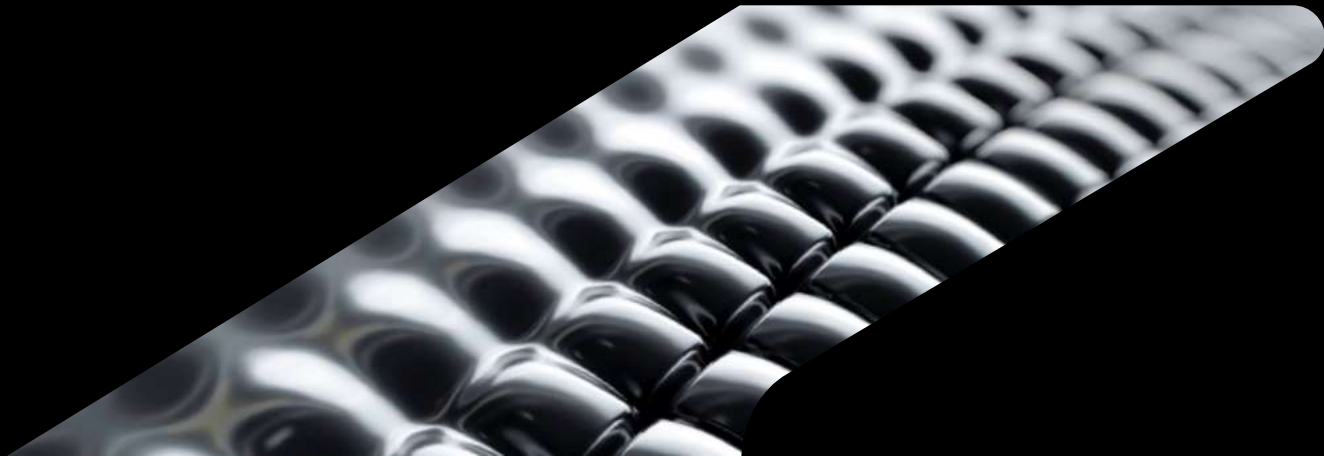




iLogic: 25 Tips and Tricks to Boost the Octane in Your Inventor Automation

MFG501293

Curtis Waguespack
Automation Solution Consultant



About Me

Curtis Waguespack
[wag] [əs] [pak]



- I've used Inventor and AutoCAD in the real-world for over 20 years.
- This has included a great deal of Inventor automation with iLogic.
- I started out modifying LISP routines and writing VBA for AutoCAD.
- I currently work as an Automation Solution Consultant, where I help professionals automate their designs.
- I teach iLogic classes, and in the past I have taught general Inventor and AutoCAD classes.
- Additionally, I've authored and co authored multiple editions of the **Mastering Autodesk Inventor** book.



About You

Intended Audience

- ✓ Some of these tips are immediately useful to new iLogic users
- ✓ Some of these tips are more relevant to experienced iLogic users
- ✓ I think all of these tips are worth knowing about, so that you can work them into your automation efforts as the needs arise, regardless of experience level



Questions and Comments

- ✓ Class format = 90 minutes:
 - 60 minutes of presentation
 - 30 minutes of Q & A
- ✓ Please hold questions and comments until the end
 - Note the number of the tip, for reference later



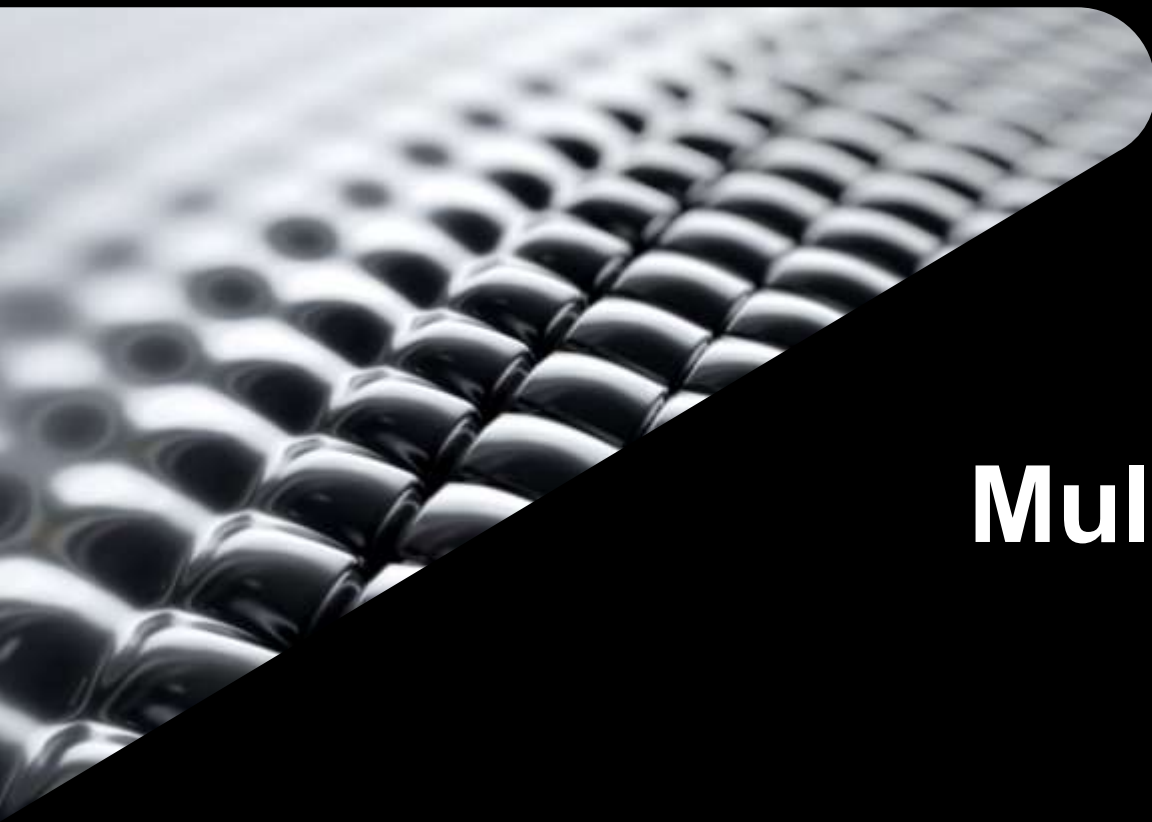
Objectives

1. Tips and techniques using the **iLogic editor**
2. Finding our way around the **Inventor API**
3. Explore development and **error handling** tools and techniques
4. Create a better **interface** between your iLogic automation and the users who employ it



Objective:

Tips and Techniques Using the iLogic Editor



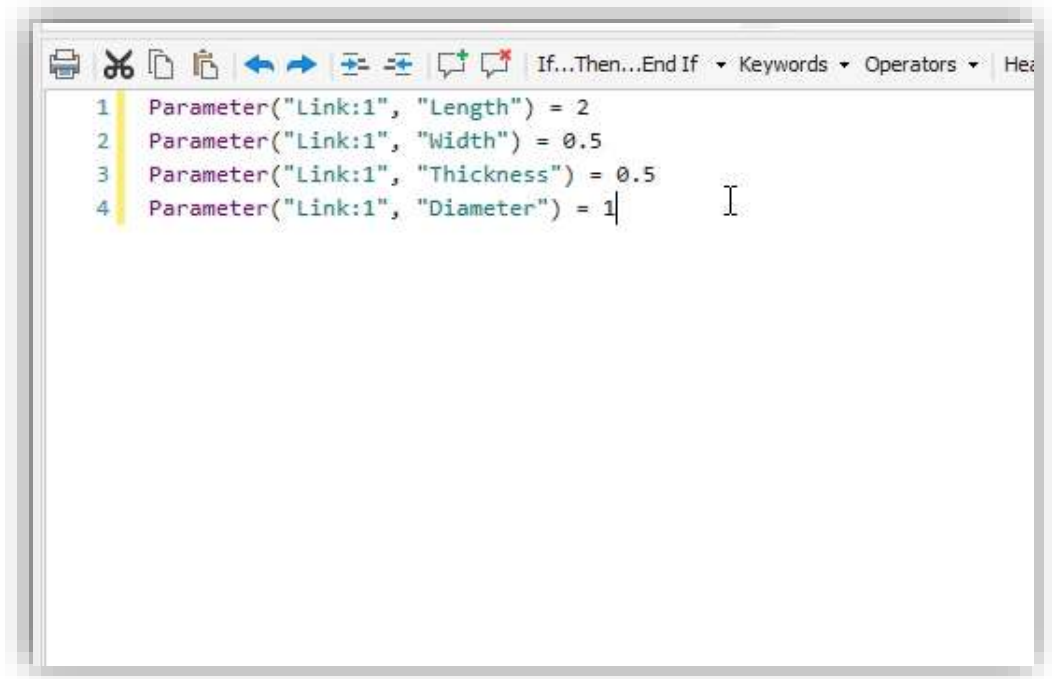
Multi-Line Typing

Tip #1

Multi-line editing

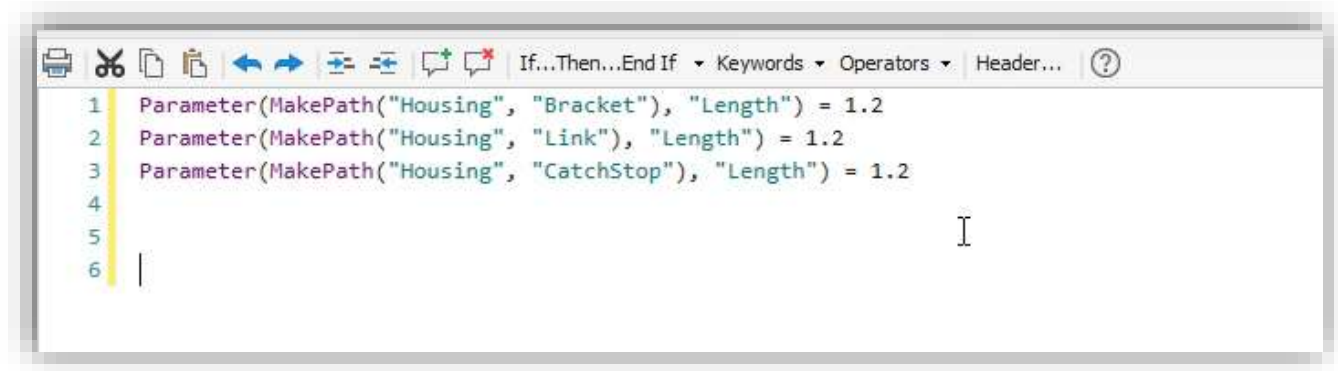
- When copying code or changing multiple lines you can enter text to multiple lines at once
- Place your cursor on the first line
- Then hold the **CTRL** key and then select multiple lines to create a multi-line cursor
- Type as you would normally

★ Tip provided by **Scott Hallmark**



Multi-location line editing

- We can do this in multiple places within the same line as well



The screenshot shows a code editor window with a toolbar at the top containing icons for print, cut, copy, paste, undo, redo, and comment. The toolbar also includes dropdown menus for 'If...Then...End If', 'Keywords', 'Operators', and 'Header...', along with a help icon. The code is written in a syntax-highlighted language and is as follows:

```
1 Parameter(MakePath("Housing", "Bracket"), "Length") = 1.2
2 Parameter(MakePath("Housing", "Link"), "Length") = 1.2
3 Parameter(MakePath("Housing", "CatchStop"), "Length") = 1.2
4
5
6 |
```

A vertical yellow line is positioned at the start of line 6. A cursor is located at the end of line 6, and a small 'I' icon is visible to the right of the cursor.

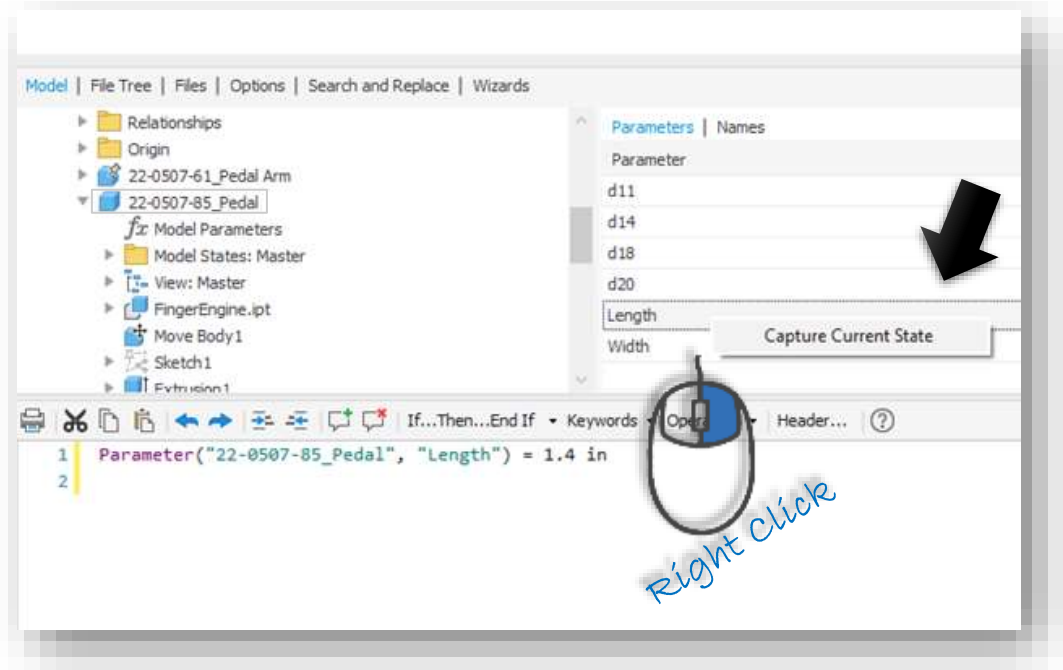


Extract to be exact!

Tip #2

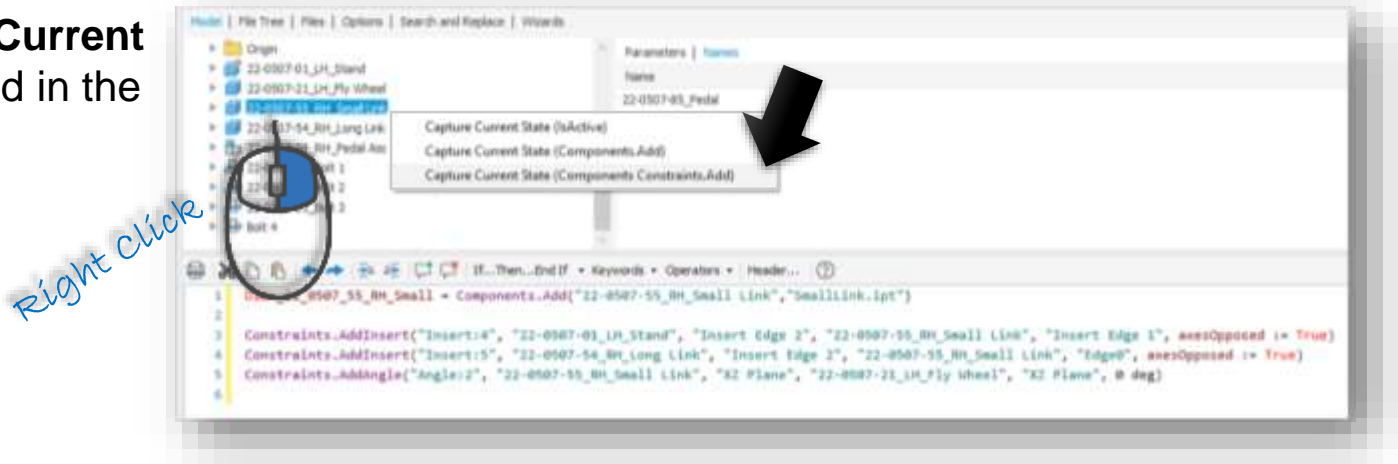
Extracting a Parameter from a sub-component

- The iLogic Editor allows us to create code by extracting it rather than typing it.
- Use this to your advantage to :
 - Gain efficiency
 - Reduce typo errors
 - Avoid omissions
 - Eliminate syntax issues



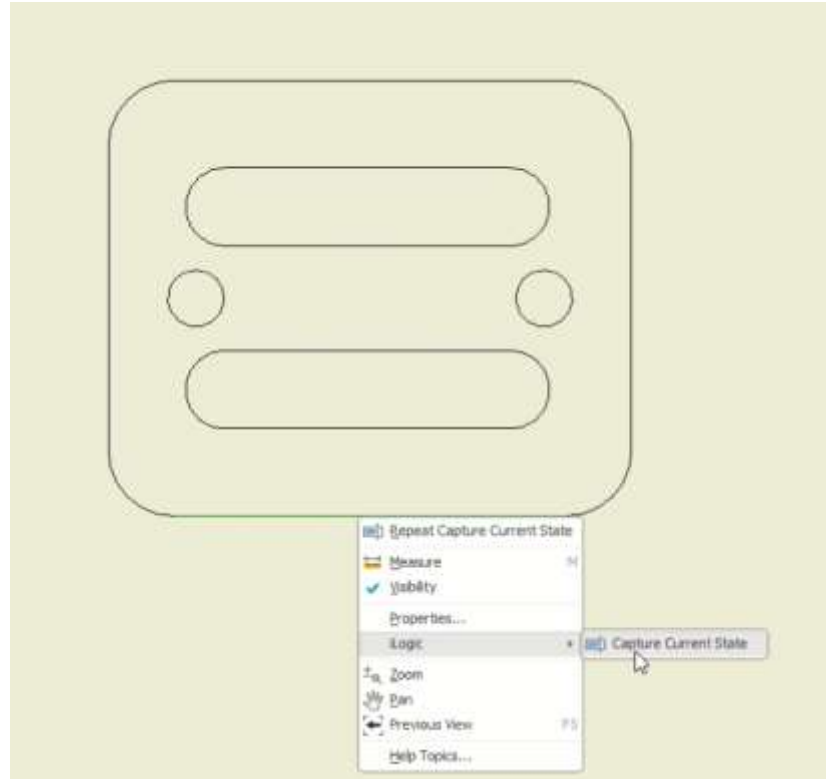
Extracting the code to place and constrain a sub-component

- There are several different **Capture Current State** options found in the right click menu



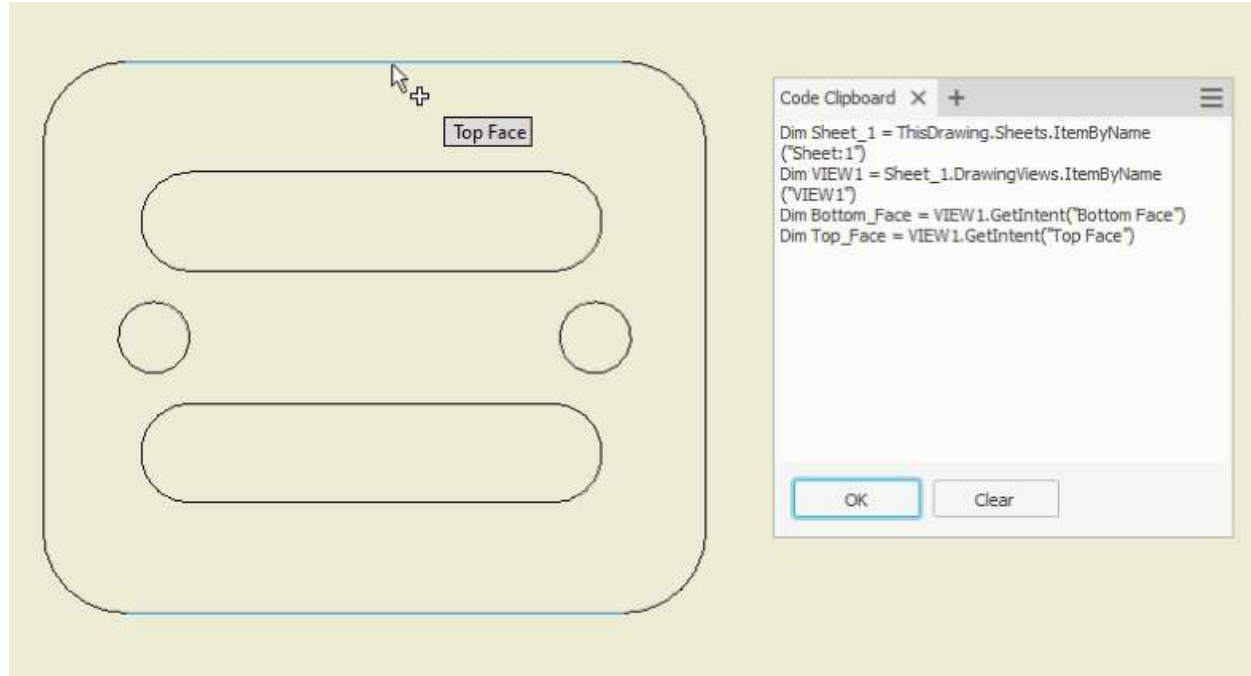
Extract code from drawing edges

- In a drawing you can right click an edge to extract the code needed to place a dimension



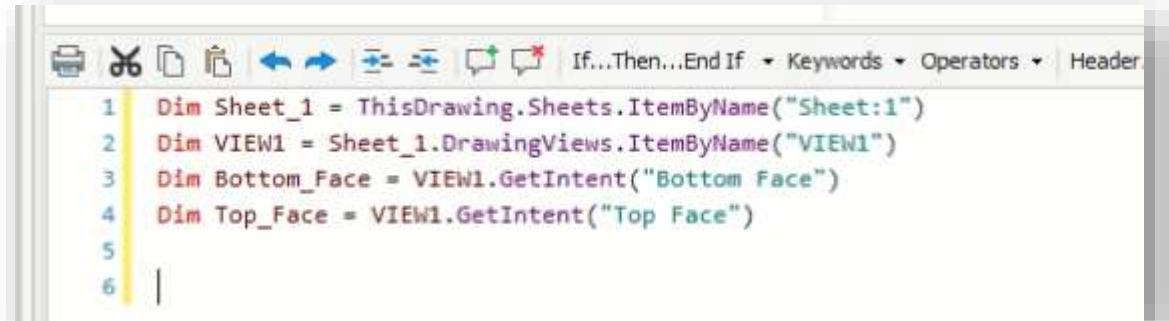
Extract code from drawing edges

- If there are named geometry entities in a model, we can extract them by right clicking on the drawing edge associated with them
- If the edge does not have a named entity, iLogic will create it in the model for us



Paste the extracted code into a rule

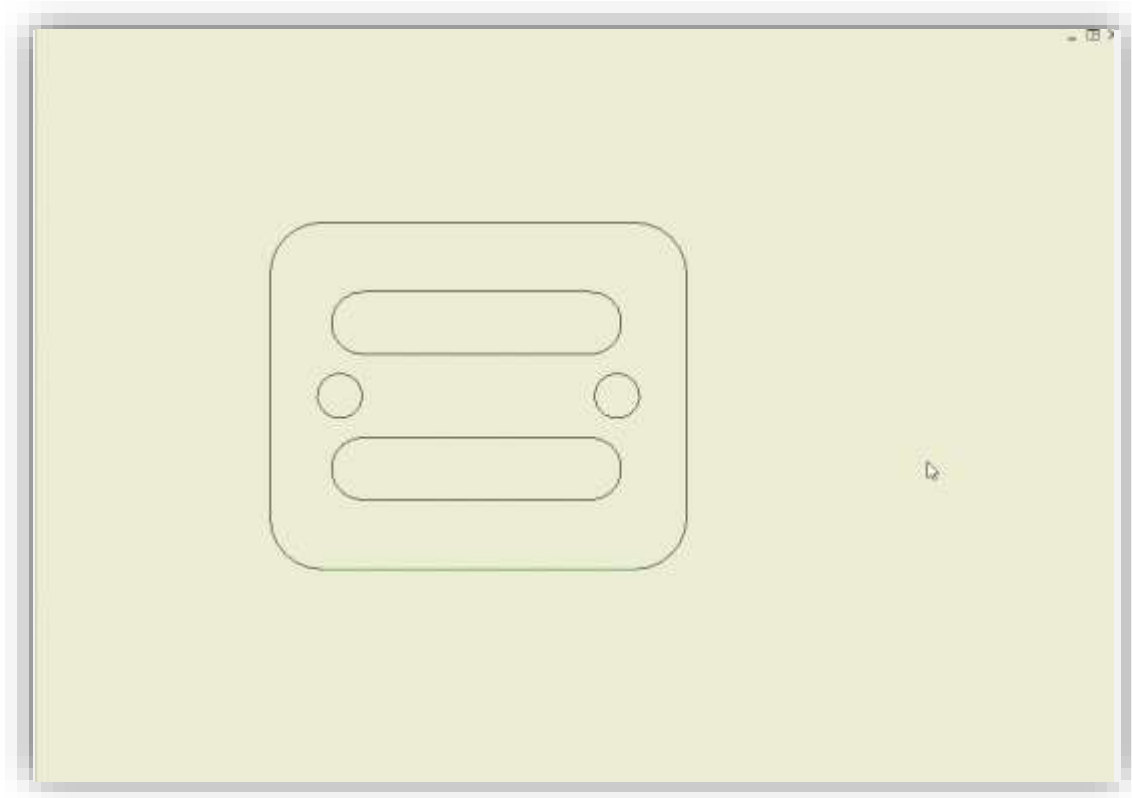
- In the iLogic Editor we can then paste in the code from the clipboard to our iLogic rule



The screenshot shows the iLogic Editor's code window. The toolbar at the top includes icons for print, cut, copy, paste, undo, redo, and comment. Below the toolbar, the code is as follows:

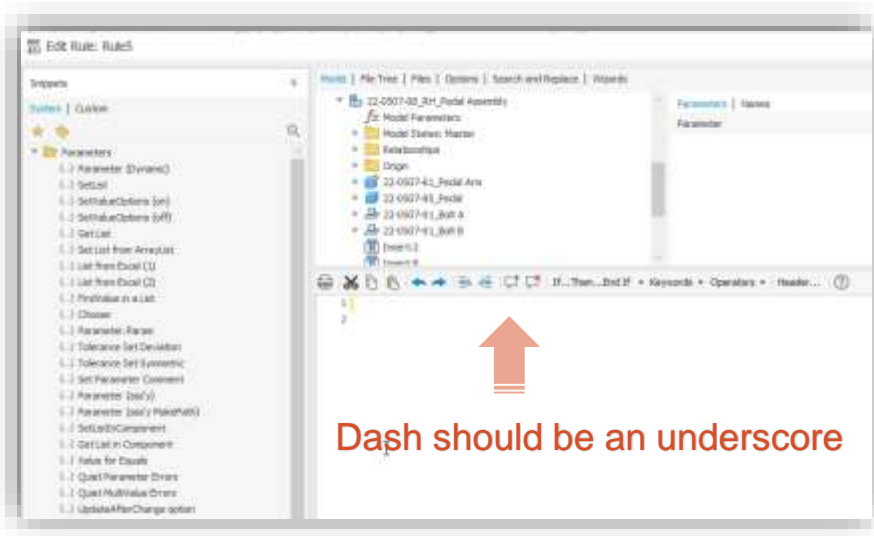
```
1 Dim Sheet_1 = ThisDrawing.Sheets.ItemByName("Sheet:1")
2 Dim VIEW1 = Sheet_1.DrawingViews.ItemByName("VIEW1")
3 Dim Bottom_Face = VIEW1.GetIntent("Bottom Face")
4 Dim Top_Face = VIEW1.GetIntent("Top Face")
5
6 |
```

Extracting code to place dimensions

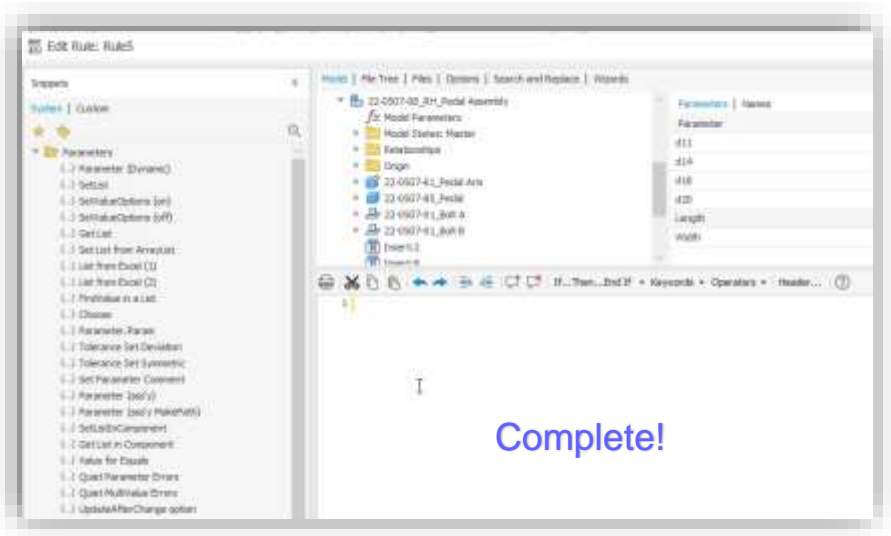


Extracting Comparison

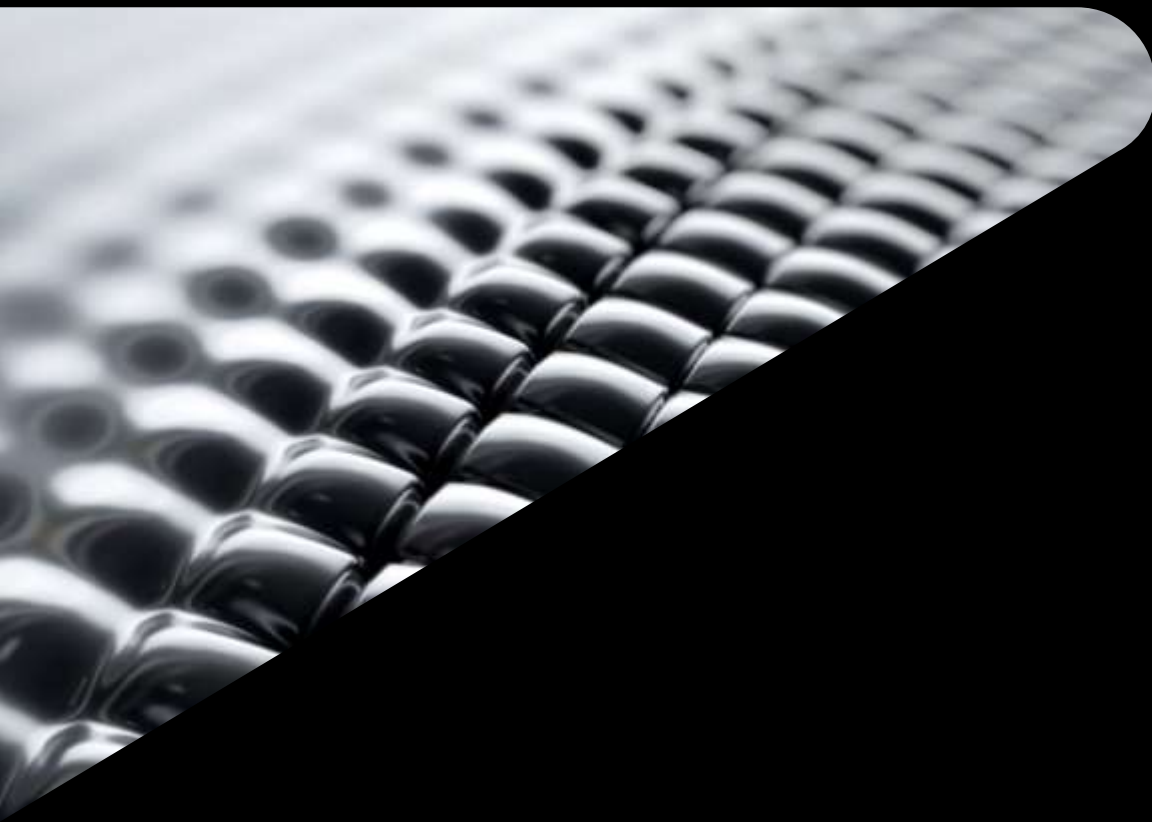
Typing vs extracting



Typing the code (with typo)



Extracting the code



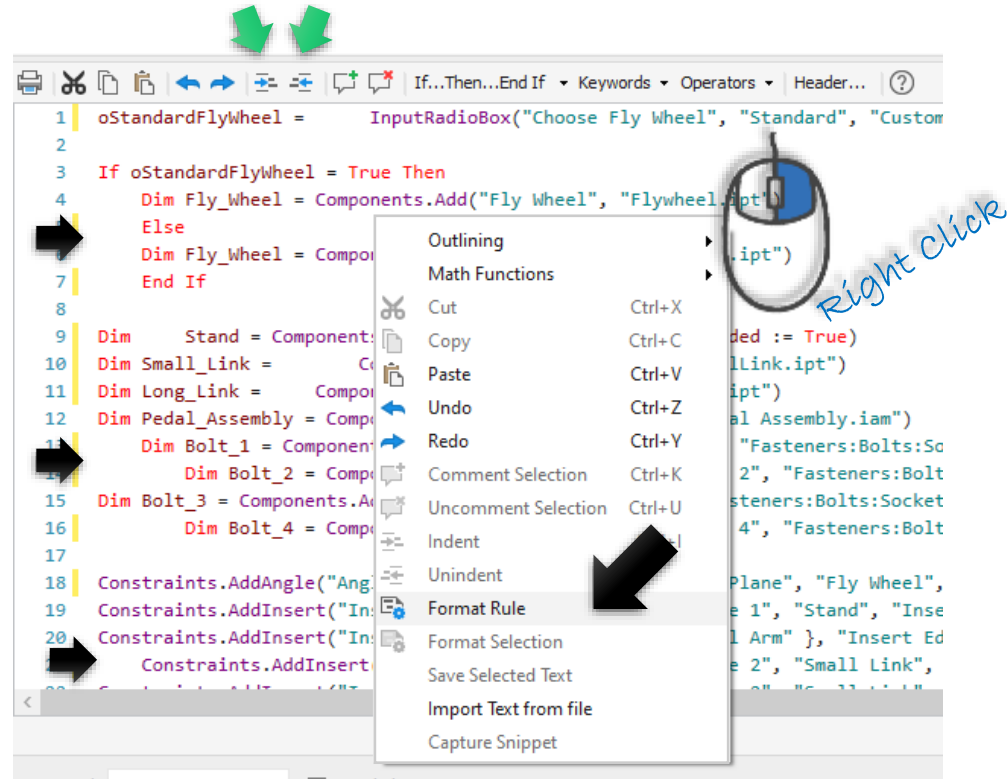
Right Click: Format Rule

Tip #3

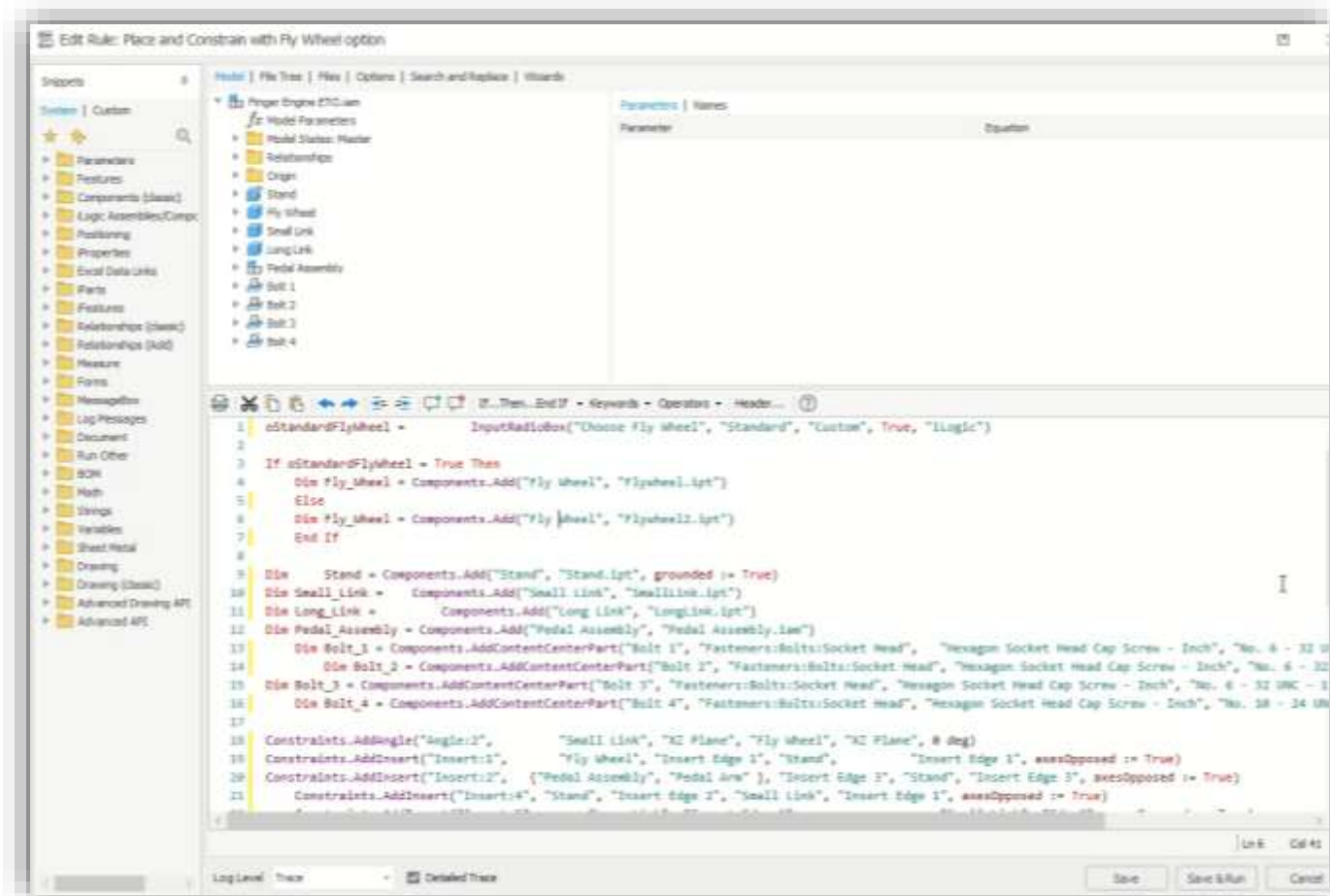
Format Rule

- Rather than manually fixing poorly formatted code, you can right click and choose Format Rule
- Remove extra “White Space”
- Fix indents
- Improve your code’s readability
- Save time

Indent buttons



Format Rule



Format Rule

- Be aware that there is a slight bug with the Format Rule tool, in how it handles these two else statements


- **Else If** (with space) formats unexpectedly
- **Elseif** (without space) formats as expected

- Both work, as far as the code running correctly, but the Format Rule tool “sees” them differently



```
8 If oSize < 3 Then
9   Parameter("Link Arm", "Length") = 7
10  Parameter("Link Arm", "Width") = 1
11  Else If oSize < 7 Then
12    Parameter("Link Arm", "Length") = 9
13    Parameter("Link Arm", "Width") = 3
14  ElseIf oSize < 9 Then
15    Parameter("Link Arm", "Length") = 11
16    Parameter("Link Arm", "Width") = 5
17  End If
```

```
8 If oSize < 3 Then
9   Parameter("Link Arm", "Length") = 7
10  Parameter("Link Arm", "Width") = 1
11  Else If oSize < 7 Then
12    Parameter("Link Arm", "Length") = 9
13    Parameter("Link Arm", "Width") = 3
14  ElseIf oSize < 9 Then
15    Parameter("Link Arm", "Length") = 11
16    Parameter("Link Arm", "Width") = 5
17  End If
```

A close-up, black and white photograph of a woven mesh or fabric texture, showing a grid of small, rounded, interlocking shapes. The texture is slightly out of focus in the background, creating a sense of depth. This image is partially obscured by a black diagonal overlay that contains the text.

