Webinar:
Introduction to the Femap API

Rachel Backes, ATA Engineering

June 20th 2019

13290 Evening Creek Drive S, Suite 250, San Diego CA 92128

(858) 480-2000 www.ata-e.com

ata-engineering @ATAEngineering

ATA Provides High-Value Engineering Services

ATA Engineering helps to overcome product design challenges across a range of industries

Aerospace

Robotics & Controls

Themed Entertainment

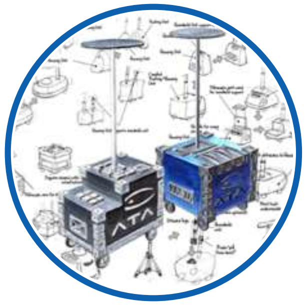
Defense

Industrial & Mining Equipment

Consumer Products

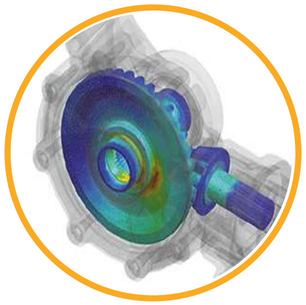
We Offer Complete, Integrated Solutions

With expertise in design, analysis, and test, ATA engineers regularly work across disciplines to find the optimal solution



Design

From initial concept development to detailed structural design



Analysis

Comprehensive structural, fluid, acoustic, and thermal analysis services

Test

Industry-leading structural test services for extreme loading environments

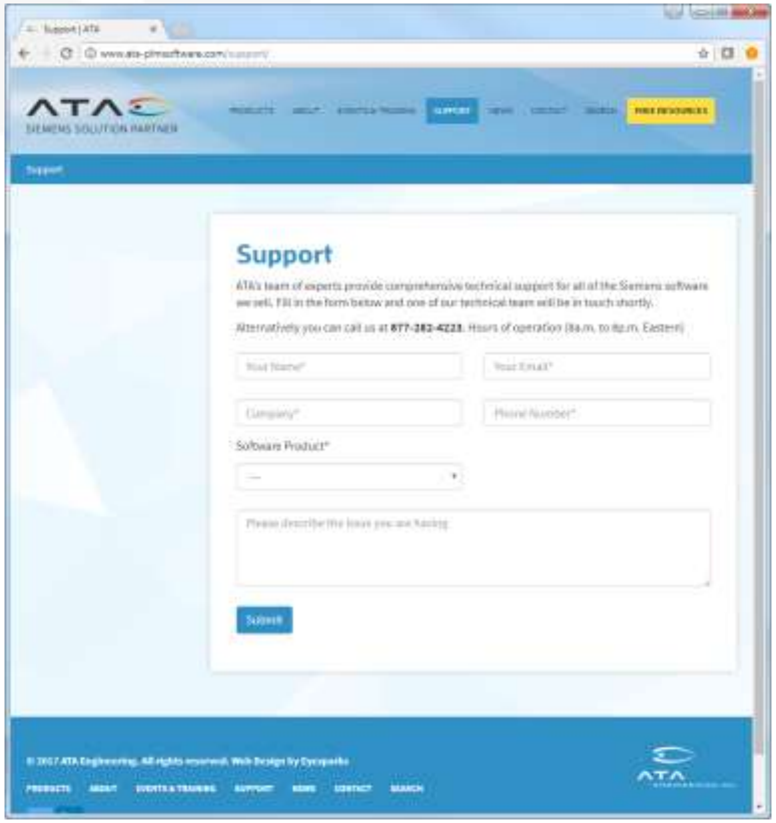
ATA is a Value-Added Reseller for Siemens PLM Software

ATA offers training, free resources, and hotline support for a variety of Siemens products.

- Siemens product lines we support include:
  - STAR-CCM+
  - Femap
  - Simcenter Nastran (formerly NX Nastran)
  - Simcenter 3D
  - NX CAD & CAM
  - Teamcenter
  - Solid Edge
- Contact the hotline at 877-ATA-4CAE or http://ata-plmsoftware.com/support
- Developer of the official Simcenter Nastran training materials
- Preferred North American provider of Simcenter Nastran training
- Recognized as Smart Expert Partner with validated expertise in Femap and STAR-CCM+

Visit Our Website for Product Information and Free Resources

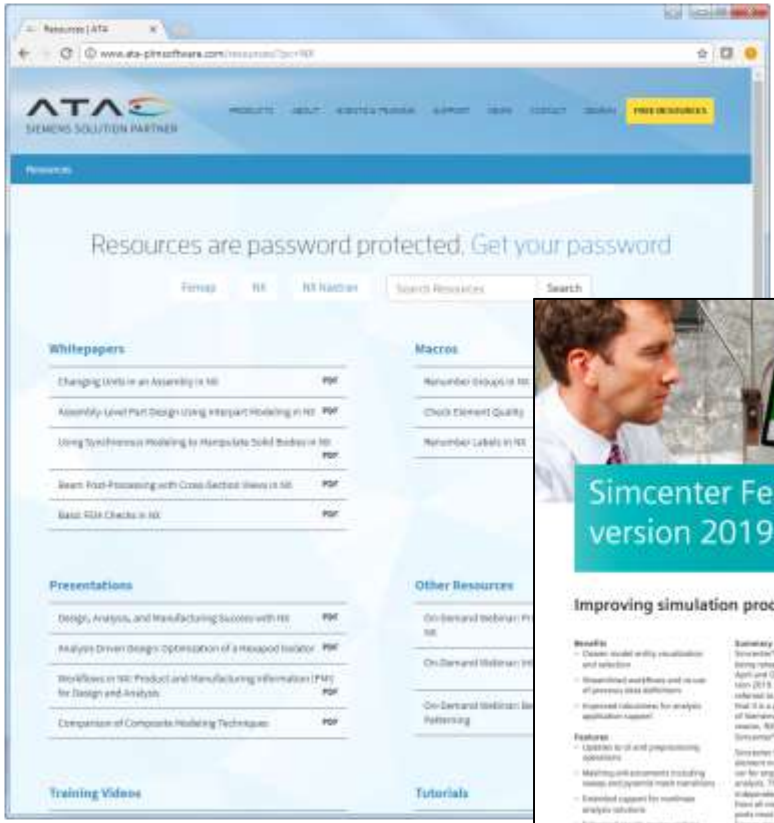
www.ata-plmsoftware.com



Support

ATA's team of experts provide comprehensive technical support for all of the Siemens software we sell. Fill in the form below and one of our technical team will be in touch shortly.

Alternatively you can call us at **877-282-4223**. Hours of operation (8a.m. to 8p.m. Eastern)



Resources are password protected. Get your password

Whitepapers
- Changing Units in an Assembly in NX — PDF
- Assembly-Level Part Design Using Interpart Modeling in NX — PDF
- Using Synchronous Modeling to Manipulate Solid Bodies in NX — PDF
- Beam Post-Processing with Cross-Section Views in NX — PDF
- Basic FEA Checks in NX — PDF

Macros
- Renumber Groups in NX
- Check Element Quality
- Renumber Labels in NX

Presentations
- Design, Analysis, and Manufacturing Success with NX — PDF
- Analysis Driven Design: Optimization of a Hexapod Isolator — PDF
- Workflows in NX: Product and Manufacturing Information (PMI) for Design and Analysis — PDF
- Comparison of Composite Modeling Techniques — PDF

Other Resources
- On-Demand Webinar: [illegible]
- On-Demand Webinar: [illegible]
- On-Demand Webinar: [illegible] Patterning

Training Videos

Tutorials

SIEMENS
Ingenuity for life

Simcenter Femap version 2019.1

Improving simulation productivity

Benefits
- Clearer model entity visualization and selection
- Streamlined workflows and re-use of previous data definitions
- Improved calculations for analysis application support

Features
- Updates to UI and preprocessing operations
- Meshing enhancements including sweep and pyramid mesh transitions
- Extended support for nonlinear analysis solutions
- Enhanced results review options for partial results vectors

