal orient_vec as esp_Vector) as Long

Change type of elements in Idlist using a new property ID. If an orientation vector is required for the new type use the orient_vec. Element geometries must match.

Declare Function esp_ModSurfDiv App (ByVal IDlist as Long, ByVal s_div as Long, ByVal t_div as Long, ByVal tolerance as Double) as Long

Change number of s,t divisions on surfaces in Idlist. Tolerance should be between 0 and 1, a lower tolerance will increase the number of facets used to draw the surface.

Declare Function esp_ModMatlAngle App (ByVal IDlist as Long, ByVal method as Long, ByVal CSys as Long, ByVal Axis as Long, ByVal Angle as Single, ByVal Vec as esp_Vector) as Long

Change the material angle of the elements in Idlist.

Declare Function esp_ModElReverse App (ByVal IDlist as Long, ByVal rev_opt as

Long, ByRef Vec as esp_Vector) as Long

Change the normal direction of the elements in Idlist. If rev_opt is 0 the normals are reversed, if it's 1 they are pointed outward, if it's 2 they are pointed inward, if it's 3 use the vec to specify the normal direction

Declare Function esp_ModElOrder App (ByVal IDlist as Long, ByVal to_para as Long, ByVal mid_nodes as Long) as Long

Change the order of elements in Idlist. If to_param is true they are made parabolic and if mid_nodes is true midside nodes are created. If false they are made linear.

Declare Function esp_ModElSplitQuads App (ByVal IDlist as Long) as Long

Splits quad elements in Idlist into triangle elements.

Declare Function esp_ModSplineOrder App (ByVal IDlist as Long, ByVal order as Long) as Long

Change the order of the splines in IDlist

Declare Function esp_ModSplineKnots App (ByVal IDlist as Long, ByRef loc as esp_Coord) as Long

Insert a spline knot at loc into the spline in IDlist

Declare Function esp_ModColor App (ByVal IDlist as Long, ByVal ent_type as Long, ByVal colorID as Long) as Long

Change the color of entities of ent_type in Idlist.

Declare Function esp_ModLayer App (ByVal IDlist as Long, ByVal ent_type as Long, ByVal layerID as Long) as Long

Change the layer of entities of ent_type in Idlist.

Declare Function esp_ModBoundSurf App (ByVal BoundID as Long, ByVal map as Long, ByVal surfID as Long) as Long

Change the underlying surface of a boundary. If map is true the boundary is mapped onto the surface, if false it is unmapped.

Declare Function esp_ModMoveBy App (ByVal IDlist as Long, ByVal ent_type as Long, ByRef Vec as esp_Vector) as Long

Move entities of ent_type in Idlist along vec.

Declare Function esp_ModRotateBy App (ByVal IDlist as Long, ByVal ent_type as Long, ByRef Axis as esp_Vector, ByVal angle as Double, ByVal dist as Double) as Long

Rotate entities of ent_type in Idlist an angle about axis. If dist is non-zero they will be translated as well

Declare Function esp_ModRotateTo App (ByVal IDlist as Long, ByVal ent_type as Long, ByRef Axis as esp_Vector, ByRef loc1 as esp_Coord, ByRef loc2 as esp_Coord) as Long

Rotate entities of ent_type in Idlist from loc1 to loc2 about axis. If dist is non-zero they will be translated as well

Declare Function esp_ModProjOnCurve App (ByVal IDlist as Long, ByVal ent_type as Long, ByVal cu_ID as Long) as Long

Project entities of ent_type(point or node) in Idlist onto curve cu_ID.

Declare Function esp_ModProjOnSurf App (ByVal IDlist as Long, ByVal ent_type as Long, ByVal su_ID as Long) as Long

Project entities of ent_type(point or node) in Idlist onto surface su_ID.

Declare Function esp_ModScale App (ByVal IDlist as Long, ByVal ent_type as Long, ByRef scale_factors as esp_Coord, ByRef s_loc as esp_Coord, ByVal csysID as Long) as Long

Scale entities of ent_type using scale_factors about s_loc in the coordinate system csysID.

Declare Function esp_ModRenumber App (ByVal IDlist as Long, ByVal ent_type as Long, ByVal startID as Long) as Long

Renumber entities of ent_type in Idlist using startID as the beginning.

Delete Function

This functions is used to delete FEMAP entities.

Declare Function esp_DeleteEnt App (ByVal IDlist as Long, ByVal ent_type as Long) as Long

Delete entities of ent_type in Idlist. Hierarchical delete rules still apply.

Declare Function esp_DeleteEntSingle App (ByVal ID as Long, ByVal ent_type as Long) as Long

Delete entities of ent_type at Id. Hierarchical delete rules still apply.

Coordinate System Functions

These functions are used to create and manipulate coordinate systems.

Declare Function esp_CSysCreateVec App (ByRef base as esp_Coord, ByRef xdir as

esp_Vector, ByRef ydir as esp_Vector) as Long

Create a coordinate system using base and x and y axis vectors. Returns the ID of the new coordinate system.