Add Code Structure with Sub Procedures

Tip #4

What is a Sub Procedure ?

- A Sub Procedure is a collection of statements enclosed by the **Sub** and **End Sub** statements
- The Sub procedure performs a task and then returns control to the code from which it was called
- When using a Sub, we must include a **Sub Main**
- But it does not return a value to the calling code

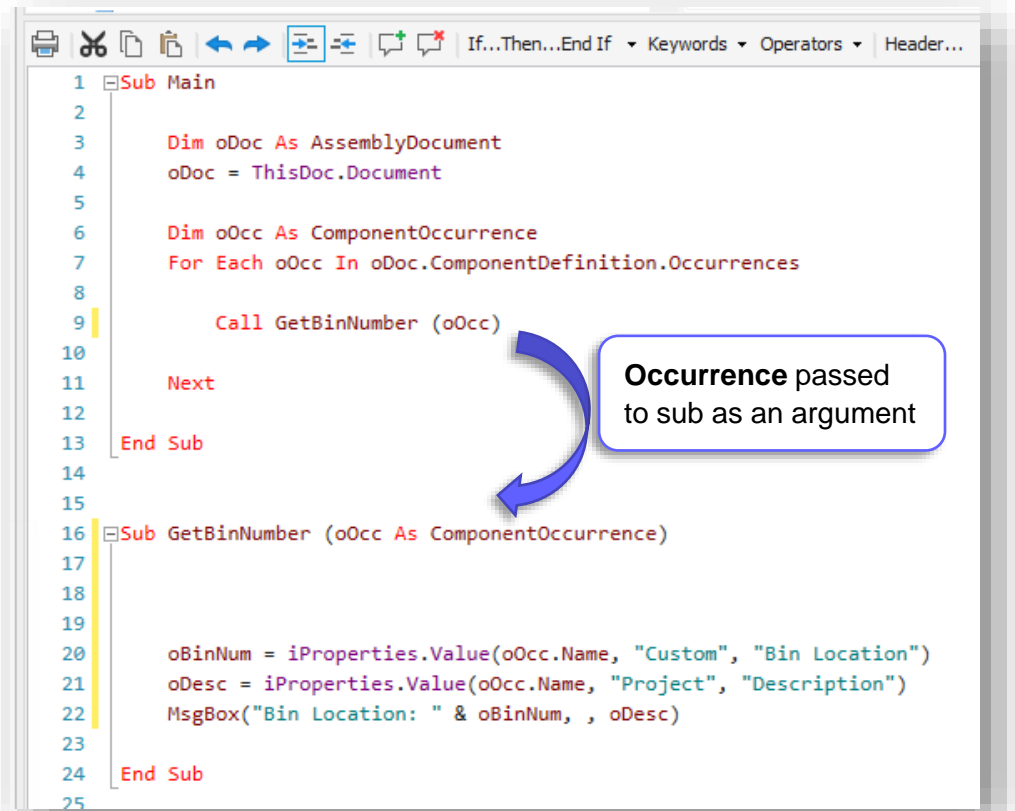


```
1 Sub Main
2
3     Call SayHello
4
5 End Sub
6
7 Sub SayHello
8
9     MessageBox.Show("Hello World!", "iLogic")
10
11 End Sub
```

The diagram illustrates the execution flow of a Sub Procedure. A blue curved arrow originates from the 'Call SayHello' statement on line 3 of the 'Sub Main' block and points to the 'Sub SayHello' block starting at line 7. Another blue curved arrow originates from the end of the 'Sub SayHello' block (line 11) and points back to the 'End Sub' statement on line 5 of the 'Sub Main' block, indicating that control returns to the calling code after the sub procedure completes its execution.

Passing information to the Sub

- Often, we pass information to a Sub procedure in the form of arguments
- The sub requires the argument's data type to be declared
 - technically the argument is called a parameter at the sub level
- Syntax for calling a Sub procedure:
 - Call SubName (Argument1, Argument2)
- The use of the **Call** keyword is optional

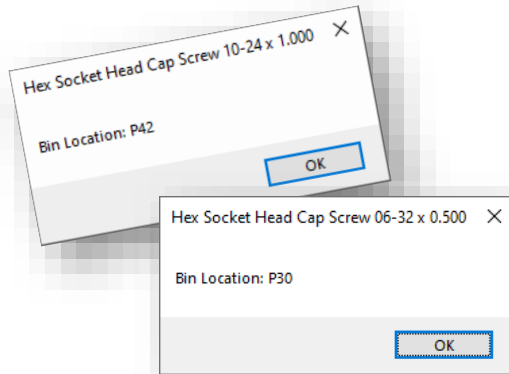


```
1 Sub Main
2
3     Dim oDoc As AssemblyDocument
4     oDoc = ThisDoc.Document
5
6     Dim oOcc As ComponentOccurrence
7     For Each oOcc In oDoc.ComponentDefinition.Occurrences
8
9         Call GetBinNumber (oOcc)
10
11     Next
12
13 End Sub
14
15
16 Sub GetBinNumber (oOcc As ComponentOccurrence)
17
18
19
20     oBinNum = iProperties.Value(oOcc.Name, "Custom", "Bin Location")
21     oDesc = iProperties.Value(oOcc.Name, "Project", "Description")
22     MsgBox("Bin Location: " & oBinNum, , oDesc)
23
24 End Sub
25
```

Occurrence passed to sub as an argument

Diagramming a Simple Sub Procedure

- Sub Name
- Argument/Parameter
- Sub Statement Code

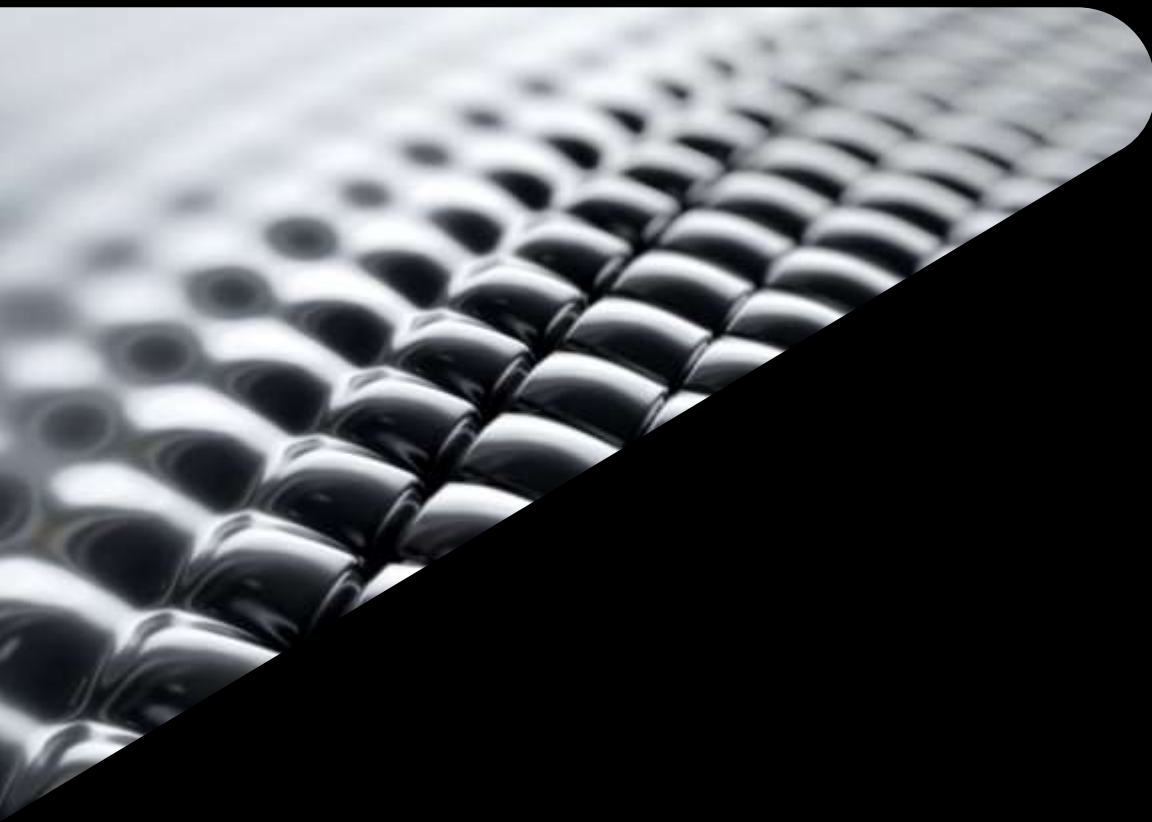


```
1 Sub Main
2
3   Dim oDoc As AssemblyDocument
4   oDoc = ThisDoc.Document
5
6   Dim oOcc As ComponentOccurrence
7   For Each oOcc In oDoc.ComponentDefinition.Occurrences
8
9       Call GetBinNumber(oOcc)
10
11   Next
12
13 End Sub
14
15 Sub GetBinNumber(oOcc As ComponentOccurrence)
16
17
18
19
20   oBinNum = iProperties.Value(oOcc.Name, "Custom", "Bin Location")
21   oDesc = iProperties.Value(oOcc.Name, "Project", "Description")
22   MsgBox("Bin Location: " & oBinNum, , oDesc)
23
24 End Sub
25
```

Why use Sub Procedures ?

- Subdividing large amount of code into smaller subs make your code more:
 - Readable
 - Maintainable
- Procedures are useful for performing repeated or shared tasks, such as:
 - Calculations
 - Formatting
 - Data operations
- You can use procedures as building blocks for your iLogic rules.
- Allows us to reuse, rather than copy/paste the same code over and over

```
1 Sub Main
2
3     Dim oDrawDoc = ThisDrawing.Document
4     Dim oModel = ThisDrawing.ModelDocument
5
6     Call ProcessAssembly(oModel)
7     Call Publish_PDF(oDrawDoc)
8     Call DXF_Out(oDrawDoc)
9     Call SetApprovalStamp(oDrawDoc)
10    Call ProcessAssembly(oModel)
11 End Sub
12
13 Sub ProcessAssembly(oMDoc As AssemblyDocument)
14
15
16
17 Sub Publish_PDF(oDraw As DrawingDocument)
18
19
20
21 Sub DXF_Out(oDraw As DrawingDocument)
22
23
24
25
26 Sub SetApprovalStamp(oDraw As DrawingDocument)
```



Use Function Procedures

Tip #5

What is a Function Procedure ?

- A Function procedure is a collection of statements enclosed by the **Function** and **End Function** statements
- Like a Sub, a Function performs a task and then returns control to the code from which it was called
- Unlike a Sub, a Function returns a value to the calling code
- When using a Function, we must include a **Sub Main**

```
1 Sub Main
2
3     oThickness = Parameter("Thickness")
4     oHoleSize = Parameter("Diameter")
5
6     oOffset = CalculateOffset(oThickness, oHoleSize)
7     MessageBox.Show("Offset: " & oOffset, "iLogic")
8
9 End Sub
10
11 Function CalculateOffset (oThick As Double , oHole As Double)
12
13     If oThick < 3 Then
14         oShimSize = 0.32
15     ElseIf oThick < 5 Then
16         oShimSize = 0.41
17     Else
18         oShimSize = 0.533
19     End If
20
21     oOffset = oThick + oHole - oShimSize*2
22     Return oOffset
23 End Function
```

Passing information to the Function

- We pass information into a Function the same as we do a Sub
- The Function requires the argument's data type to be declared
 - technically the argument is called a parameter at the function level
- Syntax for calling a Function procedure:
 - Capturing Variable = FunctionName (Argument1, Argument2)

```
1 Sub Main
2
3     oThickness = Parameter("Thickness")
4     oHoleSize = Parameter("Diameter")
5
6     oOffset = CalculateOffset(oThickness, oHoleSize)
7     MessageBox.Show("Offset: " & oOffset, "iLogic")
8
9 End Sub
10
11 Function CalculateOffset (oThick As Double , oHole As Double)
12
13     If oThick < 3 Then
14         oShimSize = 0.32
15     ElseIf oThick < 5 Then
16         oShimSize = 0.41
17     Else
18         oShimSize = 0.533
19     End If
20
21     oOffset = oThick + oHole - oShimSize*2
22     Return oOffset
23 End Function
```

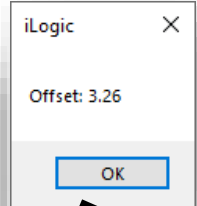
Parameter values
passed to function as
arguments

Calculated value
passed back from the
function to the calling
code

Diagramming a Simple Function Procedure

- Function Name
- Arguments/Parameters
- Function Statement Code
- Return Statement

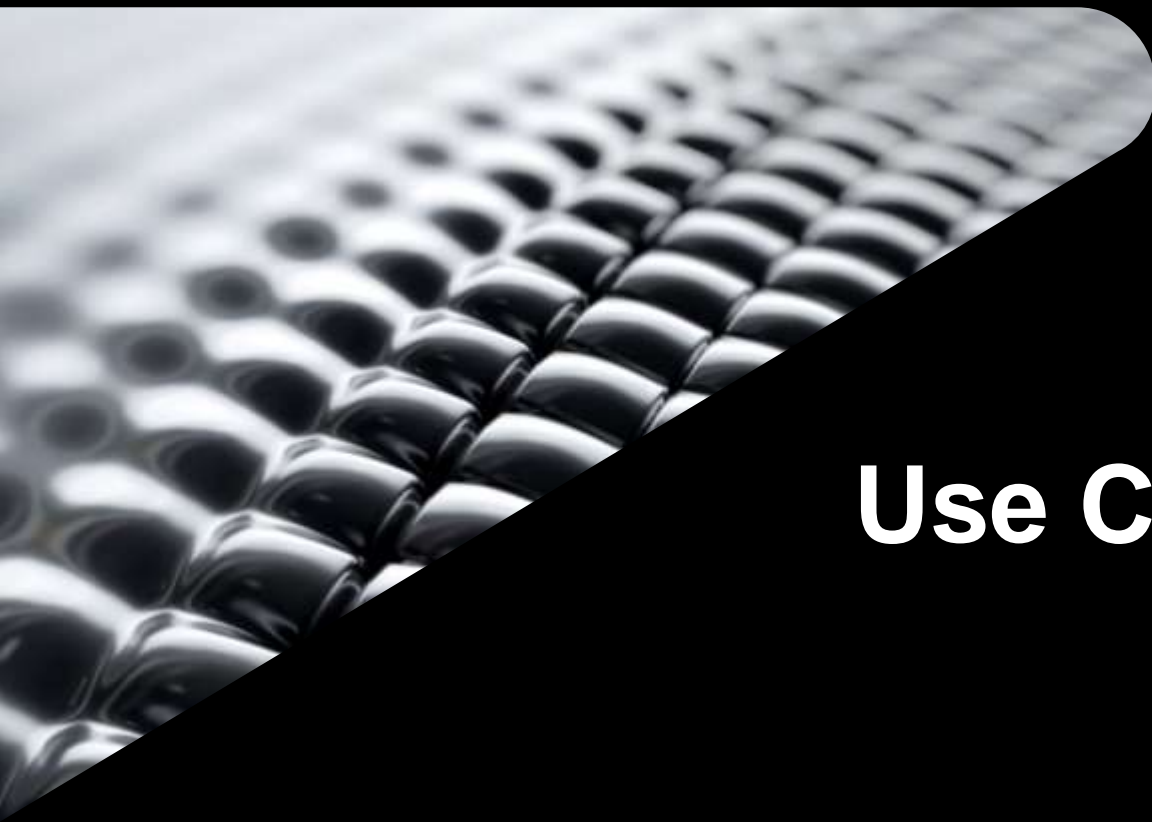
```
1 Sub Main
2
3     oThickness = Parameter("Thickness")
4     oHoleSize = Parameter("Diameter")
5
6     oOffset = CalculateOffset(oThickness, oHoleSize)
7     MessageBox.Show("Offset: " & oOffset, "iLogic")
8
9
10 End Sub
11
12 Function CalculateOffset(oThick As Double, oHole As Double)
13
14     If oThick < 3 Then
15         oShimSize = 0.32
16     ElseIf oThick < 5 Then
17         oShimSize = 0.41
18     Else
19         oShimSize = 0.533
20     End If
21
22     oOffset = oThick + oHole - oShimSize * 2
23
24     Return oOffset
25 End Function
```



Why use Function Procedures ?

- Generally, we use functions to do some task or calculation, and return one or more values to the calling code.
- Often a Function is called multiple times within the same code but is passed different values each time.
- Allows us to reuse, rather than copy/paste the same code over and over

```
1 Sub Main
2
3     oThickness = Parameter("Thickness")
4     oHoleSize = Parameter("Diameter")
5
6     oOffset = CalculateOffset(oThickness, oHoleSize)
7
8     MessageBox.Show("Offset: " & oOffset, "iLogic")
9
10 End Sub
11
12 Function CalculateOffset(oThick As Double, oHole As Double)
13
14     If oThick < 3 Then
15         oShimSize = 0.32
16     ElseIf oThick < 5 Then
17         oShimSize = 0.41
18     Else
19         oShimSize = 0.533
20     End If
21
22     oOffset = oThick + oHole - oShimSize * 2
23
24     Return oOffset
25 End Function
```



Use Class ThisRule

Tip #6

What is an iLogic rule Class ?

- If you add a **Sub Main** procedure to your iLogic rule, the rule immediately takes the form of a standard VB.net class.
- A class is just a blueprint for a data type.
- In our case the data type is an iLogic Rule.

```
Sub Main
    Dim oDoc As AssemblyDocument
    oDoc = ThisDoc.Document

    Dim oOcc As ComponentOccurrence
    For Each oOcc In oDoc.ComponentDefinition.Occurrences

        oBinNum = iProperties.Value(oOcc.Name, "Custom", "Bin Location")
        oDesc = iProperties.Value(oOcc.Name, "Project", "Description")
        MsgBox("Bin Location: " & oBinNum, , oDesc)

    Next
End Sub
```

What is an iLogic rule Class ?

- By Default, the **Class** and **End Class** statements are not visible.
- We can add them as needed

```
Class ThisRule

Sub Main

    Dim oDoc As AssemblyDocument
    oDoc = ThisDoc.Document

    Dim oOcc As ComponentOccurrence
    For Each oOcc In oDoc.ComponentDefinition.Occurrences

        oBinNum = iProperties.Value(oOcc.Name, "Custom", "Bin Location")
        oDesc = iProperties.Value(oOcc.Name, "Project", "Description")
        MsgBox("Bin Location: " & oBinNum, , oDesc)

    Next

End Sub

End Class
```

What is an iLogic rule Class ?

- One of the most common uses of this within an iLogic rule, is to allow us to use a **shared** variable to pass information between procedures without the need to do so with arguments.

```
Class ThisRule

Sub Main

    Dim oDoc As AssemblyDocument
    oDoc = ThisDoc.Document

    Dim oOcc As ComponentOccurrence
    For Each oOcc In oDoc.ComponentDefinition.Occurrences

        oBinNum = iProperties.Value(oOcc.Name, "Custom", "Bin Location")
        oDesc = iProperties.Value(oOcc.Name, "Project", "Description")
        MsgBox("Bin Location: " & oBinNum, , oDesc)

    Next

End Sub

End Class
```

Using “Class ThisRule”

```
1 Class ThisRule
2
3   Shared oOcc As ComponentOccurrence
4   Shared oBinNum As String
5
6   Sub Main
7
8       Dim oDoc As AssemblyDocument
9       oDoc = ThisDoc.Document
10
11       For Each oOcc In oDoc.ComponentDefinition.Occurrences
12
13           'call the sub routine to get bin number
14           Call GetBinNumber(oOcc)
15
16           oDesc = iProperties.Value(oOcc.Name, "Project", "Description")
17
18           MsgBox("Bin Location: " & oBinNum, , oDesc)
19       Next
20   End Sub
21
22   Sub GetBinNumber(oOcc As ComponentOccurrence)
23
24       oBinNum = iProperties.Value(oOcc.Name, "Custom", "Bin Location")
25
26   End Sub
27
28 End Class
29
```

Value is held at the rule level

Occurrence is held at the rule level

```
Sub Main

    Dim oDoc As AssemblyDocument
    oDoc = ThisDoc.Document

    Dim oOcc As ComponentOccurrence
    For Each oOcc In oDoc.ComponentDefinition.Occurrences

        'call the sub routine to get bin number
        Call GetBinNumber(oOcc)

    Next

End Sub

Sub GetBinNumber(oOcc As ComponentOccurrence)

    oBinNum = iProperties.Value(oOcc.Name, "Custom", "Bin Location")
    oDesc = iProperties.Value(oOcc.Name, "Project", "Description")
    MsgBox("Bin Location: " & oBinNum, , oDesc)

End Sub
```

Occurrence passed to sub as an argument, and the iProperty value is held in the sub

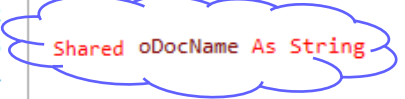
Example

- Here a variable called oDocName is captured at the beginning of the Main sub
- And then used over and over by the Sub and Function procedures

Be Aware:

- Shared variable values at the class level persist in rule for until the file is closed

```
1  Class ThisRule
2
3  Shared oDocName As String
4
5  Sub Main
6
7      oDocName = ThisDoc.FileName
8
9      Call GetPartNumber
10     Call GetRelatedComps
11     Call CalcOffset
12
13     Call GetPartNumber
14     Call GetRelatedComps
15     Call CalcOffset
16
17     Call GetPartNumber
18     Call GetRelatedComps
19     Call CalcOffset
20
21 End Sub
22
23 Sub GetPartNumber...
26
27 Sub GetRelatedComps...
30
31 Function CalcOffset...
34
35 End Class
```

A blue oval highlights the 'Shared oDocName As String' declaration on line 3. Three blue arrows originate from this oval and point to the 'Call' statements in the 'Main' sub (lines 9, 13, 17) and the three sub-procedures (lines 23, 27, 31), illustrating that all these procedures share the same oDocName variable.

Note that you don't need to call it ThisRule

- We typically just call the class ThisRule
- But it can be named anything you like
- Both examples work the same:



```
1 Class ThisRule
2
3     Dim oOcc As ComponentOccurrence
4     Dim oBinNum As String
5
6 Sub Main
7
8     Dim oDoc As AssemblyDocument
9     oDoc = ThisDoc.Document
10
```



```
1 Class Carl_The_iLogic_Rule
2
3     Dim oOcc As ComponentOccurrence
4     Dim oBinNum As String
5
6 Sub Main
7
8     Dim oDoc As AssemblyDocument
9     oDoc = ThisDoc.Document
10
```



Create collapsible code groups

Tip #7

Create collapsible code groups

- The syntax is simply, apostrophe & an opening square bracket

' ['

- Followed by, apostrophe & a closing square bracket:

']

```
7  [- '[ Get iProps
8      oPN = iProperties.Value(oOcc.Name, "Proj
9      oRev = iProperties.Value(oOcc.Name, "Pro
10     oDesc = iProperties.Value(oOcc.Name, "Pr
11     oTitle = iProperties.Value(oOcc.Name, "S
12     oBinNum = iProperties.Value(oOcc.Name, "
13     ']
```


The same rule with outlining collapsed

```
1
2 Dim oDoc As AssemblyDocument
3 oDoc = ThisDoc.Document
4
5 For Each oOcc In oDoc.ComponentDefinition.Occurrences
6
7     '[ Get iProps
8     oPN = iProperties.Value(oOcc.Name, "Project", "Part Number")
9     oRev = iProperties.Value(oOcc.Name, "Project", "Revision Number")
10    oDesc = iProperties.Value(oOcc.Name, "Project", "Description")
11    oTitle = iProperties.Value(oOcc.Name, "Summary", "Title")
12    oBinNum = iProperties.Value(oOcc.Name, "Custom", "Bin Location")
13    ' ]
14
15    '[ Write to logger
16    Logger.Info("Part Number: " & oPN)
17    Logger.Info("Revision Number: " & oRev)
18    Logger.Info("Description: " & oDesc)
19    Logger.Info("oTitle: " & oTitle)
20    Logger.Info("Bin Location: " & oBinNum)
21    Logger.Info("")
22    ' ]
23 Next
```

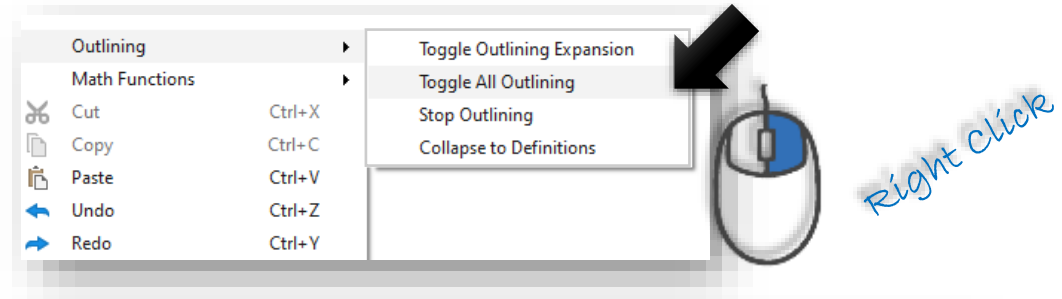
```
1
2 Dim oDoc As AssemblyDocument
3 oDoc = ThisDoc.Document
4
5 For Each oOcc In oDoc.ComponentDefinition.Occurrences
6
7     + [ Get iProps
14
15     + [ Write to logger
23 Next
```

You can collapse or expand the groups individually by clicking the

+ or -

Use the Right Click Outlining Options

- These code groups can be collapsed using the right click Outlining options



```
1
2 Dim oDoc As AssemblyDocument
3 oDoc = ThisDoc.Document
4
5 For Each oOcc In oDoc.ComponentDefinition.Occurrences
6
7   (+) Get iProps
14
15   (+) Write to logger
23 Next
24
```

Note the difference in outlining using Sub Main()

Sub Main ()



```
1 Sub Main()  
2  
3 Dim oDoc As AssemblyDocument  
4 oDoc = ThisDoc.Document  
5  
6 For Each oOcc In oDoc.ComponentDefinition.Occurrences  
7  
8 + Get iProps  
15  
16 + Write to logger  
24 Next  
25  
26 End Sub
```

With parenthesis, no outlining for
Main sub

Sub Main

```
1 Sub Main  
2  
3 Dim oDoc As AssemblyDocument  
4 oDoc = ThisDoc.Document  
5  
6 For Each oOcc In oDoc.ComponentDefinition.Occurrences  
7  
8 + Get iProps  
15  
16 + Write to logger  
24 Next  
25  
26 End Sub
```

Without parenthesis, outlining included for
Main sub



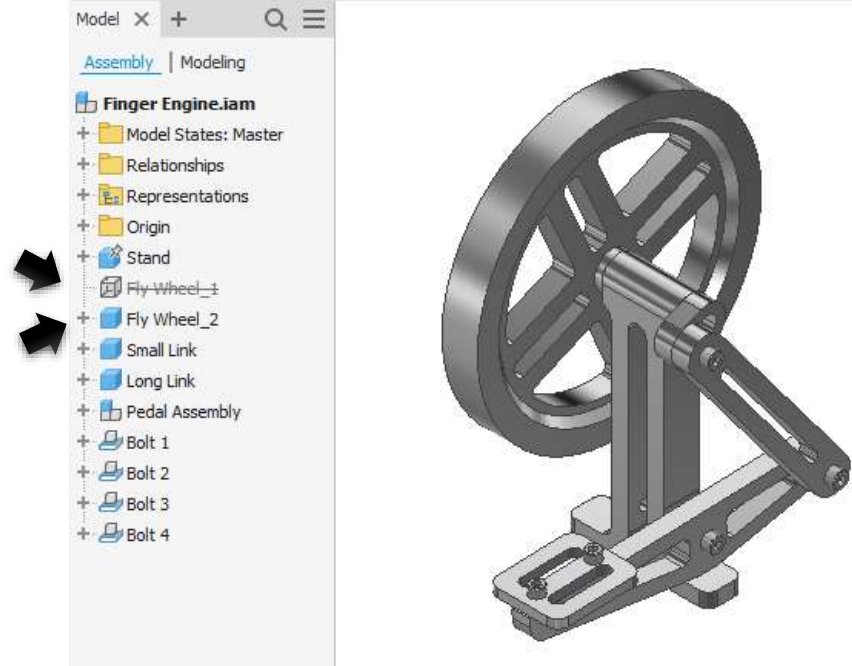
iLogic Place and Constrain vs Suppress/Unsuppress

Tip #8

Configuring Active and Inactive Components

TRADITIONAL ILOGIC APPROACH

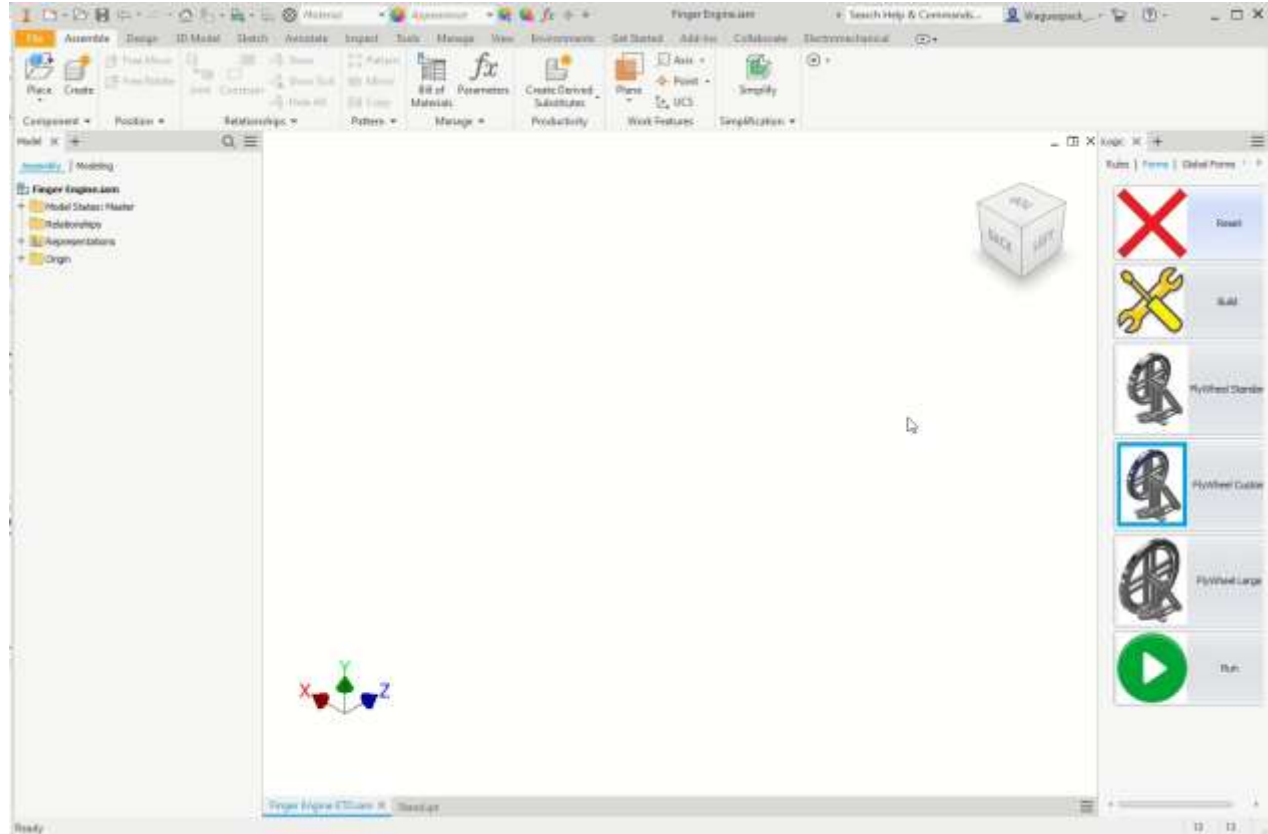
- In the past our primary mode of operation withing in iLogic was to create configurations by placing all of the components in an assembly, and then making some active and some inactive in order to configure
- This created some extra overhead, as far as dealing with Level of Details, Bill of Materials, Drawings, etc.



