# SmartPlant 3D Programming II

*Student Workbook*

Process, Power & Marine

INTERGRAPH

**Version 7 SP4** **February 2008** **N/A**

## Copyright

Copyright © 2004-2008 Intergraph Corporation. All Rights Reserved.

Including software, file formats, and audiovisual displays; may be used pursuant to applicable software license agreement; contains confidential and proprietary information of Intergraph and/or third parties which is protected by copyright law, trade secret law, and international treaty, and may not be provided or otherwise made available without proper authorization.

Portions of this software are owned by Spatial Corp. © 1986-2008. All Rights Reserved.

## Restricted Rights Legend

Use, duplication, or disclosure by the government is subject to restrictions as set forth below. For civilian agencies: This was developed at private expense and is "restricted computer software" submitted with restricted rights in accordance with subparagraphs (a) through (d) of the Commercial Computer Software - Restricted Rights clause at 52.227-19 of the Federal Acquisition Regulations ("FAR") and its successors, and is unpublished and all rights are reserved under the copyright laws of the United States. For units of the Department of Defense ("DoD"): This is "commercial computer software" as defined at DFARS 252.227-7014 and the rights of the Government are as specified at DFARS 227.7202-3.

Unpublished – rights reserved under the copyright laws of the United States.
Intergraph Corporation
Huntsville, Alabama 35894-0001

## Warranties and Liabilities

All warranties given by Intergraph Corporation about equipment or software are set forth in your purchase contract, and nothing stated in, or implied by, this document or its contents shall be considered or deemed a modification or amendment of such warranties. Intergraph believes the information in this publication is accurate as of its publication date.

The information and the software discussed in this document are subject to change without notice and are subject to applicable technical product descriptions. Intergraph Corporation is not responsible for any error that may appear in this document.

The software discussed in this document is furnished under a license and may be used or copied only in accordance with the terms of this license.

No responsibility is assumed by Intergraph for the use or reliability of software on equipment that is not supplied by Intergraph or its affiliated companies. THE USER OF THE SOFTWARE IS EXPECTED TO MAKE THE FINAL EVALUATION AS TO THE USEFULNESS OF THE SOFTWARE IN HIS OWN ENVIRONMENT.

Intergraph is not responsible for the accuracy of delivered data including, but not limited to, catalog, reference and symbol data. Users should verify for themselves that the data is accurate and suitable for their project work.

## Trademarks

Intergraph, the Intergraph logo, PDS, SmartPlant, FrameWorks, I-Convert, I-Export, I-Sketch, SmartMarine, IntelliShip, INtools, ISOGEN, MARIAN, SmartSketch, SPOOLGEN, SupportManager, and SupportModeler are trademarks or registered trademarks of Intergraph Corporation or its subsidiaries in the United States and other countries. Microsoft and Windows are registered trademarks of Microsoft Corporation. ACIS is a registered trademark of SPATIAL TECHNOLOGY, INC. Infragistics, Presentation Layer Framework, ActiveTreeView Ctrl, ProtoViewCtl, ActiveThreed Ctrl, ActiveListBar Ctrl, ActiveSplitter, ActiveToolbars Ctrl, ActiveToolbars Plus Ctrl, and ProtoView are trademarks of Infragistics, Inc. Portions of 2D DCM, 3D DCM, and HLM from D-Cubed Limited are incorporated. All rights reserved. Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other brands and product names are trademarks of their respective owners.

# Table of Contents

# Introduction

The Student workbook is designed as an aid for students attending the SP3D Programming II class presented by Intergraph Corporation, and it's a supplement to the standard product documentation.

## Objective

This document is designed to provide a guideline for people who need to customize the server-based interference check post-processing rule and design smart occurrence symbol definitions for the SmartPlant 3D application. This workbook includes, but is not limited to the following:

- Provides an overview of customization with the SmartPlant 3D software using standard Windows™ programming tools and languages like Visual Basic™.
- Customize the server-based interference check post-processing rule
- Describes some of the automation components that can be used to design new symbol smart occurrence entities
- Provides examples of workflow customization.

Assumptions are made here that the user has a prerequisite knowledge of the SmartPlant 3D reference data.

## Course description

- Visual Basic Smart Occurrence Symbol Creation

## Course Reference Material

SmartPlant 3D Programmer's Guide
SmartPlant 3D Symbols Reference Data Guide
SmartPlant 3D Reference Data Guide

# Lab 1: Interference Check Post-Processing Rule

## Objectives

After completing this lab, you will be able to:

- Understand the post-processing interference checking rule
- Assign the interference object to a permission group
- Add a note to the interference object based on the object type of the colliding objects
- Add a rule to eliminate the creation of interference objects for Handrail-to-Slab collisions
- Add a rule to avoid creating interference objects where colliding objects belong to a test permission group

### Exercise 1: Interference object permission group property

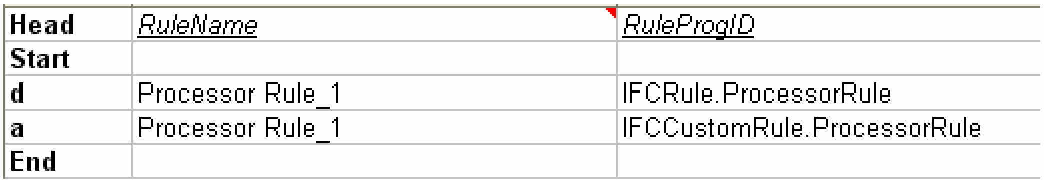
1. Create the following directories:

   *c:\train\IFCRule*

2. Copy the delivered IFC Post-Processing Visual Basic files to *c:\train\IFCRule*

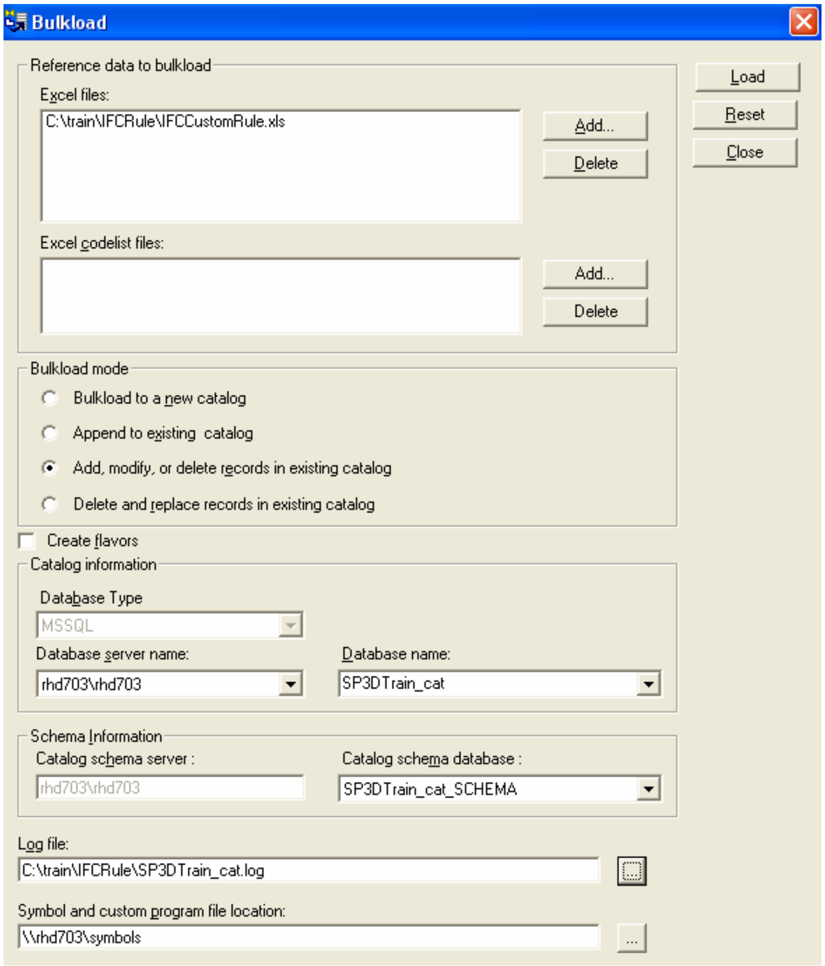
   **Note:**
   - The IFC Post-Processing Visual Basic project is delivered under [Installation]\Programming\ExampleCode\Rules\InterferenceRules

3. Open the IFCRule.xls under [Installation]\CatalogData\BulkLoad\Datafiles
4. Remember to delete the existing record and add the letter A to the new record.
5. Add a new IFC rule name and the ProgID as shown here:



| Head | RuleName | RuleProgID |
|---|---|---|
| Start | | |
| d | Processor Rule_1 | IFCRule.ProcessorRule |
| a | Processor Rule_1 | IFCCustomRule.ProcessorRule |
| End | | |

6. Save the Excel sheet as IFCCustomRule.xls under c:\train\IFCRule. Exit Excel.
7. Run Bulkload Utility (START Menu -> Intergraph SmartPlant 3D -> Database Tools -> Bulkload Reference Data)
8. Set the bulkload to A/M/D mode.
9. Select Load button to add the new IFC rule into the training catalog.



Bulkload

Reference data to bulkload

Excel files: C:\train\IFCRule\IFCCustomRule.xls

Excel codelist files:

Bulkload mode

○ Bulkload to a new catalog

○ Append to existing catalog

● Add, modify, or delete records in existing catalog

○ Delete and replace records in existing catalog

□ Create flavors

Catalog information

Database Type: MSSQL

Database server name: rhd703\rhd703

Database name: SP3DTrain_cat

Schema Information

Catalog schema server : rhd703\rhd703

Catalog schema database : SP3DTrain_cat_SCHEMA

Log file: C:\train\IFCRule\SP3DTrain_cat.log

Symbol and custom program file location: \\rhd703\symbols

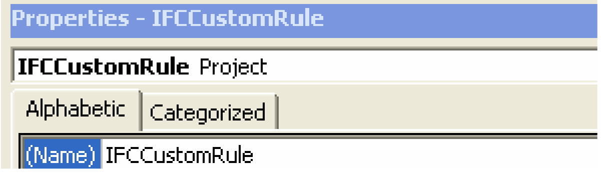
10. Place/route some structures members, pipelines, cabletrays and standard equipments in the model such that they interfere with one-another as shown below:

    a) Piping against structure
    b) Equipment against structure
    c) Piping against cabletray
    d) Structure against cabletray

11. Navigate to *c:\train\IFCRule* and remove the Read-only flag from all files.

12. Open the IFCRule.vbp project.

13. Go to the Properties Window and change the name of the Project as shown here:



Properties - IFCCustomRule

**IFCCustomRule** Project

(Name) IFCCustomRule

14. Go to the Visual Basic Explorer Window and select the Project node. Select *File -> Save Project As* option to save the project as IFCCustomRule.vbp under the *c:\train\IFCRule* directory.

15. Open the PostProcessorRule.cls class and navigate to the Class_Initialize subroutine.

16. Note the names of the permission groups:

```
m_strPermissionGroups(0) = "IFC Supports"
m_strPermissionGroups(1) = "IFC Conduits"
m_strPermissionGroups(2) = "IFC Cableway"
m_strPermissionGroups(3) = "IFC HVAC"
m_strPermissionGroups(4) = "IFC Piping"
m_strPermissionGroups(5) = "IFC Structure"
m_strPermissionGroups(6) = "IFC Equipment"
m_strPermissionGroups(7) = "IFC Volumes"
```

17. These are the names of the permission groups that the interferences will be assigned to. You will need to create these permission groups using the Project Management task.

18. Notice that in the Get Permission Index subroutine defines the ranking on the basis of which permission groups will be assigned.

```
Select Case (strParentType)

Case "Pipe Supports", "Cable Tray Supports", "Duct Supports"
   GetPermissionGroupIndex = 0

Case "Conduit Components", "Conduits"
   GetPermissionGroupIndex = 1

Case "Cable Tray Components", "Cableway Along Leg", "Cableway Straight", _
         "Cable Trays", "Cableway Turn"
   GetPermissionGroupIndex = 2

Case "HVAC Components", "Ducts"
   GetPermissionGroupIndex = 3

Case "Pipes", "Piping Welds", "Piping Components", "Piping Instruments", _
         "Piping Specialty Items"
   GetPermissionGroupIndex = 4

Case "Member Part Linear", "Member Part Curve", "Slab", _
          "Equipment Foundation", "Footing", "Stairs", "Ladders", "Handrails"
   GetPermissionGroupIndex = 5

Case "Legacy Equipment", "Legacy Designed Equipment", "Equipment"
   GetPermissionGroupIndex = 6
```

```
Case "Interference Volumes"
   GetPermissionGroupIndex = 7

Case Default
   GetPermissionGroupIndex = -1
End Select
```

**Note** This is the hierarchy of object types. If an object that is lower in the hierarchy (lower permissiongroupindex) interferes with an object higher in the hierarchy (higher permissiongroupindex), the interference will be assigned to the permission group of the object lower in the hierarchy.

19. Go to the IJDInterferenceRule_CreateInterference subroutine and uncomment the following lines:

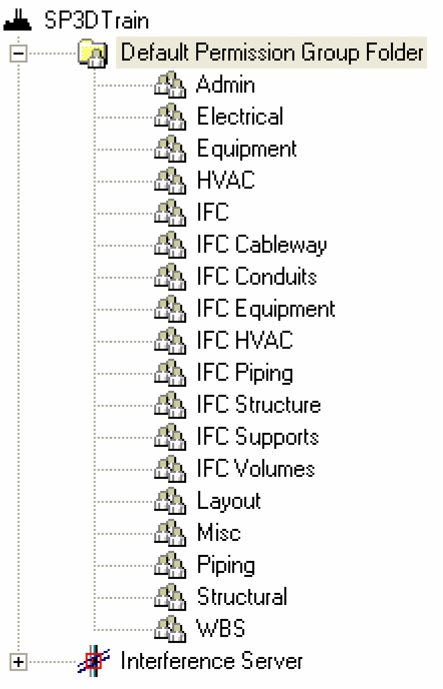
    ```
    If IfcType = IfcServerInterference Then
    'assign a permission group to the IFC object based on rule
       AssignIFCPermissionGroup pInterferenceObj, strParentType1, strParentType2
    End If
    ```

20. Update the binary compatibility of your program to IFCCustomRule-ref.dll

    Note: One of the most important steps in Visual Basic programming is to preserve the binary compatibility of your program. Save the final version of your dll file to be binary compatibility in order to preserve the CLSID

21. Re-compile the program using File -> Make IFCCustomRule.dll

22. Start the Project Management task and make permission groups named "IFC Cableway", "IFC Piping" and "IFC Structure", etc. as shown here:



SP3DTrain
Default Permission Group Folder
Admin
Electrical
Equipment
HVAC
IFC
IFC Cableway
IFC Conduits
IFC Equipment
IFC HVAC
IFC Piping
IFC Structure
IFC Supports
IFC Volumes
Layout
Misc
Piping
Structural
WBS
Interference Server

23. Start the interference check service.

24. Select the Plant and select permission group IFC as the group to assign interferences to.

25. Click Start to start the interference detection process. The process starts and begins running.

26. After 5 minutes, refresh/define workspace that includes objects placed in the above step. You should see interferences created between the objects.
27. Check the permission group of the interference object. You will see the following:

| Interfering Objects | Permission Group |
|---|---|
| Piping against structure | IFC Piping |
| Equipment against structure | IFC Structure |
| Piping against cabletray | IFC Cableway |
| Structure against cabletray | IFC Cableway |

28. Stop the interference detection process.

### Exercise 2: Interference object remark property

Add a note to the interference object remark property where the colliding objects are both structure objects (linear member part).

1. Open the IFCCustomRule.vbp project.

2. Go to the PostProcessorRule.cls

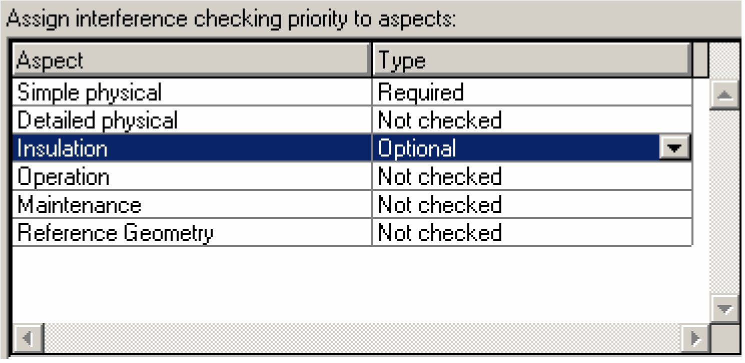
3. Go to the IJDInterferenceRule_CreateInterference subroutine and add the following lines

```
Dim strNotes As String
strNotes = ""
If IfcType = IfcServerInterference Then
   If strParentType1 Like "Member Part Linear" And _
      strParentType2 Like "Member Part Linear" Then
      strNotes = "Call Structure Design leader"
   pInterferenceObj.InterferenceRemark = strNotes
   End If
End If
```

4. Re-compile the program using File -> Make IFCCustomRule.dll

5. Start the interference check service.

6. Change the interference checking process criteria by assigning the interference priority to the Insulation Aspect as Optional.



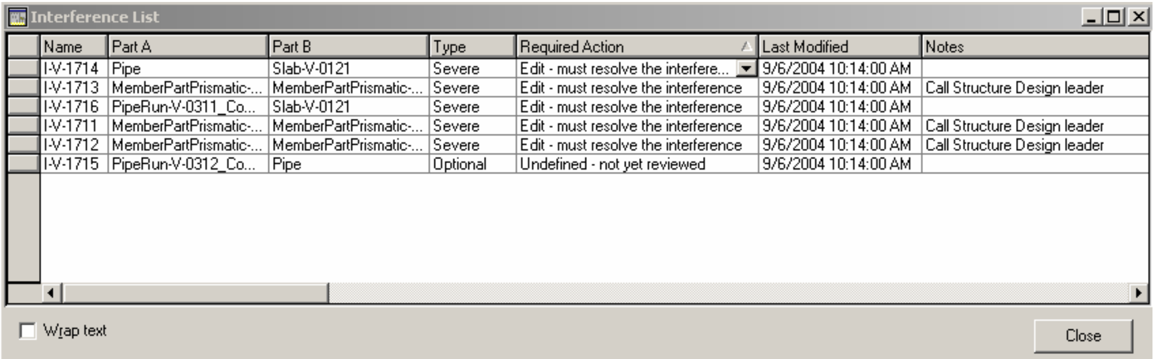
Assign interference checking priority to aspects:

| Aspect | Type |
|---|---|
| Simple physical | Required |
| Detailed physical | Not checked |
| Insulation | Optional |
| Operation | Not checked |
| Maintenance | Not checked |
| Reference Geometry | Not checked |

7. Click Start button to start the interference detection process. Select Yes to re-check the entire Model. Select OK button for the process starts and begins running.

8. After 5 minutes, refresh/define workspace. You should see interferences created between the objects. Stop the interference detection process.