Summary

[illegible]

Visualization and user interface

[illegible]

siemens.com/femap

Introduction to the Femap API

Questions we'll answer

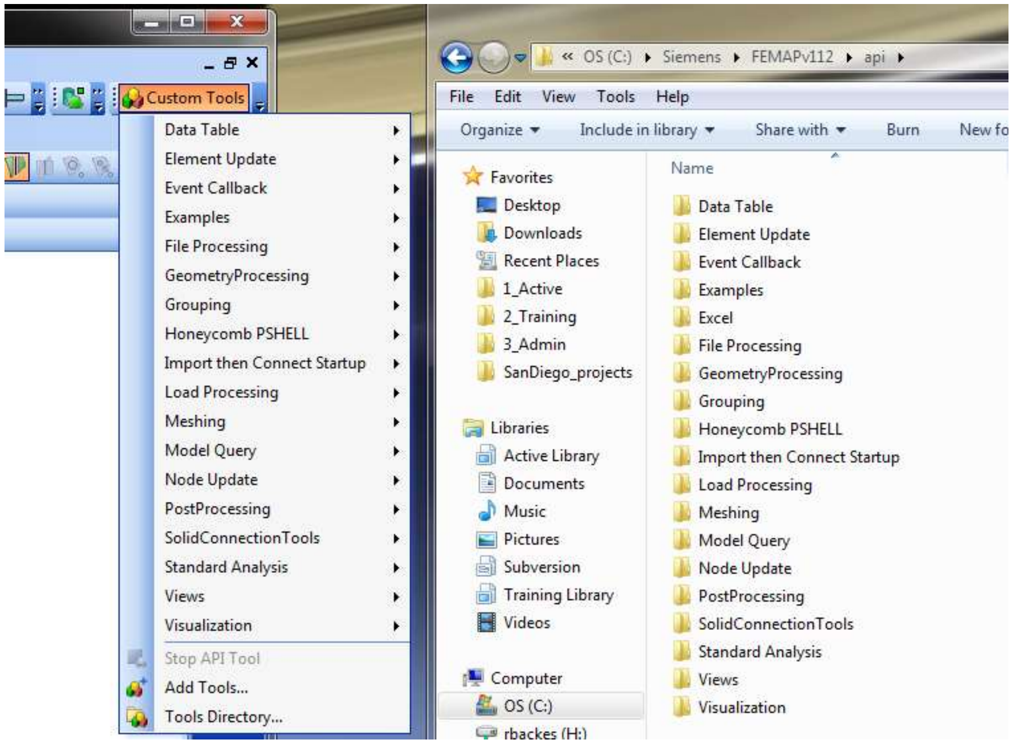
- What is an API?
- Why is it useful?
- What types of objects are used?
- Where do I write an API?
- What if I need help?
- How do I…
  - edit nodes and elements?
  - get user-selected nodes and loop through them?
  - write an if statement?
  - list data to the messages pane?
  - use arrays and variants?
  - use return codes?
- What do I do if I get an error message?

API Basics – What is an API?

- API (Application Program Interface)
  - The term API technically refers to the language and libraries that can control Femap
  - It's also used to refer to individual API programs
- Instead of opening Femap and clicking on tools or entering data manually, an API does the same tasks either inside Femap or from another program like Excel
  - Uses Microsoft's OLE/COM framework
  - Native language is Visual Basic (VB), but other languages can be used (Python, C#...)
  - Contains virtually every command in Femap
- FEMAP API enables the user to automate repetitive or tedious tasks via computer code by providing access to the objects in the model and FEMAP functionalities

API Basics – Custom Tools

- Every Custom Tool is an API
  - APIs that are useful to many users come with Femap and can be found in the Custom Tools menu
  - You can open, read, and edit the API codes
    - <Femap Install>\api



Why API? – Repetitive Tasks & Consistency

- Example:
  - Sarah just started analyzing a new design for an electronics enclosure. She knows that she and the designer will have to analyze this part many times before they have a final design. Each time, she'll need to take an RSS of her X, Y, and Z gravitational acceleration results and add a thermal result. She must enter a factor of safety of 1.5 for the gravitational cases and 1.2 for the thermal cases.
- By writing an API to do her load combinations, she'll…
  - save time in the long run
  - ensure that the combinations are performed exactly the same way every time
  - eliminate mistakes, such as a typo in the factors of safety

She expects to do something repetitively

She has to do almost the same thing every time

She has to enter values by hand

Why API? – Do something Femap doesn’t have a built-in tool for

- Example:
  - Jim’s lead analyst gave him an old model of an assembly and asked him to perform some mesh updates. Unfortunately, the model came from many bulk data files and contains 100+ components. He needs to figure out what groups/bulk data files his elements are in.
- Femap can tell you what elements are in a group, but not what groups an element is in.
- He can write an API to do this task and…
  - save time on this project
  - have a new tool that he can use on similar tasks in the future
  - share his tool with his colleagues so everyone is more efficient

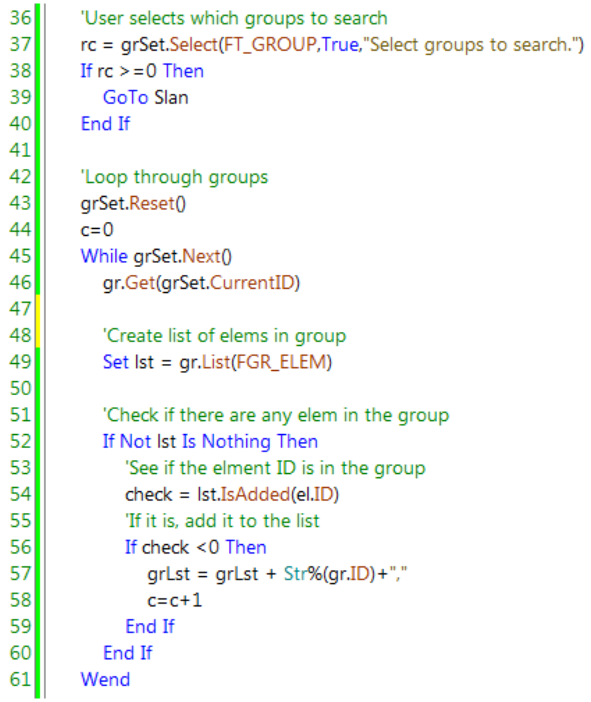
He’d needs to sort through a lot things (Something computers can do really quickly)

Femap doesn’t have a built-in tool to do what he needs

A well written API can be used on many different models and can be shared with others

DEMO

- Let's see Jim's API in action…
- This API is called **Find Element Groups.BAS** and is available on the ATA PLM Software website (along with the other examples in this webinar)
  - https://ata-plmsoftware.com



```
'User selects which groups to search
rc = grSet.Select(FT_GROUP,True,"Select groups to search.")
If rc >=0 Then
    GoTo Slan
End If

'Loop through groups
grSet.Reset()
c=0
While grSet.Next()
    gr.Get(grSet.CurrentID)

    'Create list of elems in group
    Set lst = gr.List(FGR_ELEM)

    'Check if there are any elem in the group
    If Not lst Is Nothing Then
        'See if the elment ID is in the group
        check = lst.IsAdded(el.ID)
        'If it is, add it to the list
        If check <0 Then
            grLst = grLst + Str%(gr.ID)+","
            c=c+1
        End If
    End If
Wend
```