Place and Constrain

NEW ILOGIC APPROACH

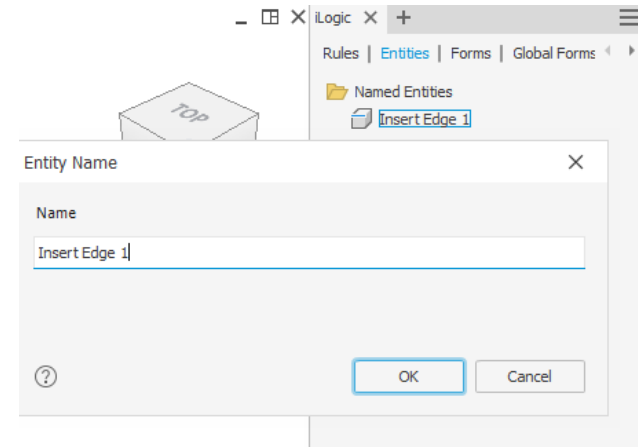
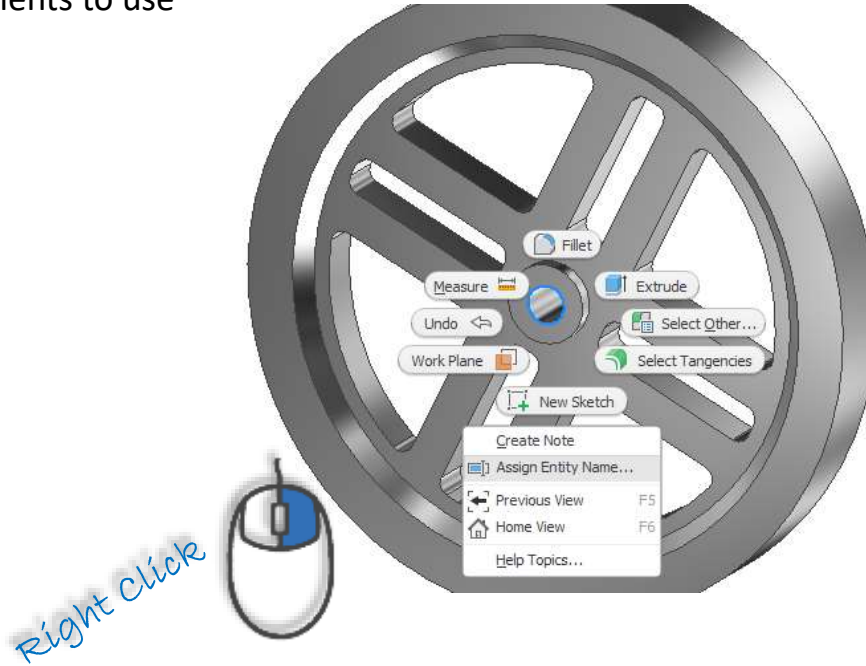
- Inventor 2019 introduced the ability to add components and constraints quickly and easily, by extracting the code from an assembly.
- This allows us to start with an empty template assembly and place only the components needed.
- Or swap out components as needed
- The advantage is there are no need to manage suppressed or invisible components for drawings, Bill of Materials, etc.



Place and Constrain

NEW ILOGIC APPROACH

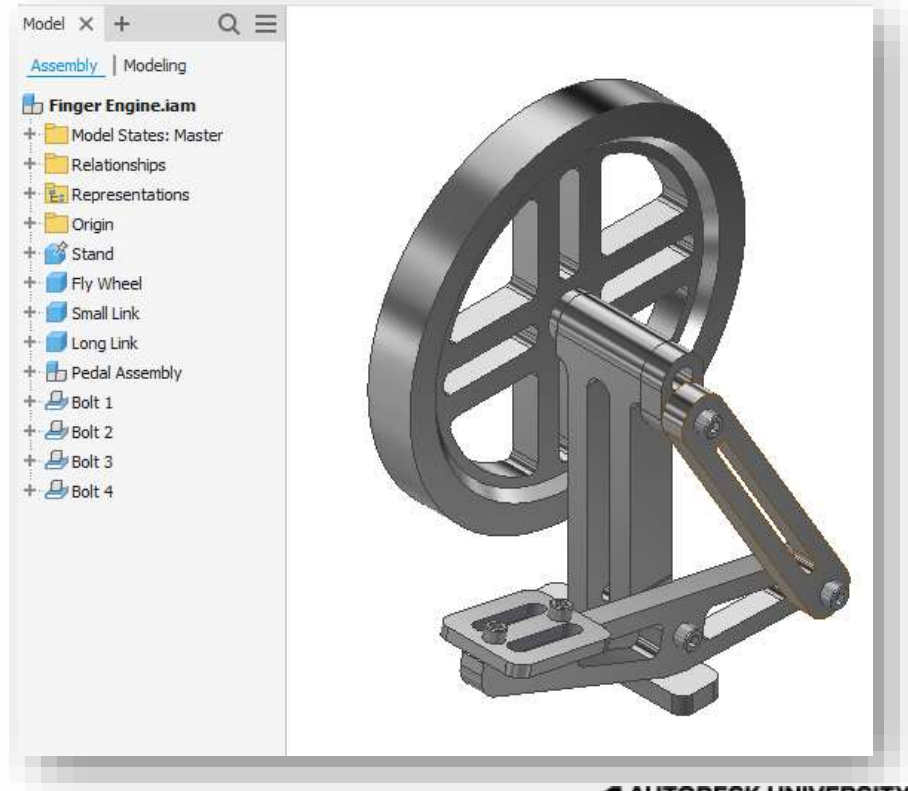
1. Create Named Geometry Entities in the components to use for constraints



Place and Constrain

NEW ILOGIC APPROACH

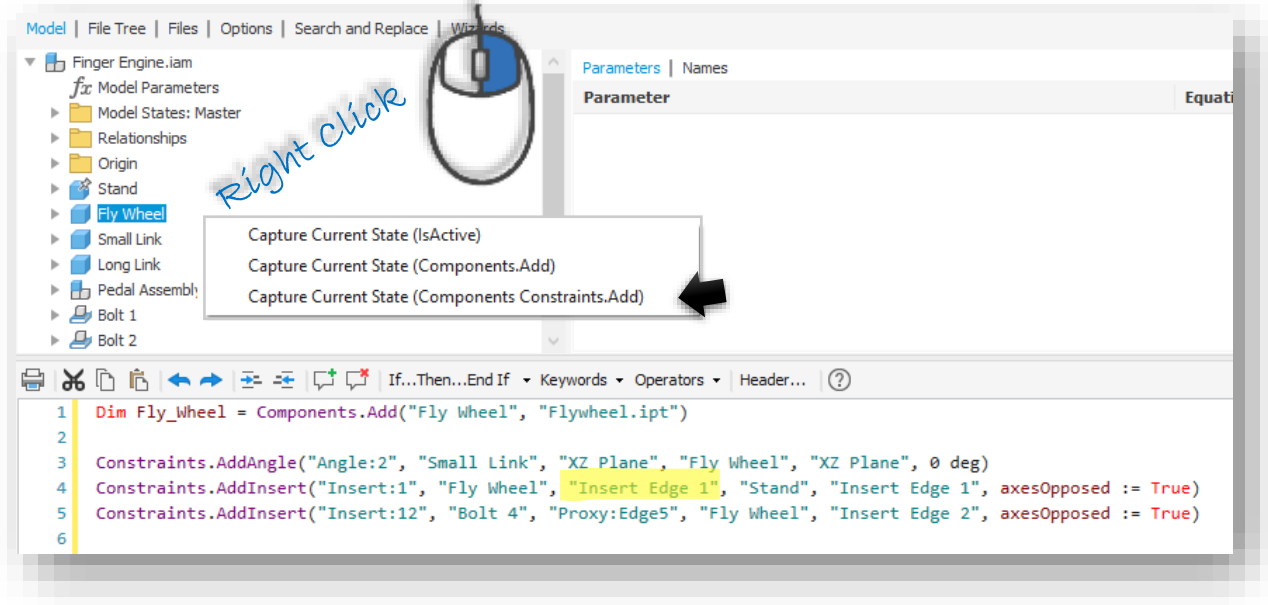
1. Create Named Geometry
Entities in the components to use
for constraints
2. Create and constrain your
assembly



Extract the Code

NEW ILOGIC APPROACH

1. Create Named Geometry Entities in the components to use for constraints
2. Create and constrain your assembly
3. Extract the code from your fully assembled model

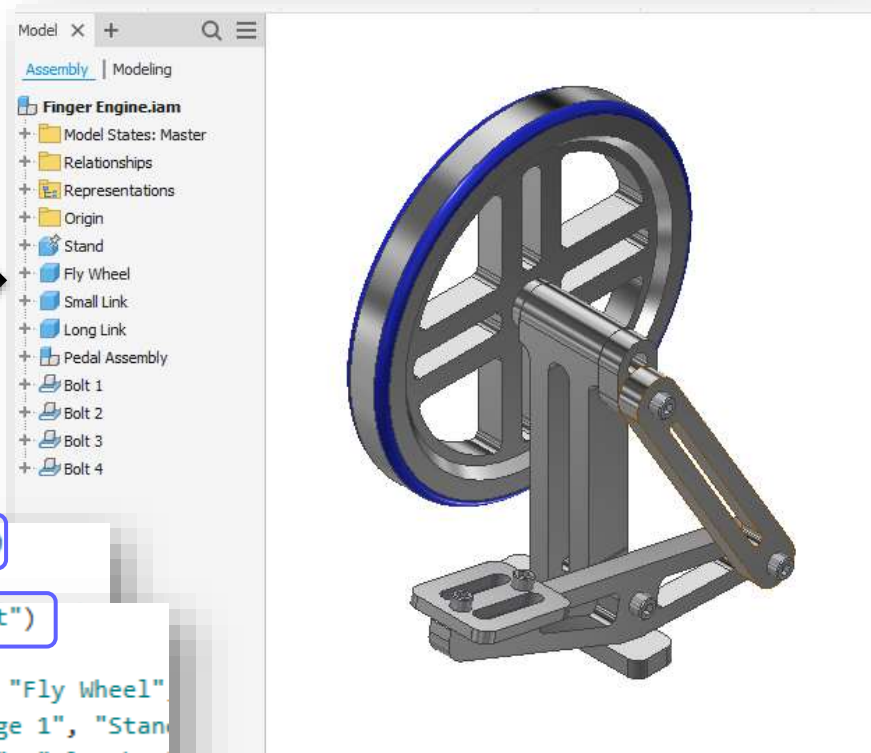


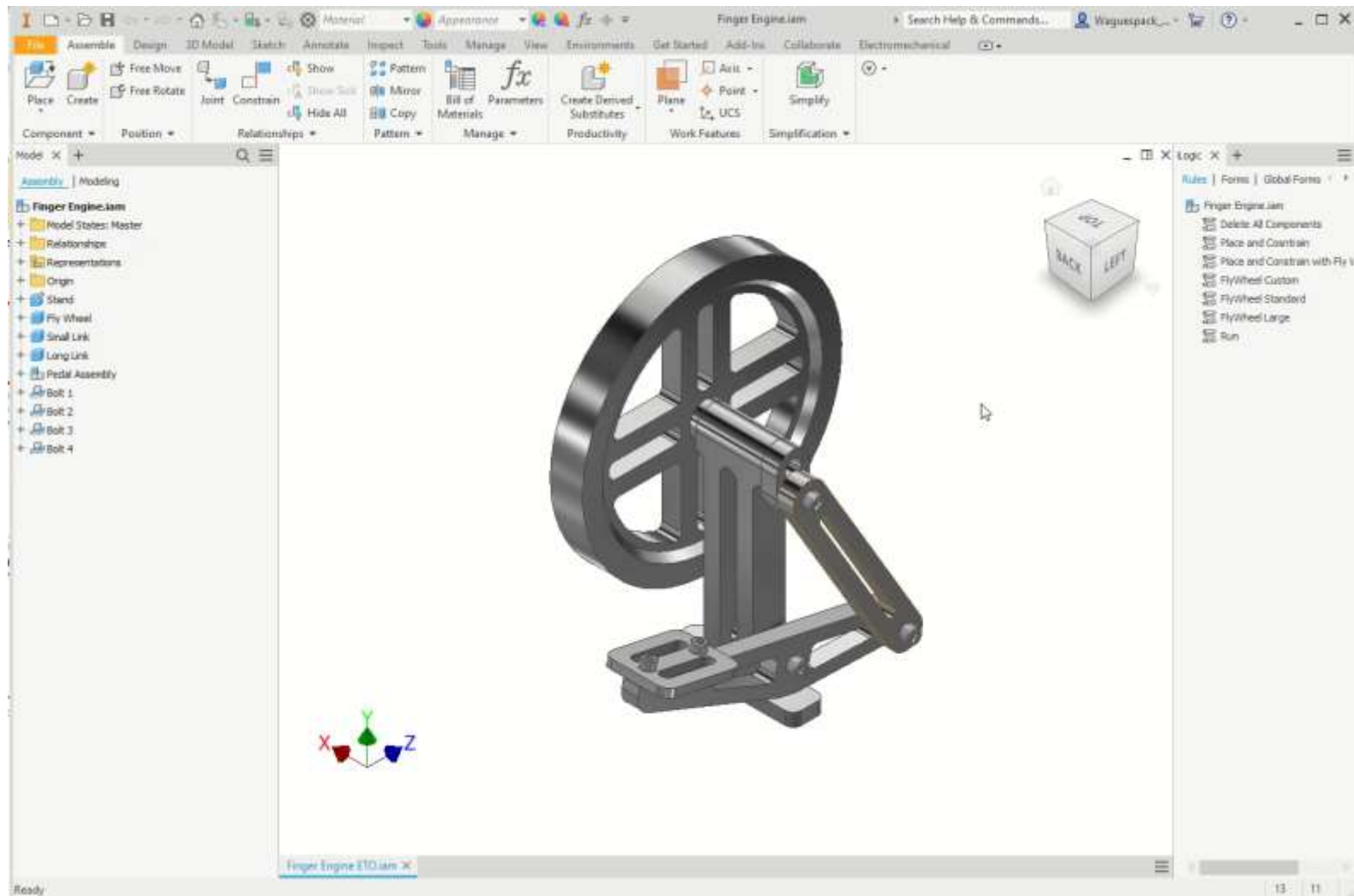
Configure the rule

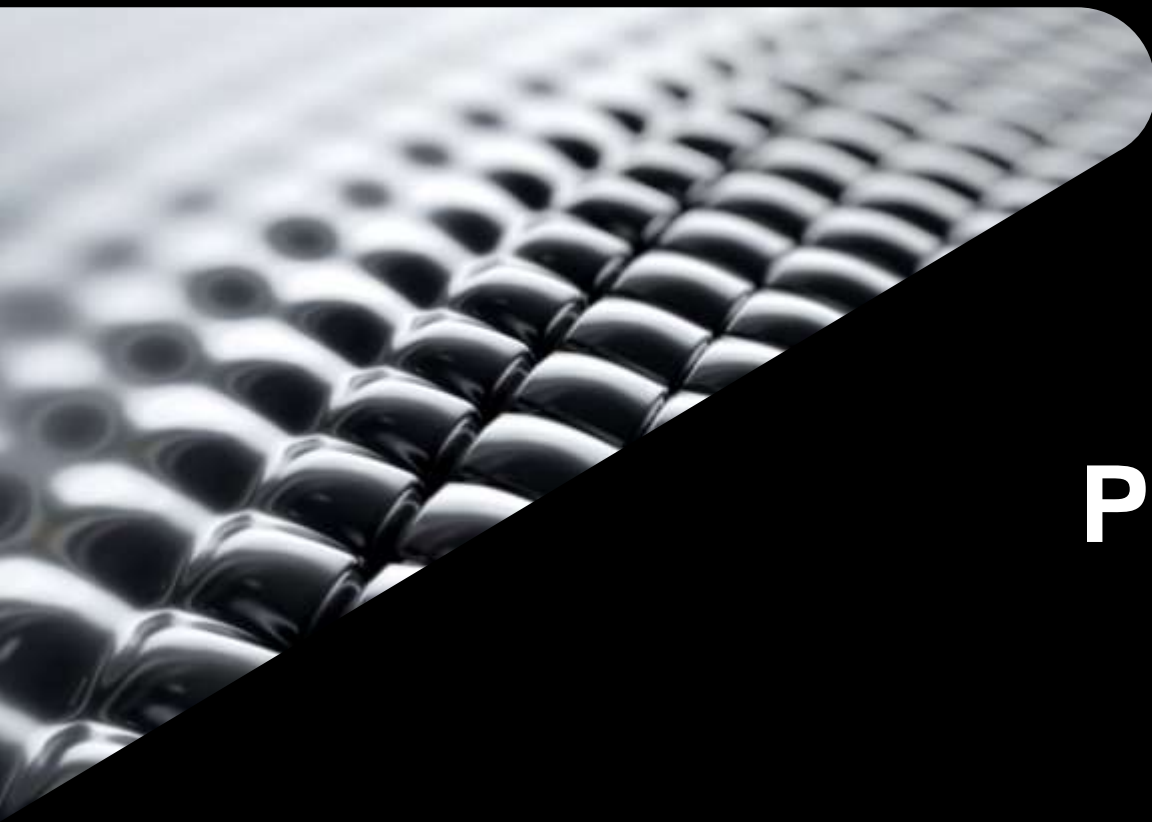
NEW ILOGIC APPROACH

1. Create Named Geometry Entities in the components to use for constraints
2. Create and constrain your assembly
3. Extract the code from your fully assembled model
4. Configure rules to swap out component files as needed.

```
1 Dim Fly_Wheel = Components.Add("Fly Wheel", "Flywheel.ipt")
2
3 Dim Fly_Wheel = Components.Add("Fly Wheel", "Flywheel2.ipt")
4
5 Constraints.AddAngle("Angle:2", "Small Link", "XZ Plane", "Fly Wheel")
6 Constraints.AddInsert("Insert:1", "Fly Wheel", "Insert Edge 1", "Stand")
7 Constraints.AddInsert("Insert:12", "Bolt 4", "Proxy:Edge5", "Fly Wheel")
```







Plan your code

Tip #9

Planning our code allows us to write much more maintainable iLogic rules

```
1 trigger = Height
2
3 IF Parameter("Height") <= 24 And Parameter("Width") <= 12 Then
4     Parameter("Size") = 4
5     Parameter("Count") = 5
6
7     Component.Visible("B04 Assembly(1)") = True
8     oComp1 = Component.InventorComponent("B04 Assembly(1)")
9     oComp1.BOPStructure = BOPStructureFromReference(BOPStructure
10     Component.Visible("B04 Assembly(1)") = True
11     oComp1 = Component.InventorComponent("B04 Assembly(1)")
12     oComp1.BOPStructure = BOPStructureFromReference(BOPStructure
13     Constraint.IsActive("B04_Hole") = True
14
15     Component.Visible("B05 Assembly(1)") = False
16     oComp1 = Component.InventorComponent("B05 Assembly(1)")
17     oComp1.BOPStructure = BOPStructureFromReference(BOPStructure
18     Component.Visible("B05 Assembly(1)") = False
19     oComp1 = Component.InventorComponent("B05 Assembly(1)")
20     oComp1.BOPStructure = BOPStructureFromReference(BOPStructure
21     Constraint.IsActive("B05_Hole") = False
22
23     Component.Visible("B06 Assembly(1)") = False
24     oComp1 = Component.InventorComponent("B06 Assembly(1)")
25
2600 oComp1 = Component.InventorComponent("B010 Assembly(1)")
2601 oComp1.BOPStructure = BOPStructureFromReference(BOPStructure
2602 Component.Visible("B010 Assembly(1)") = False
2603 oComp1 = Component.InventorComponent("B010 Assembly(1)")
2604 oComp1.BOPStructure = BOPStructureFromReference(BOPStructure
2605 Constraint.IsActive("B010_Hole") = False
2606
2607 Component.Visible("B010 Assembly(1)") = False
2608 oComp1 = Component.InventorComponent("B010 Assembly(1)")
2609 oComp1.BOPStructure = BOPStructureFromReference(BOPStructure
2610 Component.Visible("B010 Assembly(1)") = False
2611 oComp1 = Component.InventorComponent("B010 Assembly(1)")
2612 oComp1.BOPStructure = BOPStructureFromReference(BOPStructure
2613 Constraint.IsActive("B010_Hole") = False
2614
2615 Component.Visible("B010 Assembly(1)") = True
2616 oComp1 = Component.InventorComponent("B010 Assembly(1)")
2617 oComp1.BOPStructure = BOPStructureFromReference(BOPStructure
2618 Component.Visible("B010 Assembly(1)") = True
2619 oComp1 = Component.InventorComponent("B010 Assembly(1)")
2620 oComp1.BOPStructure = BOPStructureFromReference(BOPStructure
2621 Constraint.IsActive("B010_Hole") = True
2622
2623 End If
```

2510
lines of
code!



Rule to do
the same
thing, but
only 67
lines of
code!



```
1 Sub Main
2
3     trigger = Height
4
5     'step into subroutine to set Size
6     Call SetSize(Height)
7
8     For i = 4 To 20
9         Parameter("Count") = Parameter("Size") - 1
10
11         oBoolean = 1 <= Parameter("Size")
12
13         oObjectName = "B0" & i & " Assembly(1)"
14         Call Set_Visible(oObjectName, oBoolean) 'step into subroutine to set visibility
15
16         oObjectName = "B0" & i & " Assembly(1)"
17         Call Set_Visible(oObjectName, oBoolean) 'step into subroutine to set visibility
18     Next
19
20 End Sub
21
22 Sub Set_Visible(oObjectName As String, oBoolean As Boolean)
23
24     Component.InventorComponent(oObjectName).Visible = oBoolean
25     Constraint.IsActive("B0" & i & "_Hole") = oBoolean
26
27     'step into subroutine to set BOP Structure
28     Call BOPStructure(oObjectName)
29
30 End Sub
31
32 Sub BOPStructure(oObjectName As String)
33
34     If Component.Visible(oObjectName) = True Then
35         Component.InventorComponent(oObjectName).BOPStructure = _
36         BOPStructureFromReference(BOPStructure
37     Else If Component.Visible(oObjectName) = False Then
38         Component.InventorComponent(oObjectName).BOPStructure = _
39         BOPStructureFromReference(BOPStructure
40     End If
41
42 End Sub
43
44 Sub SetSize(Height As Integer)
45
46     If Parameter("Height") <= 24 And Parameter("Height") <= 31 Then : Parameter("Size") = 4
47     Else If Parameter("Height") <= 31 And Parameter("Height") <= 40 Then : Parameter("Size") = 5
48     Else If Parameter("Height") <= 40 And Parameter("Height") <= 48 Then : Parameter("Size") = 6
49     Else If Parameter("Height") <= 48 And Parameter("Height") <= 56 Then : Parameter("Size") = 7
50     Else If Parameter("Height") <= 56 And Parameter("Height") <= 64 Then : Parameter("Size") = 8
51     Else If Parameter("Height") <= 64 And Parameter("Height") <= 72 Then : Parameter("Size") = 9
52     Else If Parameter("Height") <= 72 And Parameter("Height") <= 80 Then : Parameter("Size") = 10
53     Else If Parameter("Height") <= 80 And Parameter("Height") <= 88 Then : Parameter("Size") = 11
54     Else If Parameter("Height") <= 88 And Parameter("Height") <= 96 Then : Parameter("Size") = 12
55     Else If Parameter("Height") <= 96 And Parameter("Height") <= 104 Then : Parameter("Size") = 13
56     Else If Parameter("Height") <= 104 And Parameter("Height") <= 112 Then : Parameter("Size") = 14
57     Else If Parameter("Height") <= 112 And Parameter("Height") <= 120 Then : Parameter("Size") = 15
58     Else If Parameter("Height") <= 120 And Parameter("Height") <= 128 Then : Parameter("Size") = 16
59     Else If Parameter("Height") <= 128 And Parameter("Height") <= 136 Then : Parameter("Size") = 17
60     Else If Parameter("Height") <= 136 And Parameter("Height") <= 144 Then : Parameter("Size") = 18
61     Else If Parameter("Height") <= 144 And Parameter("Height") <= 152 Then : Parameter("Size") = 19
62     Else If Parameter("Height") <= 152 And Parameter("Height") <= 160 Then : Parameter("Size") = 20
63     End If
64
65     RuleParametersOutput()
66     InventorVb.DocumentUpdate()
67 End Sub
```

How to plan our iLogic rules

- Write out some of it, such as just a single size of the configuration to **get things working**
- Often this involves a lot of **copy/paste**, which creates repetitive lines
- Stop and look for a **pattern**
- **Recognize** this as an opportunity

```
1  oTrigger = Height
2  If Parameter("Height") >= 24 And Parameter("Height") <= 32 Then
3      Parameter("Size") = 4
4      Parameter("Count") = 3
5
6      Component.Visible("NG4 AssemblyLH:1") = True
7      oCompOcc = Component.InventorComponent("NG4 AssemblyLH:1")
8      oCompOcc.BOMStructure = BOMStructureEnum.kDefaultBOMStructure
9      Component.Visible("NG4 AssemblyRH:1") = True
10     oCompOcc = Component.InventorComponent("NG4 AssemblyRH:1")
11     oCompOcc.BOMStructure = BOMStructureEnum.kDefaultBOMStructure
12     Constraint.IsActive("NG4 _Mate") = True
13
14     Component.Visible("NG5 AssemblyLH:1") = False
15     oCompOcc = Component.InventorComponent("NG5 AssemblyLH:1")
16     oCompOcc.BOMStructure = BOMStructureEnum.kReferenceBOMStructure
17     Component.Visible("NG5 AssemblyRH:1") = False
18     oCompOcc = Component.InventorComponent("NG5 AssemblyRH:1")
19     oCompOcc.BOMStructure = BOMStructureEnum.kReferenceBOMStructure
20     Constraint.IsActive("NG5 _Mate") = False
21
22     Component.Visible("NG6 AssemblyLH:1") = False
23     oCompOcc = Component.InventorComponent("NG6 AssemblyLH:1")
```

Evaluate and outline what you have

```
1 oTrigger = Height
2 If Parameter("Height") >= 24 And Parameter("Height") <= 32 Then
3     Parameter("Size") = 4
4     Parameter("Count") = 3
5
6     Component.Visible("NG4 AssemblyLH:1") = True
7     oCompOcc = Component.InventorComponent("NG4 AssemblyLH:1")
8     oCompOcc.BOMStructure = BOMStructureEnum.kDefaultBOMStructure
9     Component.Visible("NG4 AssemblyRH:1") = True
10    oCompOcc = Component.InventorComponent("NG4 AssemblyRH:1")
11    oCompOcc.BOMStructure = BOMStructureEnum.kDefaultBOMStructure
12    Constraint.IsActive("NG4 _Mate") = True
13
14    Component.Visible("NG5 AssemblyLH:1") = False
15    oCompOcc = Component.InventorComponent("NG5 AssemblyLH:1")
16    oCompOcc.BOMStructure = BOMStructureEnum.kReferenceBOMStructure
17    Component.Visible("NG5 AssemblyRH:1") = False
18    oCompOcc = Component.InventorComponent("NG5 AssemblyRH:1")
19    oCompOcc.BOMStructure = BOMStructureEnum.kReferenceBOMStructure
20    Constraint.IsActive("NG5 _Mate") = False
21
22    Component.Visible("NG6 AssemblyLH:1") = False
23    oCompOcc = Component.InventorComponent("NG6 AssemblyLH:1")
```

20 instances of RH & 20 instances of LH assemblies (40 assemblies)

17 Height ranges (4 thru 20) ... 1 thru 3 are static

If Height >= (some number) and Height <= (some other number) Then

Size = 4

Count = 3

LH size 4 and RH size 4 Assemblies visible w/ default BOM structure,
all others are invisible and reference

Else If Height >= (some number) and Height <= (some other number) Then

Size = 5

Count = 4

LH size 4 and RH size 4 Assemblies visible w/ default BOM structure,
all others are invisible and reference

Else if....

$$140 \times 17 = 2380$$

Use your outline to write “pseudo” code

- Plan out your code structure
- Create Sub procedures and Functions to reuse code where we're doing the same task for different components

Trigger = Height

~ Call Sub to set Size

~ {Loop i = 4 to 20,

~ Set Count = Size -1

~ BooleanVariable = i <= Parameter("Size")

~ Call Sub to set Visibility

~ Call Sub to set BOM structure

Next}

{Sub to set Size

If Height >= (X) and Height <= (Y) Then

Set Size

Else....

End Sub}

{Sub to set Visibility

~ Set visibility

~ Set constraint active

End Sub}

{Sub to set BOM structure

~ if component is visible, then default BOM,

Else reference BOM

End Sub}


```

1  Sub Main
2
3      cTrigger = Height
4
5      'step into subroutine to set Size
6      Call SetSize(mHeight)
7
8      For i = 4 To 20
9          Parameter("Count") = Parameter("Size") -1
10
11          mBoolean = 1 <> Parameter("Size")
12
13          oOccName = "HG" & i & " - AssemblyHt:1"
14          Call Set_Visible(oOccName, mBoolean) 'step into subroutine to set visibility
15
16          oOccName = "HG" & i & " - AssemblyBt:1"
17          Call Set_Visible(oOccName, mBoolean) 'step into subroutine to set visibility
18      Next
19
20  End Sub
21
22  Sub Set_Visible(oOccName As String, mBoolean As Boolean)
23
24      Component.InventorComponent(oOccName).Visible = mBoolean
25      Constraint.IsActive("HG" & i & " - Ht:1") = mBoolean
26
27      'step into subroutine to set BOP Structure
28      Call BOPStructure(oOccName)
29
30  End Sub
31
32  Sub BOPStructure(oOccName As String)
33
34      If Component.Visible(oOccName) = True Then
35          Component.InventorComponent(oOccName).BOPStructure = _
36              BOPStructureEnum.kDefaultBOPStructure
37      ElseIf Component.Visible(oOccName) = False Then
38          Component.InventorComponent(oOccName).BOPStructure = _
39              BOPStructureEnum.kParameterizedBOPStructure
40      End If
41
42  End Sub
43
44  Sub SetSize(mHeight As Integer)
45
46      If Parameter("Height") <= 24 And Parameter("Height") <= 32 Then : Parameter("Size") = 4
47      ElseIf Parameter("Height") > 32 And Parameter("Height") <= 40 Then : Parameter("Size") = 5
48      ElseIf Parameter("Height") > 40 And Parameter("Height") <= 48 Then : Parameter("Size") = 6
49      ElseIf Parameter("Height") > 48 And Parameter("Height") <= 56 Then : Parameter("Size") = 7
50      ElseIf Parameter("Height") > 56 And Parameter("Height") <= 64 Then : Parameter("Size") = 8
51      ElseIf Parameter("Height") > 64 And Parameter("Height") <= 72 Then : Parameter("Size") = 9
52      ElseIf Parameter("Height") > 72 And Parameter("Height") <= 80 Then : Parameter("Size") = 10
53      ElseIf Parameter("Height") > 80 And Parameter("Height") <= 88 Then : Parameter("Size") = 11
54      ElseIf Parameter("Height") > 88 And Parameter("Height") <= 96 Then : Parameter("Size") = 12
55      ElseIf Parameter("Height") > 96 And Parameter("Height") <= 104 Then : Parameter("Size") = 13
56      ElseIf Parameter("Height") > 104 And Parameter("Height") <= 112 Then : Parameter("Size") = 14
57      ElseIf Parameter("Height") > 112 And Parameter("Height") <= 120 Then : Parameter("Size") = 15
58      ElseIf Parameter("Height") > 120 And Parameter("Height") <= 128 Then : Parameter("Size") = 16
59      ElseIf Parameter("Height") > 128 And Parameter("Height") <= 136 Then : Parameter("Size") = 17
60      ElseIf Parameter("Height") > 136 And Parameter("Height") <= 144 Then : Parameter("Size") = 18
61      ElseIf Parameter("Height") > 144 And Parameter("Height") <= 152 Then : Parameter("Size") = 19
62      ElseIf Parameter("Height") > 152 And Parameter("Height") <= 160 Then : Parameter("Size") = 20
63      End If
64
65      RuleParametersOutput()
66      InventorVb.Document.Update()
67  End Sub

```

```
1 Sub Main
```

```
2  
3 oTrigger = Height
```

```
4  
5 'step into subroutine to set Size
```

```
6 Call SetSize(oHeight)
```

```
7  
8 For i = 4 To 20
```

```
9 Parameter("Count") = Parameter("Size") - 1
```

```
10  
11 oBoolean = i <= Parameter("Size")
```

```
12  
13 oOccName = "NG" & i & " AssemblyLM:1"
```

```
14 Call Set_Visible(oOccName, oBoolean) 'step into subroutine to set visibility
```

```
15  
16 oOccName = "NG" & i & " AssemblyRM:1"
```

```
17 Call Set_Visible(oOccName, oBoolean) 'step into subroutine to set visibility
```

```
18 Next
```

```
19  
20 End Sub
```

```
1 Sub Main  
2  
3 oTrigger = Height  
4  
5 'step into subroutine to set Size  
6 Call SetSize(oHeight)  
7
```

```
36 Component.InventorComponent(oOccName).BOMStructure = _  
37 BOMStructureEnum.kDefaultBOMStructure  
38  
39 If Not Component.Visible(oOccName) = False Then  
40 Component.InventorComponent(oOccName).BOMStructure = _  
41 BOMStructureEnum.kParametricBOMStructure  
42 End If  
43  
44 End Sub  
45  
46 Sub SetSize(oHeight As Integer)  
47  
48 If Parameter("Height") <= 24 And Parameter("Height") <= 32 Then : Parameter("Size") = 4  
49 ElseIf Parameter("Height") > 32 And Parameter("Height") <= 40 Then : Parameter("Size") = 5  
50 ElseIf Parameter("Height") > 40 And Parameter("Height") <= 48 Then : Parameter("Size") = 6  
51 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 56 Then : Parameter("Size") = 7  
52 ElseIf Parameter("Height") > 56 And Parameter("Height") <= 64 Then : Parameter("Size") = 8  
53 ElseIf Parameter("Height") > 64 And Parameter("Height") <= 72 Then : Parameter("Size") = 9  
54 ElseIf Parameter("Height") > 72 And Parameter("Height") <= 80 Then : Parameter("Size") = 10  
55 ElseIf Parameter("Height") > 80 And Parameter("Height") <= 88 Then : Parameter("Size") = 11  
56 ElseIf Parameter("Height") > 88 And Parameter("Height") <= 96 Then : Parameter("Size") = 12  
57 ElseIf Parameter("Height") > 96 And Parameter("Height") <= 104 Then : Parameter("Size") = 13  
58 ElseIf Parameter("Height") > 104 And Parameter("Height") <= 112 Then : Parameter("Size") = 14  
59 ElseIf Parameter("Height") > 112 And Parameter("Height") <= 120 Then : Parameter("Size") = 15  
60 ElseIf Parameter("Height") > 120 And Parameter("Height") <= 128 Then : Parameter("Size") = 16  
61 ElseIf Parameter("Height") > 128 And Parameter("Height") <= 136 Then : Parameter("Size") = 17  
62 ElseIf Parameter("Height") > 136 And Parameter("Height") <= 144 Then : Parameter("Size") = 18  
63 ElseIf Parameter("Height") > 144 And Parameter("Height") <= 152 Then : Parameter("Size") = 19  
64 ElseIf Parameter("Height") > 152 And Parameter("Height") <= 160 Then : Parameter("Size") = 20  
65 End If  
66  
67 RuleParametersOutput()  
68 InventorVB.Document.Update()  
69 End Sub
```



```
[Euler] -1  
|  
|  
HLS"  
can) "step into subroutine to set visibility  
HLS"  
can) "step into subroutine to set visibility  
  
[Leave As Boolean]  
  
Visible = otherwise:  
cr") = stoclase
```

[illegible]



```

[sum] -1
[0.5]
sum) 'step into subroutine to set visibility
[0.5]
sum) 'step into subroutine to set visibility
[sum && module]

```

```
Sub SetSize(oHeight As Integer)
```

```

If Parameter("Height") <= 24 And Parameter("Height") <= 32 Then : Parameter("Size") = 4
ElseIf Parameter("Height") > 32 And Parameter("Height") <= 40 Then : Parameter("Size") = 5
ElseIf Parameter("Height") > 40 And Parameter("Height") <= 48 Then : Parameter("Size") = 6
ElseIf Parameter("Height") > 48 And Parameter("Height") <= 56 Then : Parameter("Size") = 7
ElseIf Parameter("Height") > 56 And Parameter("Height") <= 64 Then : Parameter("Size") = 8
ElseIf Parameter("Height") > 64 And Parameter("Height") <= 72 Then : Parameter("Size") = 9
ElseIf Parameter("Height") > 72 And Parameter("Height") <= 80 Then : Parameter("Size") = 10
ElseIf Parameter("Height") > 80 And Parameter("Height") <= 88 Then : Parameter("Size") = 11
ElseIf Parameter("Height") > 88 And Parameter("Height") <= 96 Then : Parameter("Size") = 12
ElseIf Parameter("Height") > 96 And Parameter("Height") <= 104 Then : Parameter("Size") = 13
ElseIf Parameter("Height") > 104 And Parameter("Height") <= 112 Then : Parameter("Size") = 14
ElseIf Parameter("Height") > 112 And Parameter("Height") <= 120 Then : Parameter("Size") = 15
ElseIf Parameter("Height") > 120 And Parameter("Height") <= 128 Then : Parameter("Size") = 16
ElseIf Parameter("Height") > 128 And Parameter("Height") <= 136 Then : Parameter("Size") = 17
ElseIf Parameter("Height") > 136 And Parameter("Height") <= 144 Then : Parameter("Size") = 18
ElseIf Parameter("Height") > 144 And Parameter("Height") <= 152 Then : Parameter("Size") = 19
ElseIf Parameter("Height") > 152 And Parameter("Height") <= 160 Then : Parameter("Size") = 20
End If

RuleParametersOutput()
InventorVb.DocumentUpdate()
Sub