9. Check the remark property of the interference objects as shown below:



| Name | Part A | Part B | Type | Required Action | Last Modified | Notes |
|---|---|---|---|---|---|---|
| I-V-1714 | Pipe | Slab-V-0121 | Severe | Edit - must resolve the interfere... | 9/6/2004 10:14:00 AM | |
| I-V-1713 | MemberPartPrismatic-... | MemberPartPrismatic-... | Severe | Edit - must resolve the interference | 9/6/2004 10:14:00 AM | Call Structure Design leader |
| I-V-1716 | PipeRun-V-0311_Co... | Slab-V-0121 | Severe | Edit - must resolve the interference | 9/6/2004 10:14:00 AM | |
| I-V-1711 | MemberPartPrismatic-... | MemberPartPrismatic-... | Severe | Edit - must resolve the interference | 9/6/2004 10:14:00 AM | Call Structure Design leader |
| I-V-1712 | MemberPartPrismatic-... | MemberPartPrismatic-... | Severe | Edit - must resolve the interference | 9/6/2004 10:14:00 AM | Call Structure Design leader |
| I-V-1715 | PipeRun-V-0312_Co... | Pipe | Optional | Undefined - not yet reviewed | 9/6/2004 10:14:00 AM | |

### Exercise 3: Interference rule for Handrails-to-Slab collisions

Place handrails and grating slabs in the model such that they interfere with one-another.

1. Open the IFCCustomRule.vbp project.

2. Go to the PostProcessorRule.cls

3. Go to the IJDInterferenceRule_CreateInterference subroutine and add the following lines:

```
If IfcType = IfcServerInterference Then
    If HandrailClashGrating(strParentType1, strParentType2, pParent1, pParent2) Then
        IJDInterferenceRule_CreateInterference = False
       Exit Function
   End If
End If
```

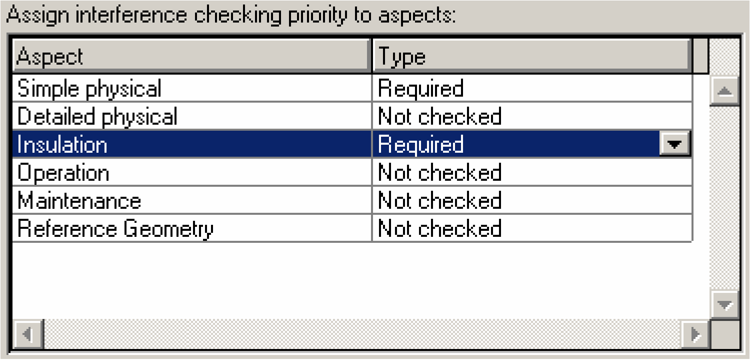
4. Create a function called HandrailClashGrating() that return a Boolean value if there is an interference between handrail and slab of type grating.

```
Private Function HandrailClashGrating(strObject1 As String _
                                   , strObject2 As String _
                                  , ByVal pParent1 As Object _
                                  , ByVal pParent2 As Object _
                                  ) As Boolean
On Error GoTo ErrHndlr

    HandrailClashGrating = False

    Dim oAttrbs As IJDAttributes
    Dim oRelationHelper As IMSRelation.DRelationHelper
    Dim oCollection As IMSRelation.DCollectionHelper
    Dim slabtype As String

    If (strObject1 Like "Slab" And strObject2 Like "Handrails") Then
    ' get the slab type from the object
        Set oRelationHelper = pParent1
        Set oCollection = oRelationHelper.CollectionRelations("ISPSSlabEntity",
"SlabEntityTypeReferenceRln_ORIG")
        If oCollection.count <> 0 Then
            Set oAttrbs = oCollection.Item(1)
            slabtype = oAttrbs.CollectionOfAttributes("IJDPart").Item("PartNumber").Value
            ' check if slab is of grating type
            If InStr(slabtype, "Grating") Then
                HandrailClashGrating = True
                Exit Function
            End If
        End If

    ElseIf (strObject1 Like "Handrails" And strObject2 Like "Slab") Then
        ' get the slab type from the object
        Set oRelationHelper = pParent2
        Set oCollection = oRelationHelper.CollectionRelations("ISPSSlabEntity",
"SlabEntityTypeReferenceRln_ORIG")
        If oCollection.count <> 0 Then
            Set oAttrbs = oCollection.Item(1)
            slabtype = oAttrbs.CollectionOfAttributes("IJDPart").Item("PartNumber").Value
            ' check if slab is of grating type
            If InStr(slabtype, "Grating") Then
                HandrailClashGrating = True
                Exit Function
            End If
        End If
    End If

Exit Function
ErrHndlr:
    Err.Clear
End Function
```

5. Re-compile the program using File -> Make IFCustomRule.dll

6. Start the interference check service.

7. Change the interference checking process criteria by assign the interference priority to the Insulation Aspect as Required.



Assign interference checking priority to aspects:

| Aspect | Type |
|---|---|
| Simple physical | Required |
| Detailed physical | Not checked |
| Insulation | Required |
| Operation | Not checked |
| Maintenance | Not checked |
| Reference Geometry | Not checked |

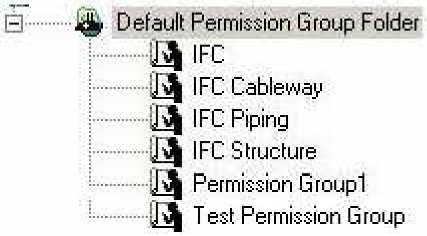
8. Click Start button to start the interference detection process. Select Yes to re-check the entire Model. Select OK button for the process starts and begins running.

9. After 5 minutes, refresh/define workspace. You should see interferences created between the objects. The interference checking process should not create an interference object between handrails and slabs of type grating.

10. Stop the interference detection process.

### Exercise 4: Interference Rule for objects belonging to a Test Permission Group

1. Start the Project Management task and create a permission group named "TESTPG".



Default Permission Group Folder
- IFC
- IFC Cableway
- IFC Piping
- IFC Structure
- Permission Group1
- Test Permission Group

2. Go to the Equipment Task. Set the active permission group to "TESTPG".
3. Place some standard equipments in the model such that they interfere with one-another.

4. Go to the Piping Task. Make sure the active permission group is set to "TESTPG". Route some pipe runs in the model such that they interfere with any objects in the model.

5. Add a rule to avoid creating interference objects where colliding objects belong to "TESTPG".

6. Open the IFCCustomRule.vbp project.

7. Go to  the PostProcessorRule.cls

8. Go to the IJDInterferenceRule_CreateInterference subroutine and add the following lines:

```
If IfcType = IfcServerInterference Then
     If objectBelongToPG("TESTPG", pParent1, pParent2) Then
         IJDInterferenceRule_CreateInterference = False
        Exit Function
    End If
End If
```

9. Create a function called objectBelongToPG("TESTPG", pParent1, pParent2) that return a Boolean value if the colliding object belong to a Permission Group called “TESTPG”.

10. Create a Private Function with the following arguments:

```
Private Function objectBelongToPG(UPPERCASE_pgName_substring As String _
                                  , ByVal pParent1 As Object _
                                  , ByVal pParent2 As Object _
                                  ) As Boolean
```

11. Next, add an error handler statement

```
On Error GoTo ErrHndlr
```

12. Set the function to return a False Boolean value by default

```
 objectBelongToPG = False
```

13. Declare and set the reference to the object variables:

```
    Dim pNum1 As Long, pNum2 As Long
    Dim strPgName1 As String, strPgName2 As String
    Dim pObject1 As IJDObject
    Dim pObject2 As IJDObject
    Set pObject1 = pParent1
    Set pObject2 = pParent2
```

14. Declare variables to store the Permission Group Name and Permission Group ID of the colliding objects.

```
    Dim pNum1 As Long, pNum2 As Long
    Dim strPgName1 As String, strPgName2 As String
```

15. Get the permission group ID from the two colliding objects

```
    pNum1 = pObject1.PermissionGroup
    pNum2 = pObject2.PermissionGroup
```

16. Get the permission group Name from the two colliding objects using a function called ConvertPGNumberToName(). The code for this function is documented later in this lab.

```
    strPgName1 = ConvertPGNumberToName(pNum1)
    strPgName2 = ConvertPGNumberToName(pNum2)
```

17. Write the test condition if the retrieved permission group name is “TESTPG”. If the condition is true then set the function to return a TRUE Boolean value.

```
    If InStr(UCase(strPgName1), UPPERCASE_pgName_substring) > 0 Or
InStr(UCase(strPgName2), UPPERCASE_pgName_substring) > 0 Then
        objectBelongToPG = True
        Exit Function
    End If
```

18. Add the exit statement to exit the function procedure

```
Exit Function
```

19. When an error occurs at run time, add the following code to handle it:

```
  Exit Function
  ErrHndlr:
      Err.Clear
```

**Note** The function should look like this:

```
Private Function objectBelongToPG(UPPERCASE_pgName_substring As String _
                                , ByVal pParent1 As Object _
                                , ByVal pParent2 As Object _
                                ) As Boolean
On Error GoTo ErrHndlr
    ' by default, foul will be created
    objectBelongToPG = False

    Dim pNum1 As Long, pNum2 As Long
    Dim strPgName1 As String, strPgName2 As String
    Dim pObject1 As IJDObject
    Dim pObject2 As IJDObject
    Set pObject1 = pParent1
    Set pObject2 = pParent2

    ' getting permission group name for parts
    pNum1 = pObject1.PermissionGroup
    pNum2 = pObject2.PermissionGroup
    strPgName1 = ConvertPGNumberToName(pNum1)
    strPgName2 = ConvertPGNumberToName(pNum2)

    If InStr(UCase(strPgName1), UPPERCASE_pgName_substring) > 0 Or InStr(UCase(strPgName2),
UPPERCASE_pgName_substring) > 0 Then
        objectBelongToPG = True
        Exit Function
    End If
```

```
Exit Function
ErrHndlr:
   Err.Clear
End Function
```

20. Next, write a Private Function that gets the permission group id and return the corresponding permission group name.

21. Create a Private Function with the following arguments:

```
Private Function ConvertPGNameToNumber(ByVal strPFName As String) As Long
```

23. Next, add an error handler statement

```
On Error GoTo ErrHndlr
```

24. Declare variables to store temporary a Permission Group Name and a Permission Group ID. Also, declare a variable to store the total number of permission group (count) define in the model.

```
   Dim CID As Long
   Dim count As Long
   Dim CIDName As String
   Dim acc As Long
   Dim i As Long
```

25. User the IJAccessControlConfiguration interface and the ApplicationContext service to retrieve the Access Control Configuration information for the current model. Declare and set the reference to the object variables:

```
   Dim oMidCtx As IJMiddleContext
   Dim oDBTypeConfig As IJDBTypeConfiguration
   Dim oDataBaseConfig As IJDataBaseConfiguration
   Dim oACConfig As IJAccessControlConfiguration
   Dim oAccessControl As IJAccessControl
   Dim bFound As Boolean

   Set oDBTypeConfig = New DBTypeConfiguration
   Set oDataBaseConfig = New DataBaseConfiguration
   Set oACConfig = New AccessControlConfiguration

   Set oMidCtx = New GSCADMiddleContext 'should come with initialzied one
   oMidCtx.GetConfigurationTablesFromMiddle oDBTypeConfig, oDataBaseConfig, oACConfig
   Set oAccessControl = oACConfig.AccessControl
   count = oACConfig.NumberConditionIDs
```

26. Create a loop to go through the permission group list. Set a Boolean variable to TRUE if the permission group id is found in the list and return the corresponding permission group name.

```
For i = 1 To count
   oACConfig.GetConditionIDByIndex i, CIDName, CID
   oAccessControl.GetAccessRight CID, acc
   If ((acc And acUpdate) = acUpdate) Then
      If CIDName Like strPFName Then
         On Error Resume Next
         ConvertPGNameToNumber = CID
         bFound = True
         Exit For
      End If
   Else
      bFound = False
   End If
Next i
```

27. If the permission group id is not found in the list, return No found string.

```
'Could not find any permission group .. assingning 0
If bFound = False Then
   ConvertPGNameToNumber = 0
End If
```

28. Add the exit statement to exit the function procedure

```
Exit Function
```

29. When an error occurs at run time, add the following code to handle it:

```
  ErrHndlr:
Err.Raise Err.Number
Debug.Assert False
```

**Note** The function should look like this:

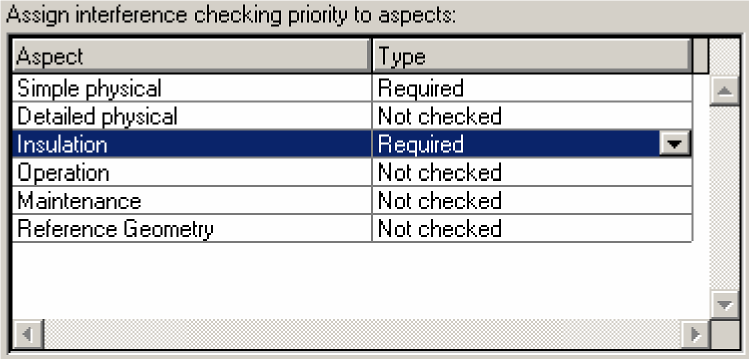
```
'Gets the PG number and return the corresponding permission group string
Private Function ConvertPGNumberToName(ByVal PGnum As Long) As String
On Error GoTo ErrHndlr
   Dim PGID As Long
   Dim count As Long
   Dim pgName As String
   Dim i As Long

   Dim oMidCtx As IJMiddleContext
   Dim oDBTypeConfig As IJDBTypeConfiguration
   Dim oDataBaseConfig As IJDataBaseConfiguration
   Dim oACConfig As IJAccessControlConfiguration
   Dim oAccessControl As IJAccessControl
```

```
    ' default
    ConvertPGNumberToName = ""

    Set oDBTypeConfig = New DBTypeConfiguration
    Set oDataBaseConfig = New DataBaseConfiguration
    Set oACConfig = New AccessControlConfiguration

    Set oMidCtx = New GSCADMiddleContext
    oMidCtx.GetConfigurationTablesFromMiddle oDBTypeConfig, oDataBaseConfig, oACConfig
    Set oAccessControl = oACConfig.AccessControl
    count = oACConfig.NumberConditionIDs
    For i = 1 To count
        oACConfig.GetConditionIDByIndex i, pgName, PGID
        If PGID = PGnum Then
            ConvertPGNumberToName = pgName
            Exit Function
        End If
    Next i

Exit Function
ErrHndlr:
    Err.Clear
    ConvertPGNumberToName = ""
End Function
```

30. Re-compile the program using File -> Make IFCCustomRule.dll

31. Start the Check interference check service.

32. Change the interference checking process criteria by assign the interference priority to the Insulation Aspect as Required.



Assign interference checking priority to aspects:

| Aspect | Type |
|---|---|
| Simple physical | Required |
| Detailed physical | Not checked |
| Insulation | Required |
| Operation | Not checked |
| Maintenance | Not checked |
| Reference Geometry | Not checked |

33. Click Start button to start the interference detection process. Select Yes to re-check the entire Model. Select OK button for the process starts and begins running.

34. After 5 minutes, refresh/define workspace. You should see interferences created between the objects. The interference checking process should not create an interference object where colliding objects belong to “TESTPG”.

35. Stop the interference detection process.

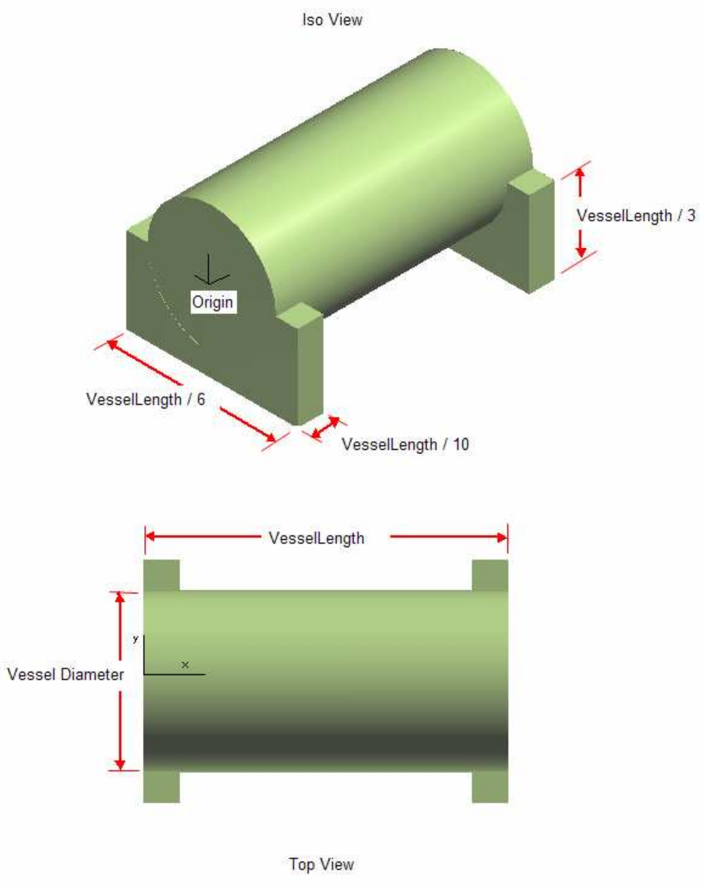
# Lab 2: Equipment Component Symbol

## Objectives

After completing this lab, you will be able to:

- Create a simple equipment component symbol
- Learn to use the Symbol Helper service to create the symbol definition
- Use the Equipment CAD Helper to define the Custom Assembly Definition
- Learn to use the IJDAggregatorDescription , IJDMemberDescriptions, and IJDPropertyDescriptions to define the behaviors of the custom assembly occurrence (CAO)
- Learn to use the Geometry Helper service to create simple geometric shapes for the symbol's output

In this lab, you will create an equipment component symbol as shown below. You start by using the SP3DEqpTemplateAsm template provided by the instructor to create the symbol. This symbol consists of three simple geometric shapes (one cylinder and two rectangular boxes) to define the symbol's output. Two inputs "VesselDiameter and "VesselLength" are required to draw this symbol.