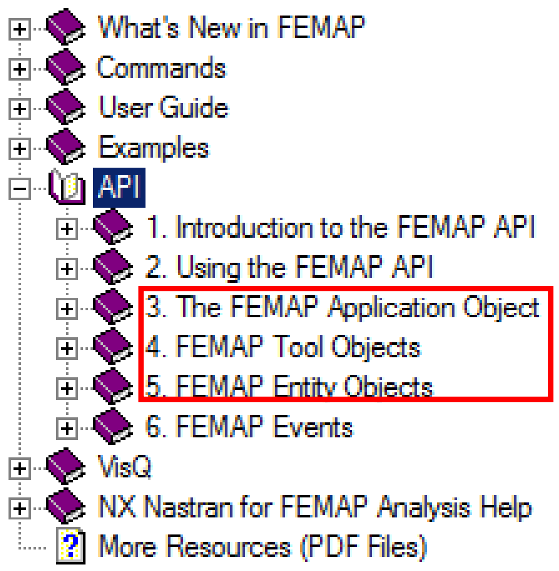
API Objects

- Femap API uses VB, which is an Object Oriented language
- Objects are model or user interface entities
  - Elements → femap.elem
  - Output Vector → femap.output
  - Groups → femap.group
  - Data Table → femap.DataTable
- Objects have properties and methods
  - Properties describe information about the object,
    - Element.topology = CQUAD4
  - Methods provide a way of interacting with an object
    - Surface.mesh()
- You can create multiple instances of an object type using Dim command; each object has its unique name.
  - Dim searchElem As Elem, myElem As Elem

Femap Objects

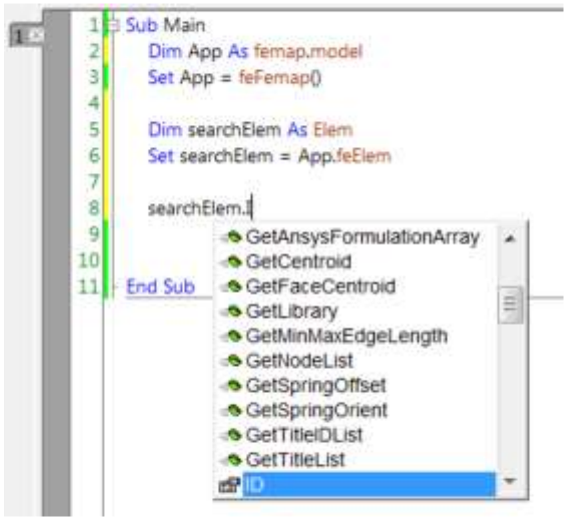
- Femap uses three categories (classes) of Objects

1. Application Object
   - Provides access to Femap GUI and menu options
   - Contains methods performed on the application level
     - Mesh Curve, Regenerate View, Delete Output, List to Messages
2. Entity Object
   - Anything created and stored in the Femap database
   - Elements, Nodes, Analysis Sets
3. Tool Object
   - A special class that provides functionality and is not stored in the model database
   - Sets, Data Table, Copy/Move tools, Beam Calculator

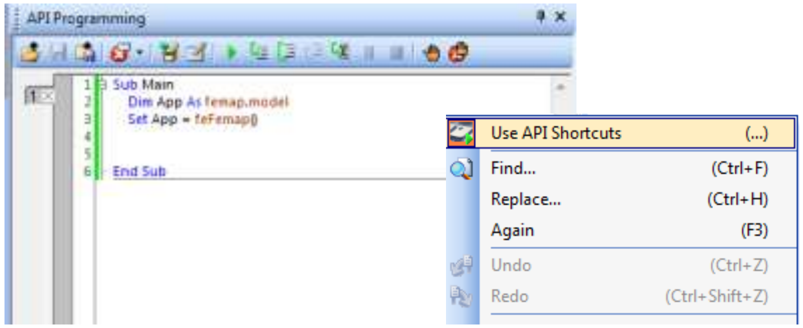


Where APIs can be written and edited

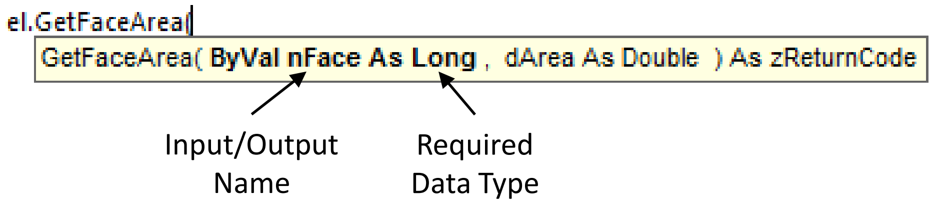
- Femap comes with a built-in API coding environment
  - Tools > Programming > API Programming
  - Uses the **Winwrap** flavor of Visual Basic
  - Write and run APIs directly
  - Interactive help pop-ups guide users
    - Property
    - Method
  - "Use API Shortcuts" toggles between Femap and Programming Pane keyboard shortcuts
    - Ctrl-Z (undo) in model vs. in API code



```
Sub Main
    Dim App As femap.model
    Set App = feFemap()

    Dim searchElem As Elem
    Set searchElem = App.feElem

    searchElem.


End Sub
```



```
Sub Main
    Dim App As femap.model
    Set App = feFemap()


End Sub
```



```
el.GetFaceArea(
GetFaceArea( ByVal nFace As Long , dArea As Double ) As zReturnCode
```

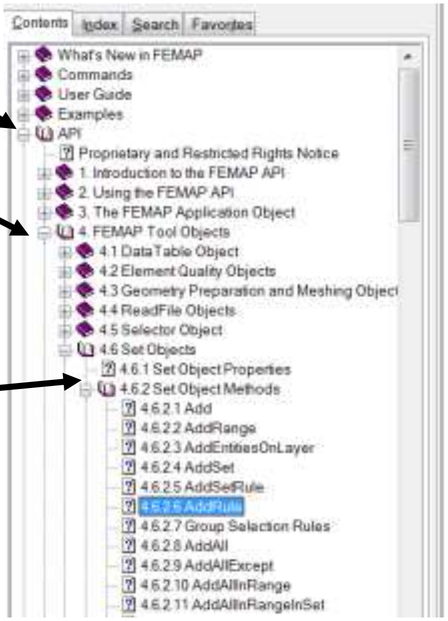
Where to get help – Femap Objects

- Help > Programming

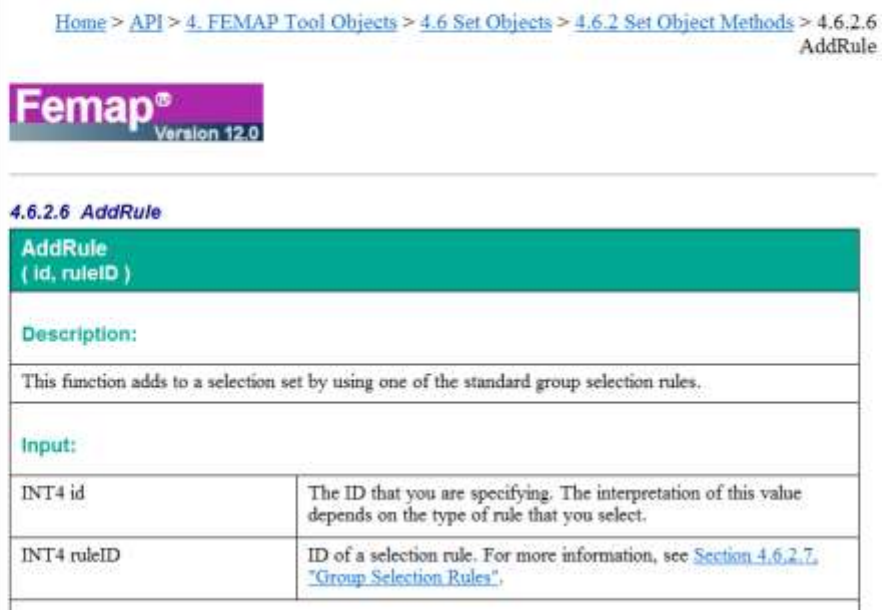
<Femap Install>\pdf\api.pdf

Organized by Object Classes
Section 3: Application
Section 4: Tool
Section 5: Entity

