' This rule is used to place the shell plates in the tank body assembly

' It uses constraints to place each shell, and staggers them so that the seam

' alternates from one side of the tank to the other

Dim intTotalShells As Integer

' This variable calculates how many total shells is needed for this assembly

' This is based on the length of the tank, combined with information in the Excel spreadsheet

intTotalShells = SHELL\_Q\_1 + SHELL\_Q\_2

' The template had two pre-built constraints that would fail after running through the automation

' I found that I could fix them by just creating them at run time, rather than having them

' pre-baked into the template

' This happens sometimes, and you need to be creative to work around certain issues

Constraints.AddMate("Mate:1", "ASME Dished Head:1", "Work Plane2", "",

"XY Plane", TANK\_L / 2 ul)

Constraints.AddMate("Flush:2", "ASME Dished Head:2", "Work Plane2",

"ASME Dished Head:1", "Work Plane2", TANK\_L)

' First create the shell files in your local directory

Dim strOldFilename As String

Dim strNewFilename1, strNewFilename2 As String

strOldFilename = TEMPLATE\_PATH & "\Tank Body Assy\Shell.ipt"

' These lines of code create copies of the shell plates, and give them filenames that include the shell width

If SHELL\_Q\_1 >= 1 Then

strNewFilename1 = PROJECT\_PATH & PROJECT\_ID & "\Tank Body Assy\Shell - " & SHELL\_W\_1 & " Wide - " & PROJECT\_ID & ".ipt"

System.IO.File.Copy(strOldFilename, strNewFilename1)

End If

' Some tank lengths don't need two different widths in order to get the length to come out properly

' In that case, we don't want to create another shell plate model if we don't need it

If SHELL\_Q\_2 >= 1 Then

strNewFilename2 = PROJECT\_PATH & PROJECT\_ID & "\Tank Body Assy\Shell - " & SHELL\_W\_2 & " Wide - " & PROJECT\_ID & ".ipt"

System.IO.File.Copy(strOldFilename, strNewFilename2)

End If

Dim strNewFilename As String

Dim intShellWidth As Integer

Dim intRunningLength As Integer

' This variable will track our progress as we move down the tank and place the individual shell plates

intRunningLength = 0

' This For loop is the meat of placing the shell plates into our assembly

For j = 1 To intTotalShells

' If j is less than the total number of shell plates with thickness 1, then we need to insert more shell

' plates with thickness 1

' Otherwise, we need to insert new shell plates with thickness 2 (represented by the "SHELL\_W\_2" parameter

If j <= SHELL\_Q\_1 Then

strNewFilename = strNewFilename1

intShellWidth = SHELL\_W\_1

Else

strNewFilename = strNewFilename2

intShellWidth = SHELL\_W\_2

End If

' Add the part to the assembly

Dim strBrowserName As String

' We are now ready to add the shell plates into our tank body assembly

' Note that we first define how we want the occurrence name of each shell plate to be listed in the model browser

' Then we place them into the assembly (at the origin)

strBrowserName = "Shell " & intShellWidth & " Wide:" & j

Dim oShell = Components.Add(strBrowserName, strNewFilename)

' The Mod operator returns the remainder of a division calculation

' In this case, if j is 1, 1 divided by 2 has a remainder of 1 (the answer is 0, with a remainder of 1)

' If j is 2, then 2 divided by 2 is 1, and there is no remainder

' This trick lets us alternate what constraints we use as we place each shell plate

If j Mod 2 = 1 Then

' Notice that we use our "intRunningLength" variable to create an offset value for each new plate that gets placed

' Then at the end of the routine, we increase the value of "intRunningLength" by the current shell width,

' so that when we place the next shell, we have the next offset value ready to go

Constraints.AddFlush("Flush:" & 8 + (j - 1) \* 3 + 1, "", "YZ Plane", strBrowserName, "YZ Plane")

Constraints.AddFlush("Flush:" & 8 + (j - 1) \* 3 + 2, "", "XZ Plane", strBrowserName, "XZ Plane")

Constraints.AddFlush("Flush:" & 8 + (j - 1) \* 3 + 3, strBrowserName, "XY Plane", "",

"Work Plane1", intRunningLength + intShellWidth / 2 in)

Else

Constraints.AddMate("Mate:" & 8 + (j - 1) \* 3 + 1, "", "YZ Plane", strBrowserName, "YZ Plane")

Constraints.AddFlush("Flush:" & 8 + (j - 1) \* 3 + 2, "", "XZ Plane", strBrowserName, "XZ Plane")

Constraints.AddMate("Mate:" & 8 + (j - 1) \* 3 + 3, "", "Work Plane1", strBrowserName,

"XY Plane", (intRunningLength + intShellWidth / 2) \* -1 in)

End If

' We don't need to push parameters for each instance if the parts are the same model

' This finds the first instance of shell width 1, and the first instance of shell width 2, and then passes

' the "TANK\_OD" and "SHELL\_W" parameters

If j = 1 Or j = SHELL\_Q\_1 + 1 Then

Parameter(strBrowserName, "TANK\_OD") = TANK\_OD

Parameter(strBrowserName, "SHELL\_W") = intShellWidth

End If

intRunningLength = intRunningLength + intShellWidth

Next

' This next bit of code uses the Inventor API to suppress the assembly cuts for the manway and gunline if they are not used

' in the assemblies

Dim oShellAssy As AssemblyDocument

oShellAssy = ThisApplication.Documents.ItemByName(PROJECT\_PATH & PROJECT\_ID & "\Tank Body Assy\Tank Body Assy - " & PROJECT\_ID & ".iam")

Dim oAssyDef As AssemblyComponentDefinition = oShellAssy.ComponentDefinition

If MANWAY = False Then oAssyDef.Features(1).Suppressed = True

If GUNLINE = False Then oAssyDef.Features(