1. Create the following directory:

   *c:\train*

2. Copy the Equipment Symbol Template Project provided by the instructor to

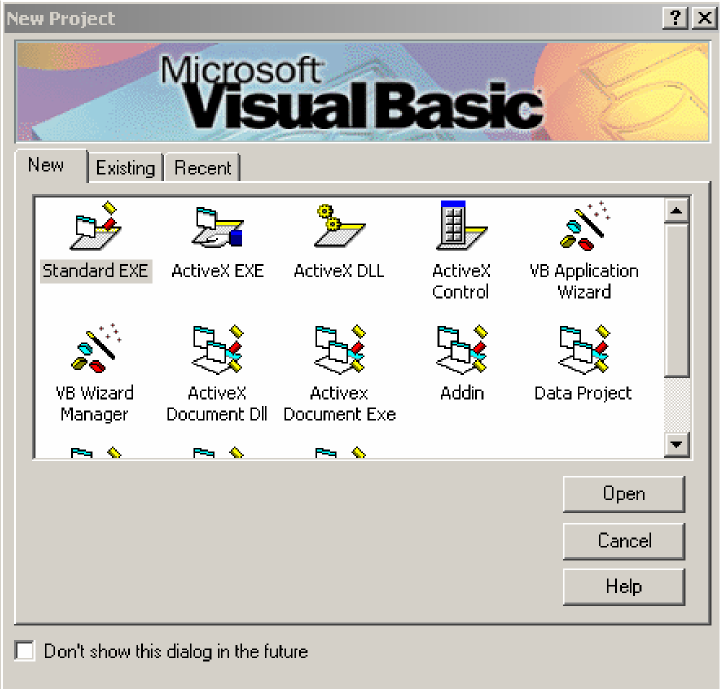
   *c:\train\EqpAsmTemplate*

   **Note:**
   - The EqpAsm template is delivered under [Installation]\Programming\ExampleCode\Symbols\EqpAsmTemplate

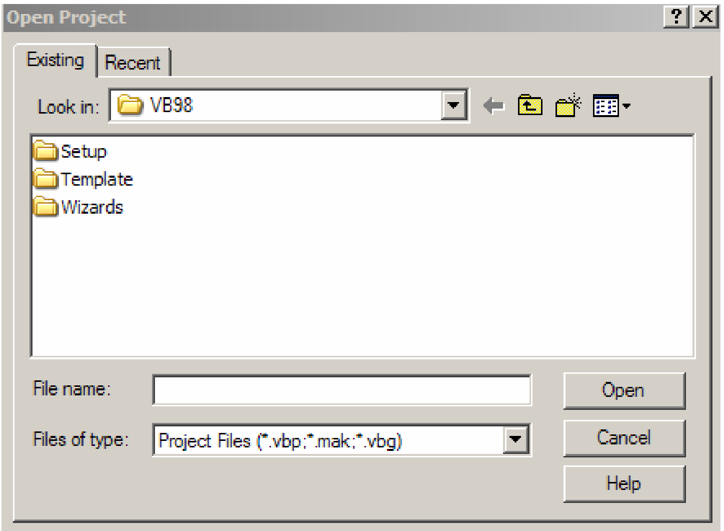
3. Create a directory called lab1 as follows:

   *c:\train\lab1*

4. Run Microsoft Visual Basic 6.0
5. Close the Microsoft New Project dialog box.



6. Select *File* -> *Open Project* option to open the Open Project Dialog box

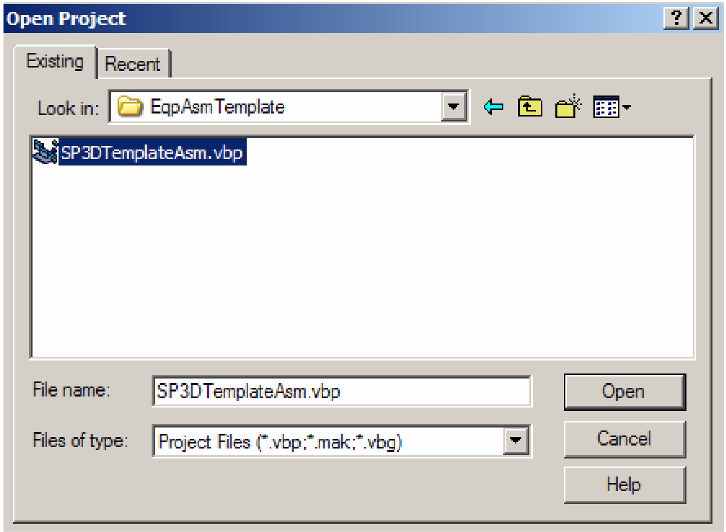


Look in: VB98

File name:

Files of type: Project Files (*.vbp;*.mak;*.vbg)

7. Navigate to c:\train\EqpAsmTemplate and open the SP3DTemplateAsm Template project

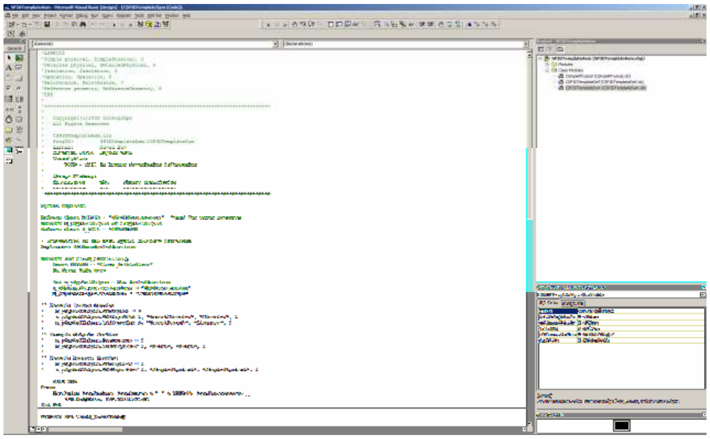


Look in: EqpAsmTemplate

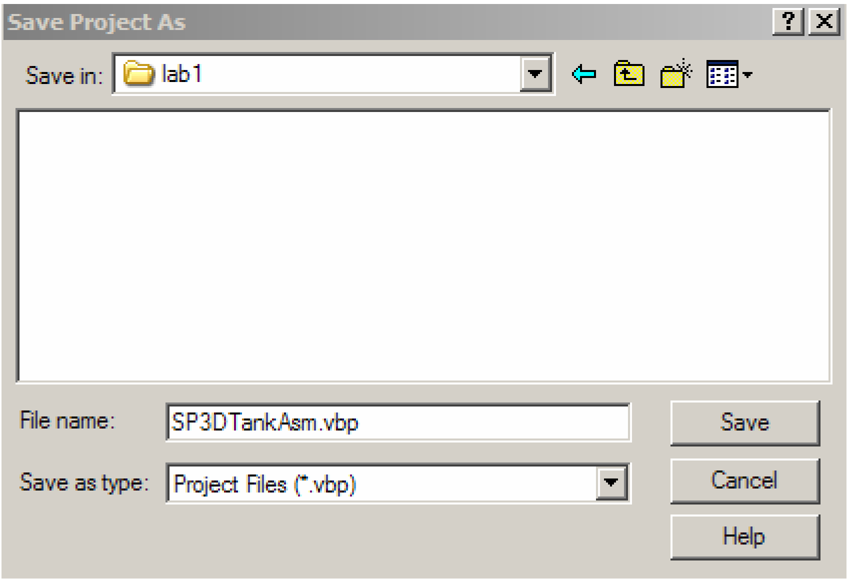
File name: SP3DTemplateAsm.vbp

Files of type: Project Files (*.vbp;*.mak;*.vbg)

8. Setup the Visual Basic Development Environment as shown below:



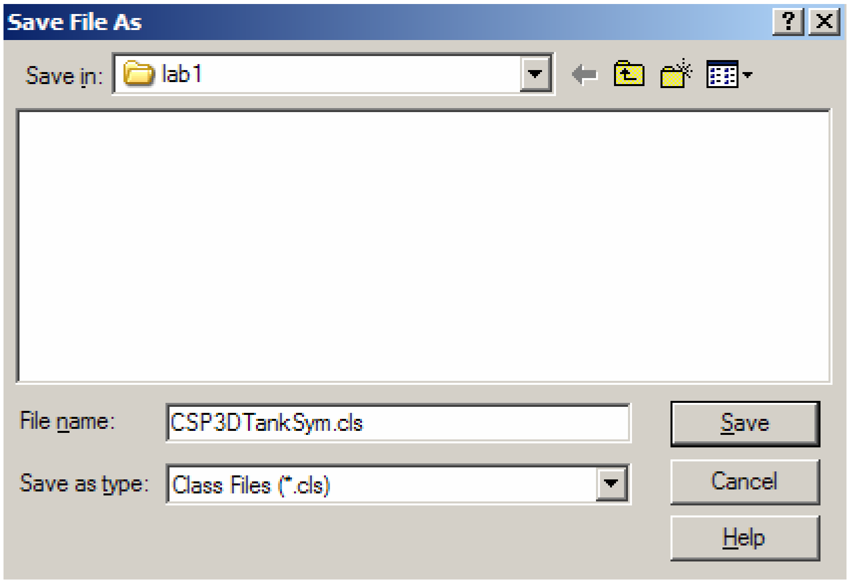
9. Go to the Visual Basic Explorer Window and select the Project node. Select *File -> Save Project As* option to save the project as SP3DTankAsm.vbp under the lab1 directory



File name: SP3DTankAsm.vbp

Save as type: Project Files (*.vbp)

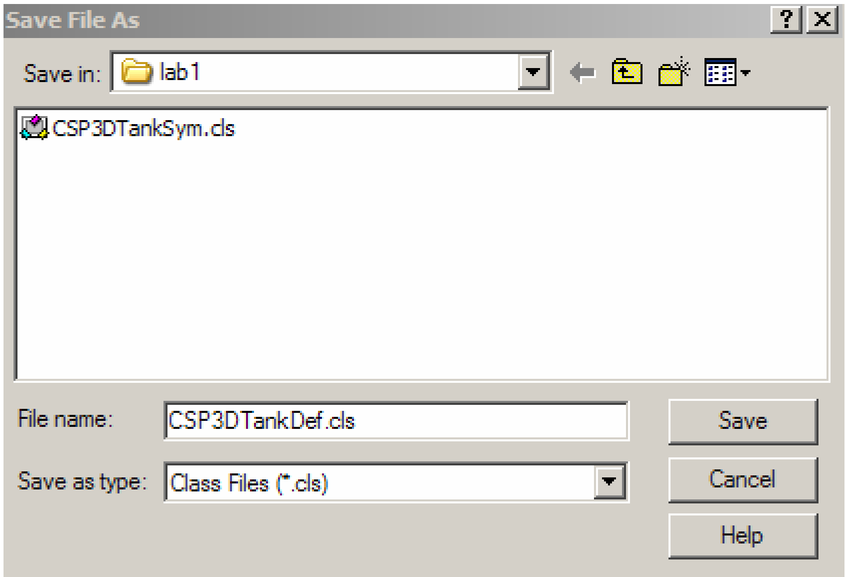
10. Go to the Visual Basic Explorer Window and select the CSP3DTemplateSym class node. Select *File -> Save CSP3DTemplateSym.cls As* option to save the class module as CSP3DTankSym.cls under lab1 directory



File name: CSP3DTankSym.cls

Save as type: Class Files (*.cls)

11. Go to the Visual Basic Explorer Window and select the CSP3DTemplateDef class node. Select *File -> Save CSP3DTemplateDef.cls As* option to save the class module as CSP3DTankDef.cls under lab1 directory



Save File As

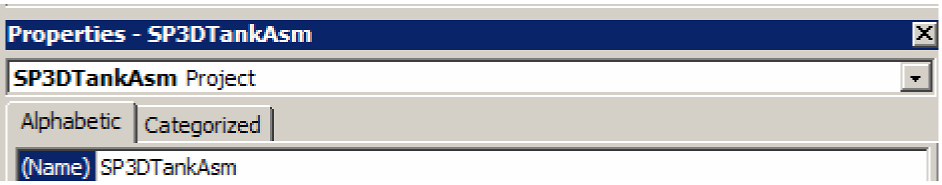
Save in: lab1

CSP3DTankSym.cls

File name: CSP3DTankDef.cls

Save as type: Class Files (*.cls)

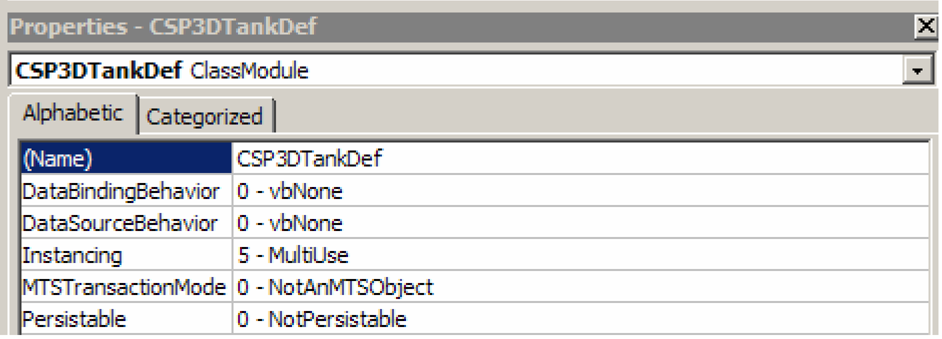
12. Go to the Visual Basic Explorer Window and select the CSimplePhysical class node. Select *File -> Save CSimplePhysical.cls As* option to save the class module as CSimplePhysical.cls under lab1 directory.
13. Go to the Properties Window and change the name of the Project and both Class Modules as shown here:



Properties - SP3DTankAsm

**SP3DTankAsm** Project

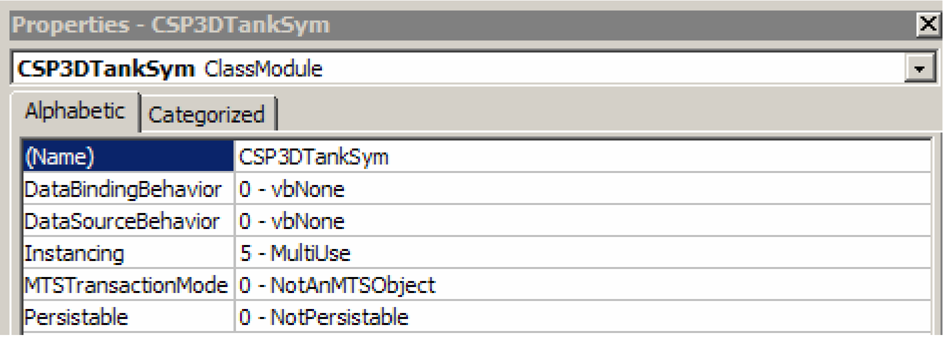
(Name) SP3DTankAsm



Properties - CSP3DTankDef

**CSP3DTankDef** ClassModule

| | |
|---|---|
| (Name) | CSP3DTankDef |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |



| | |
|---|---|
| (Name) | CSP3DTankSym |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |

14. Go to the General Declarations section in CSP3DTankSym module. Change the value of the *Constant Module variable* from *“CSP3DTemplateAsm:”* to *“CSP3DTankAsm:”*

```
Private Const MODULE = "CSP3DTankAsm:"  'Used for error messages
```

15. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
   Set m_oSymbolHelper = New SymbolServices
   m_oSymbolHelper.ProjectName = "SP3DTankAsm"
   m_oSymbolHelper.ClassName = "CSP3DTankSym"
```

16. In this Class_Initialize() routine, add the following code to define the inputs, outputs and aspects definition for this symbol.

```
'Inputs Section
   m_oSymbolHelper.NumInputs = 2
   m_oSymbolHelper.AddInputDef 1, "VesselDiameter", "Diameter", 1
   m_oSymbolHelper.AddInputDef 2, "VesselLength", "VesselLength", 1
'
'Outputs Section
   m_oSymbolHelper.NumOutputs = 3
   m_oSymbolHelper.AddOutputDef 1, "Body1", "Body1", 1
   m_oSymbolHelper.AddOutputDef 2, "Body2", "Body2", 1
   m_oSymbolHelper.AddOutputDef 3, "Body3", "Body3", 1
'
'Aspects Section
   m_oSymbolHelper.NumAspects = 1
   m_oSymbolHelper.AddAspectDef 1, "SimplePhysical", "SimplePhysical", 1
```

17. Go to the General Declarations section in CSP3DTankDef module. Change the value of the *Constant Module variable* from *"SP3DTemplateAsm:CSP3DTemplateDef"* to *"SP3DTankAsm:CSP3DTankDef"*

```
Private Const MODULE = "SP3DTankAsm:CSP3DTankDef"
```

18. Go to CSP3DTankDef Class module and rename the project name and class name as shown below:

```
   m_oEquipCADHelper.ProjectName = "SP3DTankAsm"
   m_oEquipCADHelper.ClassName = "CSP3DTankDef"
```

```
    m_oEquipCADHelper.OccurrenceRootClass = orcEquipment
```

19. Go to CSimplePhysical Class module and declare all variables for your inputs and outputs:

```
' Declare variables for Inputs and Outputs

    Dim parDiameter As Double
    Dim parLength As Double
    Dim ObjBody1 As Object
    Dim ObjBody2 As Object
    Dim ObjBody3 As Object
```

20. Use these variables to store the inputs as follows:

```
' Insert your code for inputs
    Set oPartFclt = arrayOfInputs(1)
    parDiameter = arrayOfInputs(2)
    parLength = arrayOfInputs(3)
```

21. Add code to define the first output. The following code will use the CreateCylinder() routine to create a cylinder for the Body1. The PlaceCylinder routine is defined in the geometry helper service. This function creates persistent projection of a circle based on two points and diameter.

```
' Insert your code for output (Body1)
    Dim pPos1 As IJDPosition
    Dim pPos2 As IJDPosition

    Set pPos1 = New DPosition
    Set pPos2 = New DPosition

    pPos1.Set 0, 0, 0
    pPos2.Set parLength, 0, 0
    iOutput = iOutput + 1
    Set ObjBody1 = m_oGeomHelper.CreateCylinder(arrayOfOutputs(iOutput), pPos1, pPos2, parDiameter)
```

22. Add code to define the second output. The following code will use the PlaceBox() routine to create a Box for the Body2. The PlaceBox routine is located in Geometry3d.bas module. This function takes the two opposite corners of the box as input parameters.

```
' Insert your code for output (Body2)
    pPos1.Set 0, -parLength / 3, -parLength / 3
    pPos2.Set parLength / 10, parLength / 3, 0
    Set ObjBody2 = PlaceBox(m_OutputColl, pPos1, pPos2)
    iOutput = iOutput + 1
    m_OutputColl.AddOutput arrayOfOutputs(iOutput), ObjBody2
```

23. Add code to define the third output. The following code will use the PlaceBox() routine to create a Box for the Body3. The PlaceBox routine is located in Geometry3d.bas module. This function takes the two opposite corners of the box as input parameters.

```
' Insert your code for output (Body3)
    pPos1.Set parLength - parLength / 10, -parLength / 3, -parLength / 3
```

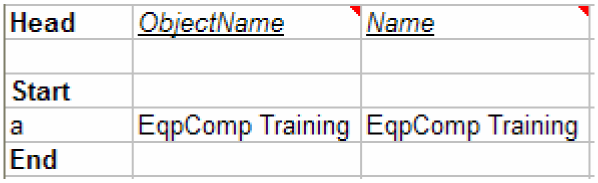
```
pPos2.Set parLength, parLength / 3, 0
Set ObjBody3 = PlaceBox(m_OutputColl, pPos1, pPos2)
iOutput = iOutput + 1
m_OutputColl.AddOutput arrayOfOutputs(iOutput), ObjBody3
```

24. Use the Set statement to clear the references from all object variables.