…then by Properties and Methods
Section X.1: Properties
Section X.2: Methods



- What's New in FEMAP
- Commands
- User Guide
- Examples
- API
  - Proprietary and Restricted Rights Notice
  - 1. Introduction to the FEMAP API
  - 2. Using the FEMAP API
  - 3. The FEMAP Application Object
  - 4. FEMAP Tool Objects
    - 4.1 DataTable Object
    - 4.2 Element Quality Objects
    - 4.3 Geometry Preparation and Meshing Object
    - 4.4 ReadFile Objects
    - 4.5 Selector Object
    - 4.6 Set Objects
      - 4.6.1 Set Object Properties
      - 4.6.2 Set Object Methods
        - 4.6.2.1 Add
        - 4.6.2.2 AddRange
        - 4.6.2.3 AddEntitiesOnLayer
        - 4.6.2.4 AddSet
        - 4.6.2.5 AddSetRule
        - 4.6.2.6 AddRule
        - 4.6.2.7 Group Selection Rules
        - 4.6.2.8 AddAll
        - 4.6.2.9 AddAllExcept
        - 4.6.2.10 AddAllInRange
        - 4.6.2.11 AddAllInRangeInSet



Home > API > 4. FEMAP Tool Objects > 4.6 Set Objects > 4.6.2 Set Object Methods > 4.6.2.6 AddRule

4.6.2.6 AddRule

AddRule
(id, ruleID)

Description:

This function adds to a selection set by using one of the standard group selection rules.

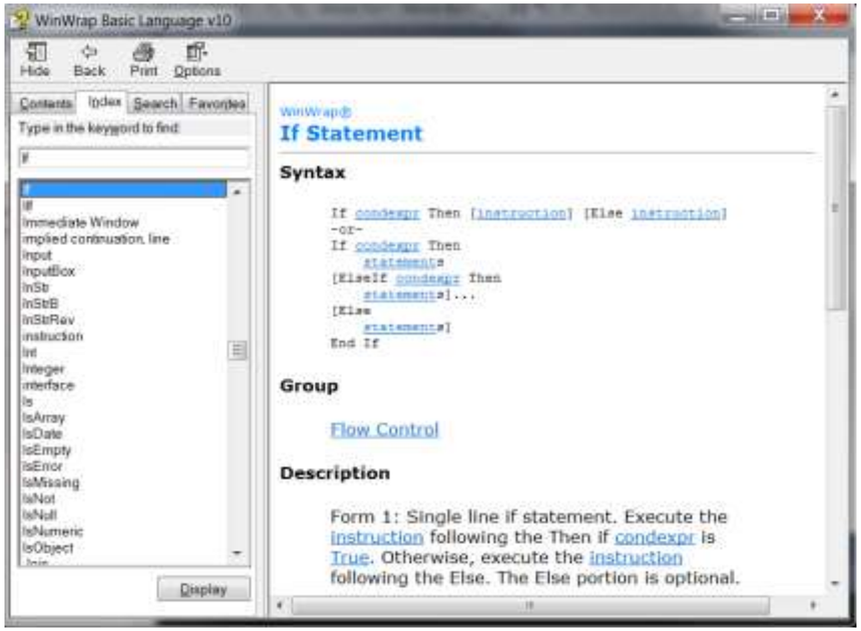
Input:

INT4 id	The ID that you are specifying. The interpretation of this value depends on the type of rule that you select.
INT4 ruleID	ID of a selection rule. For more information, see Section 4.6.2.7, "Group Selection Rules".

- Names and Searching
  - Searching is hard if you don't know what you're looking for
    - Context help or the Help File Tree are often more useful to find names for properties/methods that work with your object
  - Application methods often start with "fe"
    - feAppMessage, feViewRegenerate
  - There are common properties and methods that are used for many objects
    - ID, title, Get, Put, Next
  - Pay attention to Femap names
    - When **charting,** the user plots **data series**

Where to get help – Winwrap VB

- When working in the programming pane, press F1



WinWrap Basic Language v10

Hide Back Print Options

Contents Index Search Favorites

Type in the keyword to find

If
IIf
Immediate Window
implied continuation line
Input
InputBox
InStr
InStrB
InStrRev
instruction
Int
Integer
interface
Is
IsArray
IsDate
IsEmpty
IsError
IsMissing
IsNot
IsNull
IsNumeric
IsObject

Display

WinWrap®

If Statement

Syntax

```
If condexpr Then [instruction] [Else instruction]
-or-
If condexpr Then
    statements
[ElseIf condexpr Then
    statements]...
[Else
    statements]
End If
```

Group

Flow Control

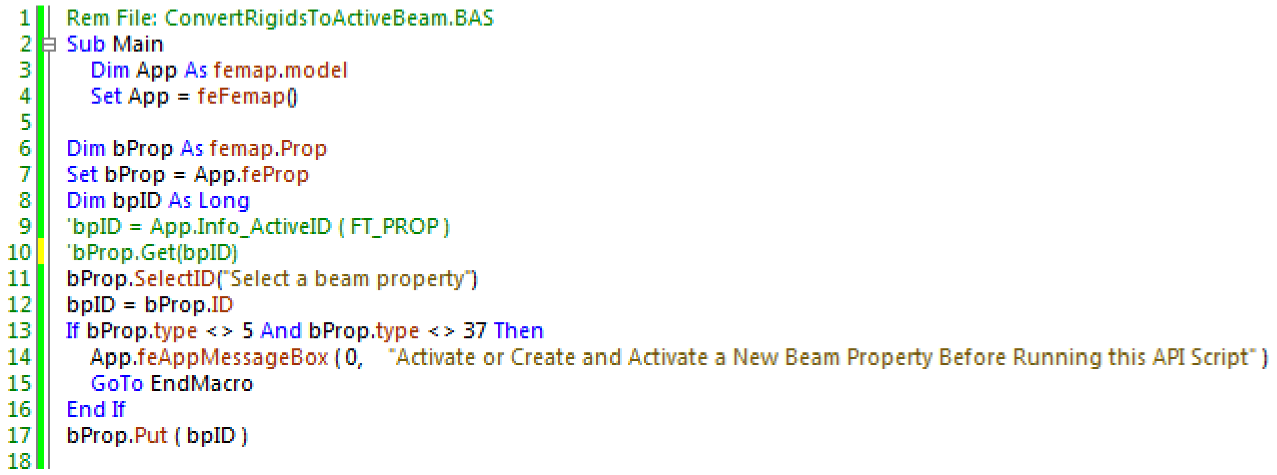
Description

Form 1: Single line if statement. Execute the instruction following the Then if condexpr is True. Otherwise, execute the instruction following the Else. The Else portion is optional.

- This help menu contains objects, properties, and methods general to Winwrap VB
  - Common variable types (Integer, Long, Double, String)
  - Math functions (Abs, Tan, Sqr)
  - Statements (If, For, While)
- When using online help (Google, StackExchange) *most* VB help will work with Winwrap, but not everything in VB is supported by Winwrap.