Declare Function esp_CSysCreatePoints App (ByRef base as esp_Coord,ByRef x_point as esp_Coord,ByRef y_point as esp_Coord) as Long

Create a coordinate system using base and points on x and y axis. Returns the ID of the new coordinate system.

Declare Function esp_XformUp App (ByVal csysID as Long, ByRef loc as esp_Coord) as Long

Transform the loc into coordinates of coordinate system csysID.

Declare Function esp_XformDown App (ByVal csysID as Long, ByRef loc as esp_Coord) as Long

Transform the loc into global coordinates from coordinate system csysID.

Declare Function esp_XformAcross App (ByVal in_ID as Long,ByVal out_ID as Long, ByRef loc as esp_Coord) as Long

Transform the loc into from coordinates of coordinate system in_ID to coordinates of coordinate system out_ID.

Declare Function esp_XformWorkplane App (ByRef loc as esp_Coord, ByRef work as esp_Coord) as Long

Transform the loc into from global coordinates to workplane coordinates.

Tools Functions

These functions are similar to those found under the FEMAP tools menu.

Declare Function esp_ChkCoincNodes App (ByVal IDlist1 as Long, ByVal IDlist2 as Long, ByVal tol as Double) as Long

Check for coincident nodes between Idlist1 and Idlist2 using tolerance. If Idlist2 is 0 Idlist1 is checked among itself.

Declare Function esp_VarCreate App (ByVal name as String, ByVal value as Double, ByVal eq as String) as Long

Create a variable of name and assigns it either value or equation.

Declare Function esp_VarGet App (ByVal name as String, ByRef value as Double, ByRef eq as String) as Long

Gets the variable of name and fills in value or equation.

Info Functions

These functions used to obtain information about the current FEMAP model.

Declare Function esp_InfoMin App (ByVal entity as Long) as Long

Returns the minimum ID of entity in the current model.

Declare Function esp_InfoMax App (ByVal entity as Long) as Long

Returns the maximum ID of entity in the current model.

Declare Function esp_InfoNext App (ByVal entity as Long) as Long

Returns the ID of the next entity to be created in the current model.

Declare Function esp_InfoNumber App (ByVal entity as Long) as Long

Returns the number of entities in the current model.

Declare Function esp_InfoActive App (ByVal entity as Long, ByVal set_to as Long) as Long

Sets the active entity to set_to. E.g. set the active group to 3.

Function Functions

These functions are used to create FEMAP functions.

Declare Function esp_FuncCreate App (ByVal ID as Long, ByVal st as String, ByVal func_type as Long) as Long

Create a function at ID with title st of func_type.

Declare Function esp_FuncAdd App (ByVal ID as Long, ByVal x as Double, ByVal y as Double) as Long

Add an x,y value to function of ID.

Global Constants

Please refer to the basichdr.esp file to see a list of the global constants that can be used to pass values by name into FEMAP.

User Defined Types

The following types are used by the FEMAP api functions to pass data back and forth between the script and FEMAP. Declare variables of the type and reference the data as shown

e.g.

```
Dim property as esp_Property
```

```
property.val1 = 1.0
```

Type esp_Property

```
val1 as Single  
val2 as Single  
val3 as Single  
val4 as Single  
val5 as Single  
val6 as Single  
val7 as Single  
val8 as Single  
val9 as Single  
val10 as Single  
val11 as Single  
val12 as Single  
val13 as Single  
val14 as Single  
val15 as Single  
val16 as Single  
val17 as Single  
val18 as Single  
val19 as Single  
val20 as Single  
val21 as Single  
val22 as Single  
val23 as Single  
val24 as Single  
val25 as Single  
val26 as Single  
val27 as Single  
val28 as Single  
val29 as Single  
val30 as Single  
val31 as Single  
val32 as Single  
val33 as Single
```

```
        val34 as Single
        val35 as Single
        val36 as Single
        val37 as Single
        val38 as Single
        val39 as Single
End Type
```

```
Type esp_Flags
    val1 as Single
    val2 as Single
    val3 as Single
    val4 as Single
End Type
```

```
Type esp_Node
    x as Single
    y as Single
    z as Single
    tx_bc as Single
    ty_bc as Single
    tz_bc as Single
    rx_bc as Single
    ry_bc as Single
    rz_bc as Single
End Type
```

```
    Type esp_Point
        x as Single
        y as Single
        z as Single
    End Type
```

```
Type esp_Curve
    type as Single
    point_1 as Single
    point_2 as Single
    point_3 as Single
    point_4 as Single
    point_5 as Single
    mesh_elem as Single
    address as Single
    engine as Single

End Type
```

```

Type esp_Surface
    type as Single
    curve_1 as Single
    curve_2 as Single
    curve_3 as Single
    curve_4 as Single
    curve_5 as Single
    curve_6 as Single
    address as Single
    engine as Single
End Type

Type esp_Boundary
    Surf_ID as Single
    Curve_ListID as Single
End Type

Type esp_Element
    type as Single
    Prop_ID as Single
    Topology as Single
    Node_1 as Single
    Node_2 as Single
    Node_3 as Single
    Node_4 as Single
    Node_5 as Single
    Node_6 as Single
    Node_7 as Single
    Node_8 as Single
    Node_9 as Single
    Node_10 as Single
    Node_11 as Single
    Node_12 as Single
    Node_13 as Single
    Node_14 as Single
    Node_15 as Single
    Node_16 as Single
    Node_17 as Single
    Node_18 as Single
    Node_19 as Single
    Node_20 as Single
    Orient_ID as Single
    Orient_x as Single
    Orient_y as Single