```
Set ObjBody1 = Nothing
Set ObjBody2 = Nothing
Set ObjBody3 = Nothing
Set pPos1 = Nothing
Set pPos2 = Nothing
```

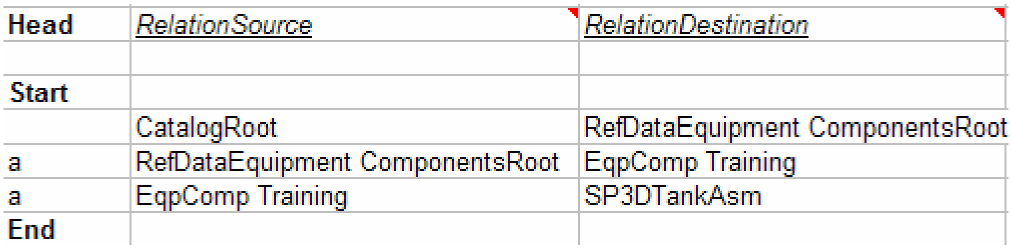
25. Compile the Visual Basic project and save the dll as SP3DTankAsm.dll in the c:\Train\lab1 Note: One of the most important steps in Visual Basic programming is to preserve the binary compatibility of your program. Save the final version of your dll file to be binary compatibility in order to preserve the CLSID.

26. Save the VB SP3DTankAsm project.
27. Open the SP3DTemplate.xls workbook. Go the ClassNodeType sheet and add the following entry.



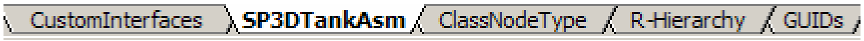
| Head | ObjectName | Name |
|---|---|---|
| | | |
| Start | | |
| a | EqpComp Training | EqpComp Training |
| End | | |

28. Go the R-Hierarchy sheet and add the following entry.



| Head | RelationSource | RelationDestination |
|---|---|---|
| | | |
| Start | | |
| | CatalogRoot | RefDataEquipment ComponentsRoot |
| a | RefDataEquipment ComponentsRoot | EqpComp Training |
| a | EqpComp Training | SP3DTankAsm |
| End | | |

29. Go to the SP3DTemplateAsm sheet and rename it as SP3DTankAsm.



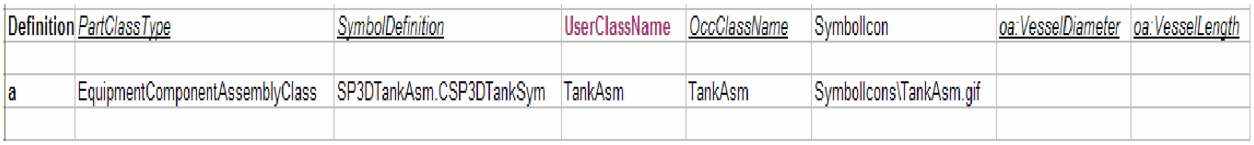
30. Go to the Class definition section and add/edit as follows:

In the Definition Section:

Notes:

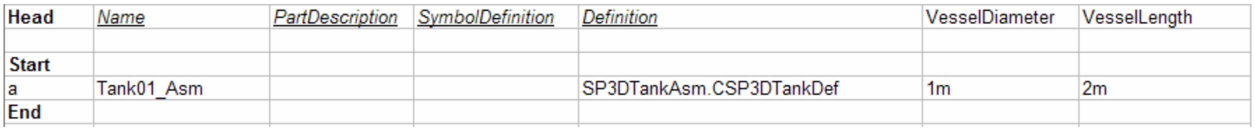
- Creating the bmp or gif file is optional. You can use Microsoft Paint to create the file and save it under your \\machine\symbols\SymbolIcons

- Make sure to change the Part Class Type to EquipmentComponentAssemblyClass



| Definition | PartClassType | SymbolDefinition | UserClassName | OccClassName | SymbolIcon | oa:VesselDiameter | oa:VesselLength |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| a | EquipmentComponentAssemblyClass | SP3DTankAsm.CSP3DTankSym | TankAsm | TankAsm | SymbolIcons\TankAsm.gif | | |
| | | | | | | | |

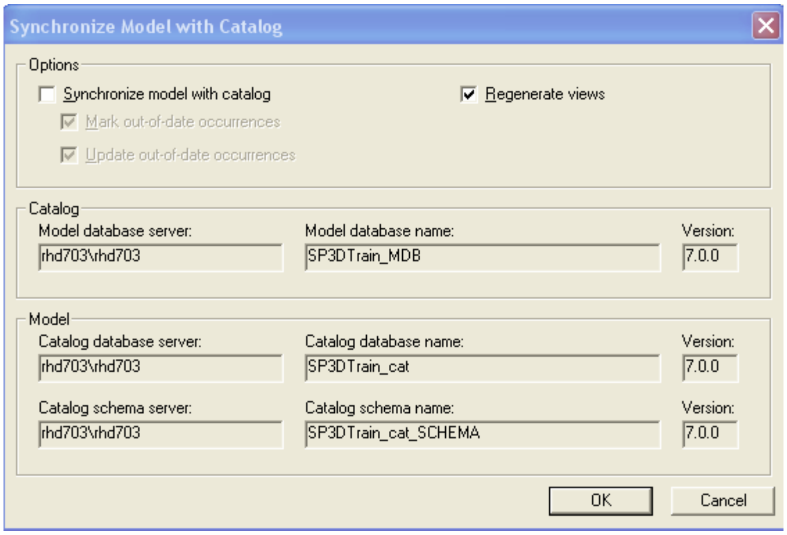
In the Part Section:



| Head | Name | PartDescription | SymbolDefinition | Definition | VesselDiameter | VesselLength |
|---|---|---|---|---|---|---|
| | | | | | | |
| Start | | | | | | |
| a | Tank01_Asm | | | SP3DTankAsm.CSP3DTankDef | 1m | 2m |
| End | | | | | | |

31. Save the Excel workbook as SP3DTankAsm.xls in the c:\Train\lab1.
32. Optional step: Create the TankAsm.gif file and place it under \\<MachineName>\Symbols\SymbolIcons
33. Load the information into the catalog using the Add/Modify/Delete Mode. Once the bulkload process is completed, review the log file.
34. Run the Project Management Task. Select the Model in the hierarchy.
35. Select Tools -> Synchronize Model with the Catalog.
36. Uncheck the Synchronize Model with the Catalog option.

***Note**: You just need to update the views in the model.*



Synchronize Model with Catalog

Options

☐ Synchronize model with catalog ☑ Regenerate views

☑ Mark out-of-date occurrences

☑ Update out-of-date occurrences

Catalog

Model database server: rhd703\rhd703 Model database name: SP3DTrain_MDB Version: 7.0.0

Model

Catalog database server: rhd703\rhd703 Catalog database name: SP3DTrain_cat Version: 7.0.0

Catalog schema server: rhd703\rhd703 Catalog schema name: SP3DTrain_cat_SCHEMA Version: 7.0.0

37. Hit “OK” Button.
38. Once the process is completed, Right click the training plant icon and select “Regenerate the Reports database” option to re-create the views in the report database.
39. Go to the Equipment Task and place the SP3DTankAsm.

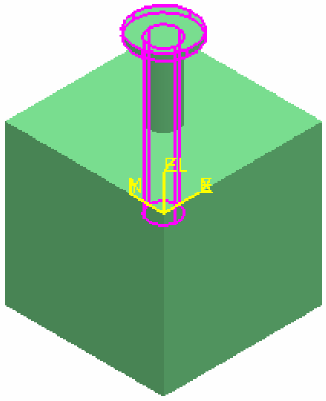
# Lab 3: Equipment Symbol with pipe port created from a placeholder

## Objectives

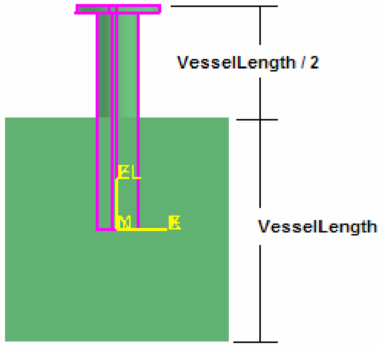
After completing this lab, you will be able to:

- Create a simple catalog equipment ymbol with pipe nozzle using a placeholder object defined in the symbol.
- Learn to use the Symbol Helper service to create the symbol definition
- Learn to use the Geometry Helper service to create simple geometric shapes for the symbol's output
- Use the Equipment CAD Helper to define the Custom Assembly Definition
- Learn to use the IJDAggregatorDescription , IJDMemberDescriptions, and IJDPropertyDescriptions to define the behaviors of the custom assembly occurrence (CAO)
- Use custom methods to create and manipulate the members within the CAD definition
- Use the IJDeletableMember interface to make the member deletable

In this lab, you will create an equipment symbol as shown below. You start by using the SP3DEqpTemplateAsm template provided by the instructor to create the symbol. This symbol consists of one geometric rectangular entity and a pipe nozzle to define the symbol's output. One input "VesselLength" is required to draw this symbol. Use the Equipment Custom Assembly Definition (CAD) Helper to create the pipe port from the placeholder defined in the symbol. The pipe port data is retrieved from the part at the given index. This type of creation is used when the position and the orientation of the pipe nozzle are driven totally or partially by the symbol.



Isometric View

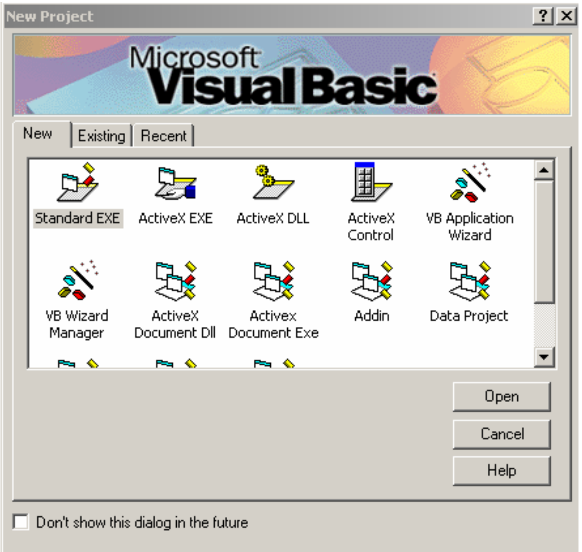


Elevation View

1. Create a directory called lab2 as follows:

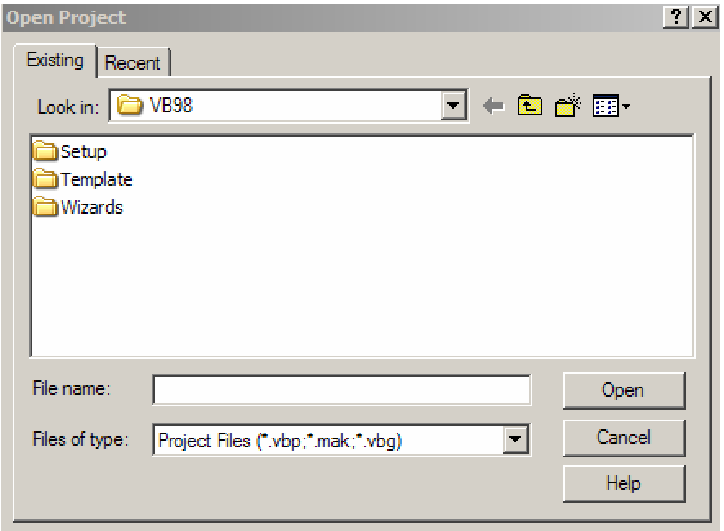
   *c:\train\lab2*

2. Run Microsoft Visual Basic 6.0
3. Close the Microsoft New Project dialog box.



□ Don't show this dialog in the future

4. Select *File* -> *Open Project* option to open the Open Project Dialog box

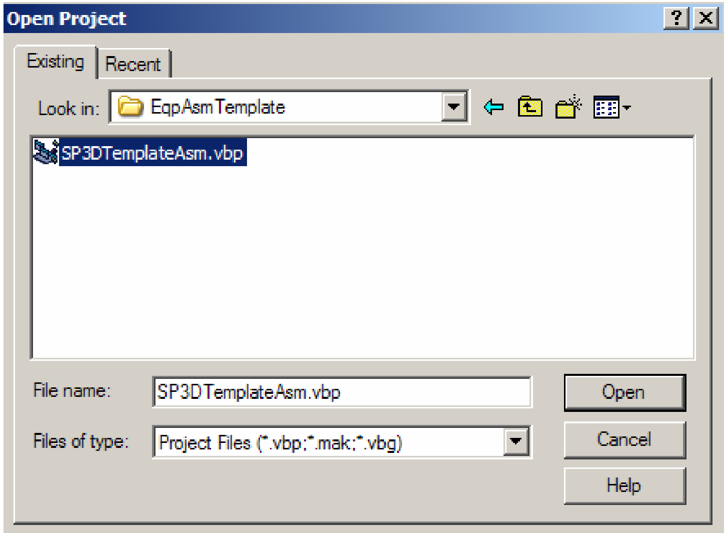


Look in: VB98

File name:

Files of type: Project Files (*.vbp;*.mak;*.vbg)

5. Navigate to c:\train\EqpAsmTemplate and open the SP3DTemplateAsm Template project.



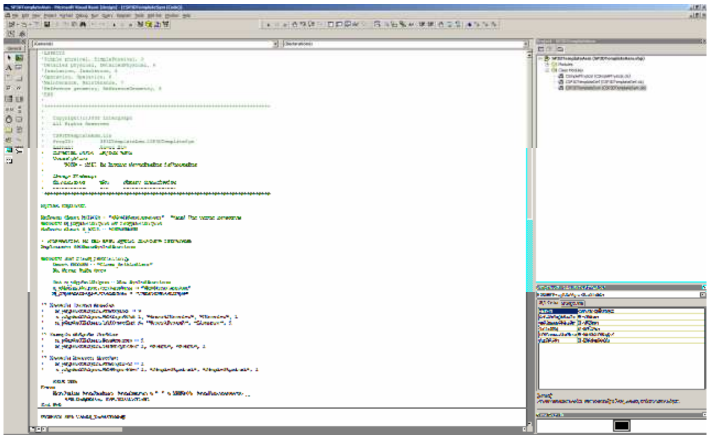
Look in: EqpAsmTemplate

SP3DTemplateAsm.vbp

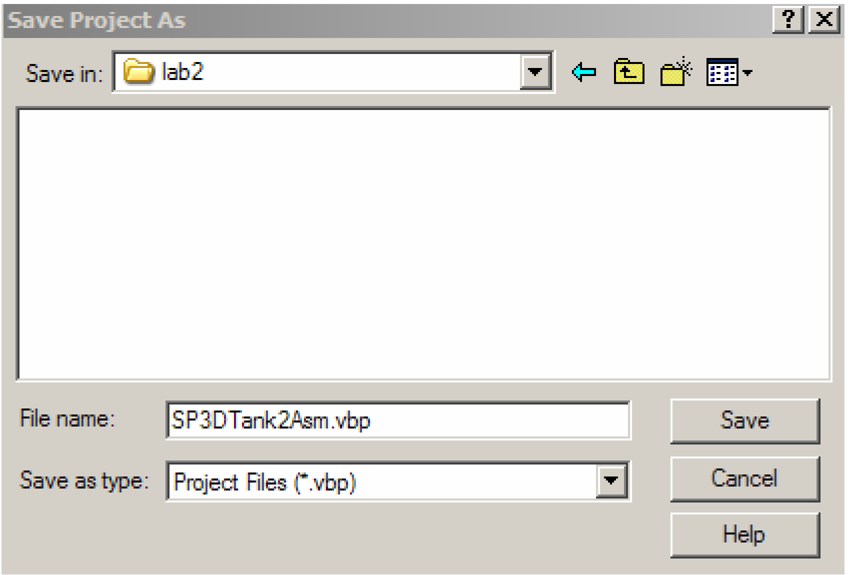
File name: SP3DTemplateAsm.vbp

Files of type: Project Files (*.vbp;*.mak;*.vbg)

6. Setup the Visual Basic Development Environment as shown below:



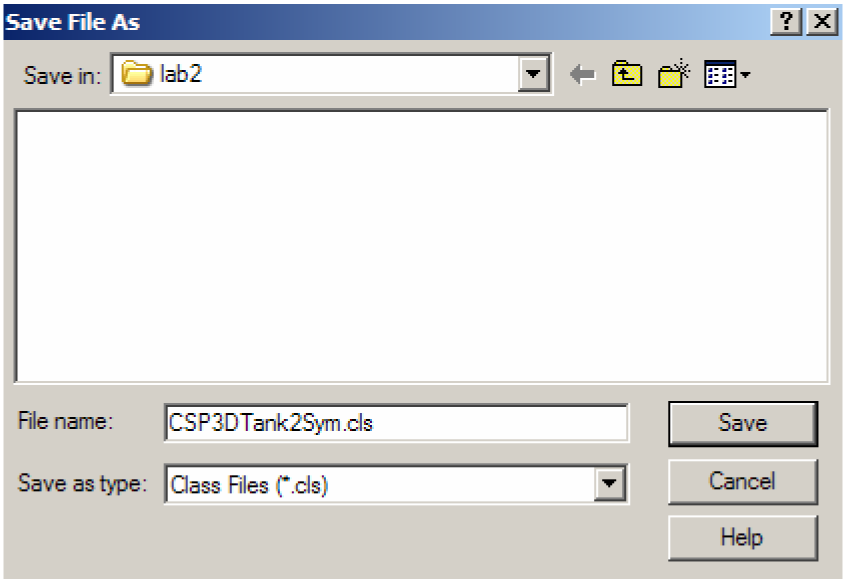
7. Go to the Visual Basic Explorer Window and select the Project node. Select *File -> Save Project As* option to save the project as SP3DTank2Asm.vbp under the lab2 directory



File name: SP3DTank2Asm.vbp

Save as type: Project Files (*.vbp)

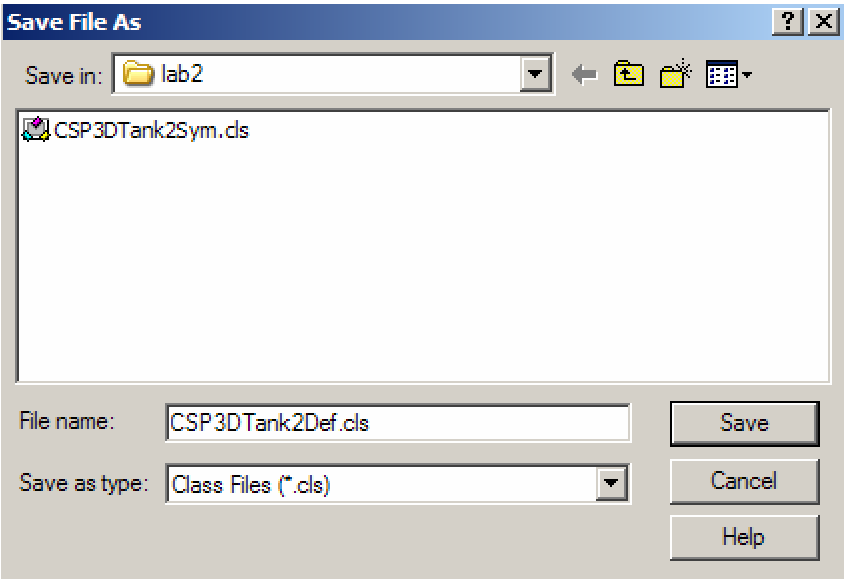
8. Go to the Visual Basic Explorer Window and select the CSP3DTemplateSym class node. Select *File -> Save CSP3DTemplateSym.cls As* option to save the class module as CSP3DTank2Sym.cls under lab2 directory



File name: CSP3DTank2Sym.cls

Save as type: Class Files (*.cls)

9. Go to the Visual Basic Explorer Window and select the CSP3DTemplateDef class node. Select *File -> Save CSP3DTemplateDef.cls As* option to save the class module as CSP3DTank2Def.cls under lab2 directory



Save in: lab2

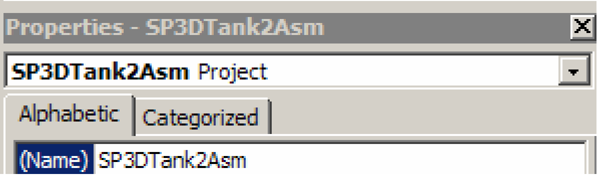
CSP3DTank2Sym.cls

File name: CSP3DTank2Def.cls

Save as type: Class Files (*.cls)

Go to the Visual Basic Explorer Window and select the CSimplePhysical class node. Select *File -> Save CSimplePhysical.cls As* option to save the class module as CSimplePhysical.cls under lab2 directory.

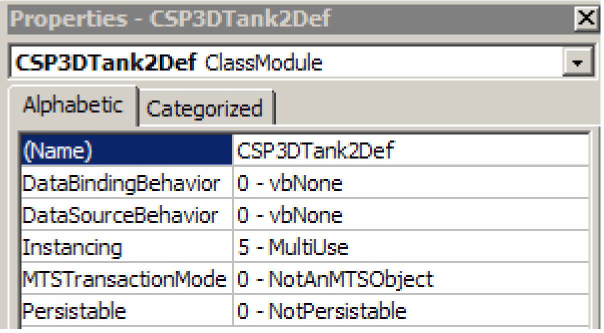
10. Go to the Properties Window and change the name of the Project and both Class Modules as follows:



Properties - SP3DTank2Asm

**SP3DTank2Asm** Project

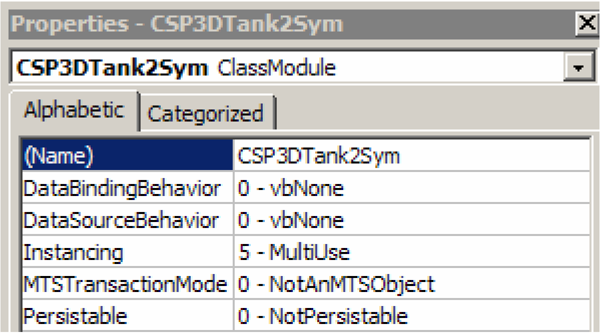
| | |
|---|---|
| (Name) | SP3DTank2Asm |



Properties - CSP3DTank2Def

**CSP3DTank2Def** ClassModule

| | |
|---|---|
| (Name) | CSP3DTank2Def |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |



| | |
|---|---|
| (Name) | CSP3DTank2Sym |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |

11. Go to the General Declarations section in CSP3DTank2Sym module and change the value of the *Constant Module variable* from *“CSP3DTemplateAsm:”* to *“CSP3DTank2Asm:”*