```

```
10 Then : Parameter["Size"] = 18
12 Then : Parameter["Size"] = 16
14 Then : Parameter["Size"] = 17
16 Then : Parameter["Size"] = 18
18 Then : Parameter["Size"] = 19
20 Then : Parameter["Size"] = 20
```

© 2006 The Authors
Journal compilation © 2006 Blackwell Publishing Ltd

```

1 Sub Main
2
3     oTrigger = Height
4
5     'step into subroutine to set Size
6     Call SetSize(oHeight)
7
8     For i = 4 To 20
9         Parameter("Count") = Parameter("Size") - 1
10
11         oBoolean = i <= Parameter("Size")
12
13         oOccName = "NG" & i & " AssemblyLM:1"
14         Call Set_Visible(oOccName, oBoolean) 'step into subroutine to set visibility
15
16         oOccName = "NG" & i & " AssemblyRM:1"
17         Call Set_Visible(oOccName, oBoolean) 'step into subroutine to set visibility
18     Next
19
20 End Sub

```

```

1 Sub Main
2
3     oTrigger = Height
4
5     'step into subroutine to set Size
6     Call SetSize(oHeight)
7

```

```

13) -1
14)
15) 'step into subroutine to set visibility
16)
17) 'step into subroutine to set visibility
18)
19)
20)
21)
22)
23)
24)
25)
26)
27)
28)
29)
30)
31)
32)
33)
34)
35)
36)
37)
38)
39)
40)
41)
42)
43)
44)
45)
46)
47)
48)
49)
50)
51)
52)
53)
54)
55)
56)
57)
58)
59)
60)
61)
62)
63)
64)
65)
66)
67)

```

```

1 Sub SetSize(oHeight As Integer)
2
3     If Parameter("Height") >= 24 And Parameter("Height") <= 32 Then : Parameter("Size") = 4
4     ElseIf Parameter("Height") > 32 And Parameter("Height") <= 40 Then : Parameter("Size") = 5
5     ElseIf Parameter("Height") > 40 And Parameter("Height") <= 48 Then : Parameter("Size") = 6
6     ElseIf Parameter("Height") > 48 And Parameter("Height") <= 56 Then : Parameter("Size") = 7
7     ElseIf Parameter("Height") > 56 And Parameter("Height") <= 64 Then : Parameter("Size") = 8
8     ElseIf Parameter("Height") > 64 And Parameter("Height") <= 72 Then : Parameter("Size") = 9
9     ElseIf Parameter("Height") > 72 And Parameter("Height") <= 80 Then : Parameter("Size") = 10
10    ElseIf Parameter("Height") > 80 And Parameter("Height") <= 88 Then : Parameter("Size") = 11
11    ElseIf Parameter("Height") > 88 And Parameter("Height") <= 96 Then : Parameter("Size") = 12
12    ElseIf Parameter("Height") > 96 And Parameter("Height") <= 104 Then : Parameter("Size") = 13
13    ElseIf Parameter("Height") > 104 And Parameter("Height") <= 112 Then : Parameter("Size") = 14
14    ElseIf Parameter("Height") > 112 And Parameter("Height") <= 120 Then : Parameter("Size") = 15
15    ElseIf Parameter("Height") > 120 And Parameter("Height") <= 128 Then : Parameter("Size") = 16
16    ElseIf Parameter("Height") > 128 And Parameter("Height") <= 136 Then : Parameter("Size") = 17
17    ElseIf Parameter("Height") > 136 And Parameter("Height") <= 144 Then : Parameter("Size") = 18
18    ElseIf Parameter("Height") > 144 And Parameter("Height") <= 152 Then : Parameter("Size") = 19
19    ElseIf Parameter("Height") > 152 And Parameter("Height") <= 160 Then : Parameter("Size") = 20
20    End If
21
22    RuleParametersOutput()
23    InventorVb.DocumentUpdate()
24 End Sub

```

```

36 Component.InventorComponent(oOccName).RDPStru
37 RDPStructureEnum.DefaultRDPStructure
38
39 ElseIf Component.Visible(oOccName) = False Then
40 Component.InventorComponent(oOccName).RDPStru
41 RDPStructureEnum.DefaultRDPStructure
42 End If
43
44 End Sub
45
46 Sub SetSize(oHeight As Integer)
47
48 If Parameter("Height") >= 24 And Parameter("Height") <= 32 Then : Parameter("Size") = 4
49 ElseIf Parameter("Height") > 32 And Parameter("Height") <= 40 Then : Parameter("Size") = 5
50 ElseIf Parameter("Height") > 40 And Parameter("Height") <= 48 Then : Parameter("Size") = 6
51 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 56 Then : Parameter("Size") = 7
52 ElseIf Parameter("Height") > 56 And Parameter("Height") <= 64 Then : Parameter("Size") = 8
53 ElseIf Parameter("Height") > 64 And Parameter("Height") <= 72 Then : Parameter("Size") = 9
54 ElseIf Parameter("Height") > 72 And Parameter("Height") <= 80 Then : Parameter("Size") = 10
55 ElseIf Parameter("Height") > 80 And Parameter("Height") <= 88 Then : Parameter("Size") = 11
56 ElseIf Parameter("Height") > 88 And Parameter("Height") <= 96 Then : Parameter("Size") = 12
57 ElseIf Parameter("Height") > 96 And Parameter("Height") <= 104 Then : Parameter("Size") = 13
58 ElseIf Parameter("Height") > 104 And Parameter("Height") <= 112 Then : Parameter("Size") = 14
59 ElseIf Parameter("Height") > 112 And Parameter("Height") <= 120 Then : Parameter("Size") = 15
60 ElseIf Parameter("Height") > 120 And Parameter("Height") <= 128 Then : Parameter("Size") = 16
61 ElseIf Parameter("Height") > 128 And Parameter("Height") <= 136 Then : Parameter("Size") = 17
62 ElseIf Parameter("Height") > 136 And Parameter("Height") <= 144 Then : Parameter("Size") = 18
63 ElseIf Parameter("Height") > 144 And Parameter("Height") <= 152 Then : Parameter("Size") = 19
64 ElseIf Parameter("Height") > 152 And Parameter("Height") <= 160 Then : Parameter("Size") = 20
65 End If
66
67 RuleParametersOutput()
68 InventorVb.DocumentUpdate()
69 End Sub

```



```

1 Sub Main
2
3     oTrigger = Height
4
5     'step into subroutine to set Size
6     Call SetSize(oHeight)
7
8     For i = 4 To 20
9         Parameter("Count") = Parameter("Size") - 1
10
11         oBoolean = i <= Parameter("Size")
12
13         oOccName = "NG" & i & " AssemblyLM:1"
14         Call Set_Visible(oOccName, oBoolean) 'step into subroutine to set visibility
15
16         oOccName = "NG" & i & " AssemblyRM:1"
17         Call Set_Visible(oOccName, oBoolean) 'step into subroutine to set visibility
18     Next
19
20 End Sub

```

```

21
22 Sub Set_Visible(oOccName As String, oBoolean As Boolean)
23
24     Component.InventorComponent(oOccName).Visible = oBoolean
25     Constraint.IsActive("NG" & i & " _Mate") = oBoolean
26
27     'step into subroutine to set BOM Structure
28     Call BOMStructure(oOccName)
29
30 End Sub

```

```

1 Sub Main
2
3     oTrigger = Height
4
5     'step into subroutine to set Size
6     Call SetSize(oHeight)
7

```

```

13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

Sub SetSize(oHeight As Integer)

```

```

If Parameter("Height") >= 24 And Parameter("Height") <= 32 Then : Parameter("Size") = 4
ElseIf Parameter("Height") > 32 And Parameter("Height") <= 40 Then : Parameter("Size") = 5
ElseIf Parameter("Height") > 40 And Parameter("Height") <= 48 Then : Parameter("Size") = 6
ElseIf Parameter("Height") > 48 And Parameter("Height") <= 56 Then : Parameter("Size") = 7
ElseIf Parameter("Height") > 56 And Parameter("Height") <= 64 Then : Parameter("Size") = 8
ElseIf Parameter("Height") > 64 And Parameter("Height") <= 72 Then : Parameter("Size") = 9
ElseIf Parameter("Height") > 72 And Parameter("Height") <= 80 Then : Parameter("Size") = 10
ElseIf Parameter("Height") > 80 And Parameter("Height") <= 88 Then : Parameter("Size") = 11
ElseIf Parameter("Height") > 88 And Parameter("Height") <= 96 Then : Parameter("Size") = 12
ElseIf Parameter("Height") > 96 And Parameter("Height") <= 104 Then : Parameter("Size") = 13
ElseIf Parameter("Height") > 104 And Parameter("Height") <= 112 Then : Parameter("Size") = 14
ElseIf Parameter("Height") > 112 And Parameter("Height") <= 120 Then : Parameter("Size") = 15
ElseIf Parameter("Height") > 120 And Parameter("Height") <= 128 Then : Parameter("Size") = 16
ElseIf Parameter("Height") > 128 And Parameter("Height") <= 136 Then : Parameter("Size") = 17
ElseIf Parameter("Height") > 136 And Parameter("Height") <= 144 Then : Parameter("Size") = 18
ElseIf Parameter("Height") > 144 And Parameter("Height") <= 152 Then : Parameter("Size") = 19
ElseIf Parameter("Height") > 152 And Parameter("Height") <= 160 Then : Parameter("Size") = 20
End If

```

```

RuleParametersOutput()
InventorVb.DocumentUpdate()
End Sub

```

```

1 Sub Main
2
3     oTrigger = Height
4
5     'step into subroutine to set Size
6     Call SetSize(oHeight)
7
8     For i = 4 To 20
9         Parameter("Count") = Parameter("Size") - 1
10
11         oBoolean = i <= Parameter("Size")
12
13         oOccName = "NG" & i & " AssemblyLM:1"
14         Call Set_Visible(oOccName, oBoolean) 'step into subroutine to set visibility
15
16         oOccName = "NG" & i & " AssemblyRM:1"
17         Call Set_Visible(oOccName, oBoolean) 'step into subroutine to set visibility
18     Next
19
20 End Sub

```

```

21
22 Sub Set_Visible(oOccName As String, oBoolean As Boolean)
23
24     Component.InventorComponent(oOccName).Visible = oBoolean
25     Constraint.IsActive("NG" & i & " _Mate") = oBoolean
26
27     'step into subroutine to set BOM Structure
28     Call BOMStructure(oOccName)
29
30 End Sub

```

```

1 Sub Main
2
3     oTrigger = Height
4
5     'step into subroutine to set Size
6     Call SetSize(oHeight)
7

```

```

130) -1
131
132 'step into subroutine to set visibility
133
134 'step into subroutine to set visibility
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

Sub SetSize(oHeight As Integer)

```

```

If Parameter("Height") >= 24 And Parameter("Height") <= 32 Then : Parameter("Size") = 4
ElseIf Parameter("Height") > 32 And Parameter("Height") <= 40 Then : Parameter("Size") = 5
ElseIf Parameter("Height") > 40 And Parameter("Height") <= 48 Then : Parameter("Size") = 6
ElseIf Parameter("Height") > 48 And Parameter("Height") <= 56 Then : Parameter("Size") = 7
ElseIf Parameter("Height") > 56 And Parameter("Height") <= 64 Then : Parameter("Size") = 8
ElseIf Parameter("Height") > 64 And Parameter("Height") <= 72 Then : Parameter("Size") = 9
ElseIf Parameter("Height") > 72 And Parameter("Height") <= 80 Then : Parameter("Size") = 10
ElseIf Parameter("Height") > 80 And Parameter("Height") <= 88 Then : Parameter("Size") = 11
ElseIf Parameter("Height") > 88 And Parameter("Height") <= 96 Then : Parameter("Size") = 12
ElseIf Parameter("Height") > 96 And Parameter("Height") <= 104 Then : Parameter("Size") = 13
ElseIf Parameter("Height") > 104 And Parameter("Height") <= 112 Then : Parameter("Size") = 14
ElseIf Parameter("Height") > 112 And Parameter("Height") <= 120 Then : Parameter("Size") = 15
ElseIf Parameter("Height") > 120 And Parameter("Height") <= 128 Then : Parameter("Size") = 16
ElseIf Parameter("Height") > 128 And Parameter("Height") <= 136 Then : Parameter("Size") = 17
ElseIf Parameter("Height") > 136 And Parameter("Height") <= 144 Then : Parameter("Size") = 18
ElseIf Parameter("Height") > 144 And Parameter("Height") <= 152 Then : Parameter("Size") = 19
ElseIf Parameter("Height") > 152 And Parameter("Height") <= 160 Then : Parameter("Size") = 20
End If

```

```

RuleParametersOutput()
InventorVb.DocumentUpdate()
End Sub

```

```

1 Sub Main
2
3   oTrigger = Height
4
5   'step into subroutine to set Size
6   Call SetSize(oHeight)
7

```

```

1 Sub Main
2
3   oTrigger = Height
4
5   'step into subroutine to set Size
6   Call SetSize(oHeight)
7
8   For i = 4 To 20
9     Parameter("Count") = Parameter("Size") - 1
10
11     oBoolean = i <= Parameter("Size")
12
13     oOccName = "NG" & i & " AssemblyLM:1"
14     Call Set_Visible(oOccName, oBoolean) 'step into subroutine to set visibility
15
16     oOccName = "NG" & i & " AssemblyRM:1"
17     Call Set_Visible(oOccName, oBoolean) 'step into subroutine to set visibility
18   Next
19
20 End Sub

```

```

21
22 Sub Set_Visible(oOccName As String, oBoolean As Boolean)
23
24   Component.InventorComponent(oOccName).Visible = oBoolean
25   Constraint.IsActive("NG" & i & " _Mate") = oBoolean
26
27   'step into subroutine to set BOM Structure
28   Call BOMStructure(oOccName)
29
30 End Sub

```

```

32 Sub BOMStructure(oOccName As String)
33
34   If Component.Visible(oOccName) = True Then
35     Component.InventorComponent(oOccName).BOMStructure = _
36     BOMStructureEnum.kDefaultBOMStructure
37   ElseIf Component.Visible(oOccName) = False Then
38     Component.InventorComponent(oOccName).BOMStructure = _
39     BOMStructureEnum.kReferenceBOMStructure
40   End If
41
42 End Sub

```

```

44 Sub SetSize(oHeight As Integer)

```

```

45
46   If Parameter("Height") >= 24 And Parameter("Height") <= 32 Then : Parameter("Size") = 4
47   ElseIf Parameter("Height") > 32 And Parameter("Height") <= 40 Then : Parameter("Size") = 5
48   ElseIf Parameter("Height") > 40 And Parameter("Height") <= 48 Then : Parameter("Size") = 6
49   ElseIf Parameter("Height") > 48 And Parameter("Height") <= 56 Then : Parameter("Size") = 7
50   ElseIf Parameter("Height") > 56 And Parameter("Height") <= 64 Then : Parameter("Size") = 8
51   ElseIf Parameter("Height") > 64 And Parameter("Height") <= 72 Then : Parameter("Size") = 9
52   ElseIf Parameter("Height") > 72 And Parameter("Height") <= 80 Then : Parameter("Size") = 10
53   ElseIf Parameter("Height") > 80 And Parameter("Height") <= 88 Then : Parameter("Size") = 11
54   ElseIf Parameter("Height") > 88 And Parameter("Height") <= 96 Then : Parameter("Size") = 12
55   ElseIf Parameter("Height") > 96 And Parameter("Height") <= 104 Then : Parameter("Size") = 13
56   ElseIf Parameter("Height") > 104 And Parameter("Height") <= 112 Then : Parameter("Size") = 14
57   ElseIf Parameter("Height") > 112 And Parameter("Height") <= 120 Then : Parameter("Size") = 15
58   ElseIf Parameter("Height") > 120 And Parameter("Height") <= 128 Then : Parameter("Size") = 16
59   ElseIf Parameter("Height") > 128 And Parameter("Height") <= 136 Then : Parameter("Size") = 17
60   ElseIf Parameter("Height") > 136 And Parameter("Height") <= 144 Then : Parameter("Size") = 18
61   ElseIf Parameter("Height") > 144 And Parameter("Height") <= 152 Then : Parameter("Size") = 19
62   ElseIf Parameter("Height") > 152 And Parameter("Height") <= 160 Then : Parameter("Size") = 20
63   End If
64
65   RuleParametersOutput()
66   InventorVb.DocumentUpdate()
67 End Sub

```


20 instances of RH & 20 instances of LH assemblies (40 assemblies)

17 Height ranges (4 thru 20) ... 1 thru 3 are static

If Height >= (some number) and Height <= (some other number)

Size = 4

Count = 3

LH size 4 and RH size 4 Assemblies visible w/
all others are invisible and reference

Else If Height >= (some number) and Height <= (some other number)

Size = 5

Count = 4

LH size 4 and RH size 4 Assemblies
all others are invisible and

Else if...

```

Trigger = Height
~ Call Sub to set Size
~ {Loop i = 4 to 20,
    ~ Set Count = Size - 1
    ~ BooleanVariable = i <= Parameter("Size")
    ~ Call Sub to set Visibility
    ~ Call Sub to set BOM structure
Next}
{Sub to set Size
    If Height >= (X) and Height <= (Y) Then
        Set Size
    Else...
End Sub}
{Sub to set Visibility
    ~ Set visibility
    ~ Set constraint active
End Sub}
{Sub to set BOM structure
    ~ if component is visible, then default BOM,
    Else reference BOM
End Sub}
  
```

```

1 Sub Main
2
3 Trigger = Height
4
5 "step into subroutine to set Size
6 Call SetSize(Height)
7
8 For i = 4 to 20
9     Parameter("Count") = Parameter("Size") - 1
10
11     nBoolean = i <= Parameter("Size")
12
13     oOccName = "HG" & i & " - AssemblyH1"
14     (Call Set_Visible(oOccName, nBoolean) "step into subroutine to set visibility
15
16     oOccName = "HG" & i & " - AssemblyH1"
17     (Call Set_Visible(oOccName, nBoolean) "step into subroutine to set visibility
18
19 Next
20
21 End Sub
22
23 Sub Set_Visible(oOccName As String, nBoolean As Boolean)
24
25 Component.InventorComponent(oOccName).Visible = nBoolean
26 Const nInt13Active("HG" & i & " - nInt13") = nBoolean
27
28 "step into subroutine to set BOM structure
29 Call BOMStructure(oOccName)
30
31 End Sub
32
33 Sub BOMStructure(oOccName As String)
34
35 If Component.Visible(oOccName) = True Then
36     Component.InventorComponent(oOccName).BOMStructure =
37     BOMStructureFrom.DefaultBOMStructure
38 ElseIf Component.Visible(oOccName) = False Then
39     Component.InventorComponent(oOccName).BOMStructure =
40     BOMStructureFrom.ReferenceBOMStructure
41 End If
42
43 End Sub
44
45 Sub SetSize(Height As Integer)
46
47 If Parameter("Height") <= 24 And Parameter("Height") <= 31 Then : Parameter("Size") = 4
48 ElseIf Parameter("Height") > 31 And Parameter("Height") <= 40 Then : Parameter("Size") = 5
49 ElseIf Parameter("Height") > 40 And Parameter("Height") <= 40 Then : Parameter("Size") = 6
50 ElseIf Parameter("Height") > 40 And Parameter("Height") <= 44 Then : Parameter("Size") = 7
51 ElseIf Parameter("Height") > 44 And Parameter("Height") <= 44 Then : Parameter("Size") = 8
52 ElseIf Parameter("Height") > 44 And Parameter("Height") <= 48 Then : Parameter("Size") = 9
53 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 10
54 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 11
55 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 12
56 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 13
57 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 14
58 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 15
59 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 16
60 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 17
61 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 18
62 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 19
63 ElseIf Parameter("Height") > 48 And Parameter("Height") <= 48 Then : Parameter("Size") = 20
64 End If
65
66 RuleParametersOutput()
67 InventorVB.DocumentUpdate()
68
69 End Sub
  
```



Use Loops when possible

Tip #10

Use Loops to eliminate repetitious lines

Unoptimized rule

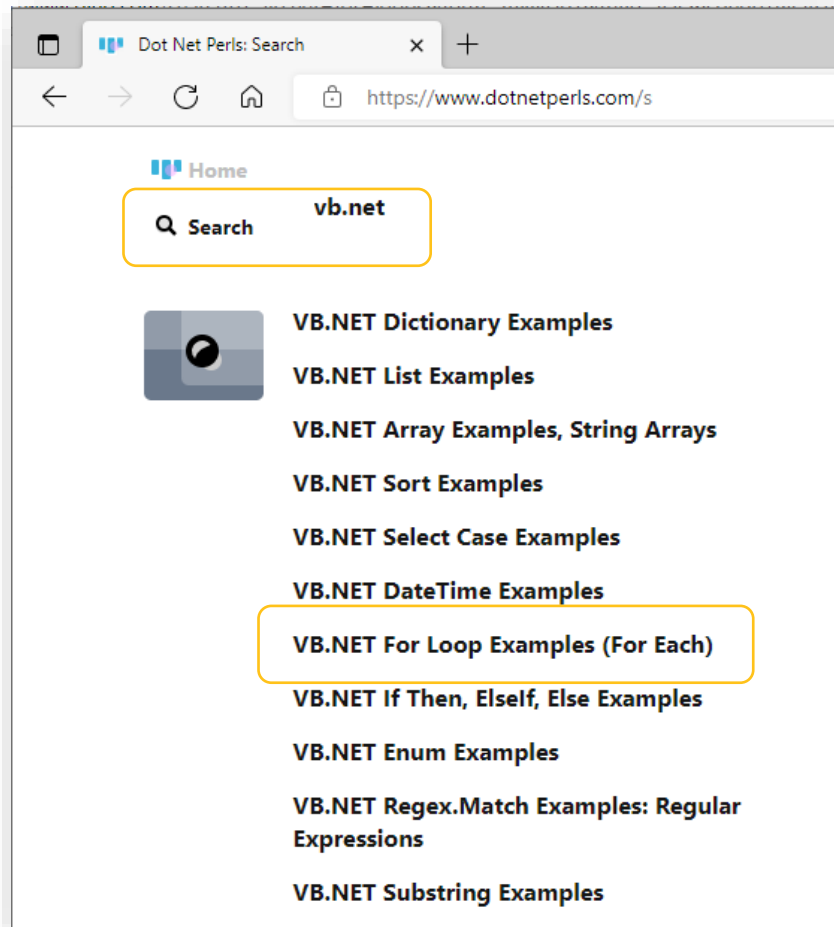
```
1 If Parameter("IsRightHand") True Then
2     Component.IsActive("RH_Brace") = True
3     Component.IsActive("RH_TopBracket") = True
4     Component.IsActive("RH_BottomBracket") = True
5     Component.IsActive("RH_Lid") = True
6     Component.IsActive("RH_Shim") = True
7
8     Component.IsActive("LH_Brace") = False
9     Component.IsActive("LH_TopBracket") = False
10    Component.IsActive("LH_BottomBracket") = False
11    Component.IsActive("LH_Lid") = False
12    Component.IsActive("LH_Shim") = False
13
14 Else
15     Component.IsActive("RH_Brace") = False
16     Component.IsActive("RH_TopBracket") = False
17     Component.IsActive("RH_BottomBracket") = False
18     Component.IsActive("RH_Lid") = False
19     Component.IsActive("RH_Shim") = False
20
21     Component.IsActive("LH_Brace") = True
22     Component.IsActive("LH_TopBracket") = True
23     Component.IsActive("LH_BottomBracket") = True
24     Component.IsActive("LH_Lid") = True
25     Component.IsActive("LH_Shim") = True
26 End If
```

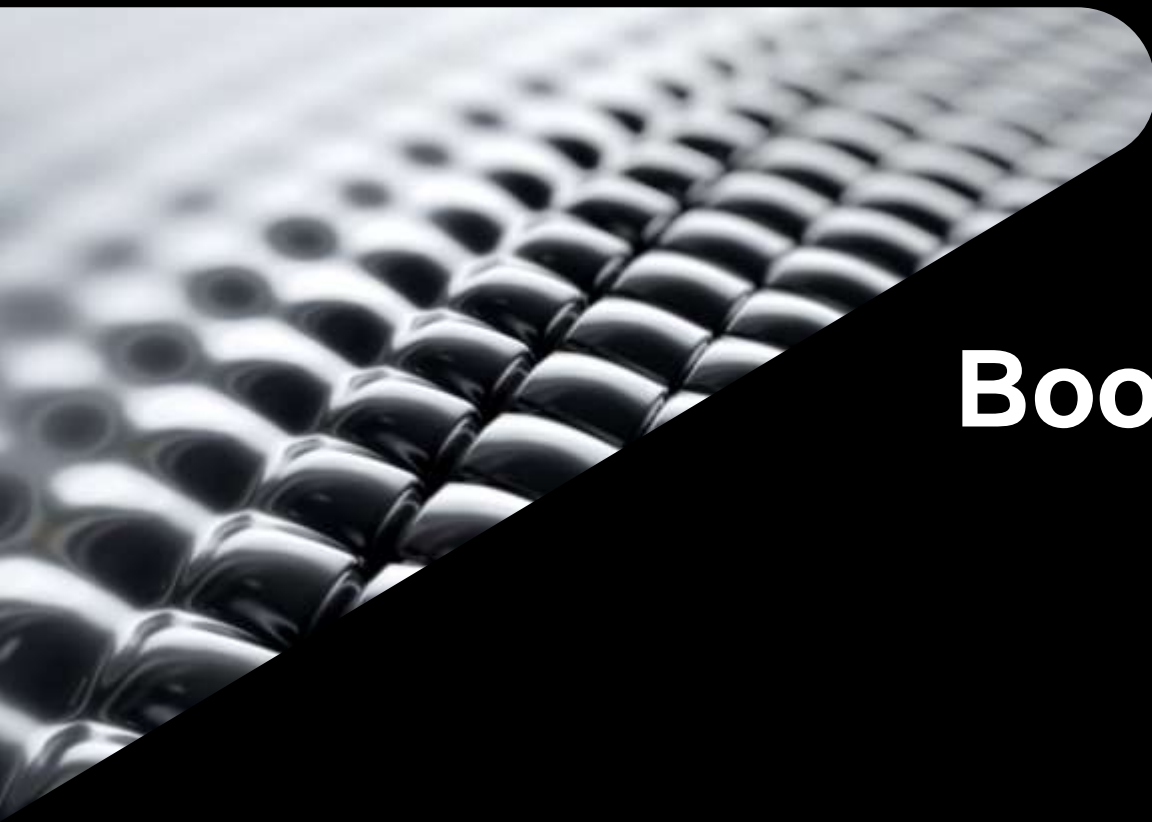
Optimized rule using a For Each loop

```
1 Dim oDoc As AssemblyDocument
2 oDoc = ThisDoc.Document
3
4 Dim oOcc As ComponentOccurrence
5 For Each oOcc In oDoc.ComponentDefinition.Occurrences
6
7     If Left(oOcc.Name, 3) = "RH_" Then
8         Component.IsActive(oOcc.Name) = Parameter("IsRightHand")
9     ElseIf Left(oOcc.Name, 3) = "LH_" Then
10        Component.IsActive(oOcc.Name) = Not Parameter("IsRightHand")
11    End If
12 Next
```

New to loops?

- Find examples online
- iLogic uses VB.net
- search online for something such as “**VB.net Loops**”
- Or “**VB.net For Each**”
- www.dotnetperls.com is a site that has a lot of examples






Booleans and the **Not** operator

Tip #11

Boolean = True/False

 **Bool·e·an**
/ˈboʊlēən/

adjective
denoting a system of algebraic notation used to represent logical propositions, especially in computing and electronics.

noun **COMPUTING**
a binary variable, having two possible values called "true" and "false".

Parameters

	Parameter Name	Consumed by	Unit/Type	Equation	No
	Model Parameters				
	User Parameters				
▶	IsCustom		True/False	True	

Skip the If/Then statements and use the **Not** operator

Parameters				
	Parameter Name	Consumed by	Unit/Type	Equation
	Model Parameters			
	User Parameters			
	IsCustom		True/False	True

```
1  If Parameter("IsCustom") = True Then
2
3      Component.IsActive("Bolt 1") = True
4      Component.IsActive("Bolt 2") = True
5      Component.IsActive("Bolt 3") = False
6      Component.IsActive("Bolt 4") = False
7
8  Else
9
10     Component.IsActive("Bolt 1") = False
11     Component.IsActive("Bolt 2") = False
12     Component.IsActive("Bolt 3") = True
13     Component.IsActive("Bolt 4") = True
14
15 End If
```

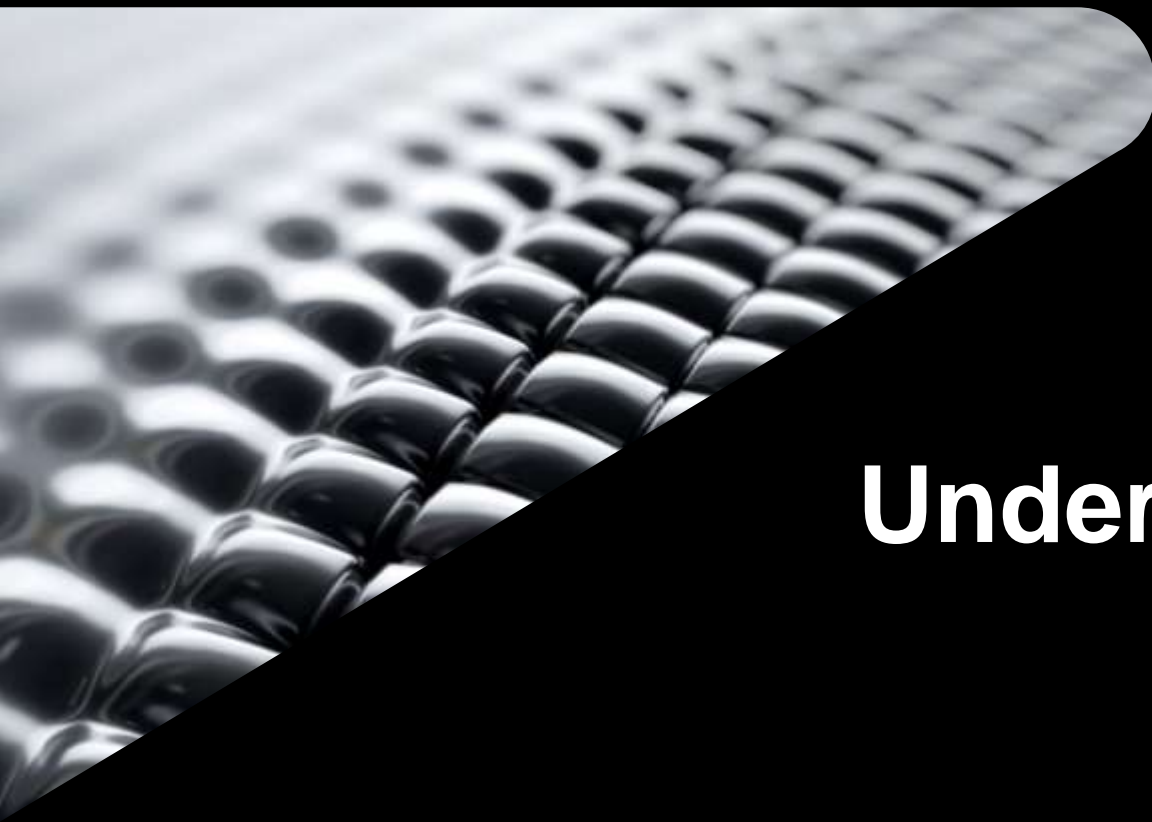
```
1  Component.IsActive("Bolt 1") = Parameter("IsCustom")
2  Component.IsActive("Bolt 2") = Parameter("IsCustom")
3  Component.IsActive("Bolt 3") = Not Parameter("IsCustom")
4  Component.IsActive("Bolt 4") = Not Parameter("IsCustom")
```

Optimized rule using the Not Operator

Objective:

Finding your way around the Inventor API

API: Application Programming Interface



Understand the API

Tip #12

An Analogy



- The chef knows how to create any dish, just as the Autodesk source code knows how to do anything in Inventor
 - Has access to all the recipes, ingredients, culinary tool and techniques
- We don't get to see the recipes, ingredients, culinary tool and techniques, and likewise we don't get to see Autodesk's source code
- We speak to the chef through the waiter, and we speak to Inventor's source code using the API
- iLogic is like the menu from which we read; in that it gives us a simplified subset of functions from which to choose

API version

```
Dim oDoc As PartDocument
oDoc = ThisApplication.ActiveDocument

Dim oCustomPropertySet As PropertySet
oCustomPropertySet = oDoc.PropertySets.Item _
    ("Inventor User Defined Properties")

Try
    'try to set the iproperty
    oProp = oCustomPropertySet.Item("Rack Number")
Catch
    'catch error when iproperty doesn't exist and create it
    oProp = oCustomPropertySet.Add("Rack Number")
End Try

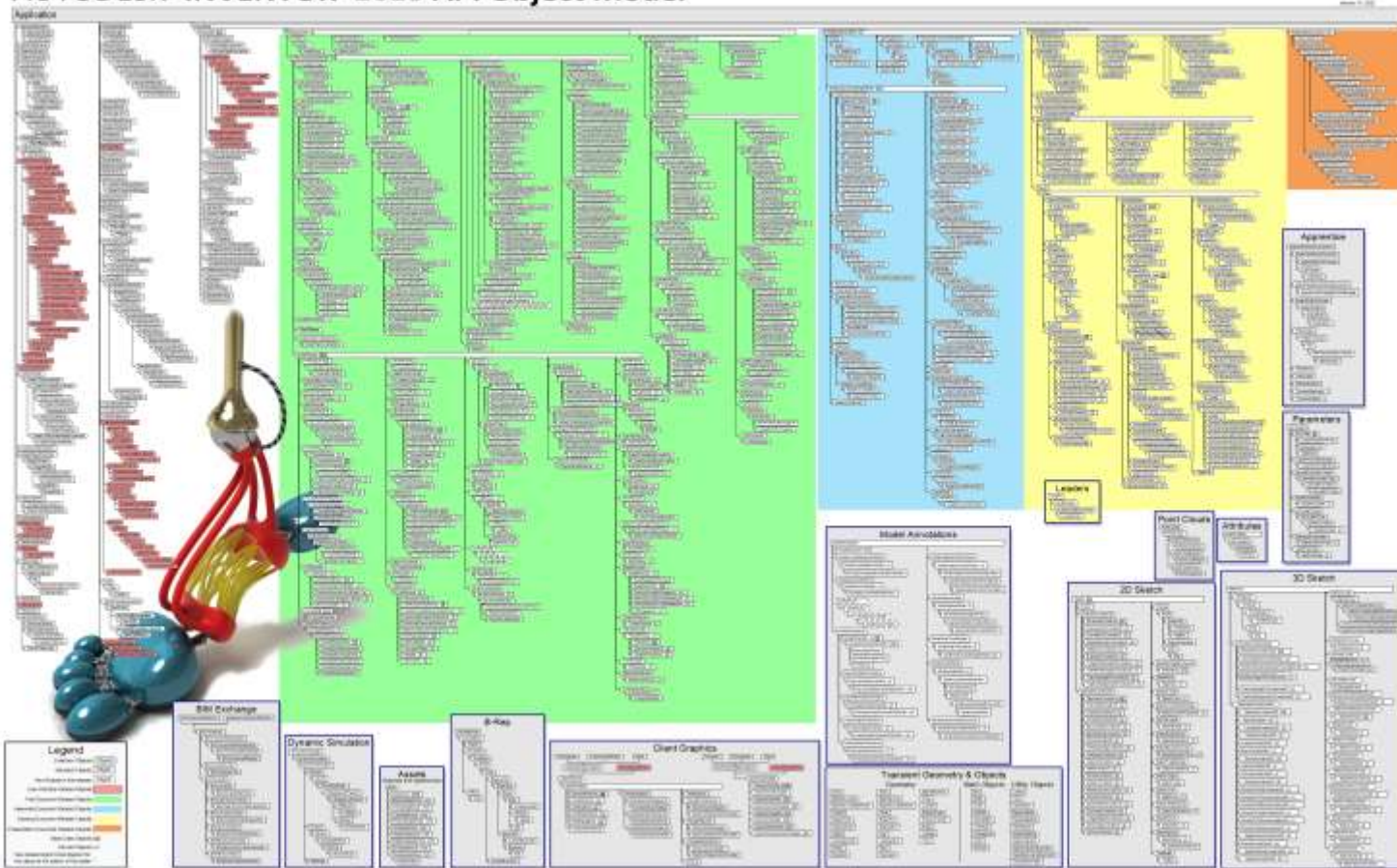
oProp.Value = "NG-4578-01"
```