DEMO

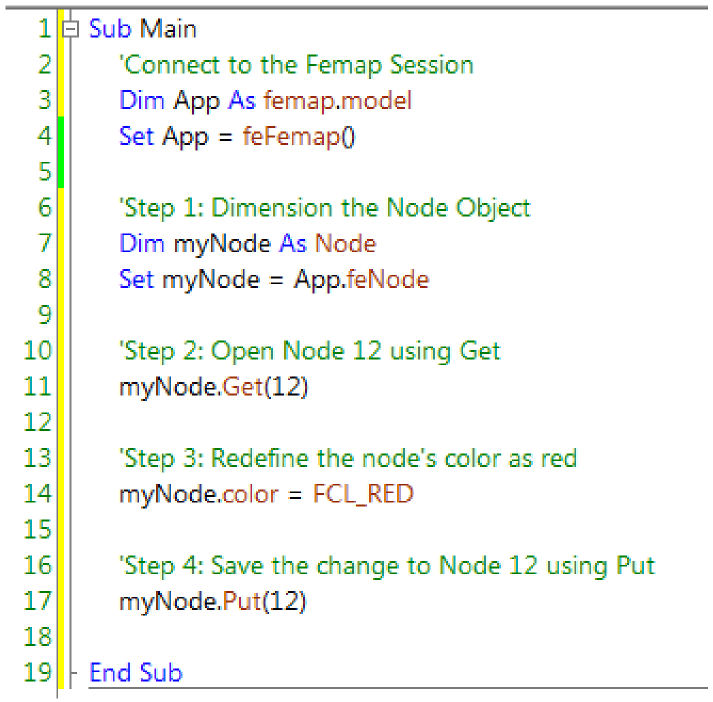
- Use the help files to edit a Custom Tool to do something different
- **Convert Rigids To Active Beam.BAS**
  - This API converts rigid elements to the activated beam property
  - It's hard to remember to activate your beam property first, so lets edit the code to allow the user to select which property they want to use.



```
Rem File: ConvertRigidsToActiveBeam.BAS
Sub Main
    Dim App As femap.model
    Set App = feFemap()

Dim bProp As femap.Prop
Set bProp = App.feProp
Dim bpID As Long
'bpID = App.Info_ActiveID ( FT_PROP )
'bProp.Get(bpID)
bProp.SelectID("Select a beam property")
bpID = bProp.ID
If bProp.type <> 5 And bProp.type <> 37 Then
    App.feAppMessageBox ( 0,   "Activate or Create and Activate a New Beam Property Before Running this API Script" )
    GoTo EndMacro
End If
bProp.Put ( bpID )
```

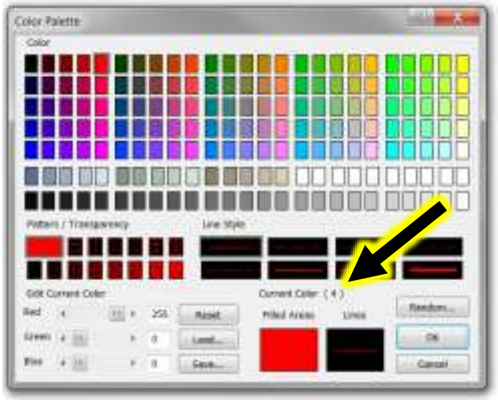
Common API Tasks – Edit an Entity

- Example: Change the color of node 12 to red
  1. Dimension a node object
     - Use ***Dim*** to associate the name with the node object type
     - Then use ***Set*** to associate the named object with your Femap session as a node
  2. Use the ***Get*** method to open the node object for editing
  3. Redefine the node ***color*** property
  4. Use the ***Put*** method to save the updated node object



```
Sub Main
    'Connect to the Femap Session
    Dim App As femap.model
    Set App = feFemap()

    'Step 1: Dimension the Node Object
    Dim myNode As Node
    Set myNode = App.feNode

    'Step 2: Open Node 12 using Get
    myNode.Get(12)

    'Step 3: Redefine the node's color as red
    myNode.color = FCL_RED

    'Step 4: Save the change to Node 12 using Put
    myNode.Put(12)

End Sub
```

<u>**Note**</u>:
- Every color in Femap is defined by an integer value.
- The integer value of any color can be found when selecting a color from the color palette
- Many color integer values have named constants associated with them. Section 3.3.7, Femap Constants, of the API help guide has a list of color names. Color names all begin with "FCL_"

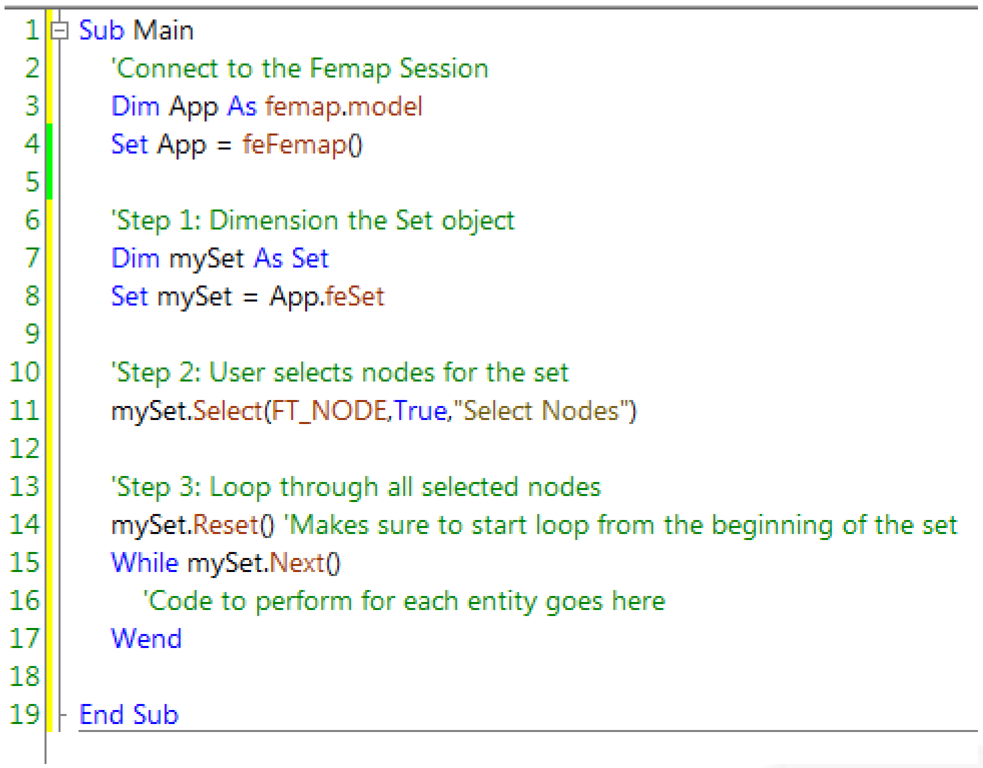


Sets

- The Femap API uses the **Set** object to hold lists of ID numbers
  - The set holds **ONLY** ID numbers
  - It is **NOT** associated with a specific entity type
- Sets are the best way to hold a list of IDs in the API
- Sets are a Tool
- Sets are used…
  - To hold user selected IDs
  - To define multiple entities when using methods
  - To loop through several entities
  - As an alternative to more clunky lists like Femap groups and array variables