```
Private Const MODULE = "CSP3DTank2Asm:"  'Used for error messages
```

12. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
    Set m_oSymbolHelper = New SymbolServices
    m_oSymbolHelper.ProjectName = "SP3DTank2Asm"
    m_oSymbolHelper.ClassName = "CSP3DTank2Sym"
```

13. In this Class_Initialize() routine, add the following code to define the inputs, outputs and aspects definition for this symbol.

```
'Inputs Section
    m_oSymbolHelper.NumInputs = 1
    m_oSymbolHelper.AddInputDef 1, "VesselLength", “VesselLength", 1
'
'Outputs Section
    m_oSymbolHelper.NumOutputs = 2
    m_oSymbolHelper.AddOutputDef 1, "Body1", "Body1", 1
    m_oSymbolHelper.AddOutputDef 2, "PipingNoz1", "PipingNoz1", 1

'Aspects Section
    m_oSymbolHelper.NumAspects = 1
    m_oSymbolHelper.AddAspectDef 1, "SimplePhysical", "SimplePhysical", 1
```

14. Go to the General Declarations section in CSP3DTank2Def module and change the value of the *Constant Module variable* from *"SP3DTemplateAsm:CSP3DTemplateDef"* to *"SP3DTank2Asm:CSP3DTank2Def"*

```
Private Const MODULE = "SP3DTank2Asm:CSP3DTank2Def"
```

15. Go to CSP3DTank2Def Class module and rename the project name and class name as shown below:

```
     m_oEquipCADHelper.ProjectName = "SP3DTank2Asm"
    m_oEquipCADHelper.ClassName = "CSP3DTank2Def"
```

16. Go to CSimplePhysical Class module and declare all variables for your inputs and outputs:

```
    Dim parLength As Double
    Dim ObjBody1 As Object

    Dim pipeDiam        As Double
    Dim flangeThick     As Double
    Dim cptOffset       As Double
    Dim flangeDiam      As Double
    Dim depth           As Double
```

17. Uses these variables to store the inputs as follows:

```
' Insert your code for inputs
    Set oPartFclt = arrayOfInputs(1)
    parLength = arrayOfInputs(2)
```

18. Go to the Insert your code for output (Body1) section. The following code will use the PlaceBox() routine to create a Box for the Body1. The PlaceBox routine is located at geometry3d.bas module. This function takes the two opposite corner points of the box as input parameters.

```
' Insert your code for output (Body1)
    Dim pPos1 As IJDPosition
    Dim pPos2 As IJDPosition

    Set pPos1 = New DPosition
    Set pPos2 = New DPosition

    pPos1.Set -parLength / 2, -parLength / 2, -parLength / 2
    pPos2.Set parLength / 2, parLength / 2, parLength / 2
    Set ObjBody1 = PlaceBox(m_OutputColl, pPos1, pPos2)
    iOutput = iOutput + 1
    m_OutputColl.AddOutput arrayOfOutputs(iOutput), ObjBody1
```

19. Declare the following variables to store the nozzle object, nozzle position and nozzle orientation.

```
    Dim oDir        As AutoMath.DVector
    Dim objNozzle   As IJDNozzle
    Set oDir = New AutoMath.DVector
    Dim CenterPos       As New AutoMath.DPosition

    RetrieveParameters 1, oPartFclt, m_OutputColl, pipeDiam, flangeThick, flangeDiam, cptOffset, depth

    CenterPos.Set 0, 0, parLength - depth + cptOffset
    oDir.Set 0, 0, 1
```

20. Use the CreateNozzlePHWithLength() to define the placeholder.

```
    Set objNozzle = CreateNozzlePHWithLength(1, oPartFclt, m_OutputColl, oDir, _
                                             CenterPos, parLength - depth + cptOffset)

' Set the output
    iOutput = iOutput + 1
```

```
m_OutputColl.AddOutput arrayOfOutputs(iOutput), objNozzle
```

21. Use the Set statement to clear the references from all object variables.

```
'Release BO 's
'
    Set ObjBody1 = Nothing
    Set pPos1 = Nothing
    Set pPos2 = Nothing
    Set objNozzle = Nothing
    Set CenterPos = Nothing
    Set oDir = Nothing
```

22. Go to CSP3DTank2Def Class module. Declare the appropriate custom methods to manage the pipe nozzle as follows

```
'Add your code here for the declaration of the Public Custom Methods used to manage new members
'Add new member(NozzleN1) to the definition

Set oMemberDescription = Nothing
Set oMemberDescription = oMemberDescriptions.AddMember("NozzleN1", 1, _
          "CMConstructNozzleN1", imsCOOKIE_ID_USS_LIB)
oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsNozzleN1"
oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructNozzleN1"
oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalNozzleN1"
oMemberDescription.SetCMCount imsCOOKIE_ID_USS_LIB, "CMCountNozzleN1"
oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseNozzleN1"

'Add properties for (NozzleN1)
Set oPropertyDescriptions = Nothing
Set oPropertyDescriptions = oMemberDescription
oPropertyDescriptions.AddProperty "NozzleN1Properties", 1, _
        IID_IJDATTRIBUTES, "CMEvaluateNozzleN1", imsCOOKIE_ID_USS_LIB
oPropertyDescriptions.AddProperty "NozzleN1GeometryProperties", 2,  _
        IID_IJDGEOMETRY, "CMEvaluateGeometryNozzleN1", imsCOOKIE_ID_USS_LIB
```

23. Go to IJEquipUserAttrMgmt_OnAttributeChange function and add the following code. This OnAttributeChange method is called each time an attribute is changed. When the “Can be Deleted” property is changed, the MakeMemberDeletable should be called to make the member deletable.

```
Private Function IJEquipUserAttrMgmt_OnAttributeChange(ByVal pIJDAttrs As IJDAttributes, ByVal
CollAllDisplayedValues As Object, ByVal pAttrToChange As IJEquipAttrDescriptor, ByVal varNewAttrValue
As Variant) As String
    Const METHOD = "IJEquipUserAttrMgmt_OnAttributeChange"
    On Error GoTo ErrorHandler

' Add code here

    Dim oMemberDescription As IJDMemberDescription
    Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)

    Select Case oMemberDescription.Name
        Case "NozzleN1"
```

```
            Select Case UCase(pAttrToChange.InterfaceName)
               Case "IJDELETABLEMEMBER"
                  If UCase(pAttrToChange.AttrName) = "CANBEDELETED" Then
                              m_oEquipCADHelper.MakeMemberDeletable oMemberDescription, pIJDAttrs,  _
                                 CBool(varNewAttrValue)
                  End If
               Case Else
                  '
            End Select
         Case Else
            '
      End Select

      IJEquipUserAttrMgmt_OnAttributeChange = ""

      Exit Function
   ErrorHandler:
      IJEquipUserAttrMgmt_OnAttributeChange = "ERROR"
      HandleError MODULE, METHOD
   End Function
```

24. Go to the end of CSP3DTank2Def Class module. Add the custom methods to manage the pipe nozzle as follows:

**Custom Method Construct**:
This method is in charge of the creation of the CAO member object (pipe nozzle). Use CreateNozzleFromPH() method to create a pipe nozzle from a nozzle place holder defined in the equipment symbol.

```
' Custom Methods for NozzleN1
Public Sub CMConstructNozzleN1(ByVal pMemberDescription As IJDMemberDescription, _
                           ByVal pResourceManager As IUnknown, _
                           ByRef pObject As Object)
   Const METHOD = "CMConstructNozzleN1"
   On Error GoTo ErrorHandler

   'Create Nozzle
   m_oEquipCADHelper.CreateNozzleFromPH pMemberDescription, pResourceManager, pObject, 1

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom Method Final:**
There is no need to add any code for this custom method

```
Public Sub CMFinalConstructNozzleN1(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMFinalConstructNozzleN1"
   On Error GoTo ErrorHandler
   Exit Sub

ErrorHandler:
   HandleError MODULE, METHOD
```

```
End Sub
```

### Custom method Inputs:

There is no need to add any code for this custom method

```
Public Sub CMSetInputsNozzleN1(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMSetInputsNozzleN1"
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

### Custom method Evaluate:

There is no need to add any code for this custom method

```
Public Sub CMEvaluateNozzleN1(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
   Const METHOD = "CMEvaluateNozzleN1"
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

### Custom method EvaluateGeometry:

This custom method will keep the nozzle’s position the same as the nozzle placeholder in the symbol.

```
Public Sub CMEvaluateGeometryNozzleN1(ByVal oPropertyDescription As IJDPropertyDescription, pObject
As Object)
   Const METHOD = "CMEvaluateGeometryNozzleN1"
   On Error GoTo ErrorHandler

   'Transform the nozzle so that it behaves like a rigid body inside the equipment
   m_oEquipCADHelper.TransformNozzleWrtPH oPropertyDescription, pObject, 1