Both do the
exact same thing

iLogic version

```
iProperties.Value("Custom", "Rack Number") = "NG-4578-01"
```

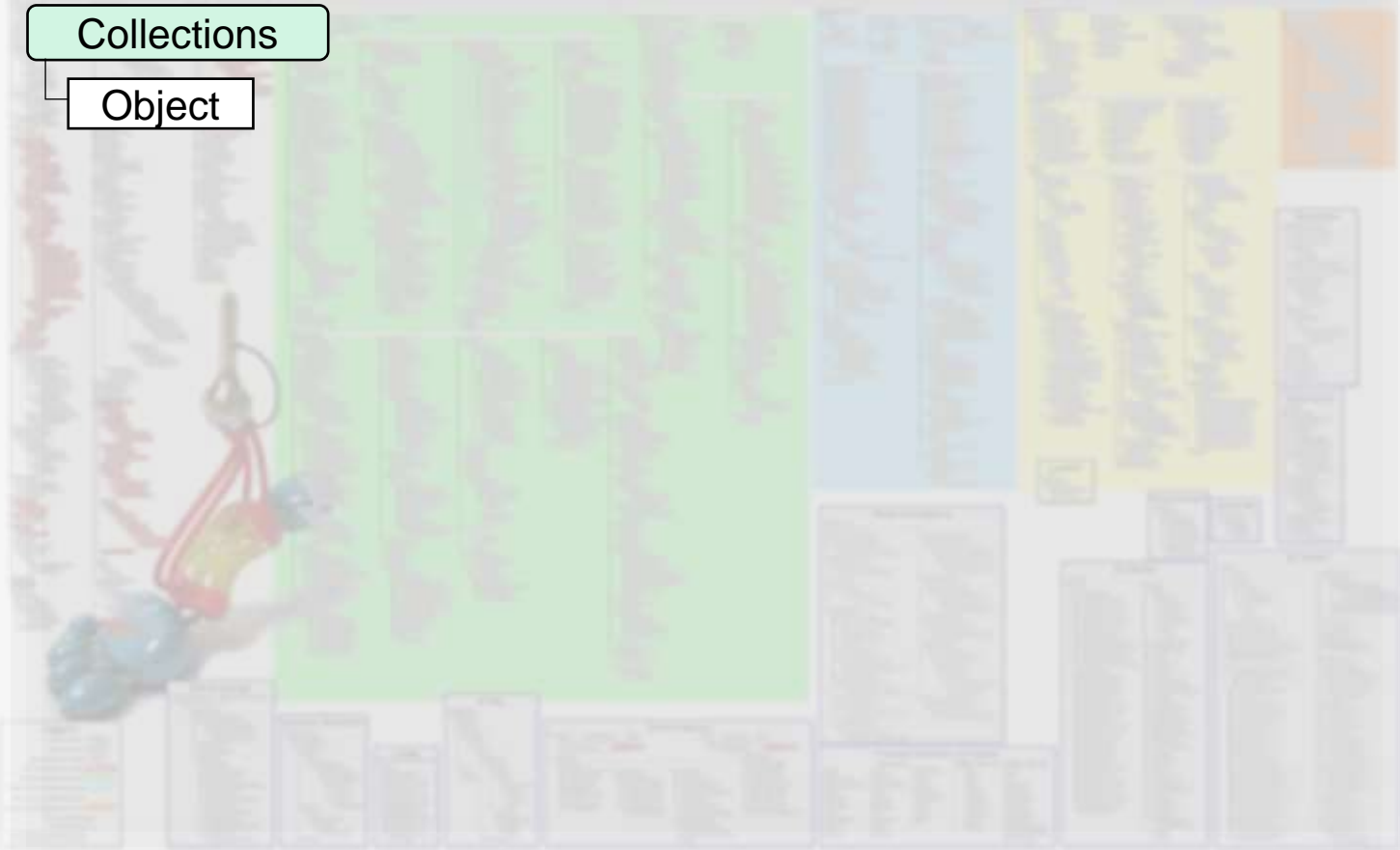
AUTODESK® INVENTOR® 2023 API Object Model

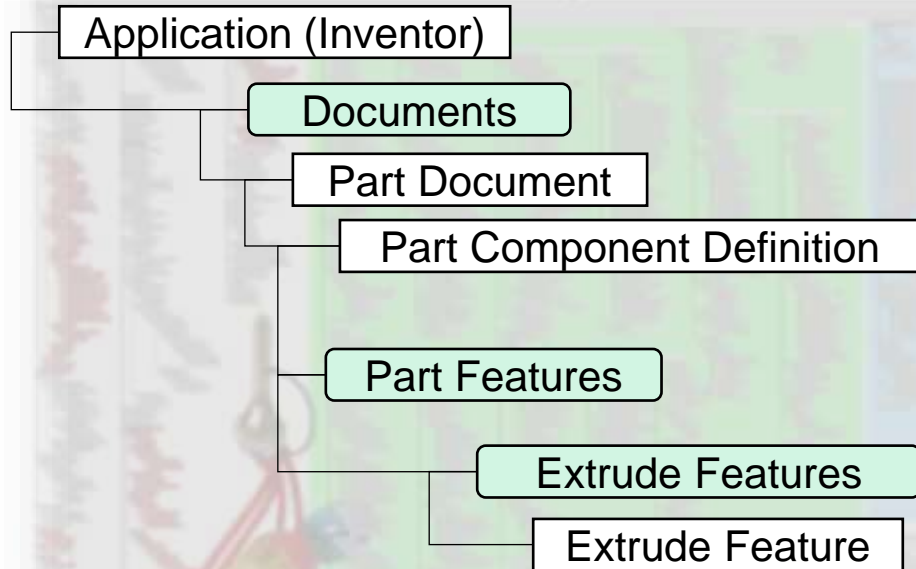


AUTODESK® INVENTOR® 2023 API Object Model

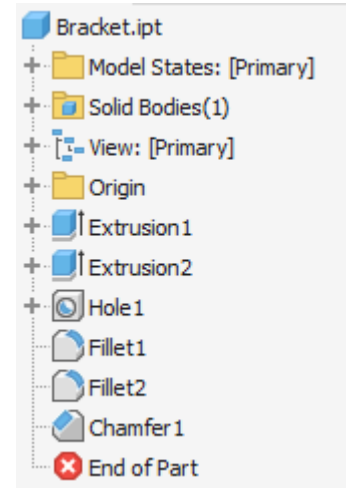
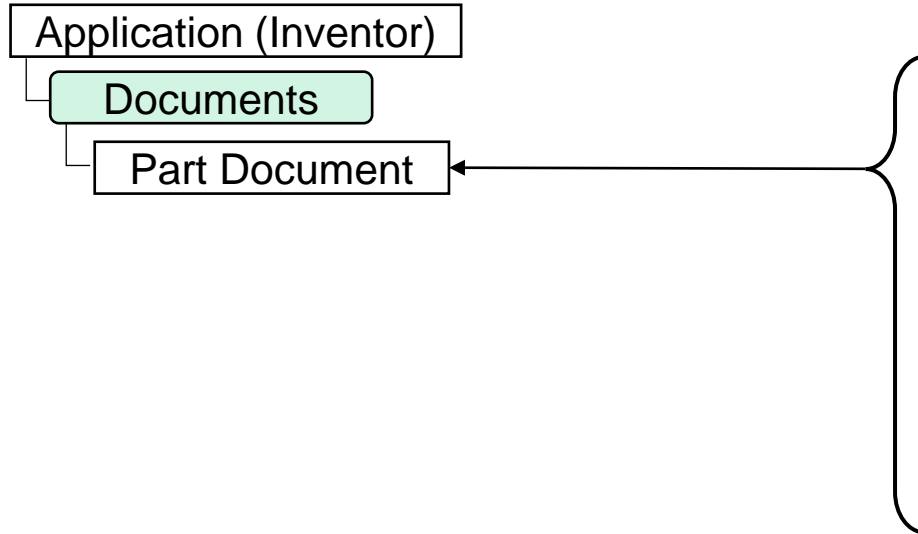
Collections

Object

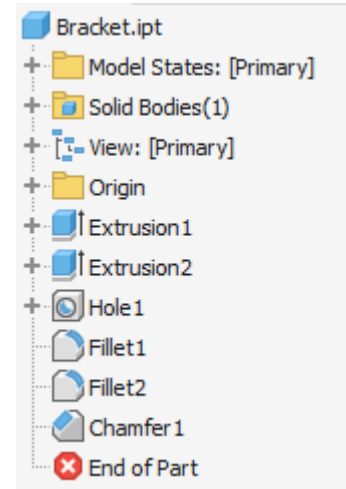
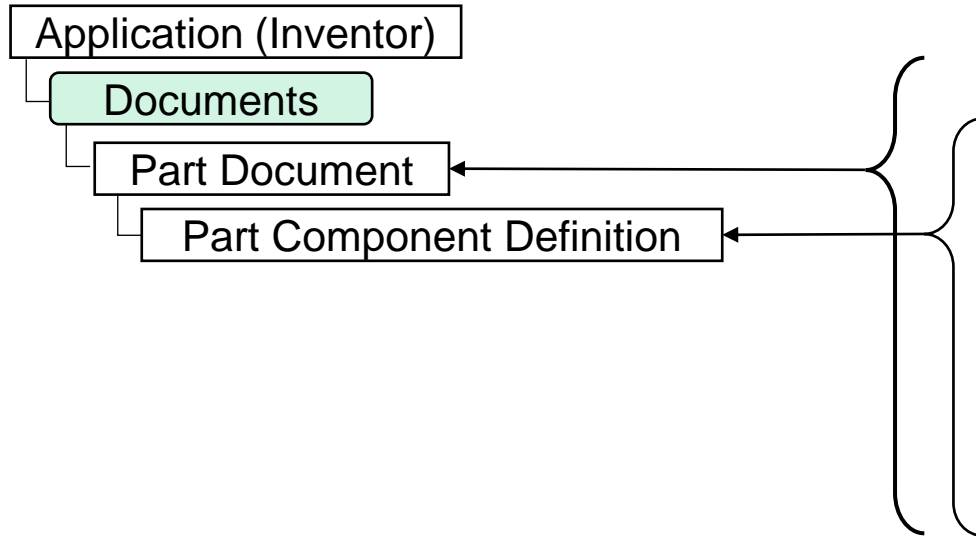




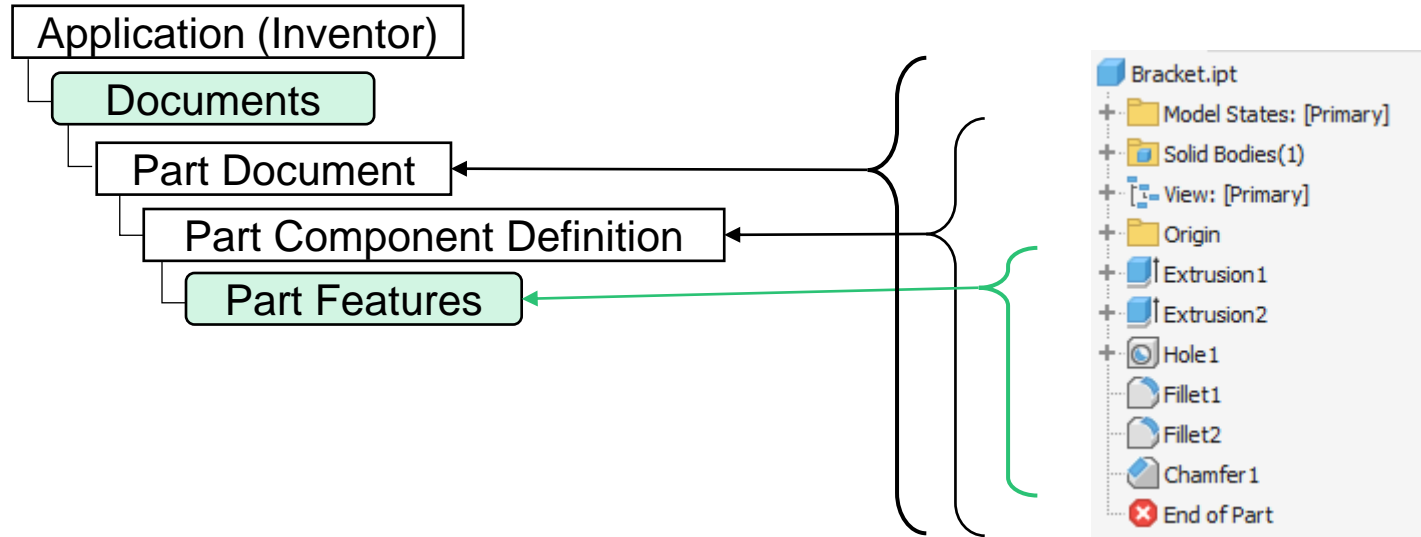
API Object Model of a part



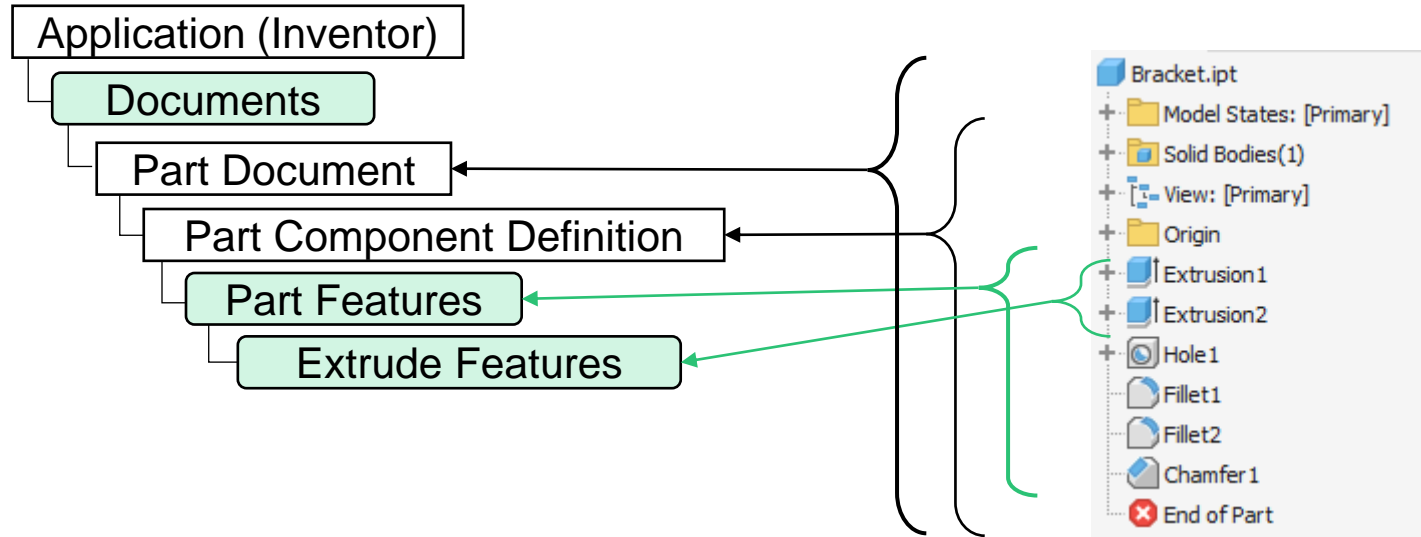
API Object Model of a part



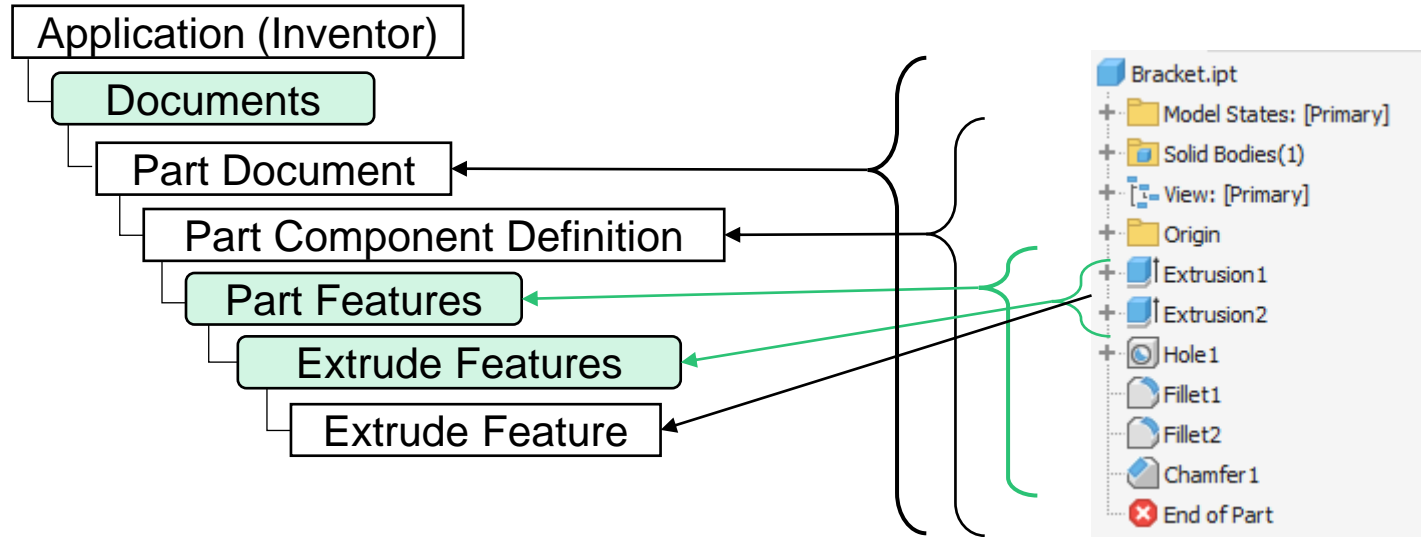
API Object Model of a part



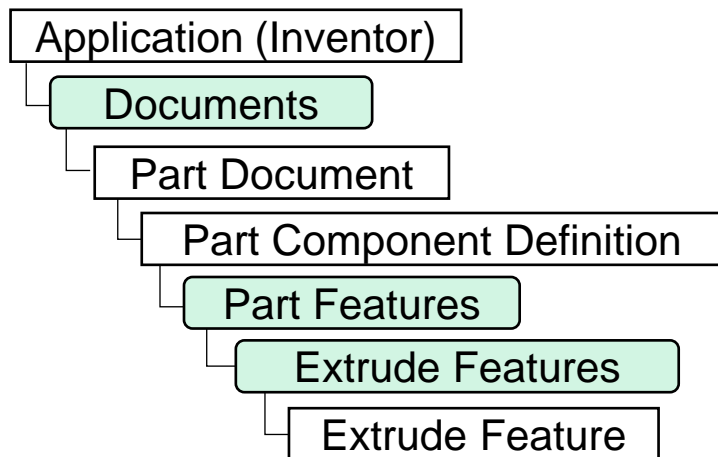
API Object Model of a part



API Object Model of a part

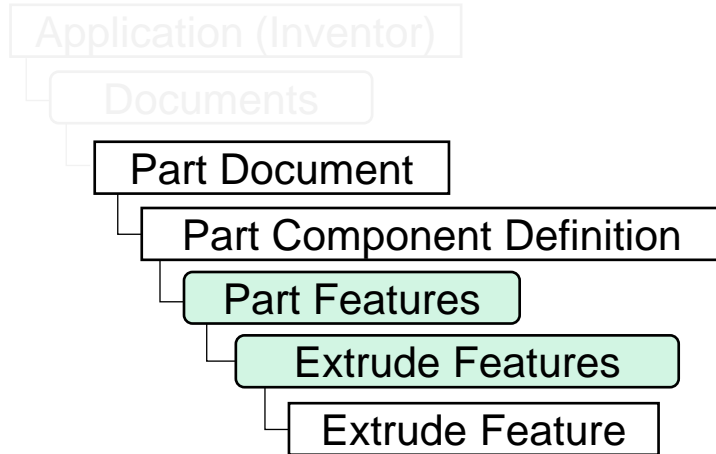


Code Comparison



```
1 Dim oApp As Inventor.Application
2 oApp = ThisApplication
3
4 Dim oDocs As Documents
5 oDocs = oApp.Documents
6
7 Dim oDoc As PartDocument
8 oDoc = oDocs.VisibleDocuments.Item(1)
9
10 Dim oDefinition As ComponentDefinition
11 oDefinition = oDoc.ComponentDefinition
12
13 Dim oFeatures As PartFeatures
14 oFeatures = oDefinition.Features
15
16 Dim oExtrusions As ExtrudeFeatures
17 oExtrusions = oFeatures.ExtrudeFeatures
18
19 Dim oExtrude1 As ExtrudeFeature
20 oExtrude1 = oExtrusions.Item("Extrusion1")
```

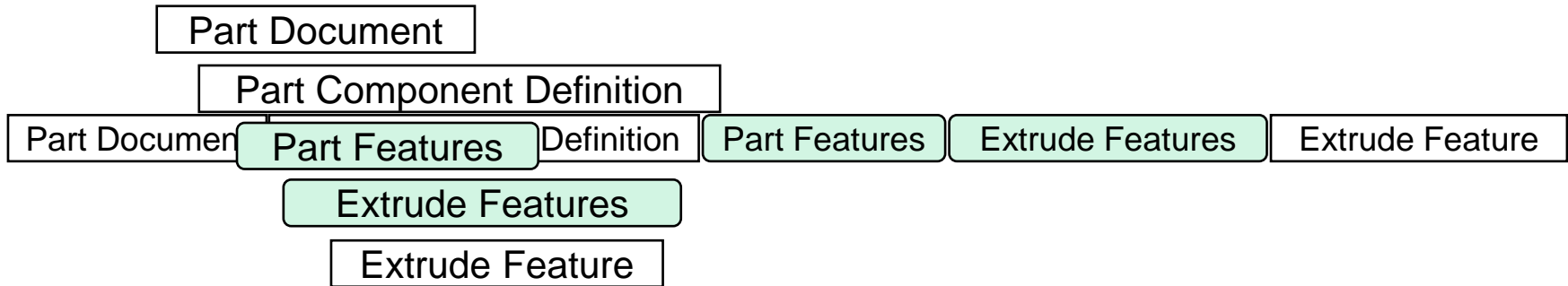
Using iLogic's **ThisDoc.Document** to get the document the rule is triggered from



```
1 Dim oDoc As PartDocument
2 oDoc = ThisDoc.Document
3
4 Dim oDefinition As ComponentDefinition
5 oDefinition = oDoc.ComponentDefinition
6
7 Dim oFeatures As PartFeatures
8 oFeatures = oDefinition.Features
9
10 Dim oExtrusions As ExtrudeFeatures
11 oExtrusions = oFeatures.ExtrudeFeatures
12
13 Dim oExtrude1 As ExtrudeFeature
14 oExtrude1 = oExtrusions.Item("Extrusion1")
```

Using a single line to get the named Extrusion

```
oExtrude1 = ThisDoc.Document.ComponentDefinition.Features.ExtrudeFeatures.Item("Extrusion1")
```

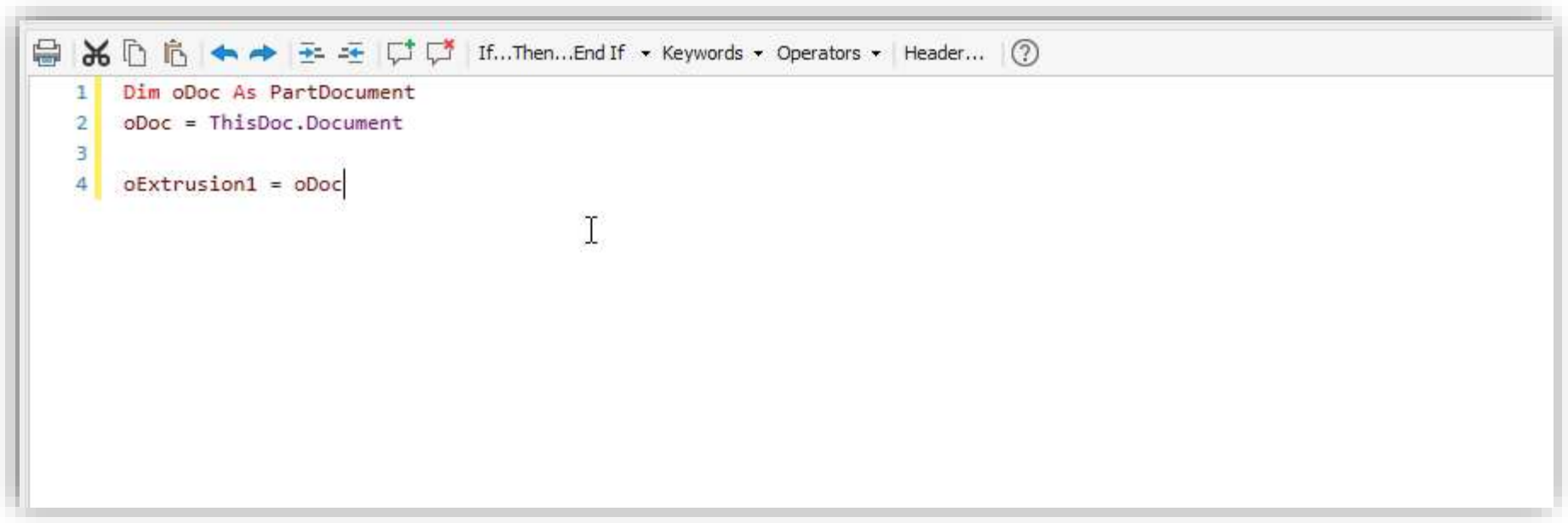




IntelliSense and Declared Variables

Tip #13

Using IntelliSense to navigate the API



Why can't I find the API object I was expecting?



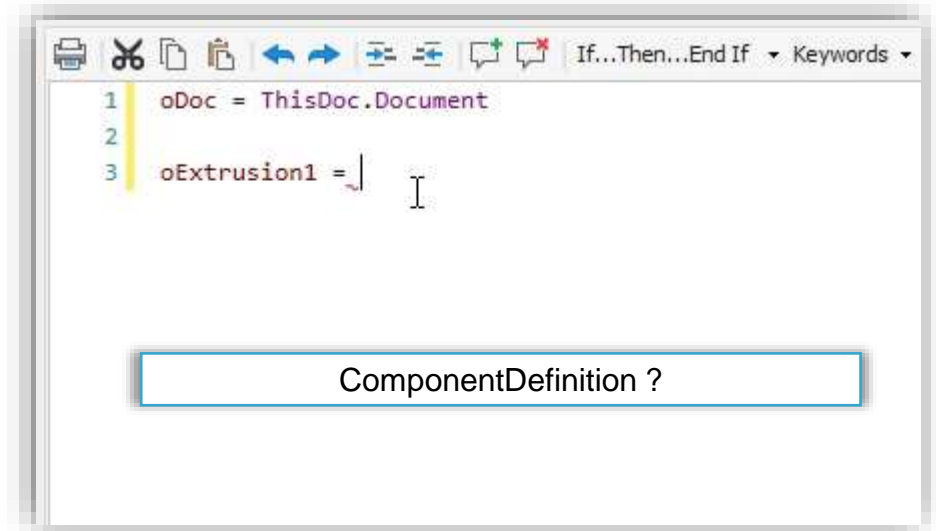
Document

Part Component Definition

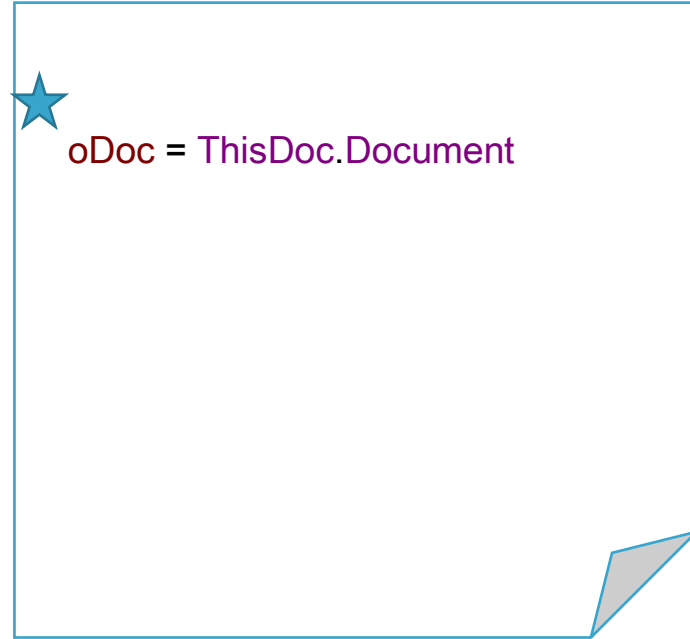
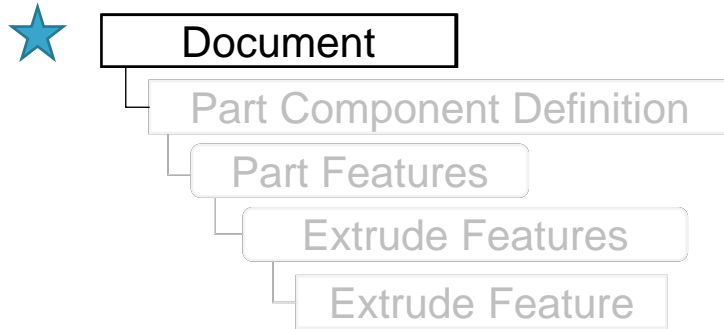
Part Features

Extrude Features

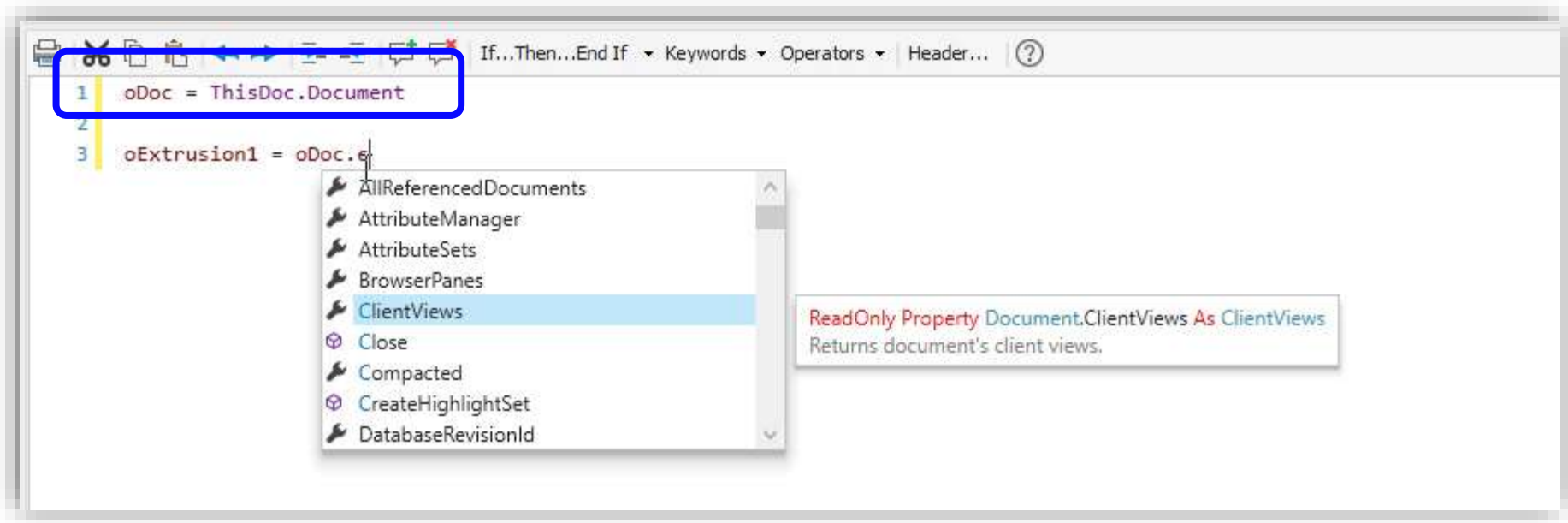
Extrude Feature



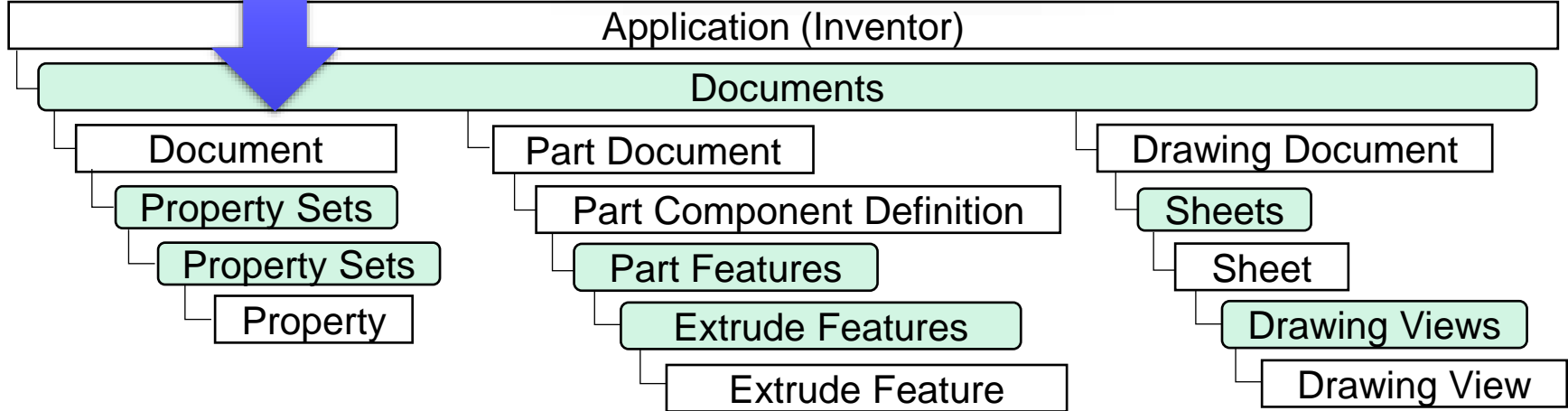
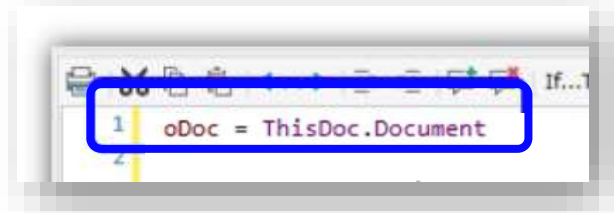
Solution: Declare the variable



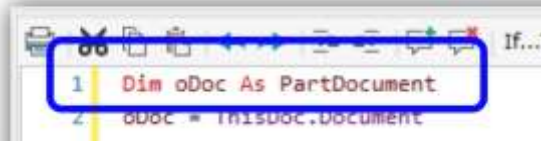
Declaring the Variable



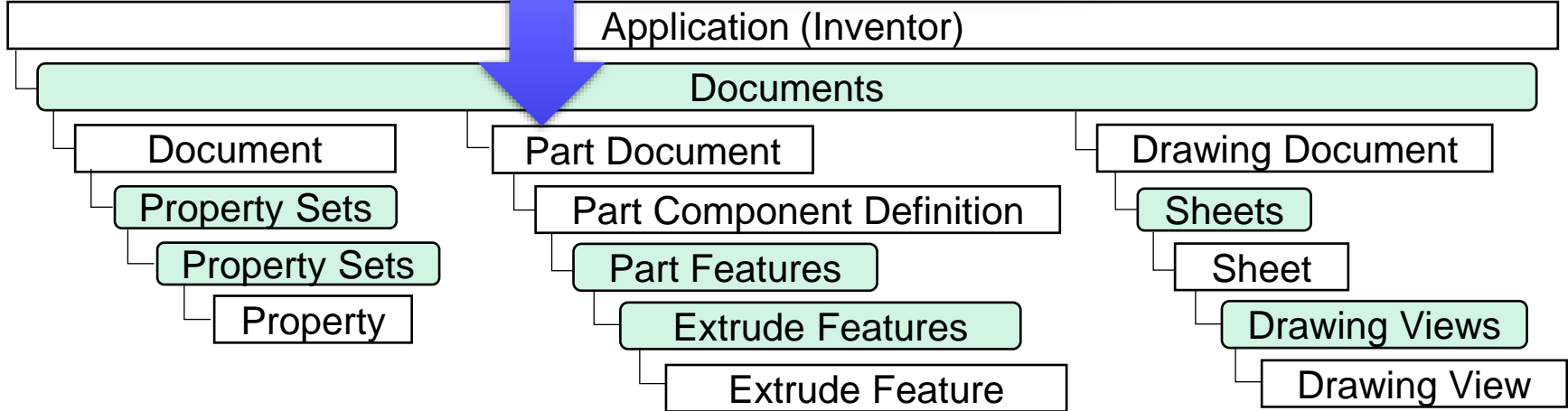
What happens when we declare the variable?