Common API Tasks – User Select & While Loop

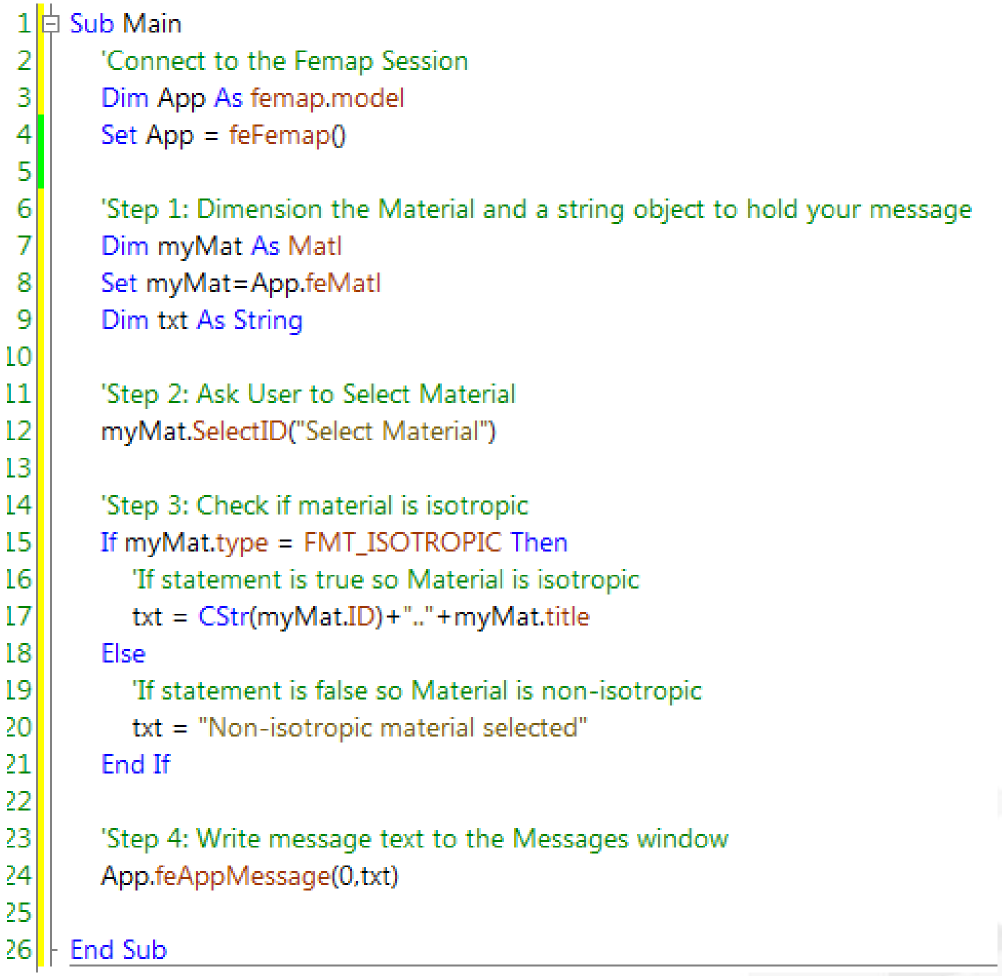
- Example: Have the user select several nodes then use a loop to perform tasks on those nodes
  1. Dimension the Set object
  2. Use the ***Select*** method to bring up the user select dialog
     - The first entry defines the entity type to select, FT_NODE
     - The second entry, if True, will clear any previous values out of the Set
     - The third entry is the text used in the selection window title bar
  3. Loop through all entities of the Set
     - First, use the ***Reset*** method to make sure it starts at the beginning
     - Use a ***While*** loop with the ***Next*** method to step through all entities in the set
     - Any code between the ***While*** line and the ***Wend*** line will be executed for each ID in the set



```
Sub Main
    'Connect to the Femap Session
    Dim App As femap.model
    Set App = feFemap()

    'Step 1: Dimension the Set object
    Dim mySet As Set
    Set mySet = App.feSet

    'Step 2: User selects nodes for the set
    mySet.Select(FT_NODE,True,"Select Nodes")

    'Step 3: Loop through all selected nodes
    mySet.Reset() 'Makes sure to start loop from the beginning of the set
    While mySet.Next()
        'Code to perform for each entity goes here
    Wend

End Sub
```

Common API Tasks – If Statements & Writing to the Messages Window

- Example: If the user selects a Isotropic material, write the material ID and title to the messages window. If the user selects a Non-isotropic material, write "Non-isotropic material selected" to the messages window
  1. Dimension objects
     - A material object
     - A string object to hold your message
  2. Ask the user to select the material
  3. Use an ***If Statement*** and the ***type*** property to see if the material is isotropic
     - Always follow ***If*** ___ with ***Then***
     - Concatenate strings with the **+** operator
     - Use the ***CStr*** method to convert numerical values to text strings
  4. Write the message text to the Messages pane using the Application method ***feAppMessage***



```
Sub Main
    'Connect to the Femap Session
    Dim App As femap.model
    Set App = feFemap()

    'Step 1: Dimension the Material and a string object to hold your message
    Dim myMat As Matl
    Set myMat=App.feMatl
    Dim txt As String

    'Step 2: Ask User to Select Material
    myMat.SelectID("Select Material")

    'Step 3: Check if material is isotropic
    If myMat.type = FMT_ISOTROPIC Then
        'If statement is true so Material is isotropic
        txt = CStr(myMat.ID)+".."+myMat.title
    Else
        'If statement is false so Material is non-isotropic
        txt = "Non-isotropic material selected"
    End If

    'Step 4: Write message text to the Messages window
    App.feAppMessage(0,txt)

End Sub
```

Arrays and Variants

- Arrays are a set of values of a single data type
  - Arrays are set using parentheses to define the size
  - The array index starts at 0, so a 3 element array has a size of 2

```
Dim ndIDs(2) As Long
ndIDs(0) = 1
ndIDs(1) = 2
ndIDs(2) = 3
```

  - To set the size of an array to the value of another variable, such as nVals…

```
Dim nVals As Long
nVals = 3
```

    …first dimension the array as dynamic using empty parentheses

```
Dim myArray() As Double
```

    …then redimension the array using the variable

```
ReDim myArray(nVals) As Double
```

  - Arrays make your code more efficient (faster and easier to write/edit)

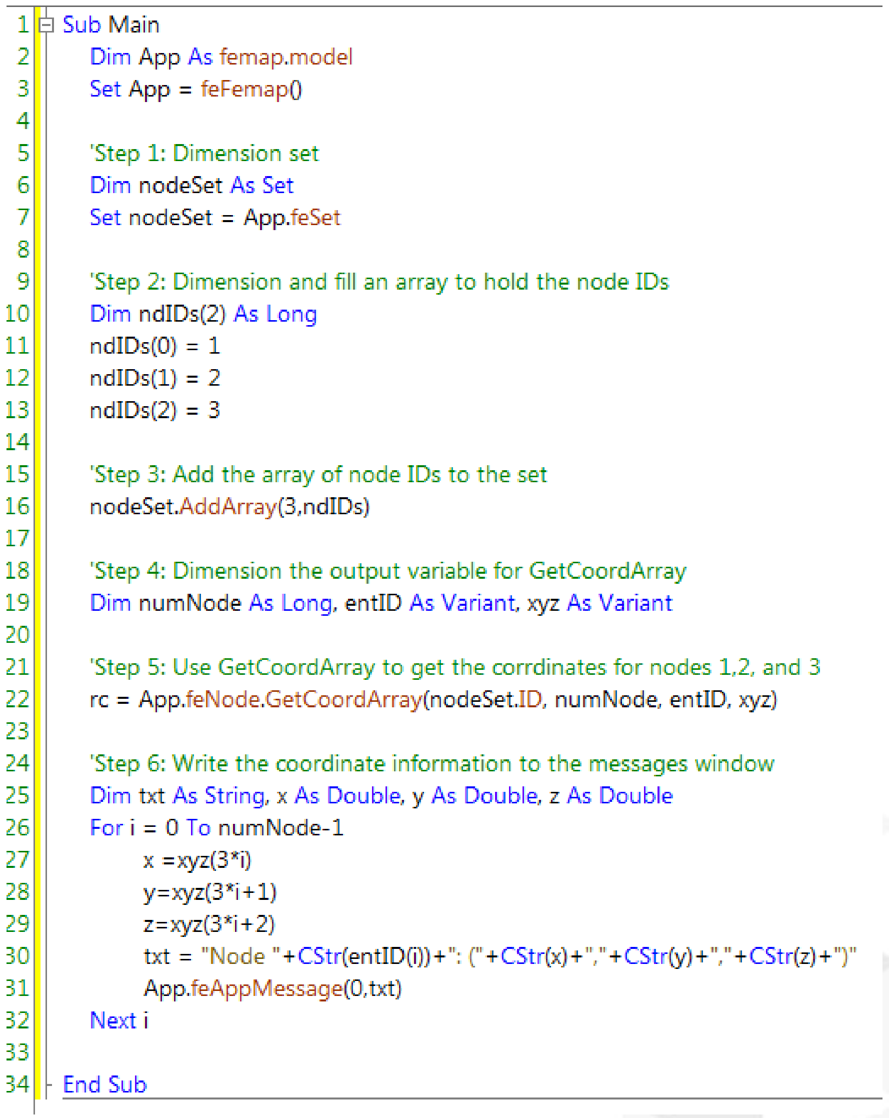
- Variants are a special data type that can hold any kind of data

```
Dim xyz As Variant
```