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

### Custom method Conditional:

This method checks if the member is conditional based on the CanBeDeleted flag. Remember, we added code to make a member deletable in the IJEquipUserAttrMgmt_OnAttributeChange function.When the property is changed, the MakeMemberDeletable is called to check the CanBeDeleted flag and whether or not to make the member deletetable.

```
Public Sub CMConditionalNozzleN1(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As
Boolean)
    Const METHOD = "CMConditionalNozzleN1"
    On Error GoTo ErrorHandler

    IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom method Count:**
There is no need to add any code for this custom method

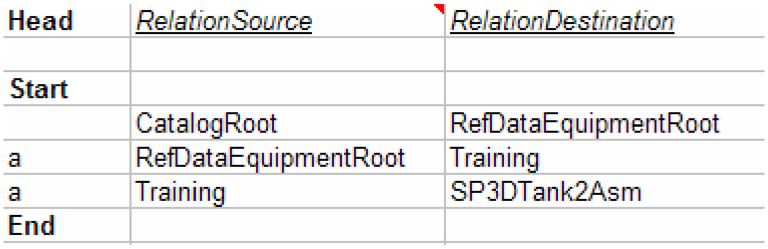
```
Public Sub CMCountNozzleN1(ByVal pMemberDesc As IJDMemberDescription, ByRef Count As Long)
    Const METHOD = "CMCountNozzleN1"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom method Release:**
There is no need to add any code for this custom method

```
Public Sub CMReleaseNozzleN1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMReleaseNozzleN1"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

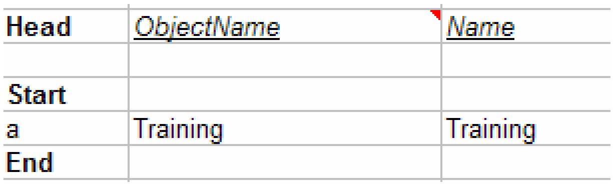
25. Compile the Visual Basic project and save the dll as SP3DTank2Asm.dll in the c:\Train\lab2 One of the most important steps in Visual Basic programming is to preserve the binary compatibility of your program. Save the final version of your dll file to be binary compatibility in order to preserve the CLSID.

26. Save the Visual Basic SP3DTank2Asm project.
27. Open the SP3DTemplate.xls workbook. Go the R-Hierarchy sheet and add the following entry.



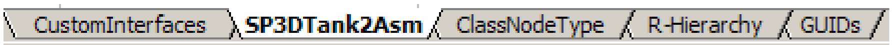
| Head | RelationSource | RelationDestination |
|---|---|---|
| | | |
| Start | | |
| | CatalogRoot | RefDataEquipmentRoot |
| a | RefDataEquipmentRoot | Training |
| a | Training | SP3DTank2Asm |
| End | | |

28. Go the ClassNodeType sheet and add the following entry.



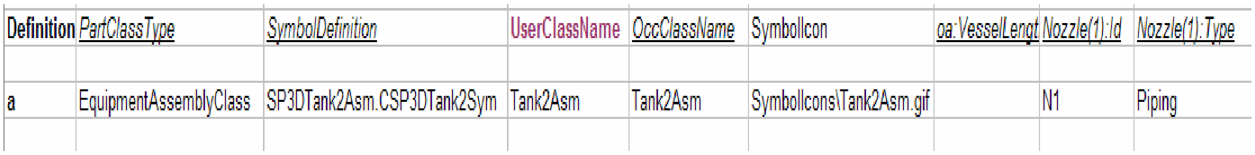
| Head | ObjectName | Name |
|---|---|---|
| | | |
| Start | | |
| a | Training | Training |
| End | | |

29. Go to the SP3DTemplateAsm sheet and rename it as SP3DTank2Asm.



30. Go to the Class definition section and add/edit as follows:

In the Class Definition Section rows:

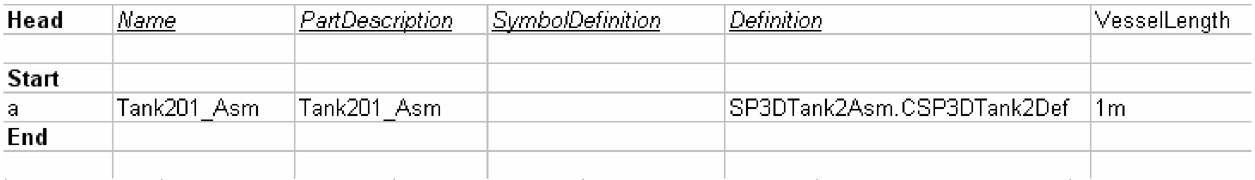


| Definition | PartClassType | SymbolDefinition | UserClassName | OccClassName | SymbolIcon | oa:VesselLengt | Nozzle(1):Id | Nozzle(1):Type |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| a | EquipmentAssemblyClass | SP3DTank2Asm.CSP3DTank2Sym | Tank2Asm | Tank2Asm | SymbolIcons\Tank2Asm.gif | | N1 | Piping |
| | | | | | | | | |

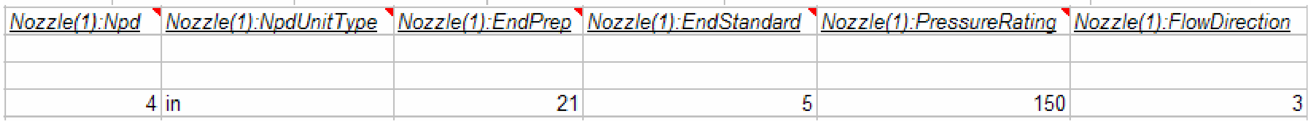
Note:

- Creating the bmp or gif file is optional. You can use Microsoft Paint to create the file and save it under your \\machine\symbols\SymbolIcons

In the Part Section rows:



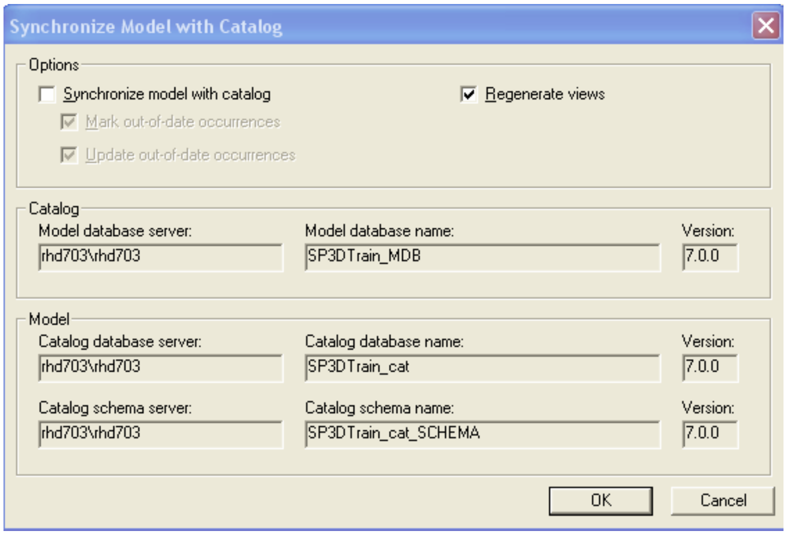
| Head | Name | PartDescription | SymbolDefinition | Definition | VesselLength |
|---|---|---|---|---|---|
| | | | | | |
| Start | | | | | |
| a | Tank201_Asm | Tank201_Asm | | SP3DTank2Asm.CSP3DTank2Def | 1m |
| End | | | | | |



| Nozzle(1):Npd | Nozzle(1):NpdUnitType | Nozzle(1):EndPrep | Nozzle(1):EndStandard | Nozzle(1):PressureRating | Nozzle(1):FlowDirection |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| 4 | in | 21 | 5 | 150 | 3 |

31. Save the Excel workbook as SP3DTank2Asm.xls in the c:\Train\lab2.
32. Optional step: Create the Tank2Asm.gif file and place it under \\<MachineName>\Symbols\SymbolIcons
33. Load the information into the catalog using the Add/Modify/Delete Mode. Once the bulkload process is completed, review the log file.
34. Run the Project Management Task. Select the Model in the hierarchy.
35. Select Tools -> Synchronize Model with the Catalog.
36. Uncheck the Synchronize Model with the Catalog option.

***Note**: You just need to update the views in the model.*



Synchronize Model with Catalog

Options

☐ Synchronize model with catalog

☑ Mark out-of-date occurrences

☑ Update out-of-date occurrences

☑ Regenerate views

Catalog

| Model database server: | Model database name: | Version: |
|---|---|---|
| rhd703\rhd703 | SP3DTrain_MDB | 7.0.0 |

Model

| Catalog database server: | Catalog database name: | Version: |
|---|---|---|
| rhd703\rhd703 | SP3DTrain_cat | 7.0.0 |

| Catalog schema server: | Catalog schema name: | Version: |
|---|---|---|
| rhd703\rhd703 | SP3DTrain_cat_SCHEMA | 7.0.0 |

37. Hit “OK” Button.
38. Once the process is completed, Right click the training plant icon and select “Regenerate the Reports database” option to re-create the views in the report database.
39. Go to the Equipment Task and place the SP3DTank2Asm.

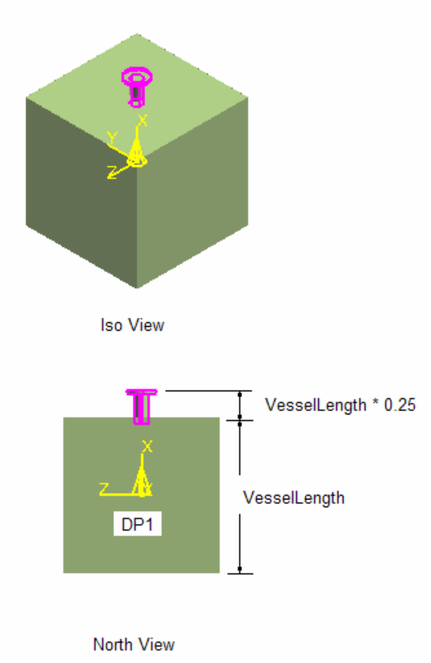
# Lab 4: Equipment Symbol with pipe port created relative to a shape

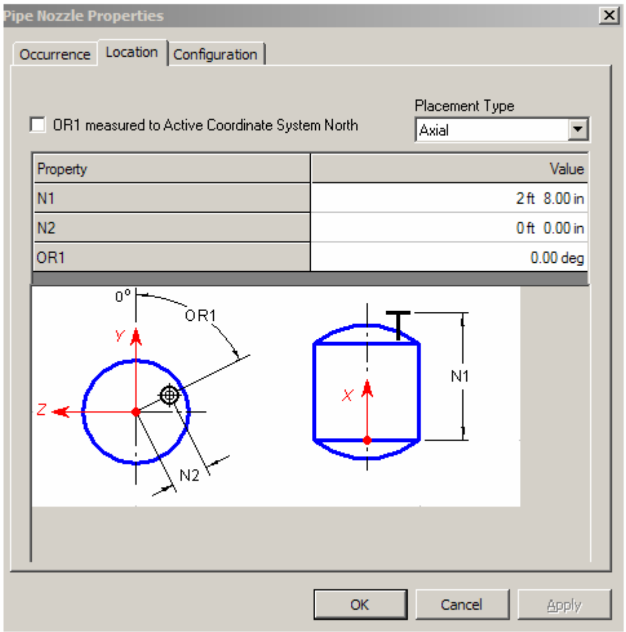
## Objectives

After completing this lab, you will be able to:

- Create a simple catalog equipment symbol with pipe nozzle created relative to a shape
- Learn to use the Symbol Helper service to create the symbol definition
- Learn to use the Geometry Helper service to create simple geometric shapes for the symbol's output
- Use the Equipment CAD Helper to define the Custom Assembly Definition
- Learn to use the IJDAggregatorDescription , IJDMemberDescriptions, and IJDPropertyDescriptions to define the behaviors of the custom assembly occurrence (CAO)
- Use the IJDeletableMember interface to make the member deletable
- Use the IJEquipUserAttrMgmt Interface to show the datum shape attributes as read only on the property page
- Use the CreateOrientationAndSetRelations() function to create the nozzle position and nozzle orientation relative to a shape

In this lab, you will create an equipment symbol as shown below. You start by using the SP3DEqpTemplateAsm template provided by the instructor to create the symbol. This symbol consists of one geometric rectangular entity to define the symbol's output. One input "VesselLength" is required to draw this symbol.  Use the Equipment Custom Assembly Definition (CAD) Helper to create the pipe port relative to the datum shape. This type of creation is used in conjunction with the creation of another member of type shape relative to which an orientation is given. This method will allow users to have more control and at the same time more freedom to positioning the nozzle.




Pipe Nozzle Properties

☐ OR1 measured to Active Coordinate System North

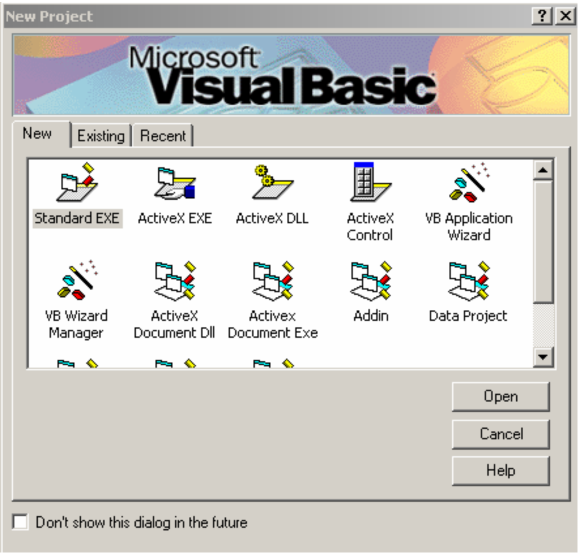
Placement Type: Axial

| Property | Value |
|---|---|
| N1 | 2ft 8.00 in |
| N2 | 0ft 0.00 in |
| OR1 | 0.00 deg |

1. Create a directory called lab3 as follows:

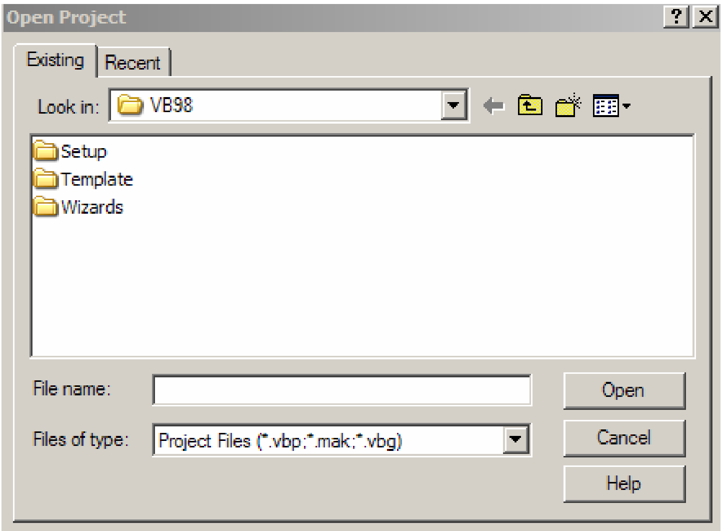
   *c:\train\lab3*

2. Run Microsoft Visual Basic 6.0
3. Close the Microsoft New Project dialog box.



☐ Don't show this dialog in the future

4. Select *File -> Open Project* option to open the Open Project Dialog box

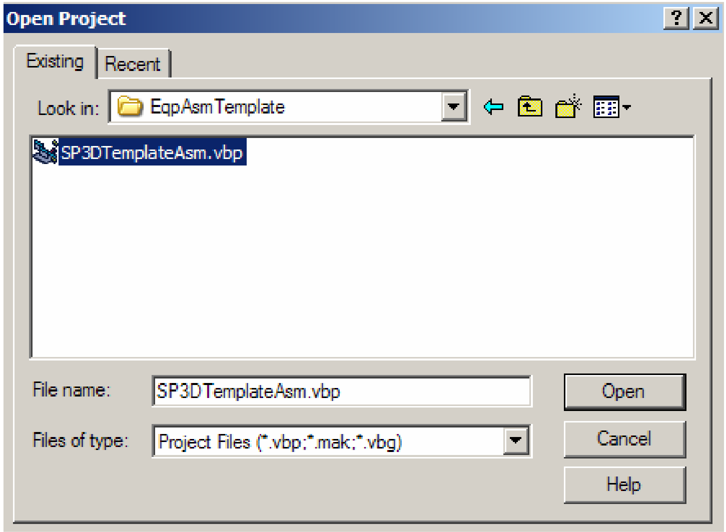


Look in: VB98

File name:

Files of type: Project Files (*.vbp;*.mak;*.vbg)

5. Navigate to c:\train\EqpAsmTemplate and open the SP3DTemplateAsm Template project

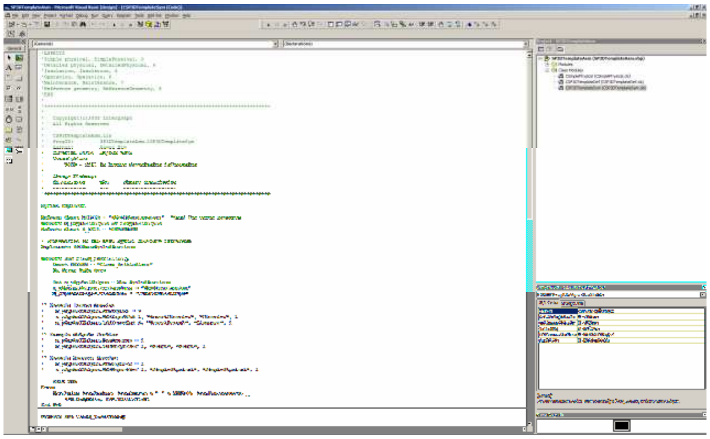


Look in: EqpAsmTemplate

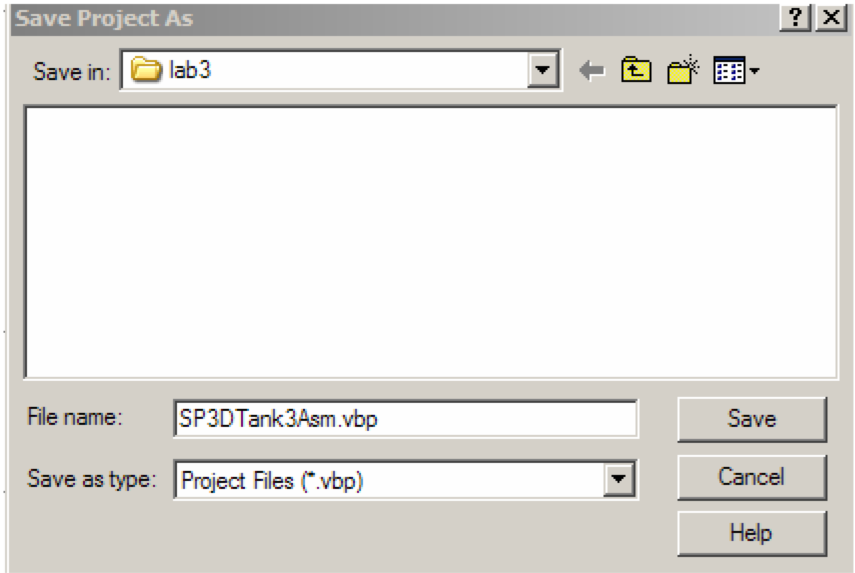
File name: SP3DTemplateAsm.vbp

Files of type: Project Files (*.vbp;*.mak;*.vbg)

6. Setup the Visual Basic Development Environment as shown below:



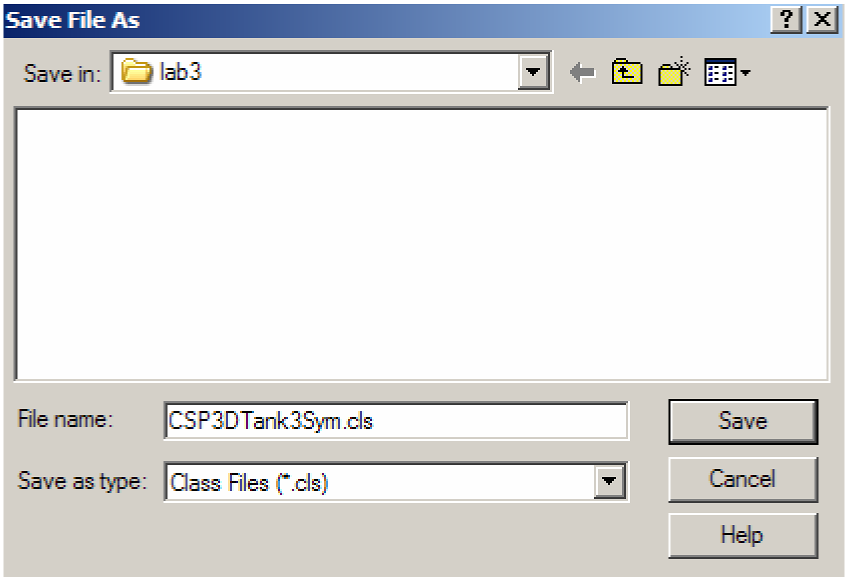
7. Go to the Visual Basic Explorer Window and select the Project node. Select *File -> Save Project As* option to save the project as SP3DTank3Asm.vbp under the lab3 directory



File name: SP3DTank3Asm.vbp

Save as type: Project Files (*.vbp)

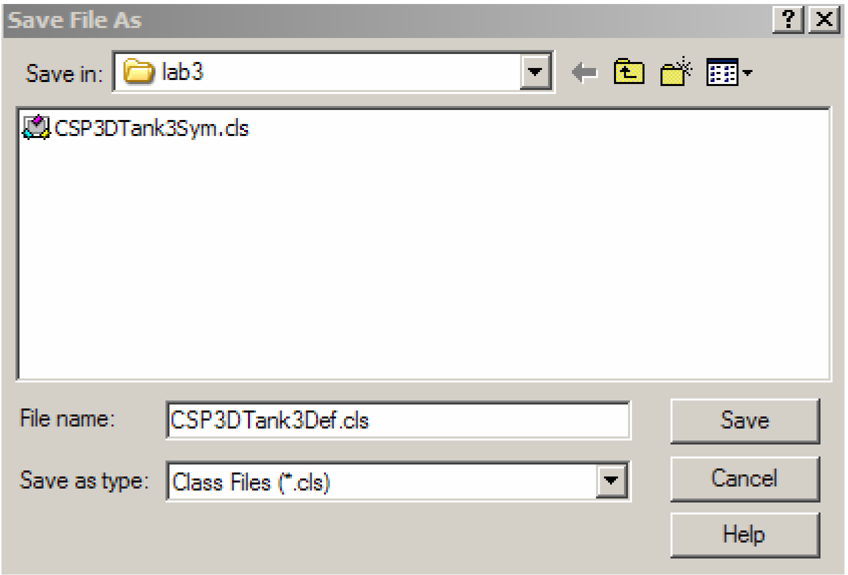
8. Go to the Visual Basic Explorer Window and select the CSP3DTemplateSym class node. Select *File -> Save CSP3DTemplateSym.cls As* option to save the class module as CSP3DTank3Sym.cls under lab3 directory



File name: CSP3DTank3Sym.cls

Save as type: Class Files (*.cls)

9. Go to the Visual Basic Explorer Window and select the CSP3DTemplateDef class node. Select *File -> Save CSP3DTemplateDef.cls As* option to save the class module as CSP3DTank3Def.cls under lab3 directory



Save File As

Save in: lab3

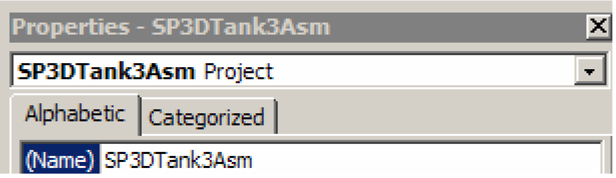
CSP3DTank3Sym.cls

File name: CSP3DTank3Def.cls

Save as type: Class Files (*.cls)

Go to the Visual Basic Explorer Window and select the CSimplePhysical class node. Select *File -> Save CSimplePhysical.cls As* option to save the class module as CSimplePhysical.cls under lab3 directory.

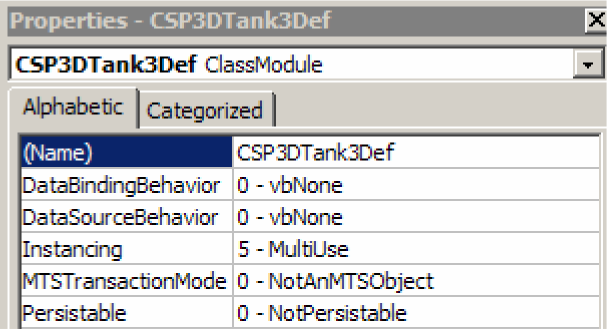
10. Go to the Properties Window and change the name of the Project and both Class Modules as follows:



Properties - SP3DTank3Asm

**SP3DTank3Asm** Project

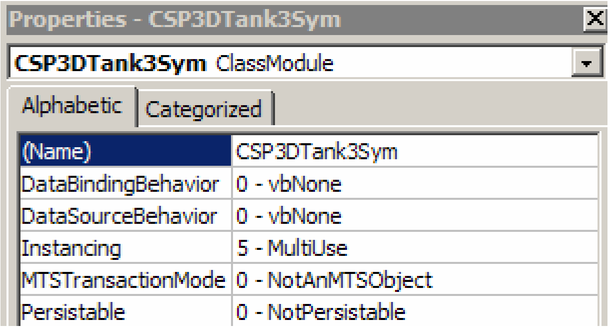
| | |
|---|---|
| (Name) | SP3DTank3Asm |



Properties - CSP3DTank3Def

**CSP3DTank3Def** ClassModule

| | |
|---|---|
| (Name) | CSP3DTank3Def |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |



| | |
|---|---|
| (Name) | CSP3DTank3Sym |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |

11. Go to the General Declarations section in CSP3DTank3Sym module and change the value of the *Constant Module variable* from *"CSP3DTemplateAsm:"* to *"CSP3DTank3Asm:"*

```
Private Const MODULE = "CSP3DTank3Asm:"  'Used for error messages
```

12. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
    Set m_oSymbolHelper = New SymbolServices
    m_oSymbolHelper.ProjectName = "SP3DTank3Asm"
    m_oSymbolHelper.ClassName = "CSP3DTank3Sym"
```

13. In this Class_Initialize() routine, add the following code to define the inputs, outputs and aspects definition for this symbol.

```
' Inputs
    m_oSymbolHelper.NumInputs = 1
    m_oSymbolHelper.AddInputDef 1, "VesselLength", "Length", 4

' Outputs
    m_oSymbolHelper.NumOutputs = 1
    m_oSymbolHelper.AddOutputDef 1, "Body1", "Body1", 1

' Aspects
    m_oSymbolHelper.NumAspects = 1
    m_oSymbolHelper.AddAspectDef 1, "SimplePhysical", "SimplePhysical", 1
```

14. Go to CSimplePhysical Class module and declare variables for your inputs and outputs:

```
' Declare variables for Inputs and Outputs

    Dim ObjBody1 As Object
    Dim parLength As Double
```

15. Uses these variables to store the inputs as follows:

```
' Insert your code for inputs

    Set oPartFclt = arrayOfInputs(1)
    parLength = arrayOfInputs(2)
```

16. Go to the Insert your code for output 1 (Body1) section. The following code will use the PlaceBox() routine to create a Box for the Body1. The PlaceBox routine is located at geometry3d.bas module. This function takes the two opposite corner points of the box as input parameters.

```
 ' Insert your code for output (Body1)

   Dim pos1 As IJDPosition
   Dim pos2 As IJDPosition

   Set pos1 = New DPosition
   Set pos2 = New DPosition

   pos1.Set -parLength / 2, -parLength / 2, -parLength / 2
   pos2.Set parLength / 2, parLength / 2, parLength/2

   Set ObjBody1 = PlaceBox(m_OutputColl, pos1, pos2)

' Set the output
   iOutput = iOutput + 1
   m_OutputColl.AddOutput arrayOfOutputs(iOutput), ObjBody1
```

17. Use the Set statement to clear the references from all object variables.

```
'Release BO 's
   Set ObjBody1 = Nothing
   Set pos1 = Nothing
   Set pos2 = Nothing
```

18. Go to the General Declarations section in CSP3DTank3Def module and change the value of the *Constant Module variable* from *“"SP3DTemplateAsm:CSP3DTemplateDef"* to *"SP3DTank3Asm:CSP3DTank3Def"*

19. Go to the top of the CSP3DTank3Def module and declare the following variables

```
Private Const MODULE = "SP3DTank3Asm:CSP3DTank3Def"

Private Const IID_IJDATTRIBUTES = "{B25FD387-CFEB-11D1-850B-080036DE8E03}"
Private Const IID_IJDGEOMETRY = "{A1732CBF-5136-11D1-9770-080036754203}"

Private m_oEquipCADHelper As IJEquipCADHelper
Private m_oEditErrors As IJEditErrors

Private m_avSymbolArrayOfInputs()   As Variant
Private m_dVesselLength As Double

Private m_oNorth                 As IJDVector
Private m_oEast                 As IJDVector
Private m_oElevation              As IJDVector
```

20. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
    m_oEquipCADHelper.ProjectName = "SP3DTank3Asm"
    m_oEquipCADHelper.ClassName = "CSP3DTank3Def"
```

21. In this Class_Initialize() routine, initialize the following variable:

```
    Set m_oEast = New DVector
    m_oEast.x = 1
    m_oEast.y = 0
    m_oEast.z = 0

    Set m_oNorth = New DVector
    m_oNorth.x = 0
    m_oNorth.y = 1
    m_oNorth.z = 0

    Set m_oElevation = New DVector
    m_oElevation.x = 0
    m_oElevation.y = 0
    m_oElevation.z = 1
```

22. Go to the Class_Terminate() routine and use the Set statement to clear the references from all object variables.