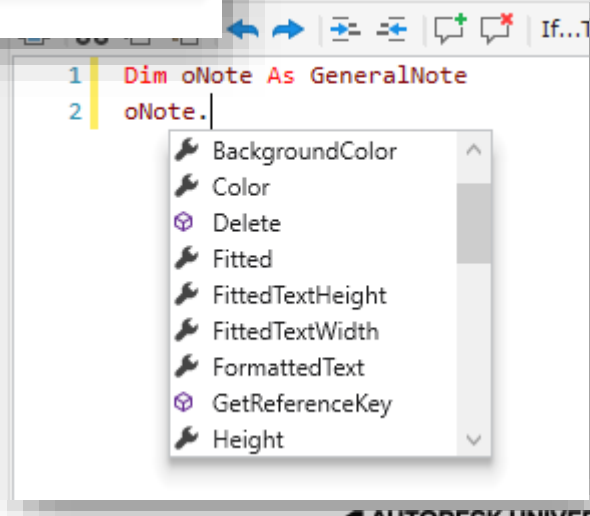
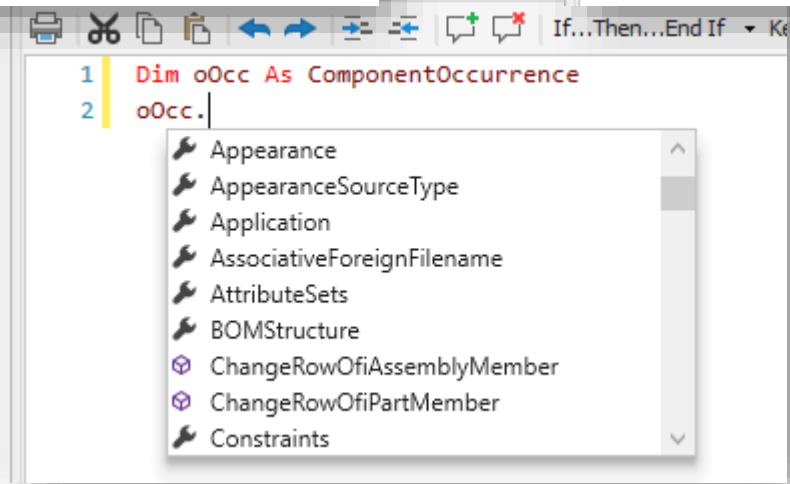
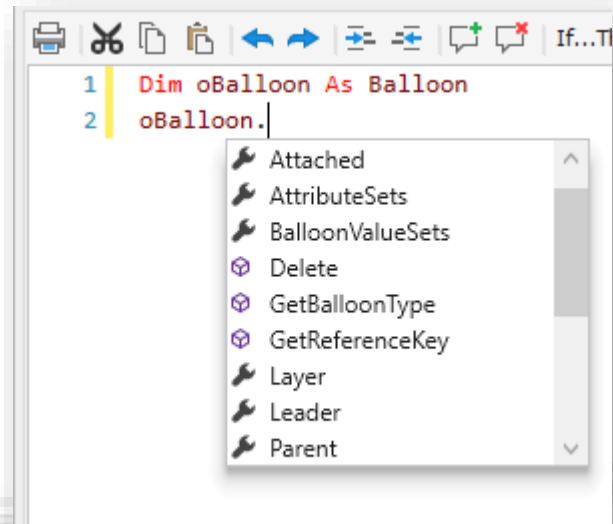
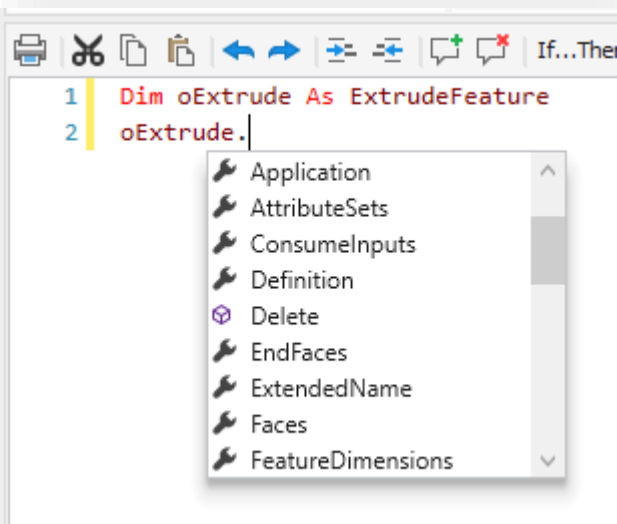


What happens when we declare the variable?

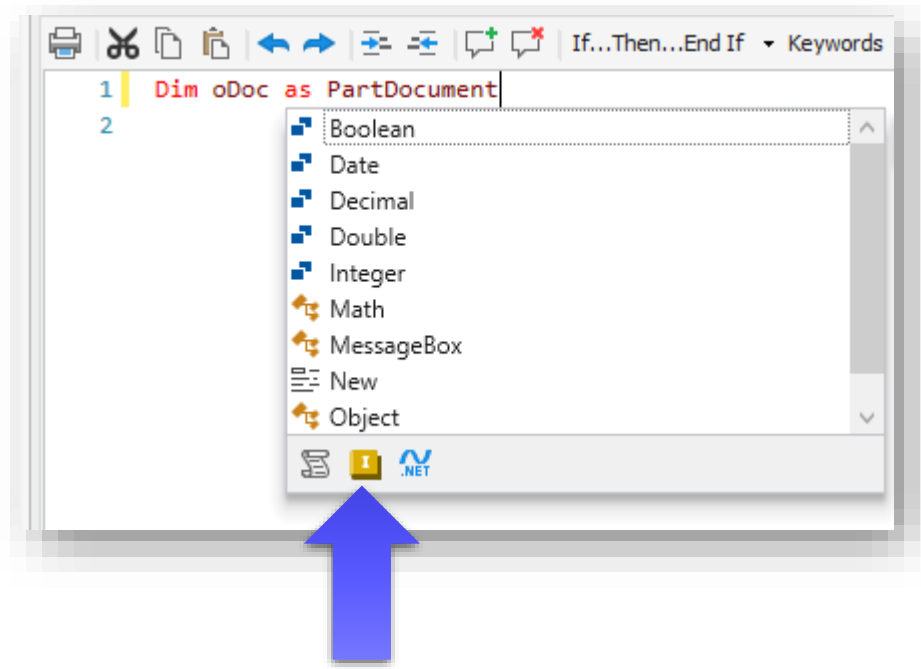


```
1 Dim oDoc As PartDocument
2 oDoc = ThisDoc.Document
```





iLogic Editor IntelliSense Filters





Access API and Programming Help Files

Tip #14

Autodesk Inventor 2023 API Help

HideLocateBackForwardPrintOptions

ContentsIndexSearch

Using the Inventor Software Development Kit

What's New in the Inventor API

Inventor API User's Manual

Inventor API Reference Manual

Sample Programs

Introduction to Using Inventor's Programming Interface

There are several resources provided to help you use Inventor's Application Programming Interface (API). These resources are all part of Inventor's Software Development Kit (SDK). The various elements of the SDK and some additional external resources are described below.

API Help

The API Help is installed with Inventor and is accessed from the Help menu as shown below.

Help

Onboarding Tutorials

Learn More

Connect

Train Web

Show Get Started Tab

Desktop Analytics

About Autodesk Inventor Professional

Help Topics

Programming/API Help

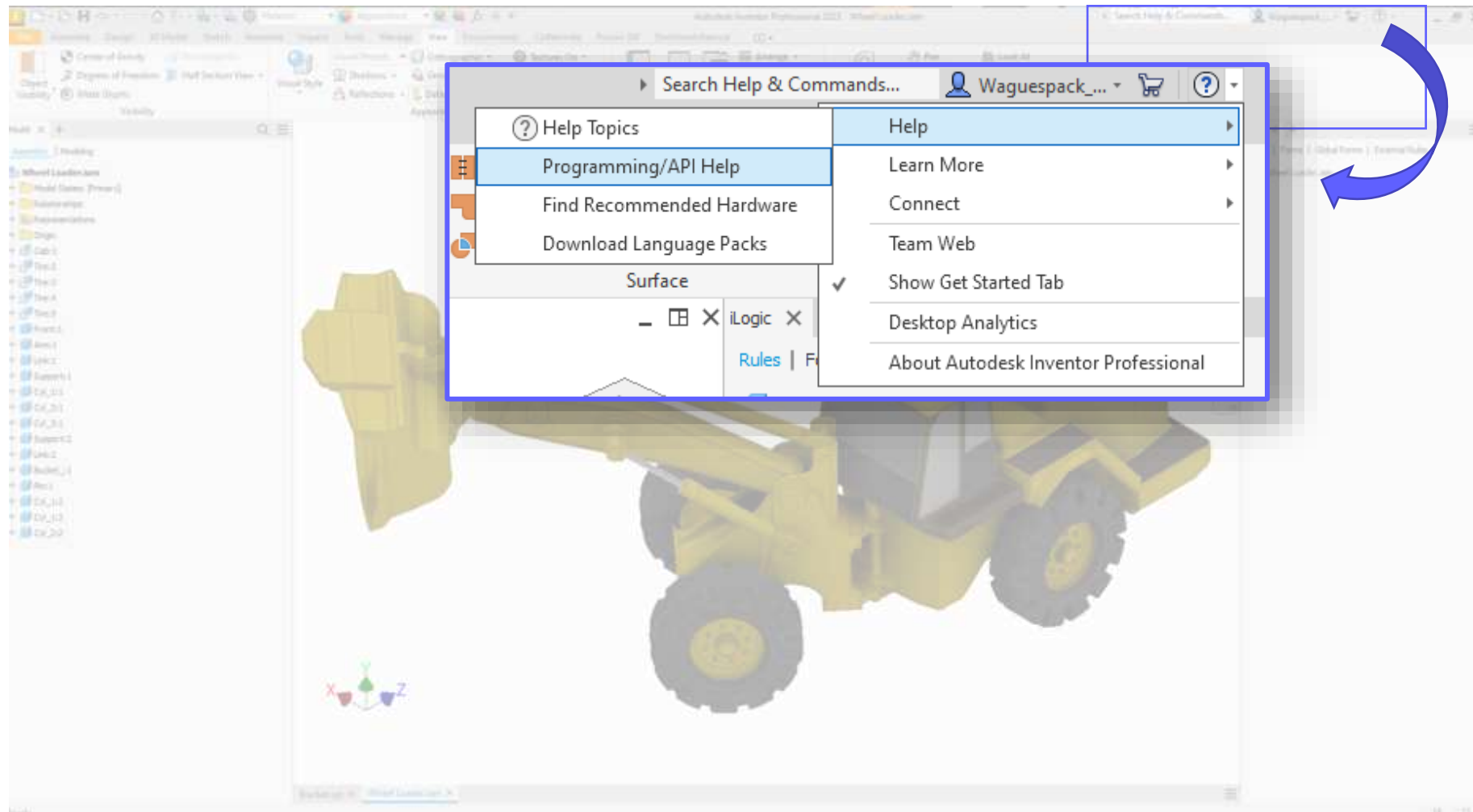
Find Recommended Hardware

Download Language Packs

The help content consists of several parts:

- Introduction to the API, which is what you're reading now.
- What's new in this release of Inventor. This lists the changes that have been made in Inventor that may require some changes to any existing programs and lists the new objects, method, properties, and events that have been added for this release.
- User's manual which provides overview topics of much of the API.
- Reference manual. The reference manual provides detailed information about every object, method, property, and event. If that topic is demonstrated in a sample, there is a link to the sample in that topic.
- Sample programs. This is a categorized list of the sample programs. These are the same samples that are also accessed through links in the reference manual topics. They are primarily VBA programs, with a few C# programs. The API is the same regardless of what language is used. It's the syntax that changes, so a program in any language can serve as an example of how to use the API.

SDK Folder



Objective:

Explore development and error handling techniques

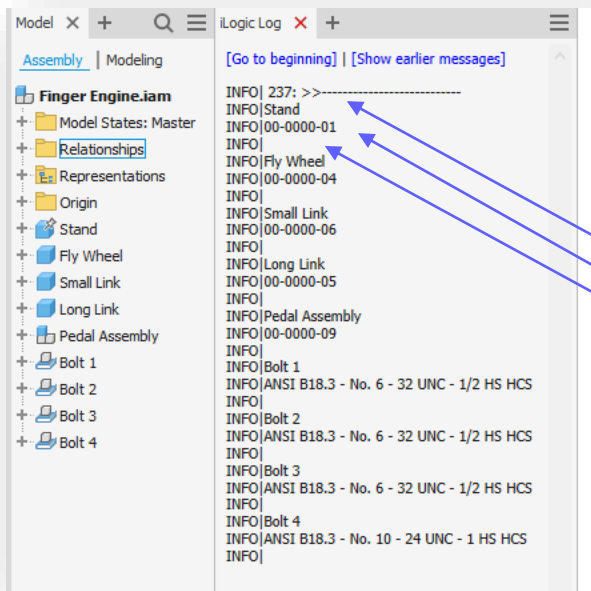
A close-up, black and white photograph of a woven mesh texture, possibly a net or a filter, with a diagonal line of light and shadow running across it. The texture is composed of many small, rounded, interconnected elements.

Use the iLogic Logger

Tip #15

What is the iLogic Logger?

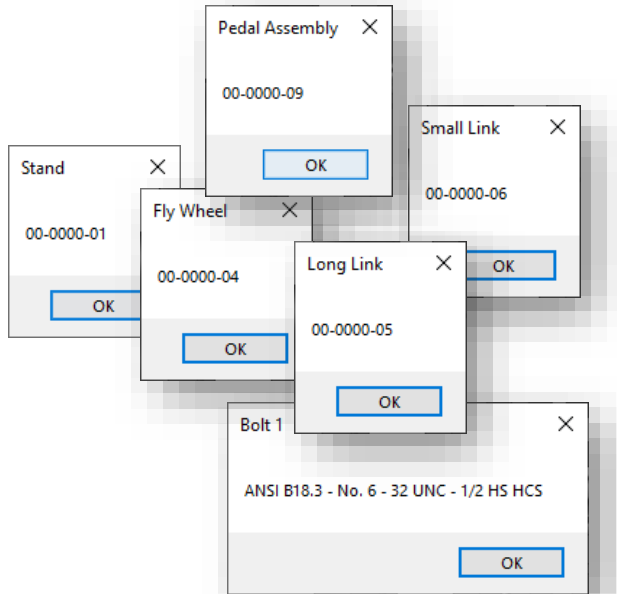
- The iLogic Logger is a tool that we can use to view or report information from our iLogic rules
- This could be model information, errors, etc
- In this example a rule writes out the occurrence name and part number to the logger



```
1 Dim oDoc As AssemblyDocument
2 oDoc = ThisDoc.Document
3
4 Dim oOcc As ComponentOccurrence
5 For Each oOcc In oDoc.ComponentDefinition.Occurrences
6
7     oPn = iProperties.Value(oOcc.Name, "Project", "Part Number")
8
9     Logger.Info(oOcc.Name)
10    Logger.Info(oPn)
11    Logger.Info("")
12
13 Next
```

Works similar to how we someone use message boxes?

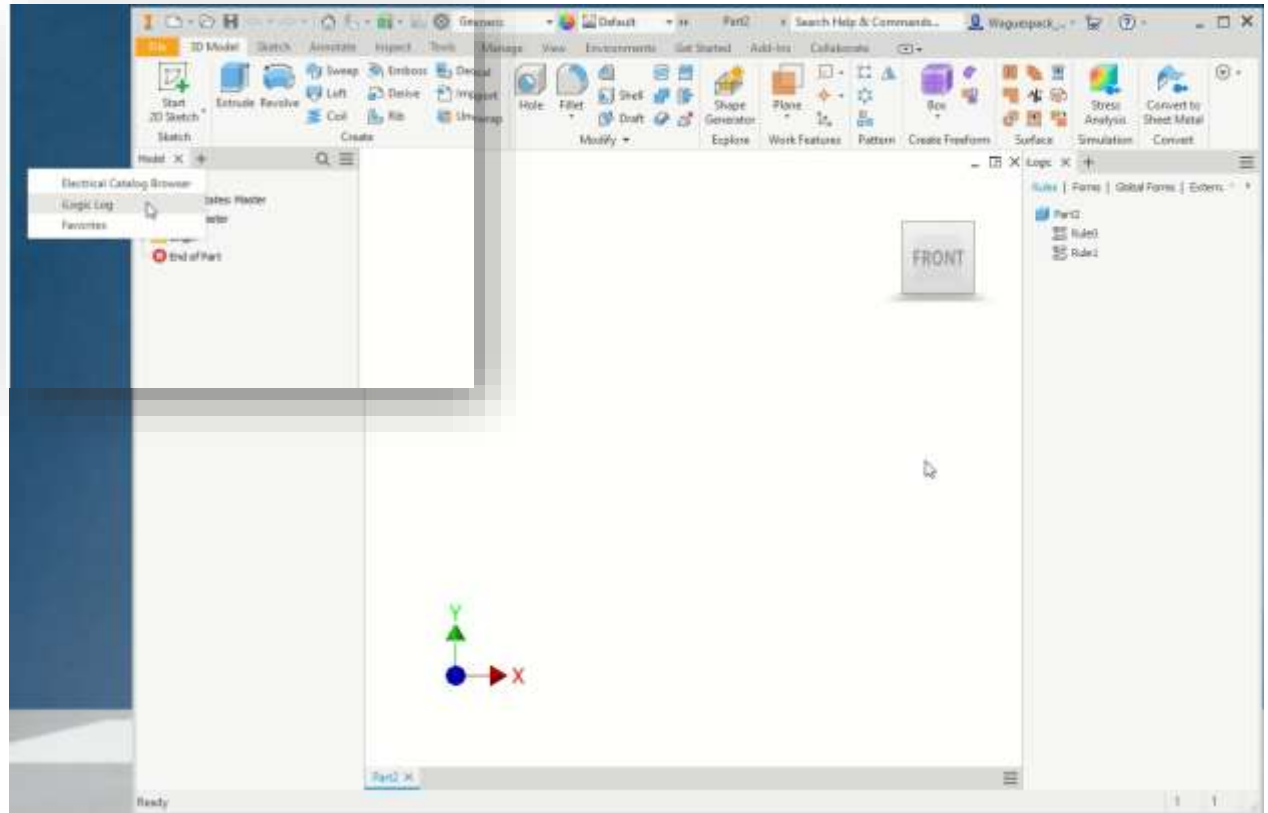
- The iLogic Logger is often a more efficient, less disruptive way to report and view information, that traditional message box sleuthing



```
1 Dim oDoc As AssemblyDocument
2 oDoc = ThisDoc.Document
3
4 Dim oOcc As ComponentOccurrence
5 For Each oOcc In oDoc.ComponentDefinition.Occurrences
6
7     oPn = iProperties.Value(oOcc.Name, "Project", "Part Number")
8     MessageBox.Show(oPn, oOcc.Name)
9
10 Next
```

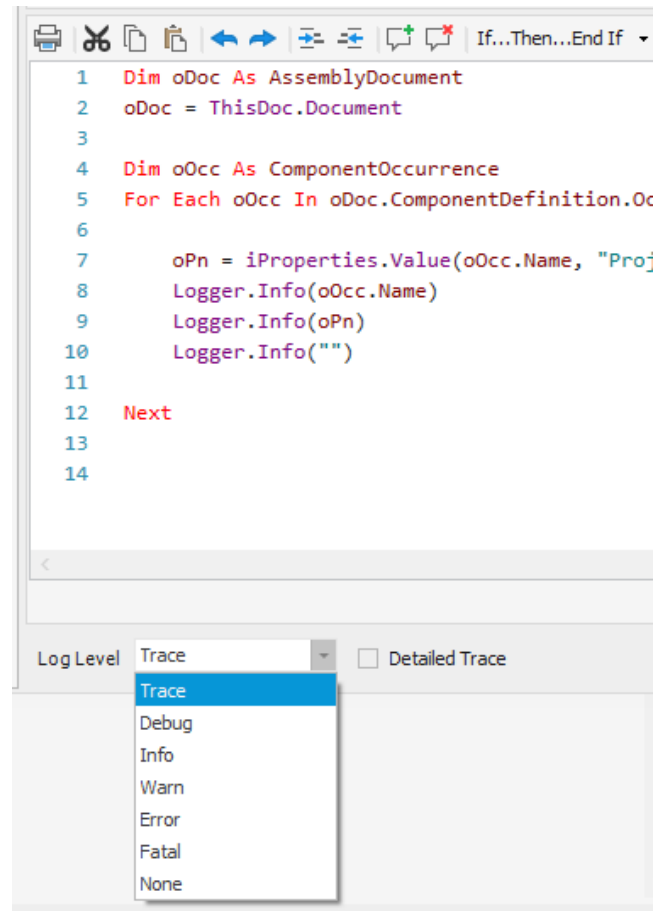
Accessing the iLogic Logger

- We access the iLogic logger using the + next the standard Model browser
- The Logger can be docked to a location and position of your liking



Logger Levels

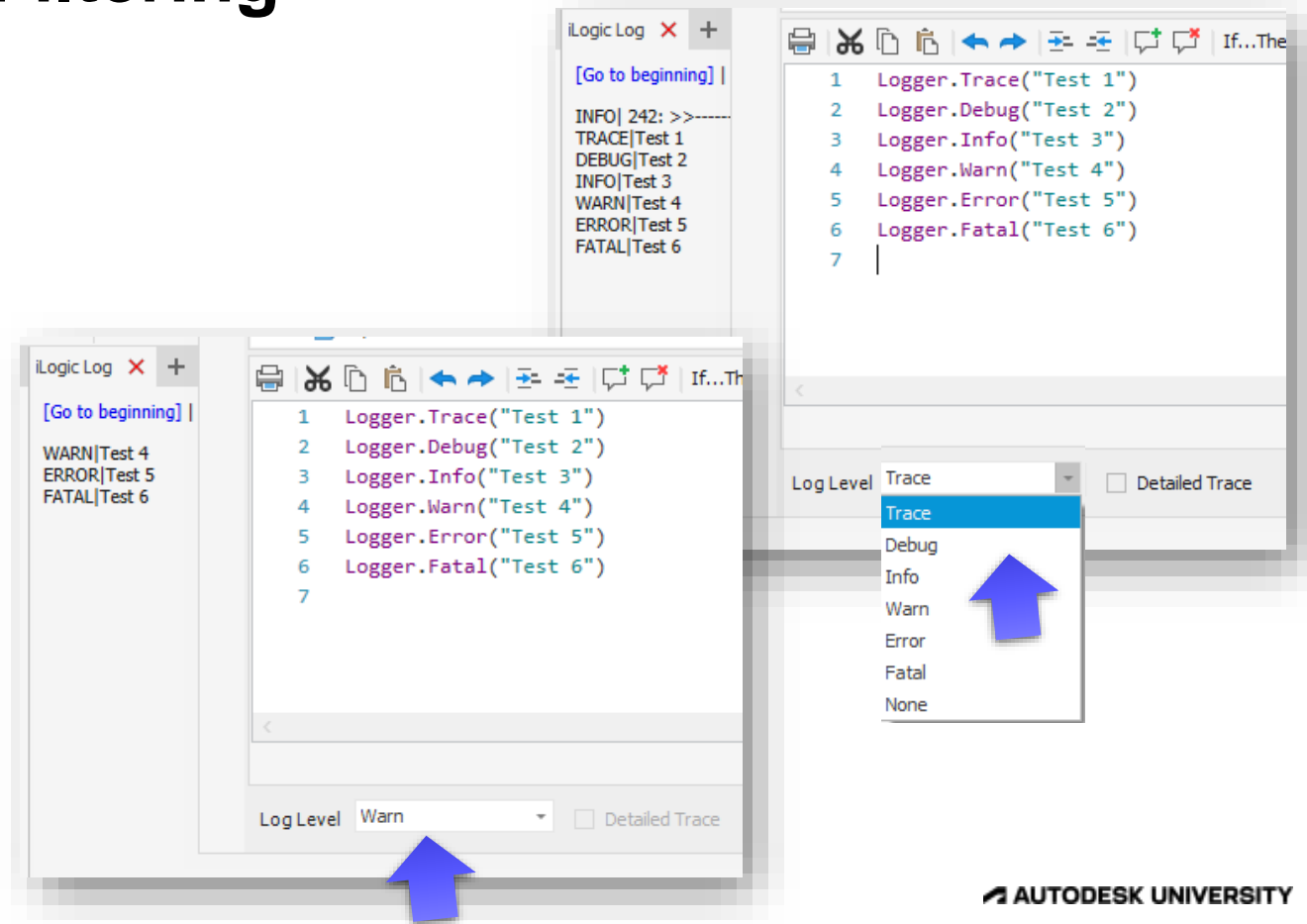
- The logger offers multiple levels of logging accessed via the Log Level drop down in the iLogic code editor



Logger Level Filtering

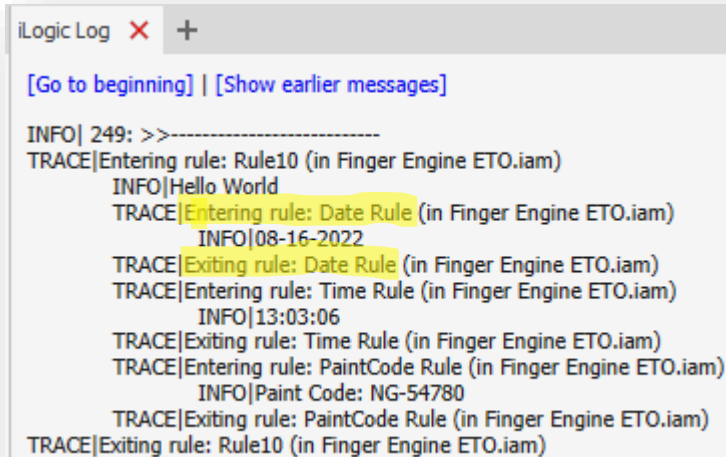
- The levels are cascading in nature, meaning that if you set the Log Level to the highest level (Trace) you will see all other levels.



- If you were to set it to a lower level (such as **Warn**) you would only see levels at or below that level



Detailed Trace

- If we turn on Detailed Trace we monitor when the automation steps into an iLogic Rule
- This can be particularly helpful when we have changes to parameters that are triggering rules excessively or unintentionally



iLogic Log  

[\[Go to beginning\]](#) | [\[Show earlier messages\]](#)

INFO| 249: >>-----

TRACE|Entering rule: Rule10 (in Finger Engine ETO.iam)

INFO|Hello World

TRACE|Entering rule: Date Rule (in Finger Engine ETO.iam)

INFO|08-16-2022

TRACE|Exiting rule: Date Rule (in Finger Engine ETO.iam)

TRACE|Entering rule: Time Rule (in Finger Engine ETO.iam)

INFO|13:03:06

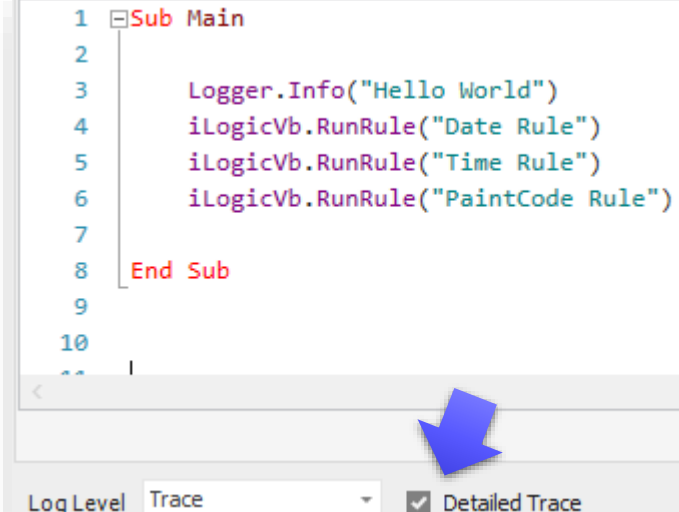
TRACE|Exiting rule: Time Rule (in Finger Engine ETO.iam)

TRACE|Entering rule: PaintCode Rule (in Finger Engine ETO.iam)

INFO|Paint Code: NG-54780

TRACE|Exiting rule: PaintCode Rule (in Finger Engine ETO.iam)

TRACE|Exiting rule: Rule10 (in Finger Engine ETO.iam)



```
1 Sub Main
2
3     Logger.Info("Hello World")
4     iLogicVb.RunRule("Date Rule")
5     iLogicVb.RunRule("Time Rule")
6     iLogicVb.RunRule("PaintCode Rule")
7
8 End Sub
9
10
```

Log Level Trace ☒ Detailed Trace



Write Logger Information to a File

Tip #16

External Logging

- It's often advantageous to have the Logger information written out to an external file
- We can do this by writing out a *.txt file
- Example provided

```
1 Class Thisrule
2   Dim oStartMarker As String
3   Dim oStartTime As DateTime
4   Dim oElapsedTime As TimeSpan
5   Dim oTimeout As Integer
6   Sub Main
7     Call StartLogging 'start logger/timer
8     Dim oDoc As AssemblyDocument
9     oDoc = ThisDoc.Document
10    Dim oOcc As ComponentOccurrence
11    For Each oOcc In oDoc.ComponentDefinition.Occurrences
12      oPn = iProperties.Value(oOcc.Name, "Project", "Part Number")
13      Logger.Info(oOcc.Name)
14      Logger.Info(oPn)
15      Logger.Info("")
16    Next
17    'Finalize
18    Call EndLogging
19  End Sub
20 Sub StartLogging
21   oStartMarker = "iLogic log"
22   Logger.Info(oStartMarker)
23   Logger.Info(DateTime.Now)
24   oStartTime = Now ' Start the timer from this point
25 End Sub
26 Sub EndLogging
27   ' Stop the timer
28   oElapsedTime = Now().Subtract(oStartTime)
29   oTime = oElapsedTime.TotalSeconds
30   oTime = Round(oTime, 2)
31   oLine1 = "iLogic Complete! "
32   oLine2 = "Total Run Time : " & oTime & " seconds "
33   oTimeout = 3.5 ' seconds
34   Logger.Info(oLine1)
35   Logger.Info(oLine2)
36   'write out the log file
37   Dim oLogFileName = "C:\Temp\iLogic log.txt"
38   iLogicVB.Automation.LogControl.SaveLogAs(oLogFileName)
39   Dim oLogText = System.IO.File.ReadAllText(oLogFileName)
40   Dim oMarkerIndex = oLogText.LastIndexOf(oStartMarker)
41   ' Replace the file contents with the portion from the last marker
42   System.IO.File.WriteAllText(oLogFileName, oLogText.Substring(oMarkerIndex))
43   ThisDoc.Launch(oLogFileName)
44 End Sub
45 End Class
```

Logic log.txt - Notepad

File Edit Format View Help

iLogic Log

INFO|8/16/2022 1:15:58 PM

INFO|Stand

INFO|00-0000-01

INFO|

INFO|Fly Wheel

INFO|00-0000-04

INFO|

INFO|Small Link

INFO|00-0000-05

Logic log.txt - Notepad

File Edit Format View Help

iLogic Log

INFO|8/16/2022 1:27:42 PM

INFO|Stand

INFO|00-0000-01

INFO|

INFO|Fly Wheel

INFO|00-0000-04

INFO|

INFO|Small Link

INFO|00-0000-06

INFO|

INFO|Long Link

INFO|00-0000-05

INFO|

INFO|Pedal Assembly

INFO|00-0000-09

INFO|

INFO|Bolt 1

INFO|ANSI B18.3 - No. 6 - 32 UNF - 1/2 HS HCS

INFO|

INFO|Bolt 2

INFO|ANSI B18.3 - No. 6 - 32 UNF - 1/2 HS HCS

INFO|

INFO|Bolt 3

INFO|ANSI B18.3 - No. 6 - 32 UNF - 1/2 HS HCS

INFO|

INFO|Bolt 4

INFO|ANSI B18.3 - No. 10 - 24 UNF - 1 HS HCS

INFO|

INFO|iLogic Complete!

INFO|Total Run Time : 0.22 seconds

External Logging



- Tip provided by **Mike Deck** of Autodesk

 **MjDeck** als Antwort auf: Curtis_Waguespack

09-24-2019 08:38 PM

 Hi **@Curtis_Waguespack**,
There's no straightforward way to do it. We could add that in a future release.
But here's a hack. Write a marker, and then delete all the text before the last marker.

```
Dim startMarker = "----- Custom iLogic Log marker -----"
Logger.Info(startMarker)

' < do some stuff here >
Logger.Info("some info about the stuff that happened starting at " & DateTime.Now)

' < do more some stuff here >
Logger.Info("some more info about the stuff that just happened...")

'write out the log file
Dim logFileName = "C:\TEMP\iLogic Log Example.txt"
iLogicVb.Automation.LogControl.SaveLogAs(logFileName)

Dim logText = System.IO.File.ReadAllText(logFileName)
Dim markerIndex = logText.LastIndexOf(startMarker)
' Replace the file contents with the portion from the last marker
System.IO.File.WriteAllText(logFileName, logText.Substring(markerIndex))
```

 **AUTODESK**
Mike Deck
Software Developer
Autodesk, Inc.

A close-up photograph of a black, textured surface with a grid of raised, rounded squares, creating a tactile, woven appearance. The image is partially obscured by a diagonal black overlay on the right side.