  - Variants are often used by the Femap API to output arrays

Common API Tasks – Use Arrays and Variants

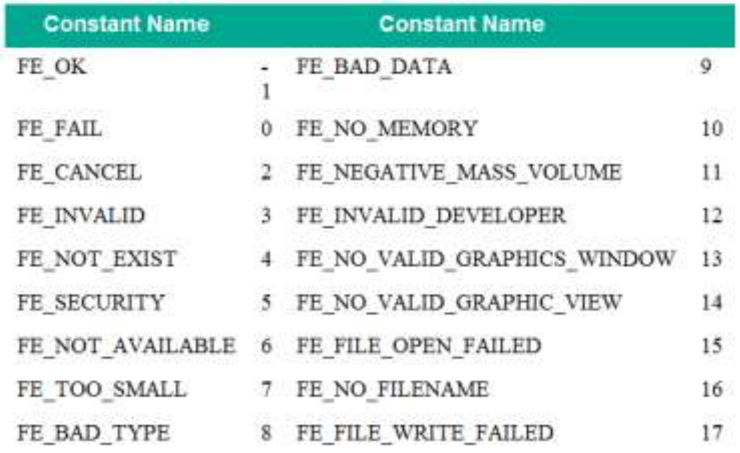
- Example: Write coordinate information for nodes 1, 2, and 3 using arrays and variants.
  1. Dimension a Set to hold the IDs
  2. Dimension and fill an array with the node IDs
  3. Add the ID array to the set using ***AddArray***
  4. Dimension the output variables for GetCoordArray
     - Use help resources to know what data types are necessary for each variable
  5. Get the coordinate data using ***GetCoordArray***
     - This is a node method, but we don't have a node object defined, so we'll use the global one ***App.feNode***
     - You don't need to store the Set ID separately, just call it as needed using the ***ID*** property
  6. Write the data to the Messages Pane
     - From the help file, we know that xyz will be a one dimensional array listing the x,y,and z coordinates of each node in entID



```
Sub Main
    Dim App As femap.model
    Set App = feFemap()

    'Step 1: Dimension set
    Dim nodeSet As Set
    Set nodeSet = App.feSet

    'Step 2: Dimension and fill an array to hold the node IDs
    Dim ndIDs(2) As Long
    ndIDs(0) = 1
    ndIDs(1) = 2
    ndIDs(2) = 3

    'Step 3: Add the array of node IDs to the set
    nodeSet.AddArray(3,ndIDs)

    'Step 4: Dimension the output variable for GetCoordArray
    Dim numNode As Long, entID As Variant, xyz As Variant

    'Step 5: Use GetCoordArray to get the corrdinates for nodes 1,2, and 3
    rc = App.feNode.GetCoordArray(nodeSet.ID, numNode, entID, xyz)

    'Step 6: Write the coordinate information to the messages window
    Dim txt As String, x As Double, y As Double, z As Double
    For i = 0 To numNode-1
            x =xyz(3*i)
            y=xyz(3*i+1)
            z=xyz(3*i+2)
            txt = "Node "+CStr(entID(i))+": ("+CStr(x)+","+CStr(y)+","+CStr(z)+")"
            App.feAppMessage(0,txt)
    Next i

End Sub
```

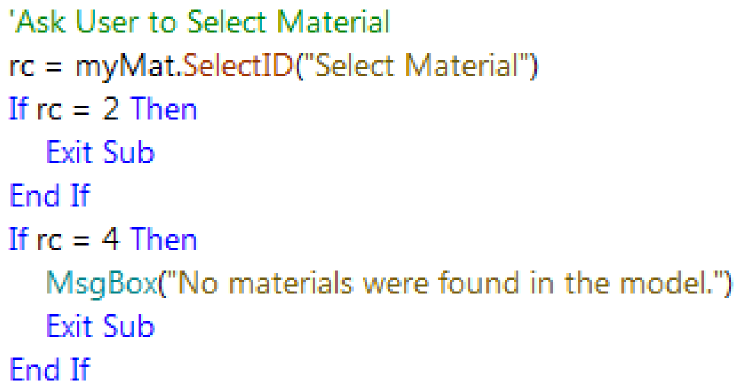
Return Codes

- Return codes are used to handle "errors" in the code
  - An "error" could be a user selecting cancel in a selection box, a method failing, etc
  - A return code = -1 indicates that the method was successful (i.e. no error)
- Return codes are Femap Constants (integers)
- To get the return code for a method, simply add "rc=" before the method execution



Constant Name		Constant Name	
FE_OK	-1	FE_BAD_DATA	9
FE_FAIL	0	FE_NO_MEMORY	10
FE_CANCEL	2	FE_NEGATIVE_MASS_VOLUME	11
FE_INVALID	3	FE_INVALID_DEVELOPER	12
FE_NOT_EXIST	4	FE_NO_VALID_GRAPHICS_WINDOW	13
FE_SECURITY	5	FE_NO_VALID_GRAPHIC_VIEW	14
FE_NOT_AVAILABLE	6	FE_FILE_OPEN_FAILED	15
FE_TOO_SMALL	7	FE_NO_FILENAME	16
FE_BAD_TYPE	8	FE_FILE_WRITE_FAILED	17

- Example: User Selection dialog box (SelectID)
- When using SelectID, three return codes are possible
  - -1 (FE_OK): No error → Continue API
  - 2 (FE_CANCEL): User selected cancel → Exit API immediately
  - 4 (FE_NOT_EXIST): No entities of the selected type exists so the dialog box could not be displayed → Pop-up message box with error message, then exit the API



```
'Ask User to Select Material
rc = myMat.SelectID("Select Material")
If rc = 2 Then
    Exit Sub
End If
If rc = 4 Then
    MsgBox("No materials were found in the model.")
    Exit Sub
End If
```

Common API Error Messages

API Error (Line 7): Type mismatch for parameter #2.

Error is in API code

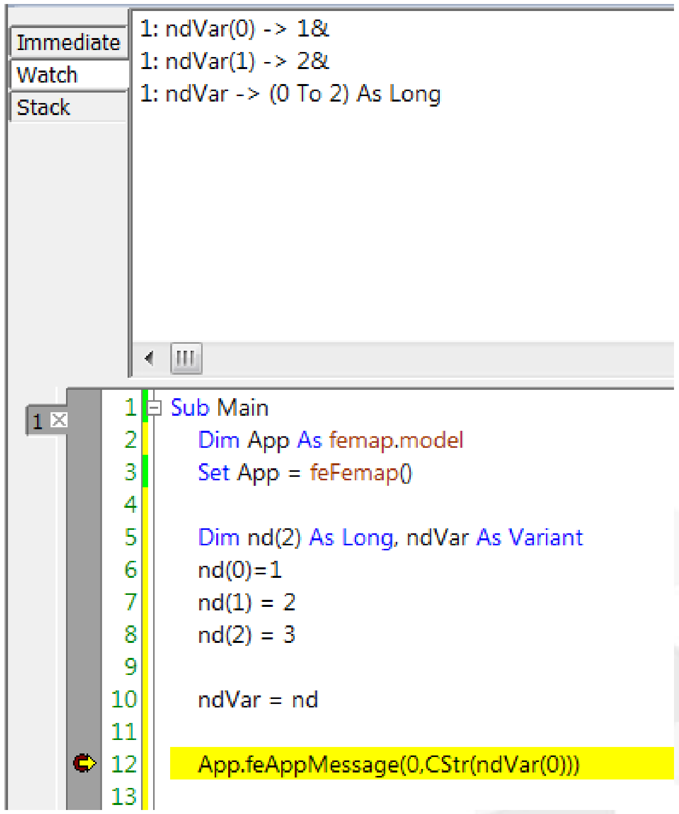
Line in which the code erred

Summary of the error

- "Type Mismatch…"
  - Either the variable was not dimensioned or was dimensioned to the wrong type
  - Check help resources to determine what type should be used
  - <u>Remember that if it's an array output, use *variant*</u>.
- "Not an object reference" or "Object var is Nothing"
  - You are referencing an object you have not defined
  - Define the object or double check your spelling
- "Parameter requires an expression. "
  - You're missing an input or output in the method definition
  - Check help resources to get a list of required inputs and outputs for the method
- "Expecting…"
  - You're missing a required character. This may be (), "", Then, etc
- "Unterminated block statement…"
  - You're missing the closing statement for a block (If, While, or For)
  - If it's missing or incorrect, add the appropriate statement (End If, Wend, or Next)
  - If you have the right statement, check for unclosed quotations around strings