```
Private Sub Class_Terminate()

    Set m_oNorth = Nothing
    Set m_oEast = Nothing
    Set m_oElevation = Nothing

    Set m_oEditErrors = Nothing
    Set m_oEquipCADHelper = Nothing
End Sub
```

23. Go to CSP3DTank3Def Class module. Declare the appropriate custom methods to manage the shape and pipe nozzle as follows:

```
    'Add your code here for the declaration of the Public Custom Methods used to manage new members
    'Add new member DP1 to the definition

    Set oMemberDescription = Nothing
    Set oMemberDescription = oMemberDescriptions.AddMember("DP1", 1, _
            "CMConstructDP1", imsCOOKIE_ID_USS_LIB)
    oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsDP1"
    oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructDP1"
    oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalDP1"
    oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseDP1"

    'Add properties for DP1
    Set oPropertyDescriptions = Nothing
    Set oPropertyDescriptions = oMemberDescription
    oPropertyDescriptions.AddProperty "DP1Properties", 1, IID_IJDATTRIBUTES, _
            "CMEvaluateDP1", imsCOOKIE_ID_USS_LIB
    oPropertyDescriptions.AddProperty "DP1GeometryProperties", 2, IID_IJDGEOMETRY,  _
            "CMEvaluateGeometryDP1", imsCOOKIE_ID_USS_LIB
```

```
 'Add new member (NozzleN1) to the definition

 Set oMemberDescription = Nothing
 Set oMemberDescription = oMemberDescriptions.AddMember("NozzleN1", 2,  _
       "CMConstructNozzleN1", imsCOOKIE_ID_USS_LIB)
 oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsNozzleN1"
 oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructNozzleN1"
 oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalNozzleN1"
 oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseNozzleN1"

'Add properties for (NozzleN1)
 Set oPropertyDescriptions = Nothing
 Set oPropertyDescriptions = oMemberDescription
  oPropertyDescriptions.AddProperty "NozzleN1Properties", 1, IID_IJDATTRIBUTES,  _
          "CMEvaluateNozzleN1", imsCOOKIE_ID_USS_LIB
 oPropertyDescriptions.AddProperty "NozzleN1GeometryProperties", 2, IID_IJDGEOMETRY, _
           "CMEvaluateGeometryNozzleN1", imsCOOKIE_ID_USS_LIB
```

24. Go to IJEquipUserAttrMgmt_OnAttributeChange function and add the following code. This OnAttributeChange method is called each time an attribute is changed. When the “Can be Deleted“property is changed, the MakeMemberDeletable should be called to make the member deletable. Set the NozzleN1 member deletable.

```
Private Function IJEquipUserAttrMgmt_OnAttributeChange(ByVal pIJDAttrs As IJDAttributes, ByVal
CollAllDisplayedValues As Object, ByVal pAttrToChange As IJEquipAttrDescriptor, ByVal varNewAttrValue
As Variant) As String
  Const METHOD = "IJEquipUserAttrMgmt_OnAttributeChange"
  On Error GoTo ErrorHandler

  Dim oMemberDescription As IJDMemberDescription
  Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)

  Select Case oMemberDescription.Name
    Case "NozzleN1"
      Select Case UCase(pAttrToChange.InterfaceName)
        Case "IJDELETABLEMEMBER"
          If UCase(pAttrToChange.AttrName) = "CANBEDELETED" Then
            m_oEquipCADHelper.MakeMemberDeletable oMemberDescription, _
        pIJDAttrs, CBool(varNewAttrValue)
          End If
        Case Else
          '
      End Select

    Case Else
      '
  End Select

  Set oMemberDescription = Nothing
    IJEquipUserAttrMgmt_OnAttributeChange = ""

  Exit Function
ErrorHandler:
  IJEquipUserAttrMgmt_OnAttributeChange = "ERROR"
  HandleError MODULE, METHOD
```

```
End Function
```

25. Go to IJEquipUserAttrMgmt_OnPreLoad function and add the following code. Use the IJEquipAttrDescriptor interface to set the datum shape properties read only.

```
Private Function IJEquipUserAttrMgmt_OnPreLoad(ByVal pIJDAttrs As IJDAttributes, ByVal
CollAllDisplayedValues As Object) As String
    Const METHOD = "IJEquipUserAttrMgmt_OnPreLoad"
    On Error GoTo ErrorHandler
    Dim oMemberDescription As IJDMemberDescription

        Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)
        Dim oAttrCollection As Collection
        Dim oAttributeDescriptor As IJEquipAttrDescriptor
        Dim m As Long

        Set oAttrCollection = CollAllDisplayedValues
        Select Case oMemberDescription.Name
        Case "DP1"
            For m = 1 To oAttrCollection.Count
                Set oAttributeDescriptor = oAttrCollection.Item(m)
                oAttributeDescriptor.AttrState = adsReadOnly
            Next
        Case Else
            '
       End Select

    Set oAttrCollection = Nothing
    Set oAttributeDescriptor = Nothing
    Set oMemberDescription = Nothing

    IJEquipUserAttrMgmt_OnPreLoad = ""
    Exit Function
ErrorHandler:
    IJEquipUserAttrMgmt_OnPreLoad = "ERROR"
    HandleError MODULE, METHOD
End Function
```

26. Go to the end of CSP3DTank3Def Class module. Add the custom methods to manage the datum shape and pipe nozzle as follows:

**Custom Method Construct**:
This method is in charge of the creation of the CAO member object (pipe nozzle). Use CreateNozzleGivenIndex() method to create the pipe nozzle given the nozzle index. The nozzle data are retrieved from the part. Use the CreateOrientationAndSetRelations() function to set the nozzle default position and nozzle default orientation.

```
Public Sub CMConstructNozzleN1(ByVal pMemberDescription As IJDMemberDescription, _
                               ByVal pResourceManager As IUnknown, _
                               ByRef pObject As Object)
    Const METHOD = "CMConstructNozzleN1"
    On Error GoTo ErrorHandler

    Dim oOrientation As IJNozzleOrientation
    Dim oNozzle As IJDNozzle

    GetDimensionsFromSymbolArray pMemberDescription.CAO

    'Create Nozzle
    m_oEquipCADHelper.CreateNozzleGivenIndex pMemberDescription, 1, pResourceManager, _
        DistribPortType_PIPE, pObject, False

    Set oNozzle = pObject
    oNozzle.Length = m_dVesselLength * 0.25

   'Create the nozzle orientation object
    Set oOrientation = m_oEquipCADHelper.CreateOrientationAndSetRelations(Nothing, oNozzle)

    'Set the default values
    oOrientation.PlacementType = Axial
    oOrientation.N1 = oNozzle.Length  + m_dVesselLength / 2
    oOrientation.N2 = 0
    oOrientation.OR1 = 0

    Set oNozzle = Nothing
    Set oOrientation = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Final:

There is no need to add any code for this custom method

```
Public Sub CMFinalConstructNozzleN1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMFinalConstructNozzleN1"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Inputs:

This method is used to set the input arguments to the member object.  Use IJDMemberDescription to get the appropriate shape (Datum shape) in order to set the relations with the pipe nozzle.

```
Public Sub CMSetInputsNozzleN1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMSetInputsNozzleN1"
    On Error GoTo ErrorHandler

        Dim l As Long
        Dim oShape As IJShape
        Dim oSmartOcc As IJSmartOccurrence
        Dim oMemberobjects As IJDMemberObjects
        Dim oMemberDesc As IJDMemberDescription
        Dim oOrientation As IJNozzleOrientation
        Set oSmartOcc = pMemberDesc.CAO

        Set oMemberobjects = oSmartOcc
        For l = 1 To oMemberobjects.Count
           Set oMemberDesc = oMemberobjects.MemberDescriptions.Item(l)
           If oMemberDesc.Name = "DP1" Then
              Set oShape = oMemberobjects.Item(l)
              Exit For
           End If
        Next l

     ' Create the nozzle relation with the shape
        Set oOrientation = m_oEquipCADHelper.CreateOrientationAndSetRelations(oShape,  _
                 pMemberDesc.Object)

        Set oShape = Nothing
        Set oSmartOcc = Nothing
        Set oMemberDesc = Nothing
        Set oOrientation = Nothing
        Set oMemberobjects = Nothing

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method Evaluate:**
There is no need to add any code for this custom method

```
Public Sub CMEvaluateNozzleN1(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
   Const METHOD = "CMEvaluateNozzleN1"
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method GeometryEvaluate:**
There is no need to add any code for this custom method

```
Public Sub CMEvaluateGeometryNozzleN1(ByVal oPropertyDescription As IJDPropertyDescription, pObject
As Object)
   Const METHOD = "CMEvaluateGeometryNozzleN1"
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method Conditional:**
This method checks if the member is conditional based on the CanBeDeleted flag.

Remember, we added code to make the member deletable in the IJEquipUserAttrMgmt_OnAttributeChange function.When the property is changed, the MakeMemberDeletable is called to check the CanBeDeleted flag and whether or not to make the member deletetable.

```
Public Sub CMConditionalNozzleN1(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As
Boolean)
    Const METHOD = "CMConditionalNozzleN1"
    On Error GoTo ErrorHandler

    IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom method Release:**
There is no need to add any code for this custom method

```
Public Sub CMReleaseNozzleN1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMReleaseNozzleN1"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Construct**:
This method is in charge of the creation of the CAO member object (datum shape). Use CreateShape() method to create the shape.

```
' Custom Methods for DP1
Public Sub CMConstructDP1(ByVal pMemberDescription As IJDMemberDescription, _
                                ByVal pResourceManager As IUnknown, _
                                ByRef pObject As Object)
    Const METHOD = "CMConstructDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler
    Dim oDatumShape As IJShape
    Dim oDesignEquipment As IJDesignEquipment

    'Create Datum Shape DP1
    Set oDatumShape = m_oEquipCADHelper.CreateShape(pMemberDescription, pResourceManager, _
                                        "DatumShape 001", "DP1")

    If Not oDatumShape Is Nothing Then
        Set pObject = oDatumShape
        oDatumShape.RepresentationId = ReferenceGeometry
        Set oDesignEquipment = pMemberDescription.CAO
        oDesignEquipment.AddShape oDatumShape
        GetDimensionsFromSymbolArray oDesignEquipment
        PositionAndOrientDP1 oDesignEquipment, oDatumShape
    End If

    Set oDesignEquipment = Nothing
    Set oDatumShape = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Final:

There is no need to add any code for this custom method

```
Public Sub CMFinalConstructDP1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMFinalConstructDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Inputs:

There is no need to add any code for this custom method

```
Public Sub CMSetInputsDP1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMSetInputsDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Evaluate:

There is no need to add any code for this custom method

```
Public Sub CMEvaluateDP1(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
    Const METHOD = "CMEvaluateDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method GeometryEvaluate:

This custom method will keep the nozzle’s position relative to the datum shape

```
Public Sub CMEvaluateGeometryDP1(ByVal oPropertyDescription As IJDPropertyDescription, pObject As
Object)
```

```
    Const METHOD = "CMEvaluateGeometryDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler
    Dim oEquipment As IJEquipment
    Dim oDatumShape As IJShape

    Set oDatumShape = oPropertyDescription.Object
    Set oEquipment = oPropertyDescription.CAO
    GetDimensionsFromSymbolArray oEquipment
    PositionAndOrientDP1 oEquipment, oDatumShape

    Set oDatumShape = Nothing
    Set oEquipment = Nothing

    Exit Sub

ErrorHandler:
    Set oDatumShape = Nothing
    Set oEquipment = Nothing
    HandleError MODULE, METHOD
End Sub
```

### Custom method Conditional:

There is no need to add any code for this custom method. The *Can be deleted* attribute is set to read only.

```
Public Sub CMConditionalDP1(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As Boolean)
    Const METHOD = "CMConditionalDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Release:

There is no need to add any code for this custom method

```
Public Sub CMReleaseDP1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMReleaseDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

27. Add the following subroutine to convert the array of inputs in a set of global variables

```
Private Sub GetDimensionsFromSymbolArray(SmartOccurrence As IJSmartOccurrence)
    Const METHOD = "GetDimensionsFromSymbolArray"
    On Error GoTo ErrorHandler
    m_avSymbolArrayOfInputs = m_oEquipCADHelper.GetSymbolArrayOfInputs(SmartOccurrence)

    'Inputs,  from equipment symbol code
    'Set m_oPartFclt = m_avSymbolArrayOfInputs(1)
    m_dVesselLength = m_avSymbolArrayOfInputs(2)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

28. Add the following subroutine to position and orient the shape with respect to the equipment.

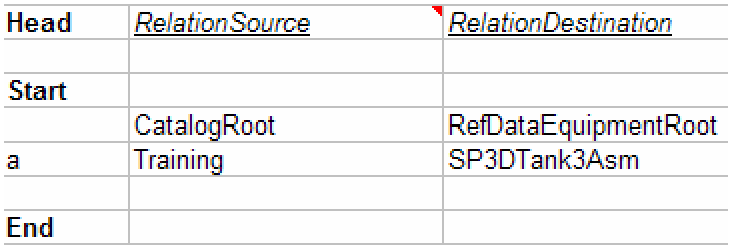
```
Private Sub PositionAndOrientDP1(Equipment As IJEquipment, Shape As IJShape)

    Dim oPosition As IJDPosition
    Set oPosition = New DPosition

    oPosition.Set 0, 0, 0
    m_oEquipCADHelper.PositionAndOrientShape Equipment, Shape, oPosition, m_oElevation, m_oNorth
    Set oPosition = Nothing

End Sub
```

29. Add the following subroutine to handle log calls.

```
Private Sub LogCalls(sMethod As String)

    If Not m_oEditErrors Is Nothing Then
        m_oEditErrors.Add 5000, m_oEquipCADHelper.ProjectName & "." & m_oEquipCADHelper.ClassName,
"Entering " & sMethod
    End If