Inventor iLogic and VB.net Forum

Tip #17

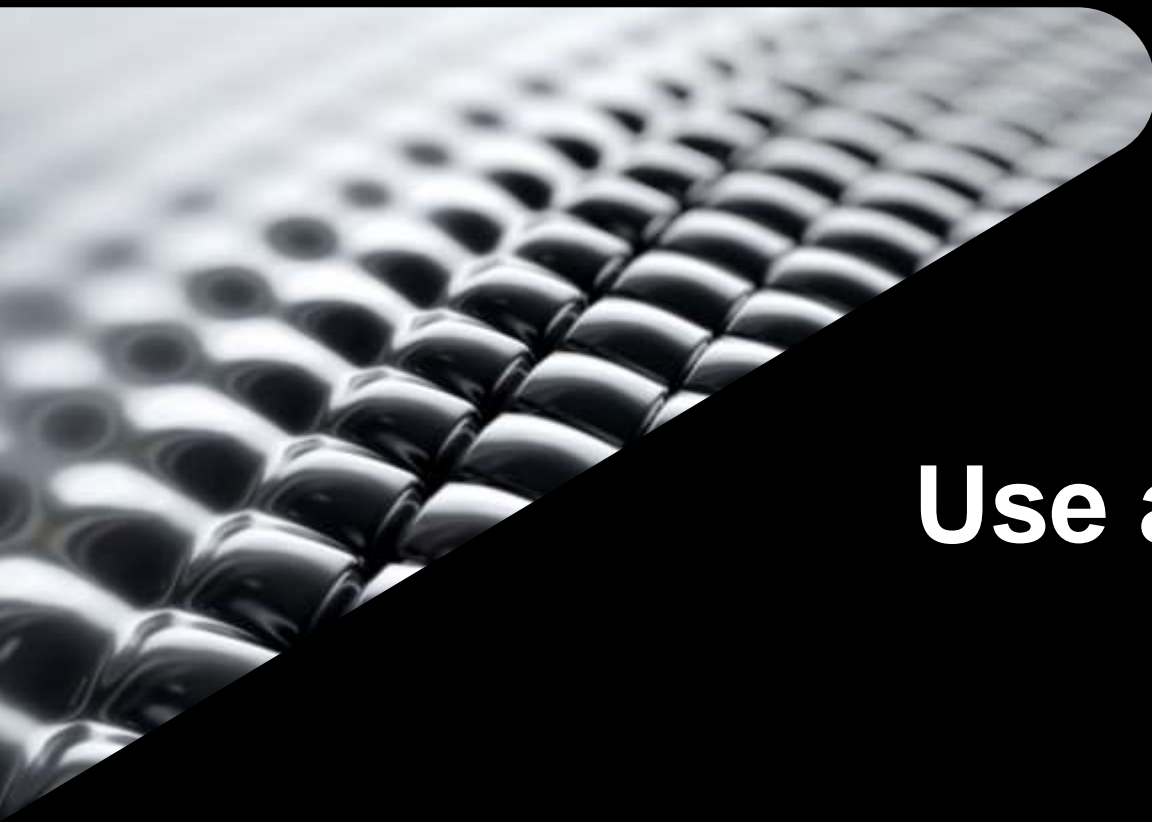
Visit the forum

- Search for solutions
- Ask questions
- Provide answers
- <https://forums.autodesk.com/t5/inventor-ilogic-and-vb-net-forum/bd-p/120>

The screenshot shows the Autodesk Knowledge Network forum page for iLogic and VB.net. The header includes the Autodesk logo and navigation links: Knowledge Network, Produkte, Support, Dazulernen, and Zugang zur Community. The main title is "Inventor iLogic and VB.net Forum" with a subtitle: "Inventor iLogic, API & VBA Forum .Share your knowledge, ask questions, and explore popular Inventor iLogic, API & VBA topics related to programming, creating add-ins, macros, working with the API or creating iLogic tools." Below the title is a search bar with a dropdown menu set to "Dieses Board" and a search button. The page is categorized under "IN FOREN VERÖFFENTLICHEN" with a link to "Zurück zur Kategorie Inventor". Navigation tabs include "Alle Beiträge", "Häufig gestellte Fragen", "Akzeptierte Lösungen", and "Nicht beantwortet". A filter section shows "OPTIONEN" and "FILTER BY LABELS". The main content area displays a list of forum posts with their titles, authors, dates, and response statistics. The posts are as follows:

Post Title	Author	Date	Answers	Views
Autodesk University 2022 – Let Us Know if You'll Be There!	von CGBenner	am 08-16-2022 11:16 AM	0	9
Accepted Solutions – What They Are, How to Find Them, and How to Accept a Solution	von CGBenner	am 05-20-2022 06:11 AM	0	182
Important for developers working with Level Of Details	von adam.nagy	am 01-22-2021 04:11 AM Zuletzt veröffentlicht am 08-23-2021 08:19 AM von Cattabiani	5	2467
New default for iLogic Excel functions in Inventor 2021	von MJDeck	am 09-10-2020 09:48 AM Zuletzt veröffentlicht am 09-25-2020 07:20 AM von MJDeck	4	3129
Check out the self-paced guide "My First Plug-in"	von wayne.brill, JelteDeJong	am 08-05-2011 01:24 PM Zuletzt veröffentlicht am 04-29-2022 02:21 PM von JelteDeJong	53	20207
Autodesk University 2022 – Let Us Know if You'll Be There!	von CGBenner	am 08-16-2022 11:16 AM	0	9
Mirrored Sheet Metal Parts	von abarilaan	am 08-16-2022 08:47 AM Zuletzt veröffentlicht am 08-16-2022 11:13 AM von abarilaan	4	28
If Statement for if a string is a number	von chainesLH36	am 08-16-2022 08:27 AM Zuletzt veröffentlicht am 08-16-2022 11:02 AM von chainesLH36	4	38

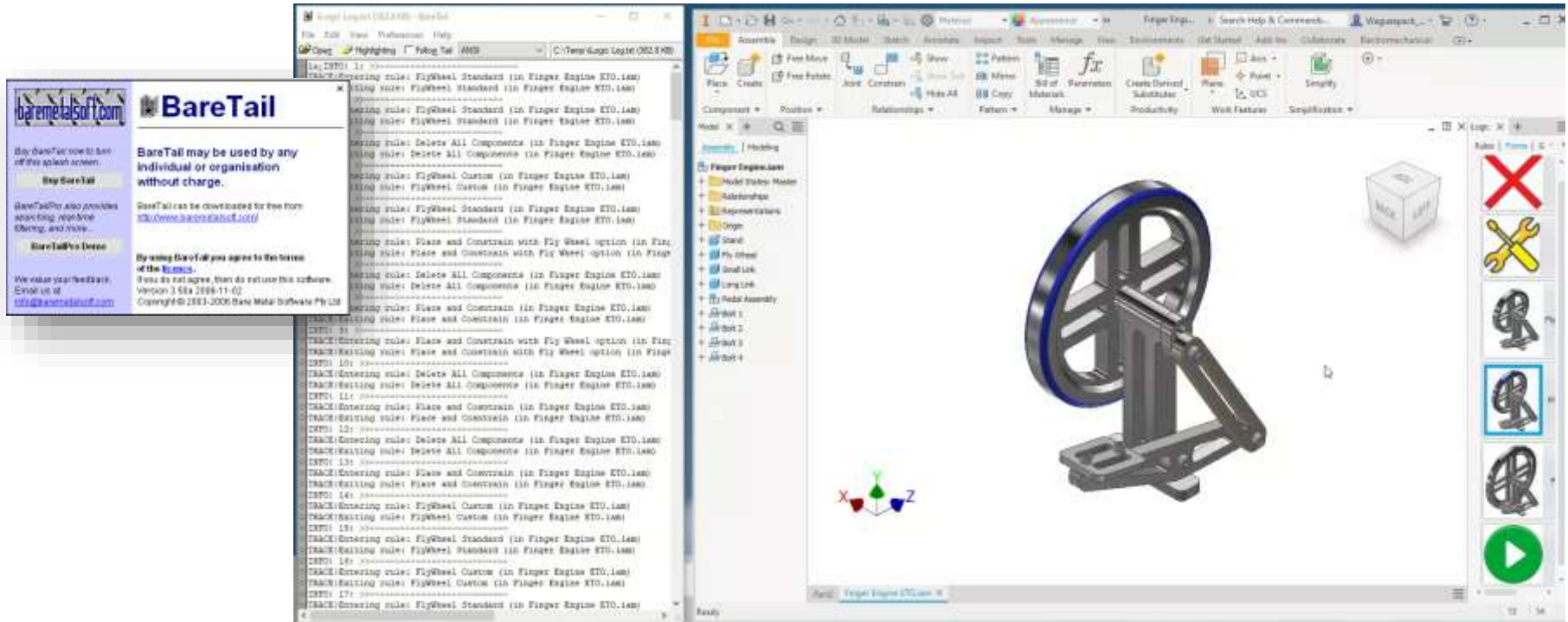
The footer of the page features the "AUTODESK UNIVERSITY" logo.



Use a Log Monitor

Tip #18

Use a log monitor such as BareTail to watch the iLogic log file



- Watch log outputs real time



Use Try Catch Statements with the iLogic Logger to catch Exceptions

Tip #19

Catch Errors/Exceptions and log the error line

- A Try Catch statement allows the rule to continue running
- Here a Parameter named Foo does not exist, and produces an error
- The Try Catch statement catches the Exception and passes it to an error handler which passes the **Error message** and the **Error line** to the iLogic Logger

```
1 Sub main
2
3     Try
4         Parameter("Foo") = 4
5     Catch ex As Exception
6         Call HandleErrors(ex)
7     End Try
8
9 End Sub
10
11 Sub HandleErrors(ex As Exception)
12
13     oStackTrace = ex.StackTrace
14     iPos = InStr(oStackTrace, iLogicVb.RuleName)
15     oErrorRuleLine = Mid(oStackTrace, iPos, Len(oStackTrace) - iPos + 1)
16     Logger.Error("Error at: " & oErrorRuleLine)
17     Logger.Info(ex.Message)
18
19 End Sub
```

iLogic Log X +

[Go to beginning] | [Show earlier messages]

INFO| 367: >>-----

TRACE|Entering rule: Load Calculations (in Part2)

ERROR|Error at: Load Calculations.vb:line 6

INFO|Parameter: Could not find a parameter named: "Foo"

TRACE|Exiting rule: Load Calculations (in Part2)

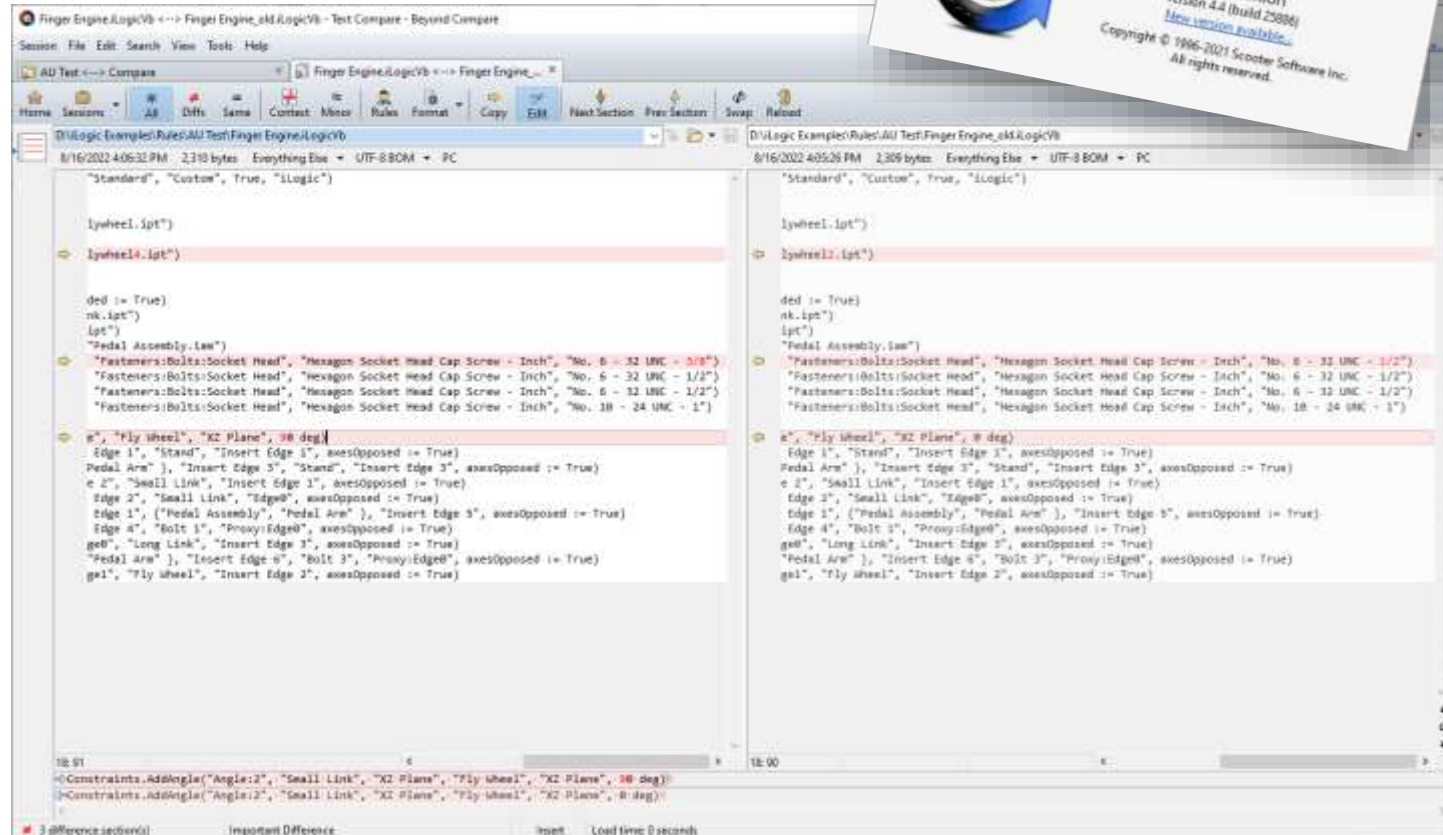


Use a Code Comparison Tool

Tip #20

Code Compare

- Assuming that you are using **External iLogic rules**, and are versioning your changes
- Use a tool such as **Beyond Compare** to track and visualize and merge changes to your code.



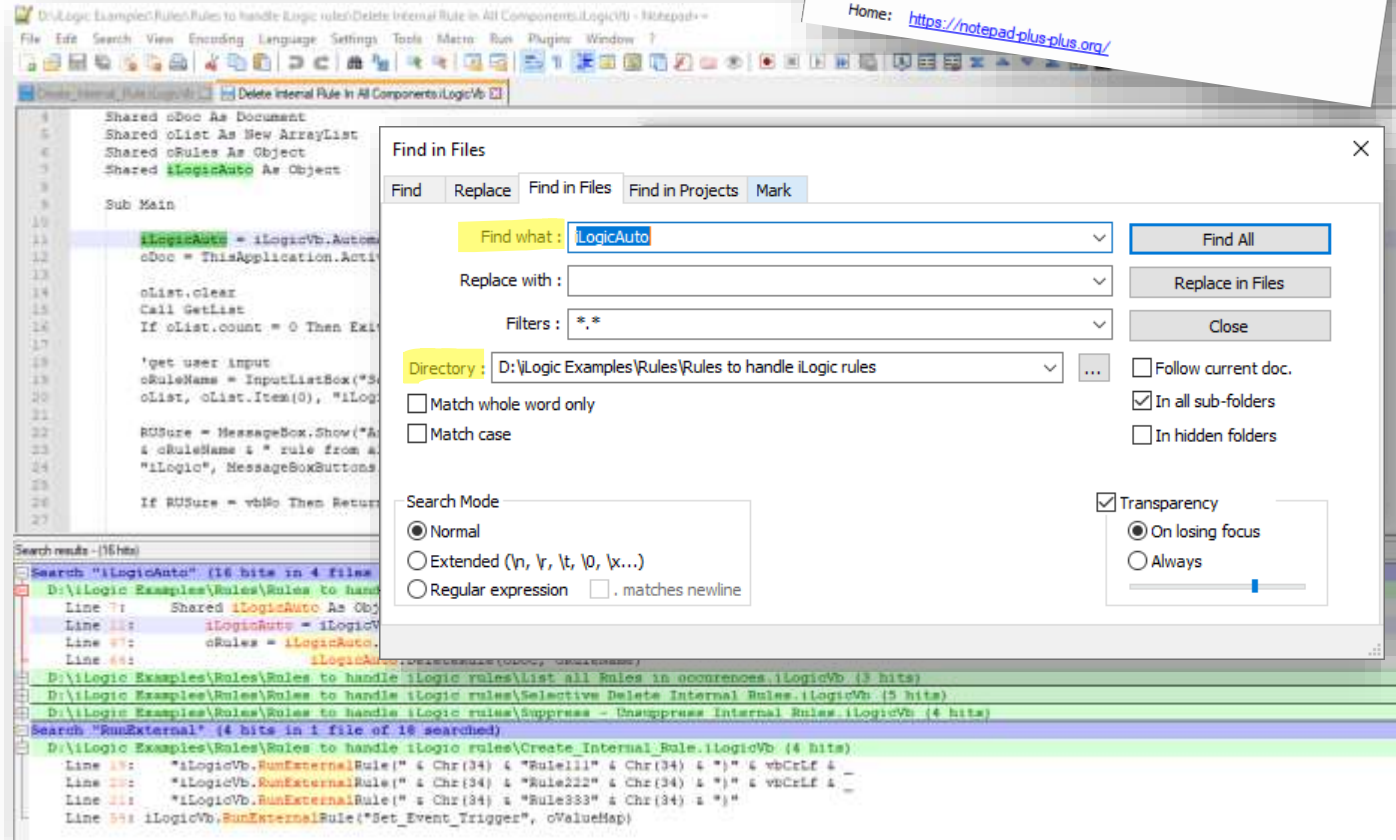


Search Within Your iLogic Rules

Tip #21

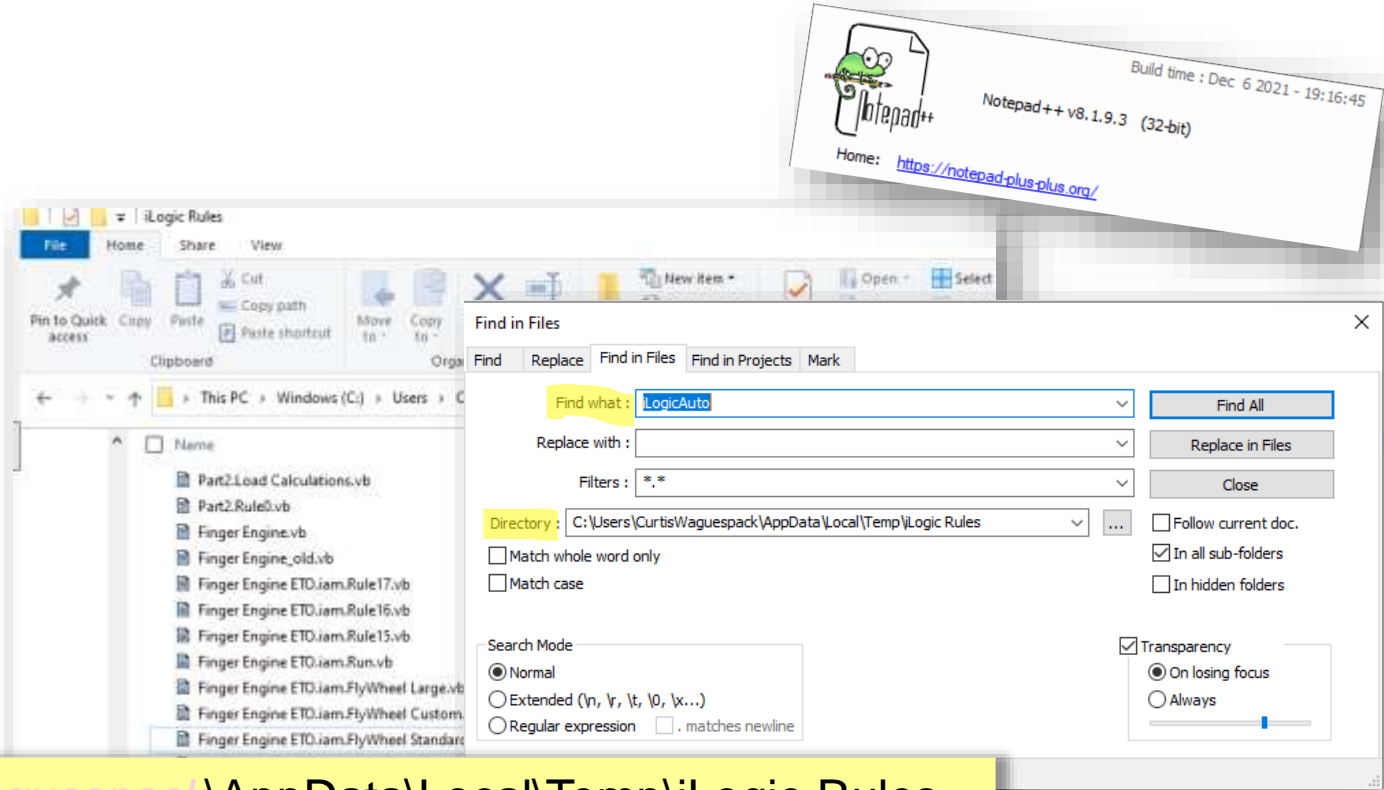
Search in iLogic Rule Files

- Use a tool such as **NotePad++** to search in your iLogic rules
- Easy to use with External rules
- Internal rules are a bit more of a challenge



Search in internal iLogic Rule Files via the Temp folder

- When you edit rules in Inventor, there is a temporary directory that creates copies of these rules as *.vb files
- We can use NotePad++ to search in these files also



C:\Users\CurtisWaguespack\AppData\Local\Temp\iLogic Rules

Objective

Create a Better Interface Between Your iLogic Automation and the Users Who Employ It

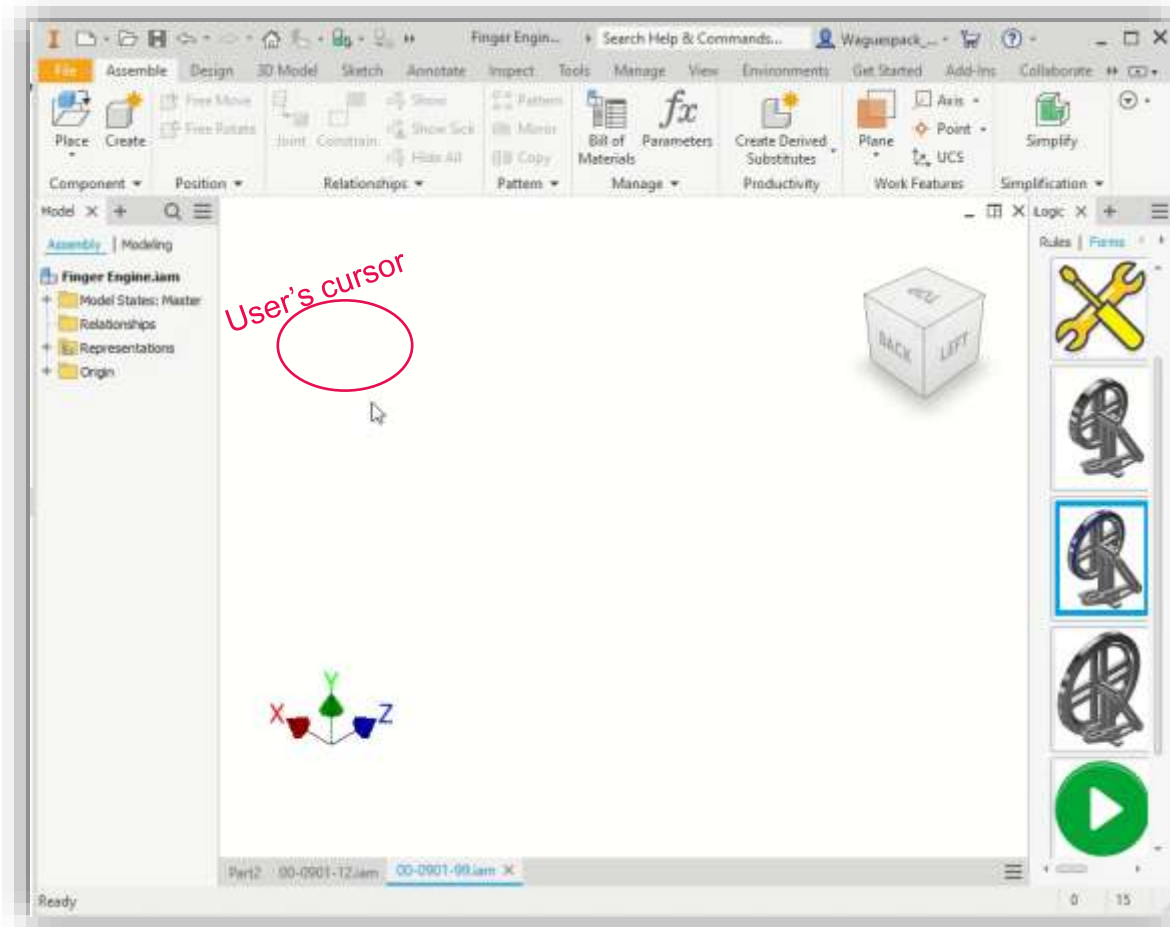


Use Timed Message Boxes

Tip #22

Use a timed message box, between rules

- The message times out, so that it doesn't interrupt the automation, by waiting for a user to click the OK button
- In this example there are 2 rules running.
 - One builds the model then displays a timed message box
 - The second runs another rule, and provided another timed message to the user



Time Message Box

★ Original code provided by **Mike Deck** of Autodesk on the Inventor iLogic and VB.net forum

- This example calls an external rule named “Timed Message”
- It passes the external rule 2 arguments:
 - Message text “Model created successfully...”
 - a “Timeout” value of 2.5 seconds

Calling Rule

```
23
24 InventorVb.DocumentUpdate()
25 ThisApplication.ActiveView.Fit
26
27 Dim oValueMap As Inventor.NameValueMap = ThisApplication.TransientObjects.CreateNameValueMap()
28 oValueMap.Add("Message", "Model created successfully." & vbCrLf & vbCrLf & "Preparing to run motion test rule.")
29 oValueMap.Add("Timeout", "2.5")
30 iLogicVb.RunExternalRule("Timed Message", oValueMap)
31
32 iLogicVb.RunRule("Run")
```

Timed Message rule

```
1 Sub Main
2     'set default
3     oTimeout = 1.5 ' seconds
4     oMsg = "Hello World"
5
6     'get value map argument from other rule
7     If Not RuleArguments("Timeout") = "" Then oTimeout = RuleArguments("Timeout")
8     If Not RuleArguments("Message") = "" Then oMsg = RuleArguments("Message")
9
10    Call TimedMessage(oMsg, oTimeout)
11
12 End Sub
13
14
15 Sub TimedMessage(oMsg As String, oTimeout As Double)
16
17     Dim oForm As New Form() With { .Enabled = True }
18     System.Threading.Tasks.Task.Delay(TimeSpan.FromSeconds(oTimeout)).ContinueWith(Sub(t)
19         oForm.Close()
20     End Sub ,
21     System.Threading.Tasks.TaskScheduler.FromCurrentSynchronizationContext())
22
23     MessageBox.Show(oForm, String.Format(oMsg, oTimeout), "iLogic")
24
25 End Sub
26
```

A close-up, black and white photograph of a woven mesh texture, possibly a screen or fabric, showing a grid of small, rounded openings. The texture is diagonal and occupies the left side of the slide, partially overlapping a black diagonal band.

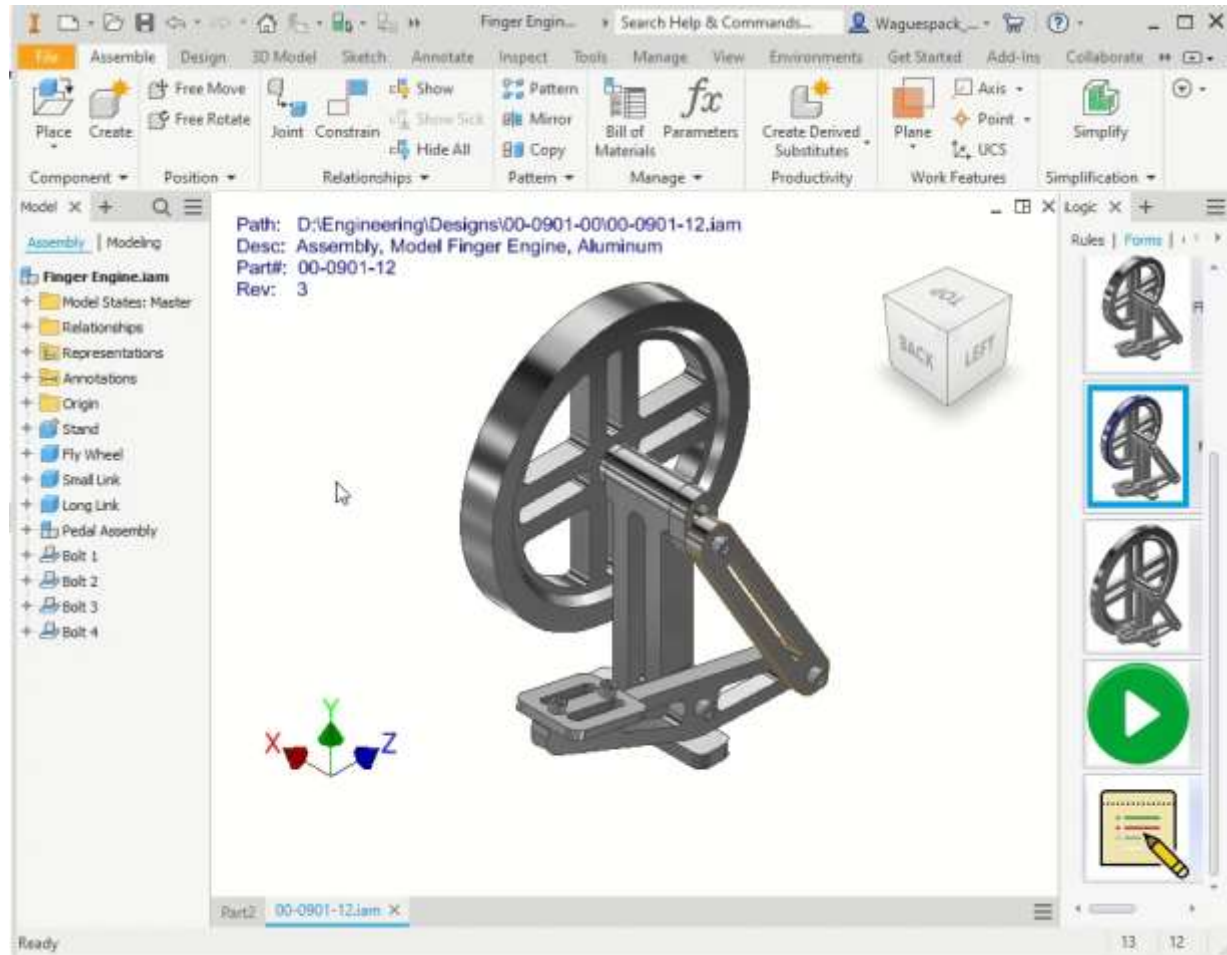
Display Information in the graphics area

Tip #23

Annotation Information

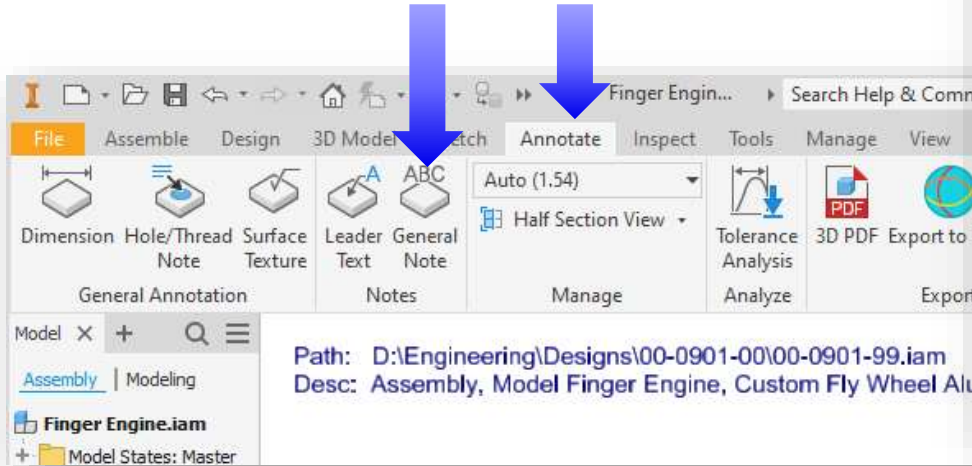
★ Tip provided by
Andrew Humiston

- Use a General Annotation Note to display file information to your users.