Debugging Tools

- Breakpoints
  - Pause the code at marked lines so you can check what's going on
  - With the cursor in the line you want to stop at, click the breakpoint icon, , or F9, to toggle the breakpoint on and off.
- Step Into
  - Runs a single line at a time
  - Use the icon, , or F11
- Watch pane
  - When your code is paused (for a breakpoint or when using step into) you can use the watch pane to evaluate variables on the fly
    - Click on the Watch tab
    - Type a variable name into the pane and hit Enter



```
1: ndVar(0) -> 1&
1: ndVar(1) -> 2&
1: ndVar -> (0 To 2) As Long
```

```
Sub Main
    Dim App As femap.model
    Set App = feFemap()

    Dim nd(2) As Long, ndVar As Variant
    nd(0)=1
    nd(1) = 2
    nd(2) = 3

    ndVar = nd

    App.feAppMessage(0,CStr(ndVar(0)))

```

Coding Recommendations

- See if there's an existing Custom Tool that does something similar and you can simply edit to meet your needs
- Start by writing comments outlining the tasks that you want the code to do
- Then just focus on one task (or even subtask) at a time
  - Write test codes to get the hang of how a technique or method works before trying to put it in your API
- Once you have a small task done, test the code and debug. When it's working, save the code and move on to the next small task
- If you break it up like this, even a beginner can write complex APIs
- If you're trying to figure out an already written API, do the same thing. Look at each small block of code and write comments describing what it's doing. Then move on to the next block. Once you understand what each small block does, you'll be able to see the whole picture.
- Leave the comments in. They'll help you remember what you did ☺

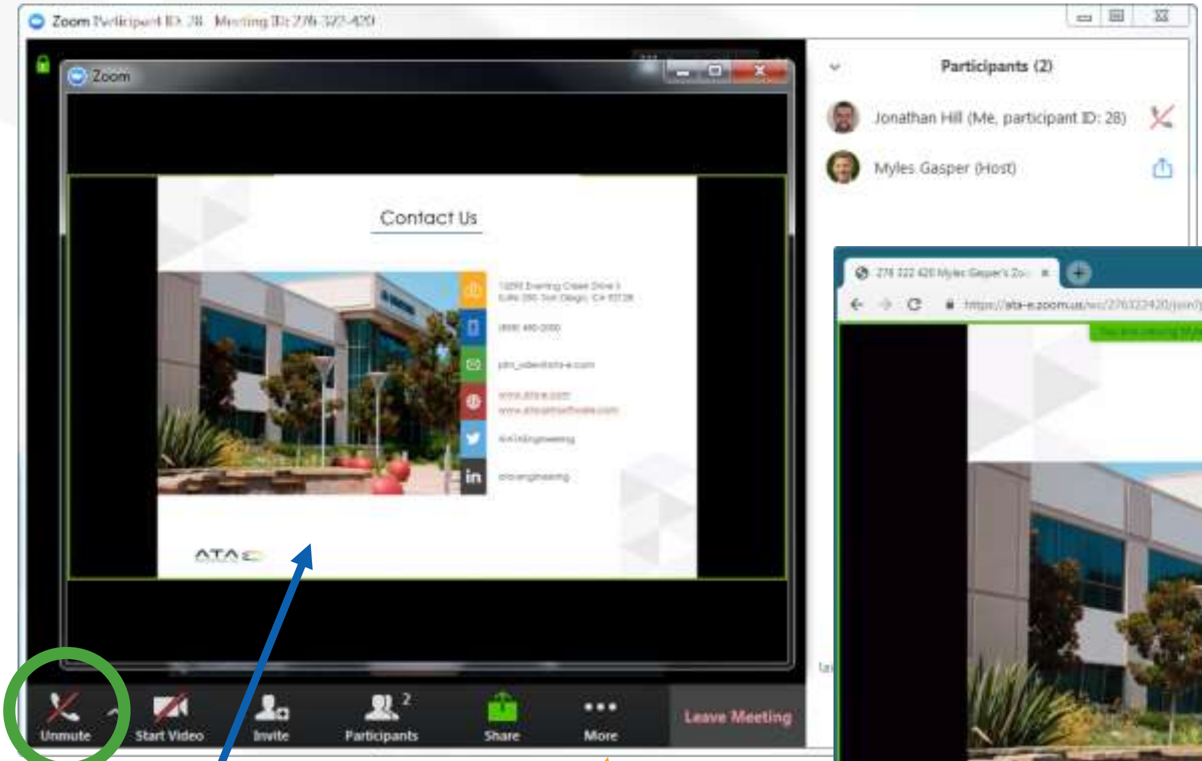
Comment every task!

Test Run after every task!

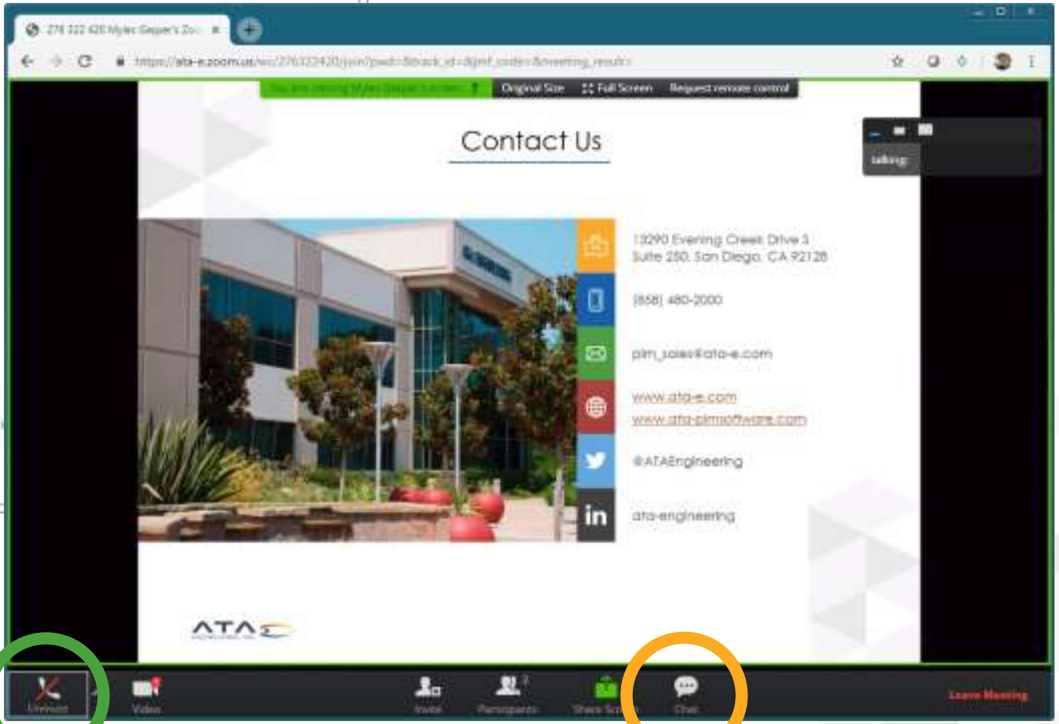
Questions?

Submit questions in the chat or unmute yourself now

Zoom Application



Web Interface



Screenshare in separate window

Chat is available under More

Contact Us

13290 Evening Creek Drive S
Suite 250, San Diego, CA 92128

(858) 480-2000

info@ata-e.com

www.ata-e.com
www.ata-plmsoftware.com

@ATAEngineering

ata-engineering