```



```
        Orient_z as Single
End Type
```

```
Type esp_BC
    tx as Single
    ty as Single
    tz as Single
    rx as Single
    ry as Single
    rz as Single
End Type
```

```
Type esp_Coord
    x as Double
    y as Double
    z as Double
End Type
```

```
Type esp_Vector
    Base_x as Double
    Base_y as Double
    Base_z as Double
    Comp_x as Double
    Comp_y as Double
    Comp_z as Double
    Length as Double
End Type
```

```
Type esp_MatI_Iso
    E as Double
    G as Double
    Nu as Double
    a as Double
    k as Double
    Cp as Double
    L_Tension as Double
    L_Comp as Double
    L_Shear as Double
    Density as Double
    Damping as Double
    Mat_Temp as Double
End Type
```

```
Type esp_MatI_Ortho_2D
    E1 as Double
```

```

E2 as Double
G12 as Double
G1z as Double
G2z as Double
Nu as Double
a1 as Double
a2 as Double
k11 as Double
k12 as Double
k13 as Double
k22 as Double
k23 as Double
k33 as Double
Cp as Double
Stress_Limits as Long
L_Tension1 as Double
L_Tension2 as Double
L_Comp1 as Double
L_Comp2 as Double
L_Shear as Double
Density as Double
Damping as Double
Mat_Temp as Double
Tsia_Wu as Double
End Type

```

```

Type esp_MatI_Ortho_3D
E1 as Double
E2 as Double
E3 as Double
G12 as Double
G23 as Double
G13 as Double
Nu12 as Double
Nu23 as Double
Nu13 as Double
a1 as Double
a2 as Double
a3 as Double
k11 as Double
k12 as Double
k13 as Double
k22 as Double
k23 as Double
k33 as Double

```

```

        Cp as Double
        L_Tension as Double
        L_Comp as Double
        L_Shear as Double
        Density as Double
        Damping as Double
        Mat_Temp as Double
End Type

```

```

Type esp_MatI_AnIso_2D
    G11 as Double
    G12 as Double
    G13 as Double
    G22 as Double
    G23 as Double
    G33 as Double
    a1 as Double
    a2 as Double
    a12 as Double
    k11 as Double
    k12 as Double
    k13 as Double
    k22 as Double
    k23 as Double
    k33 as Double
    Cp as Double
    L_Tension as Double
    L_Comp as Double
    L_Shear as Double
    Density as Double
    Damping as Double
    Mat_Temp as Double
End Type

```

```

Type esp_MatI_AnIso_3D
    G11 as Double
    G12 as Double
    G13 as Double
    G14 as Double
    G15 as Double
    G16 as Double
    G22 as Double
    G23 as Double
    G24 as Double
    G25 as Double

```

G26 as Double
G33 as Double
G34 as Double
G35 as Double
G36 as Double
G44 as Double
G45 as Double
G46 as Double
G55 as Double
G56 as Double
G66 as Double
a1 as Double
a2 as Double
a3 as Double
a4 as Double
a5 as Double
a6 as Double
k1 as Double
k2 as Double
k3 as Double
k4 as Double
k5 as Double
k6 as Double
Cp as Double
Density as Double
Damping as Double
Mat_Temp as Double
End Type

Type esp_MatI_HyperElastic
A01 as Double
A02 as Double
A03 as Double
A04 as Double
A05 as Double
A10 as Double
A11 as Double
A12 as Double
A13 as Double
A14 as Double
A20 as Double
A21 as Double
A22 as Double
A23 as Double
A30 as Double

```

    A31 as Double
    A32 as Double
    A40 as Double
    A41 as Double
    A50 as Double
    D1 as Double
    D2 as Double
    D3 as Double
    D4 as Double
    D5 as Double
    D6 as Double
    Dist_Order as Double
    Vol_Order as Double
    a as Double
    Cp as Double
    Density as Double
    Damping as Double
    Mat_Temp as Double
End Type

```

```

Type esp_Load_Value
    x as Single
    y as Single
    z as Single
    magnitude as Single
    func_ID as Long
    phase as Single
    Pfunc_ID as Long
    face_num as Long
End Type

```

```

Type esp_Load_Dir
    vec as esp_Vector
    curve_ID as Long
    surf_ID as Long
End Type

```