Annotation Note

- Creates and updates a General Annotation Note
- We use iLogic Trigger Events to update the note.
 - Example: The Before Save event



```

1 Sub Main
2     Dim oDoc As Document = ThisDoc.Document
3
4     'get general notes collection
5     sNotes = oDoc.ComponentDefinition.ModelAnnotations.ModelGeneralNotes
6
7     Dim oInfoNote As ModelGeneralNote
8     Dim oNoteDef As ModelGeneralNoteDefinition
9
10    Try : sNotes.Item("Information Note").delete : Catch : End Try
11    dFontSize = 0.12
12    oNoteDef = sNotes.CreateDefinition("", True, kUpperLeftQuadrant)
13    oNoteDef.Text.Size = dFontSize
14    oInfoNote = sNotes.Add(oNoteDef)
15    oInfoNote.Name = "Information Note"
16
17    'get existing note text
18    sString = sNotes.Item(1).Definition.Text.FormattedText
19
20    sValue = oDoc.FullFileName
21    sText = sText & "Path: " & sValue & "<Br/>"
22
23    sName = "Description"
24    sValue = iProperties.Value("Project", sName)
25    sText = sText & "Desc: " & sValue & "<Br/>"
26
27    sFormattedText = FormatText(sText, dFontSize)
28    sNotes.Item(1).Definition.Text.FormattedText = sFormattedText
29 End Sub
30
31 Function FormatText(sString1 As String, dFontSize As Double)
32
33     sFormattedText = "<StyleOverride " & "FontSize='" & dFontSize & "' " &
34     ">" & sString1 & "</StyleOverride>" & "<Br/>"
35     Return sFormattedText
36 End Function

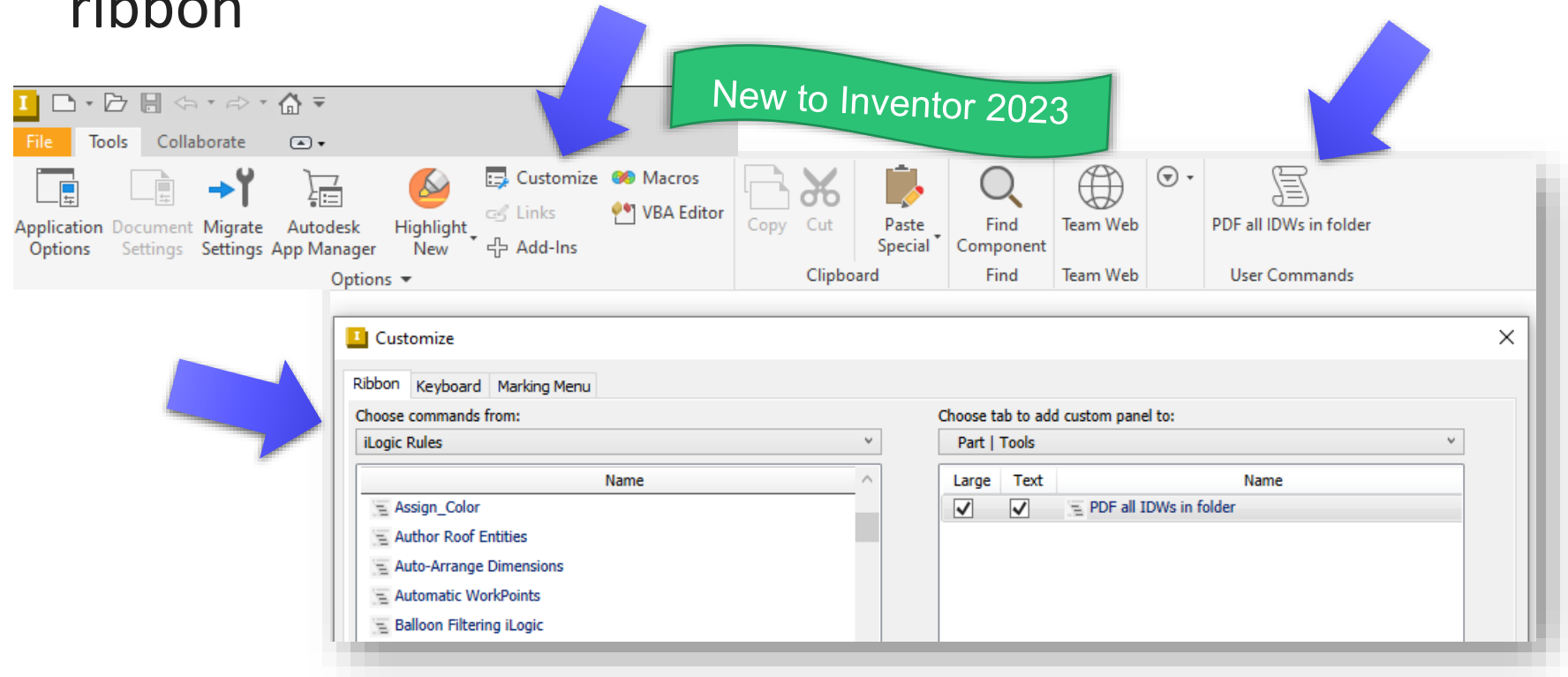
```




External Rule or Global Form Button on the Ribbon

Tip #24

Add buttons for External Rules and Global Forms to the ribbon

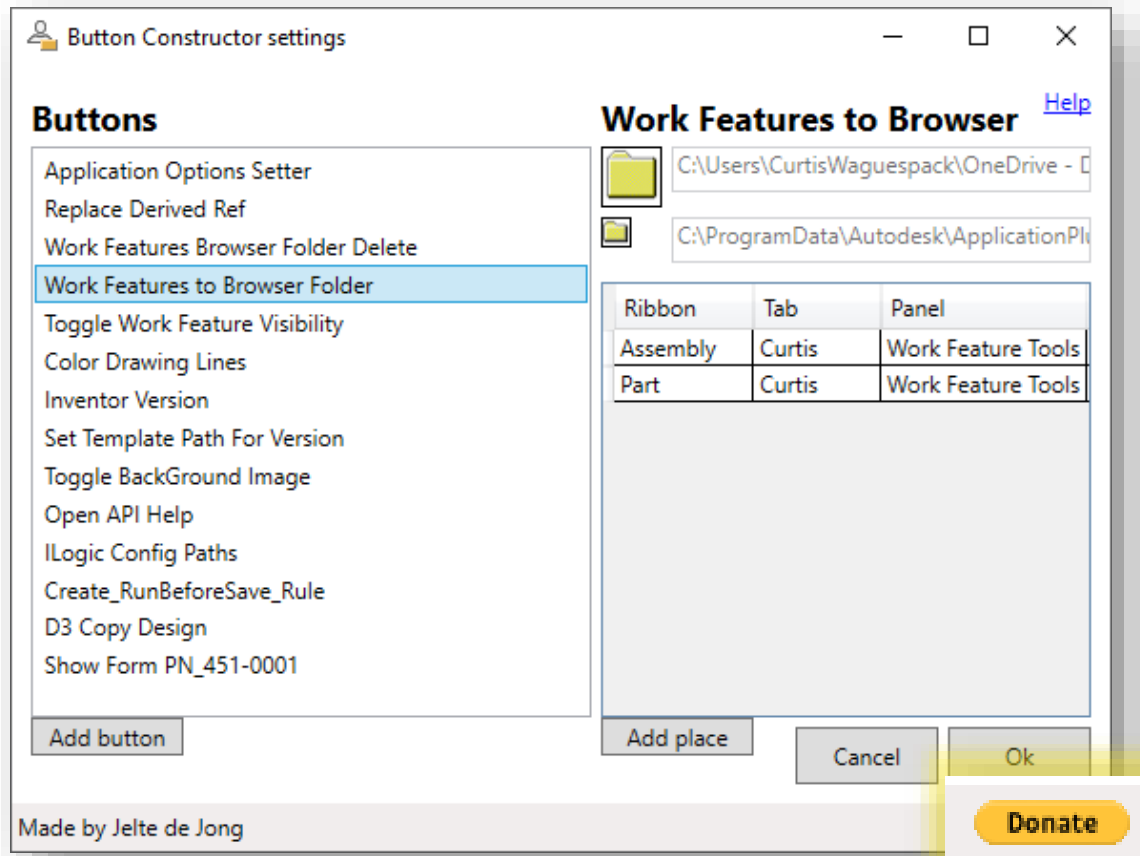


Button Constructor Add in

Get it at from the
Autodesk App Store

Created by: Jelte de Jong

www.hjalte.nl



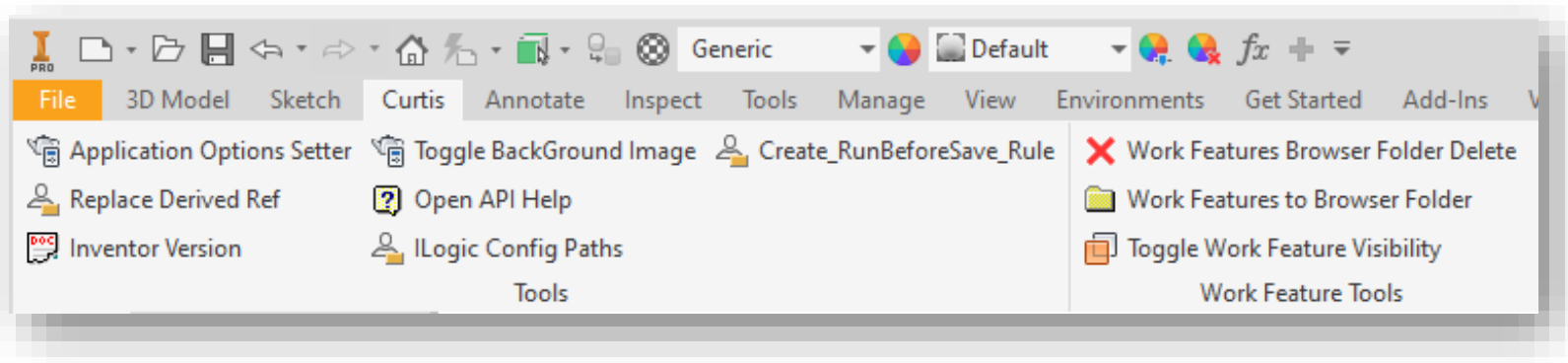
Button Constructor Add in

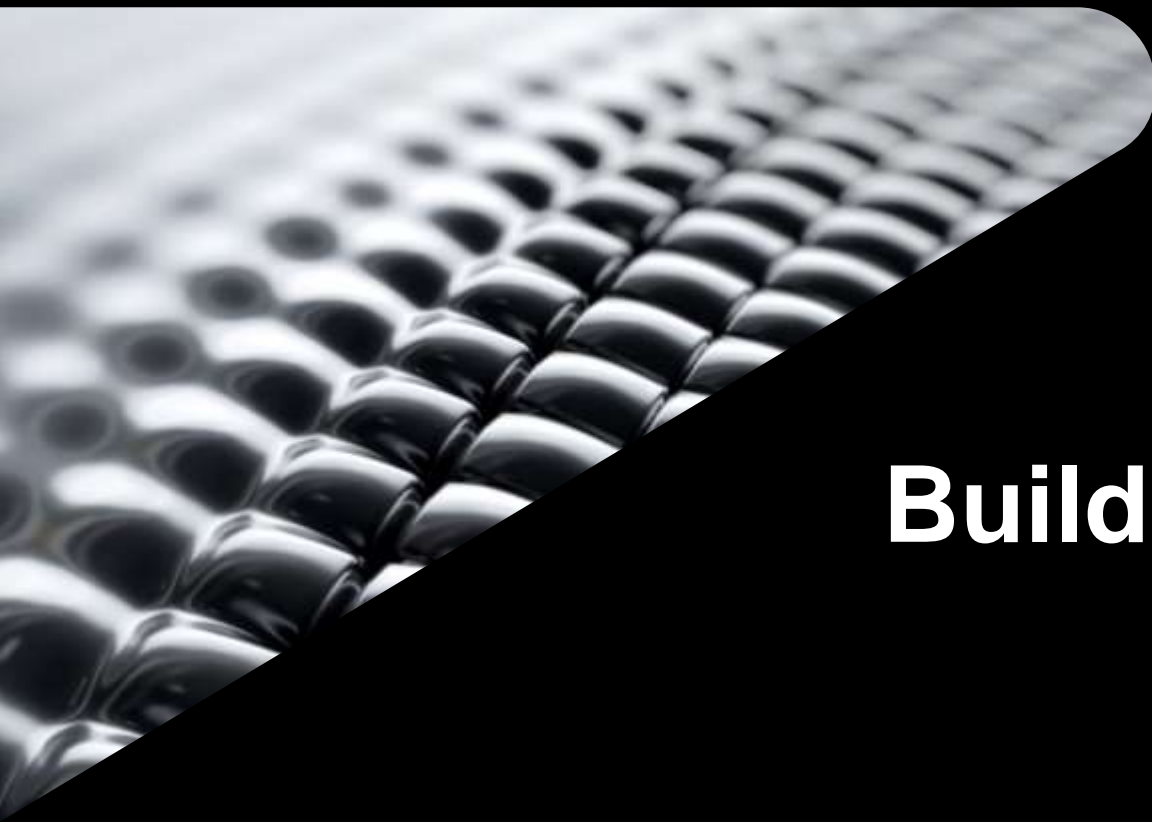
Get it at from the
Autodesk App Store

Created by: Jelte de Jong

Example:

- Custom tab called Curtis
- Contains buttons for some of my External rules





Build Better Forms



Tip #25

Make it more visual

- Add buttons to run rules
- Add icons to the buttons





Layout

  Open Job Folder

Customer (Click logo to refresh) Job (Click logo to refresh)

Save Path: S:\Active Jobs\ABC Solutions\PRODUCTION\0001\

 Select Base  Toggle Base Appearance

Appearance: Default





Place Multiple Place Single General Settings Reset Tools




Width 25.5 in



Height 120 in


Thickness 4 in

Connection Type ☒ Centered ☐ Left ☐ Right ☐ Tee

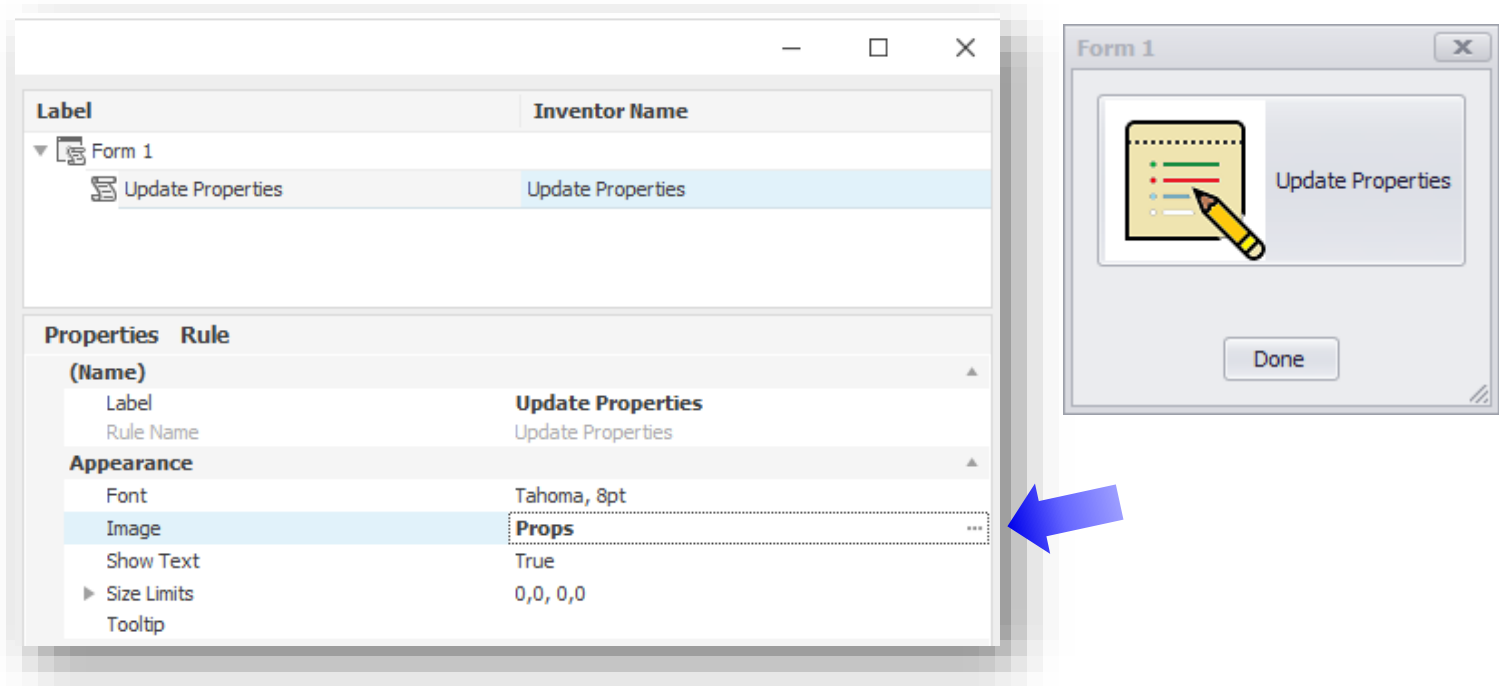
 Vertical  Wrapped  Horizontal  Angled

 Constrain Sides  Edit  Save And Replace

 Delete Unused  123 Renumber Components

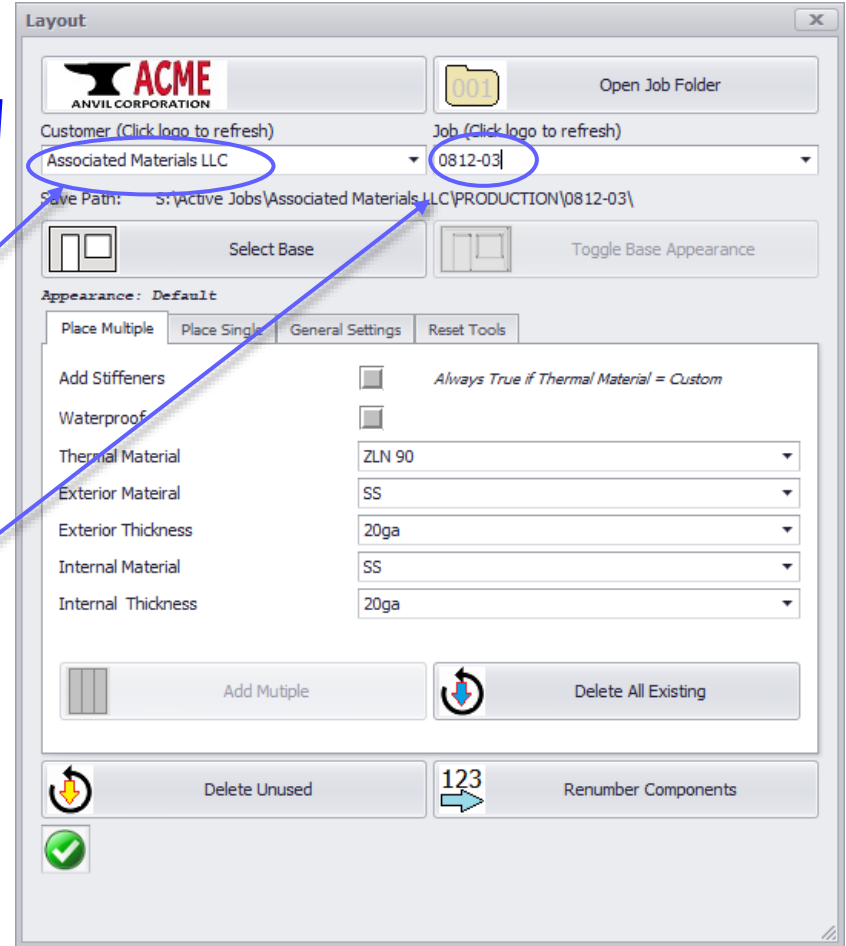
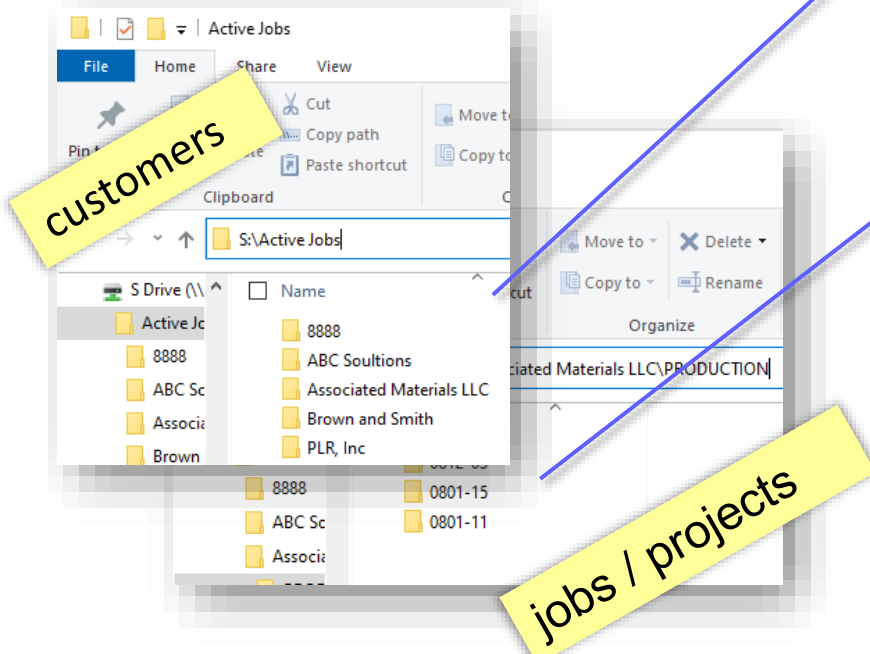


Add icons to the buttons



Make it more interactive

- Button for a rule that gets customer and job folders from directory, and **populates the lists**
- Prevents errors by disallowing invalid input



Make it more interactive

- Uses a True / False parameter as an **Enabling Parameter** to enable/disable the button
- Prevents errors by disallowing invalid options

The screenshot shows the Form Editor interface. A table lists parameters, with 'hasBaseModel' highlighted. A tooltip for 'hasBaseModel' shows its behavior as an enabling parameter.

Parameter Name	Co	Unit/Type	Equation
UnusedFileList		Text	
Status		Text	WARNING
Status_Message		Text	There are
hasBaseModel		True/False	False
hasPath		True/False	True

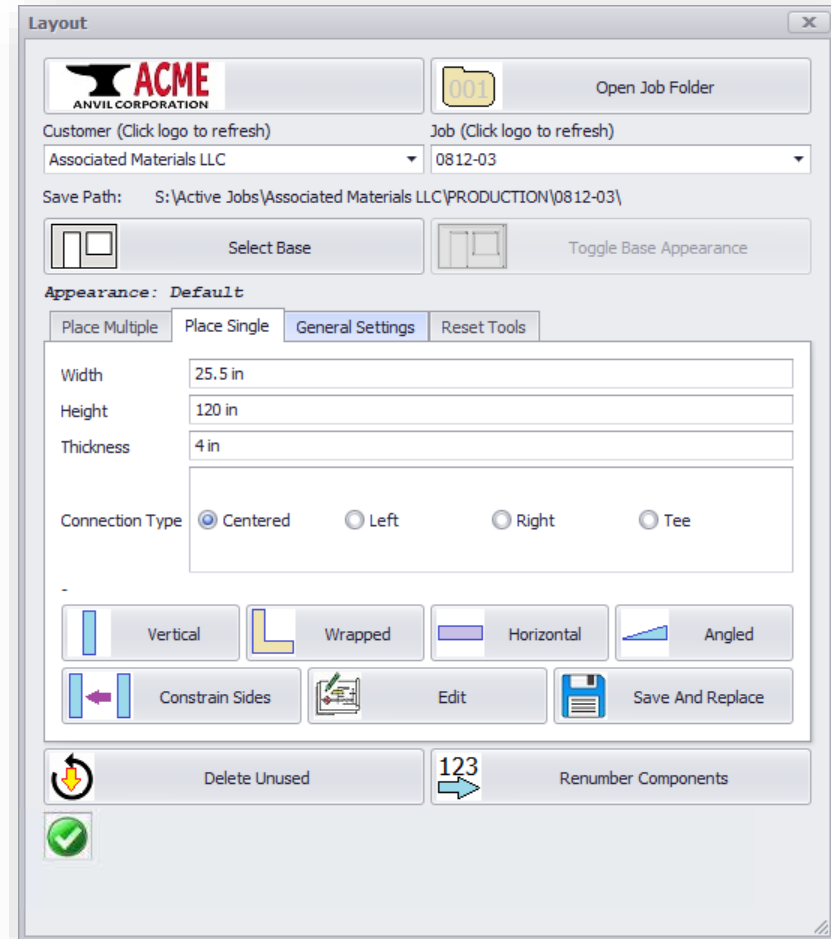
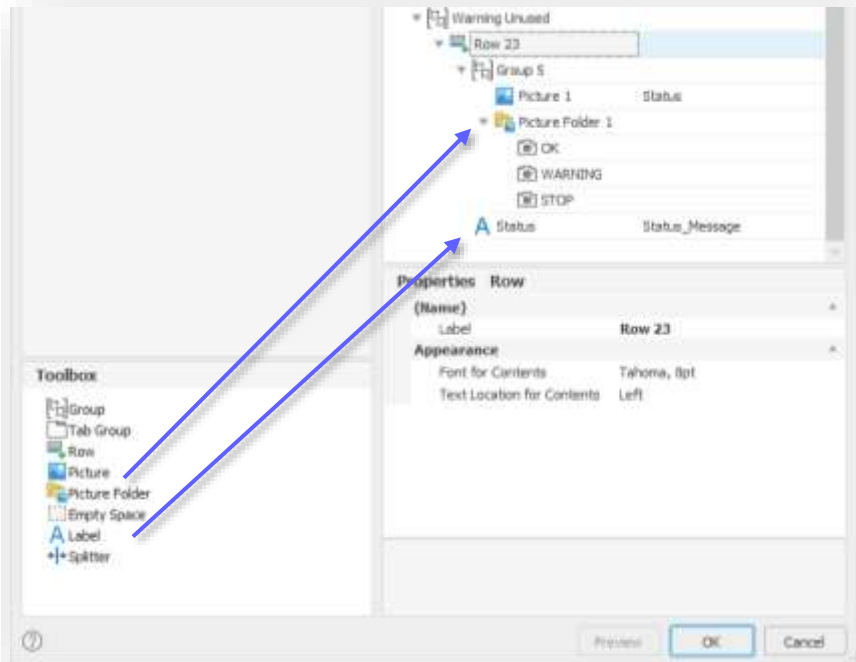
ToolTip Behavior
Enabling Parameter Name: **hasBaseModel**



The Layout window displays the ACME ANVIL CORPORATION logo and job information. It includes fields for Customer (Associated Materials LLC), Job (0812-03), and Save Path. Buttons for 'Select Base' and 'Toggle Base Appearance' are present. The 'Appearance: Default' section has tabs for 'Place Multiple', 'Place Single', 'General Settings', and 'Reset Tools'. Under 'Place Multiple', there are checkboxes for 'Add Stiffeners' (Always True if Thermal Material = Custom) and 'Waterproof'. Below these are dropdowns for 'Thermal Material' (ZLN 90), 'Exterior Mateiral' (SS), 'Exterior Thickness' (20ga), 'Internal Material' (SS), and 'Internal Thickness' (20ga). At the bottom, there are buttons for 'Add Muple', 'Delete All Existing', 'Delete Unused', and 'Renumber Components' (123). A green checkmark icon is also visible.

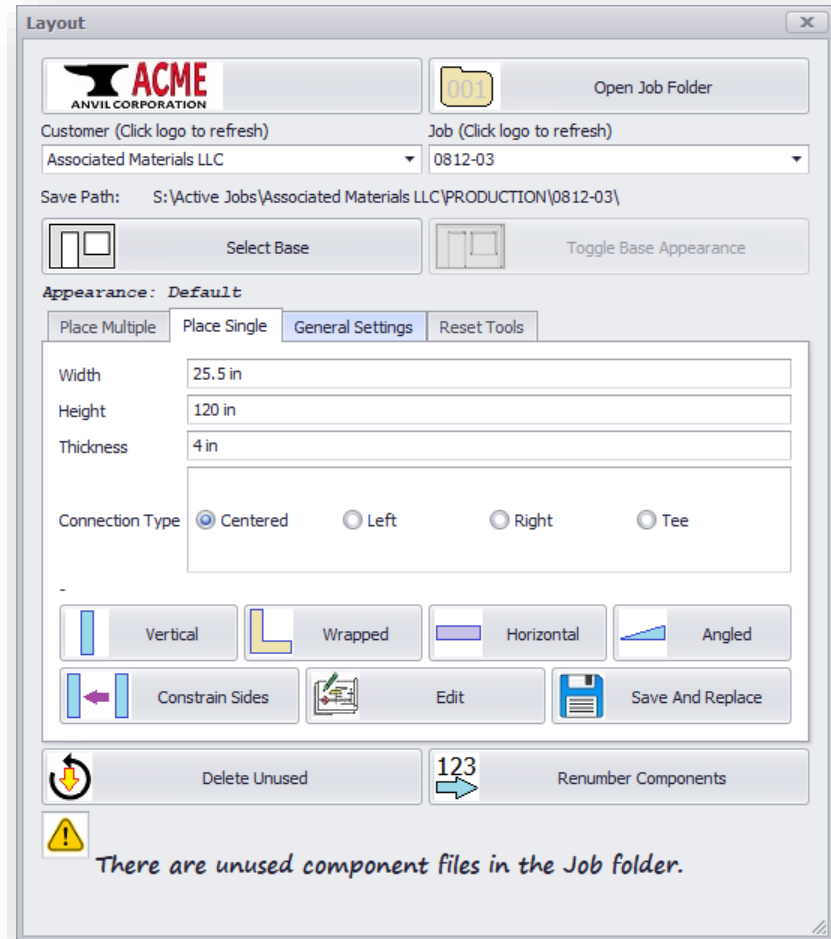
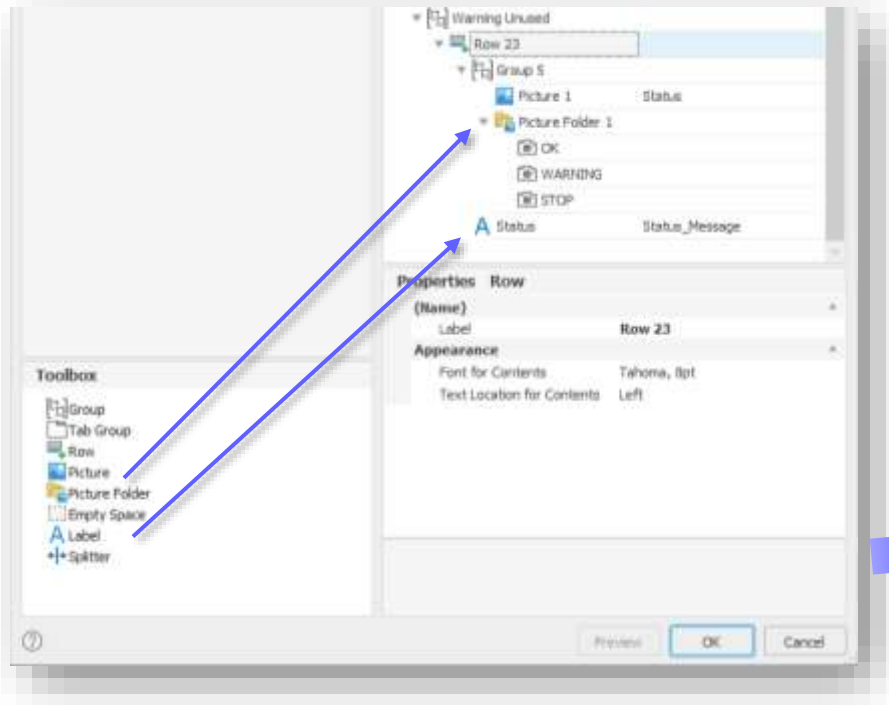
Make it more interactive

- Uses **Picture Folders and labels** to communicate interactively with the user



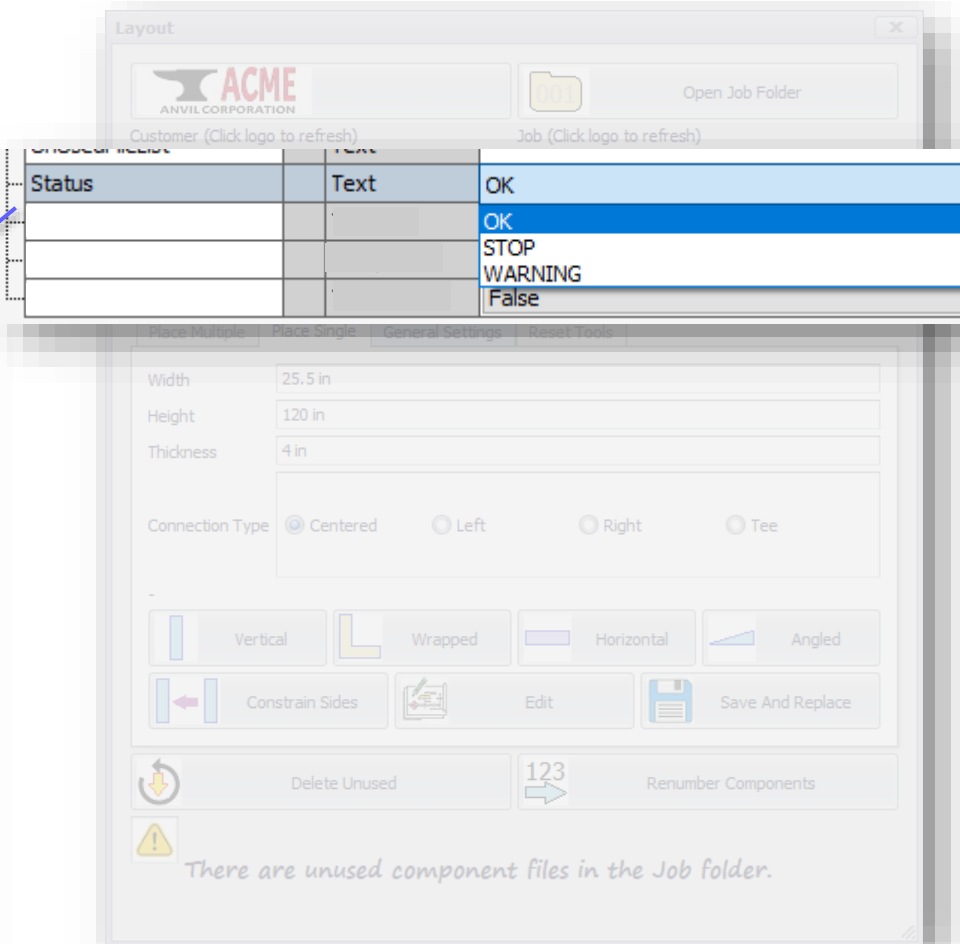
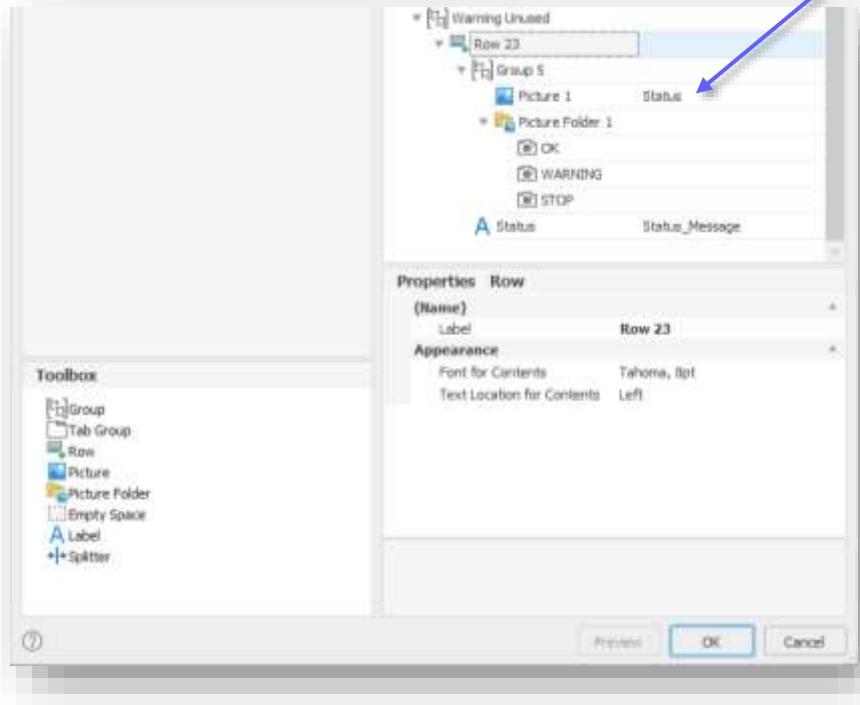
Make it more interactive

- Uses Picture Folders and labels to communicate interactively with the user



Make it more interactive

- Uses Picture Folders and labels to communicate interactively with the user

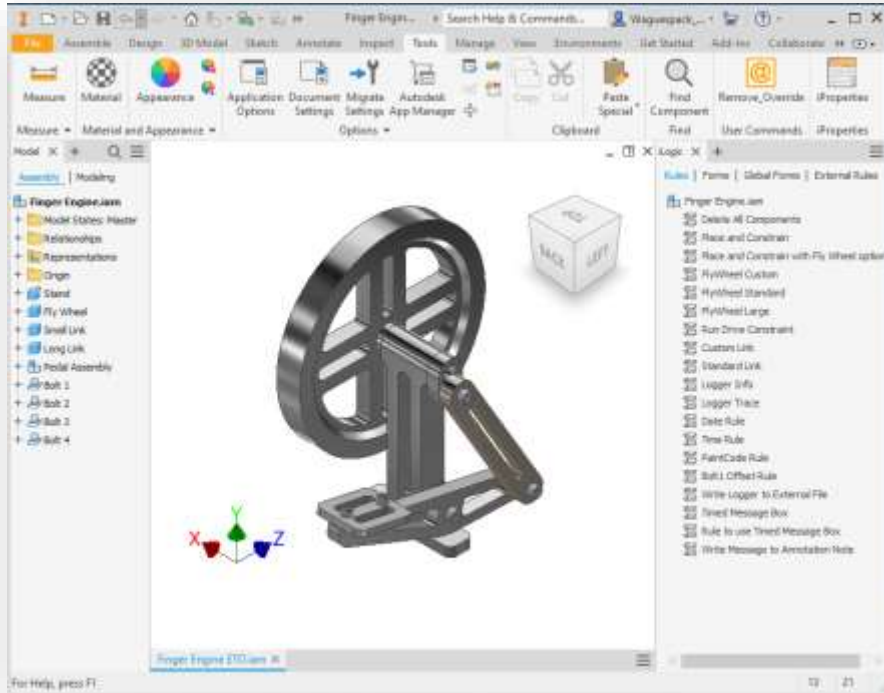




FINISH LINE IN SIGHT



Supplemental Information



> MFG501293 > Code Examples

Name

- Bin Location Class.iLogicVb
- Bin Location Function.iLogicVb
- Bin Location Sub.iLogicVb
- Bolt1 Offset Rule.iLogicVb
- Bolts Boolean Parameters with Not Version.iLogicVb
- Bolts If Then Version.iLogicVb
- Boolean Parameters with Not.iLogicVb
- Custom Link.iLogicVb
- Date Rule.iLogicVb
- Delete All Components.iLogicVb
- FlyWheel Custom.iLogicVb
- FlyWheel Large.iLogicVb
- FlyWheel Standard.iLogicVb
- Get Customer and Job Lists.iLogicVb
- Logger Info.iLogicVb
- Logger Trace.iLogicVb
- Open Jobs Folder.iLogicVb
- Outlining 1.iLogicVb
- Outlining 2.iLogicVb
- PaintCode Rule.iLogicVb
- Place and Constrain with Fly Wheel option.iLogicVb
- Place and Constrain.iLogicVb
- Planned and Optimized Rule Example.iLogicVb
- Rule to use Timed Message Box.iLogicVb
- Run Drive Constraint.iLogicVb
- Standard Link.iLogicVb
- Time Rule.iLogicVb
- Timed Message Box.iLogicVb
- Write Logger to External File.iLogicVb
- Write Message to Annotation Note.iLogicVb
- Write to Log.iLogicVb

MFG501293 > Models

Name

- Components
- Finger Engine ETO.iam
- Finger Engine.iam

Inventor 2022 files

A close-up, black and white photograph of a woven mesh texture, possibly a net or a screen, showing a grid of small, rounded, interlocking shapes. The texture is diagonal and occupies the left side of the image.

Thank you!

A close-up, black and white photograph of a woven mesh texture, possibly a net or a filter, showing a grid of diamond-shaped openings. The texture is slightly out of focus, with the foreground elements being sharper than the background.

Questions?



Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product offerings, specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2022 Autodesk. All rights reserved.