End Sub
```

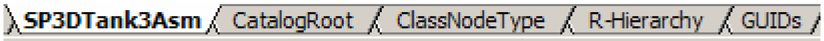
30. Compile the Visual Basic project and save the dll as SP3DTank3Asm.dll in the c:\Train\lab3 Note: One of the most important steps in Visual Basic programming is to preserve the binary compatibility of your program. Save the final version of your dll file to be binary compatibility in order to preserve the CLSID.

31. Save the Visual Basic SP3DTank3Asm project.
32. Open the SP3DTemplate.xls workbook. Go the R-Hierarchy sheet and add the following entry.



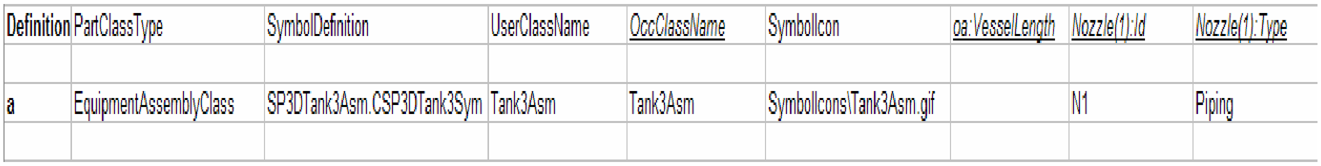
| Head | RelationSource | RelationDestination |
|---|---|---|
| | | |
| Start | | |
| | CatalogRoot | RefDataEquipmentRoot |
| a | Training | SP3DTank3Asm |
| | | |
| End | | |

33. Go to the SP3DTemplateAsm sheet and rename it as SP3DTank3Asm.



34. Go to the Class definition section and add/edit as follows:

35. In the Definition Section rows:

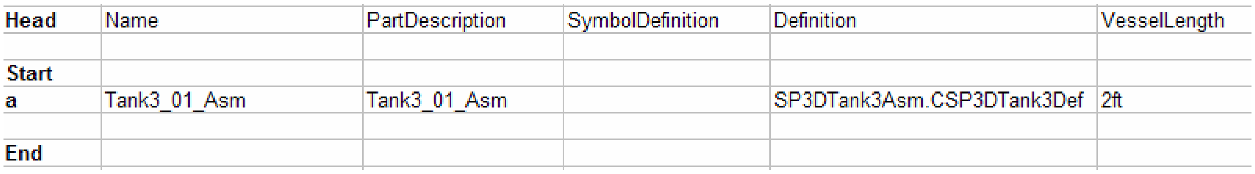


| Definition | PartClassType | SymbolDefinition | UserClassName | OccClassName | SymbolIcon | oa:VesselLength | Nozzle(1):Id | Nozzle(1):Type |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| a | EquipmentAssemblyClass | SP3DTank3Asm.CSP3DTank3Sym | Tank3Asm | Tank3Asm | SymbolIcons\Tank3Asm.gif | | N1 | Piping |
| | | | | | | | | |

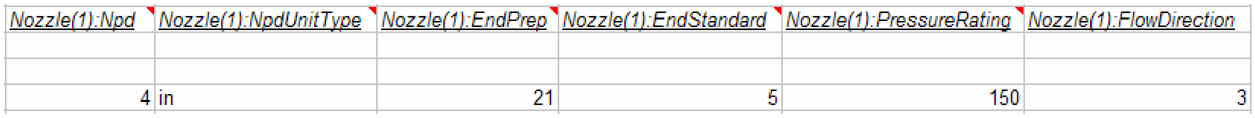
Note:

- Creating the bmp or gif file is optional. You can use Microsoft Paint to create the file and save it under your \\machine\symbols\SymbolIcons

36. In the Part Section rows:



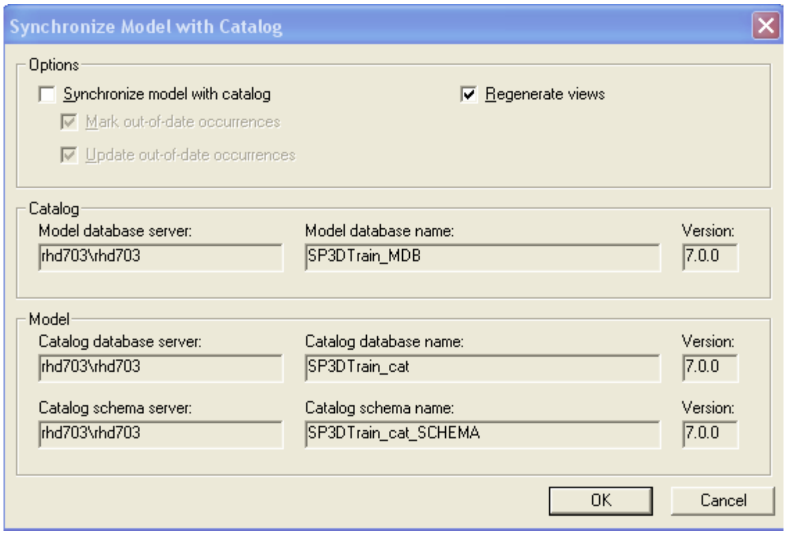
| Head | Name | PartDescription | SymbolDefinition | Definition | VesselLength |
|---|---|---|---|---|---|
| | | | | | |
| Start | | | | | |
| a | Tank3_01_Asm | Tank3_01_Asm | | SP3DTank3Asm.CSP3DTank3Def | 2ft |
| | | | | | |
| End | | | | | |



| Nozzle(1):Npd | Nozzle(1):NpdUnitType | Nozzle(1):EndPrep | Nozzle(1):EndStandard | Nozzle(1):PressureRating | Nozzle(1):FlowDirection |
|---|---|---|---|---|---|
| | | | | | |
| 4 | in | 21 | 5 | 150 | 3 |

37. Save the Excel workbook as SP3DTank3Asm.xls in the c:\Train\lab3.
38. Create the Tank3Asm.gif file and place it under \\<MachineName>\Symbols\SymbolIcons
37. Load the information into the catalog using the Add/Modify/Delete Mode. Once the bulkload process is completed, review the log file.
38. Run the Project Management Task. Select the Model in the hierarchy.
39. Select Tools -> Synchronize Model with the Catalog.
40. Uncheck the Synchronize Model with the Catalog option.

***Note****: You just need to update the views in the model.*



Synchronize Model with Catalog

Options

☐ Synchronize model with catalog

☑ Mark out-of-date occurrences

☑ Update out-of-date occurrences

☑ Regenerate views

Catalog

Model database server: rhd703\rhd703

Model database name: SP3DTrain_MDB

Version: 7.0.0

Model

Catalog database server: rhd703\rhd703

Catalog database name: SP3DTrain_cat

Version: 7.0.0

Catalog schema server: rhd703\rhd703

Catalog schema name: SP3DTrain_cat_SCHEMA

Version: 7.0.0

41. Hit “OK” Button.
42. Once the process is completed, Right click the training plant icon and select “Regenerate the Reports database” option to re-create the views in the report database.
43. Go to the Equipment Task and place the SP3DTank3Asm.

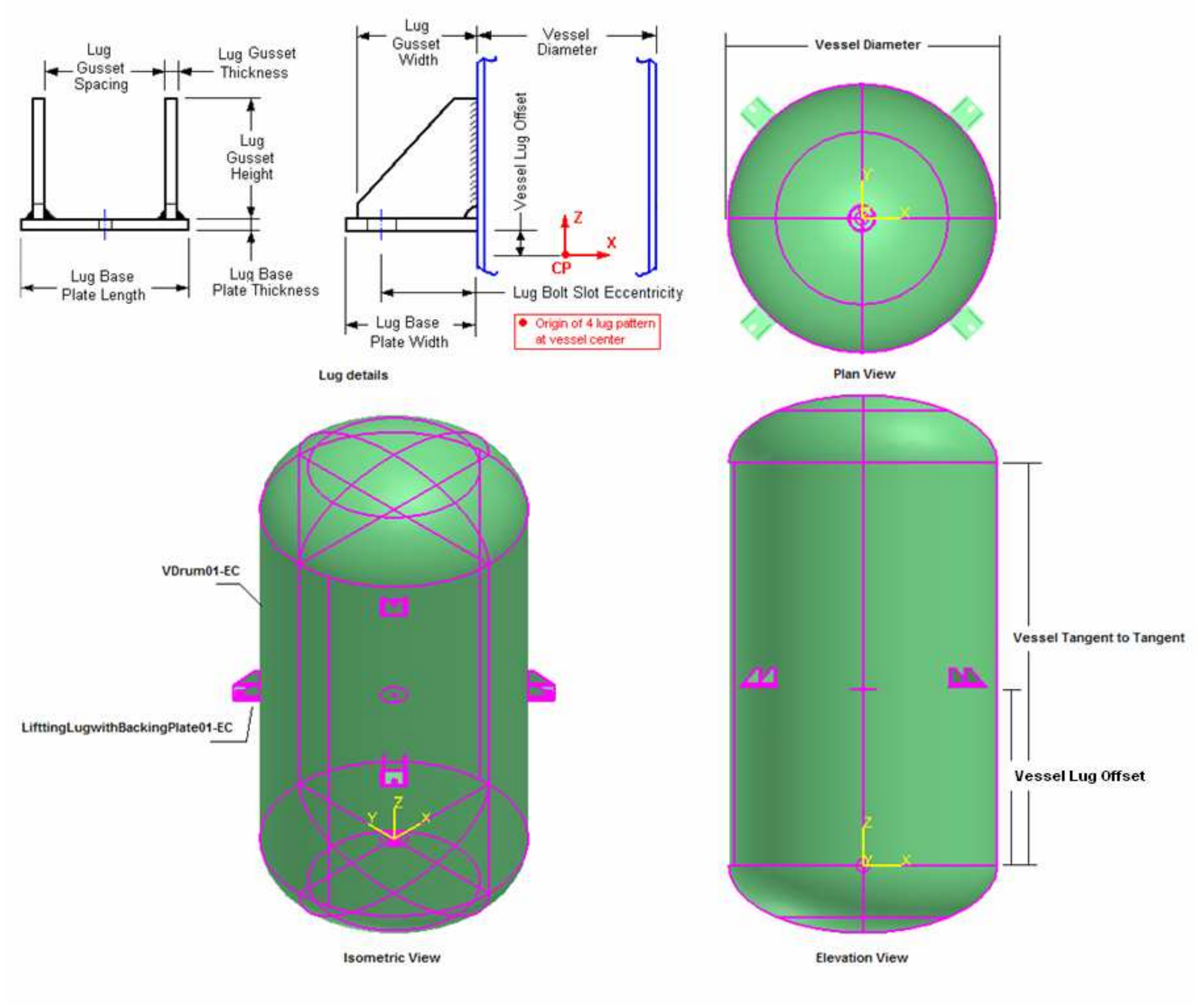
# Lab 5: Equipment components as members of the Equipment

## Objectives

After completing this lab, you will be able to:

- Create a custom assembly occurrence symbol made of equipment components
- Learn to use the Symbol Helper service to create the symbol definition
- Use the Equipment CAD Helper to define the Custom Assembly Definition
- Learn to use the IJDAggregatorDescription , IJDMemberDescriptions, and IJDPropertyDescriptions to define the behaviors of the custom assembly occurrence (CAO)
- Use the MakeMemberDeletable method to make the member deletable
- Use the IJEquipUserAttrMgmt Interface to show the equipment component attributes as read only in the property page
- Use IJDNamingRulesHelper interface to create the naming relations between a naming rule and the object
- Use IJDAttributes interface to get a collection of attributes property
- Use IJDAttribute to get an object's attribute

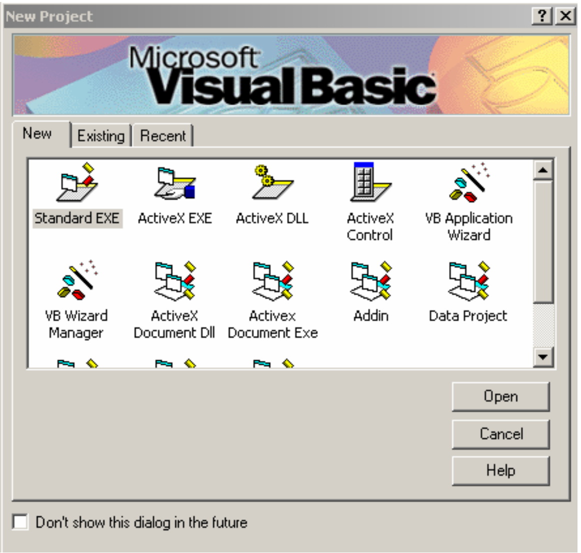
In this lab, you will create an equipment symbol as shown below. You start by using the SP3DEqpTemplateAsm template provided by the instructor to create the symbol. This symbol consists of two equipment components to define the symbol’s output. Use the Equipment Custom Assembly Definition (CAD) Helper to create the members.



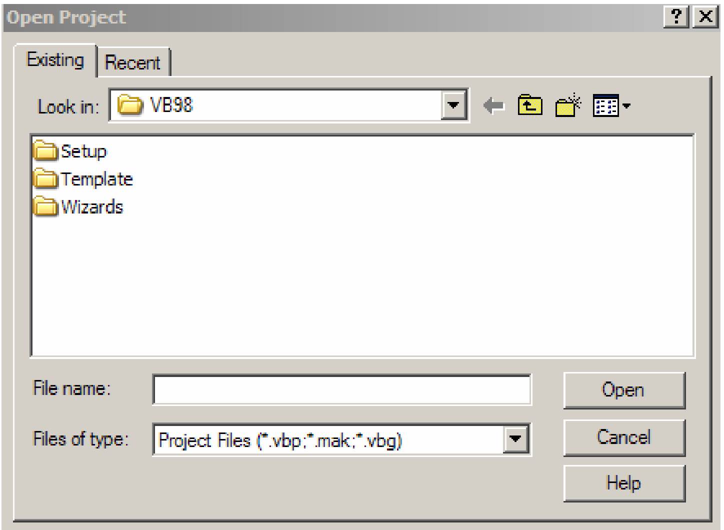
1. Create a directory called lab4 as follows:

   *c:\train\lab4*

2. Run Microsoft Visual Basic 6.0
3. Close the Microsoft New Project dialog box.



4. Select *File -> Open Project* option to open the Open Project Dialog box

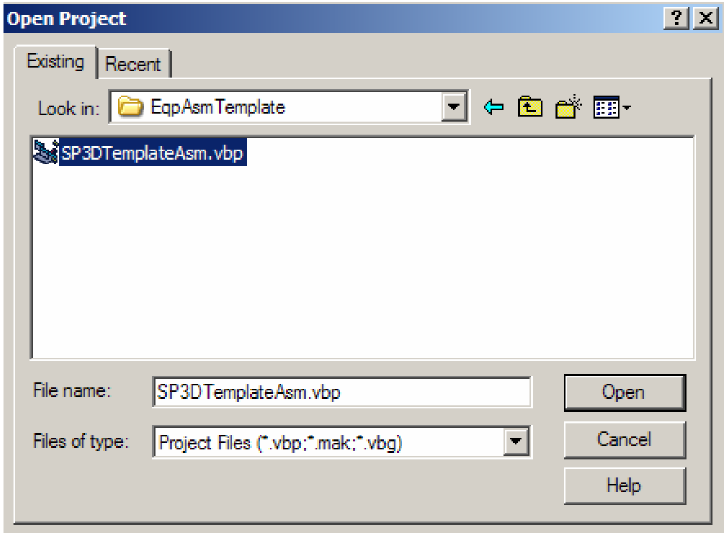


Look in: VB98

File name:

Files of type: Project Files (*.vbp;*.mak;*.vbg)

5. Navigate to c:\train\EqpAsmTemplate and open the SP3DTemplateAsm Template project

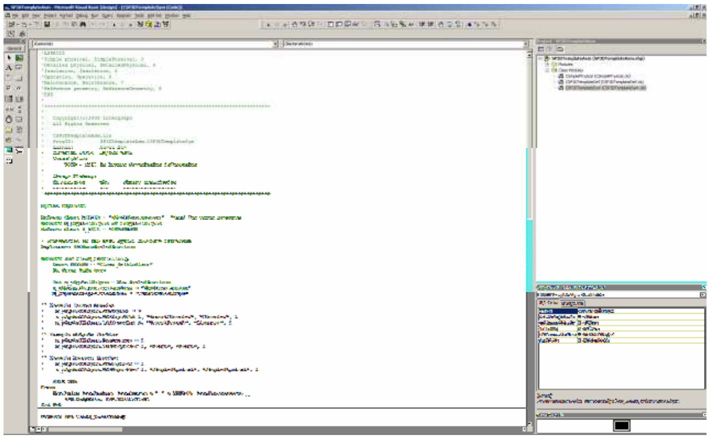


Look in: EqpAsmTemplate

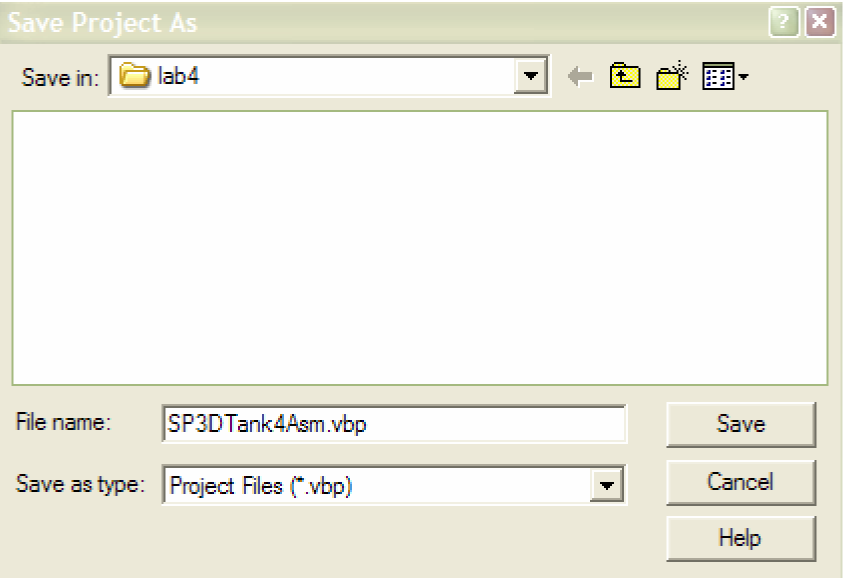
File name: SP3DTemplateAsm.vbp

Files of type: Project Files (*.vbp;*.mak;*.vbg)

6. Setup the Visual Basic Development Environment as shown below:



7. Go to the Visual Basic Explorer Window and select the Project node. Select *File* -> *Save Project As* option to save the project as SP3DTank4Asm.vbp under the lab4 directory

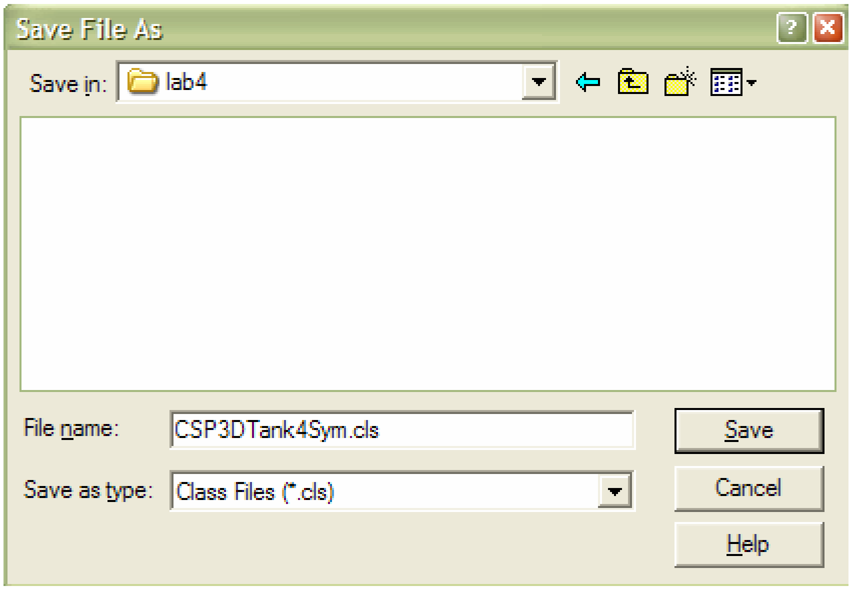


Save in: lab4

File name: SP3DTank4Asm.vbp

Save as type: Project Files (*.vbp)

8. Go to the Visual Basic Explorer Window and select the CSP3DTemplateSym class node. Select *File* -> *Save CSP3DTemplateSym.cls As* option to save the class module as CSP3DTank4Sym.cls under lab4 directory

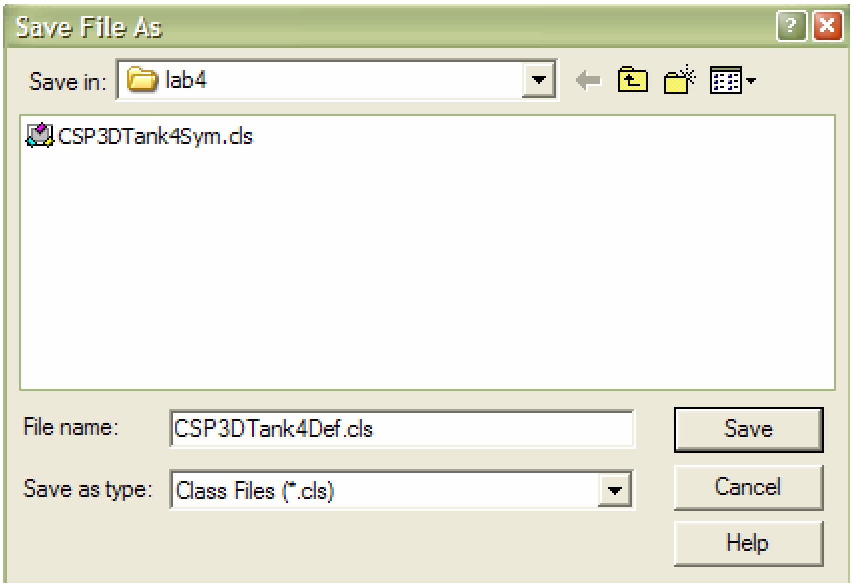


Save in: lab4

File name: CSP3DTank4Sym.cls

Save as type: Class Files (*.cls)

9. Go to the Visual Basic Explorer Window and select the CSP3DTemplateDef class node. Select *File* -> *Save CSP3DTemplateDef.cls As* option to save the class module as CSP3DTank4Def.cls under lab4 directory



Save in: lab4

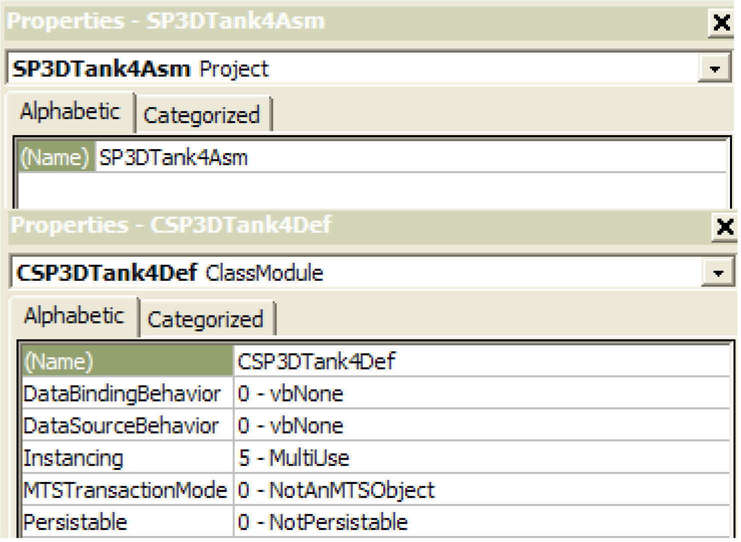
CSP3DTank4Sym.cls

File name: CSP3DTank4Def.cls

Save as type: Class Files (*.cls)

10. Go to the Visual Basic Explorer Window and select the CSimplePhysical class node. Select *File -> Save CSimplePhysical.cls As* option to save the class module as CSimplePhysical.cls under lab4 directory.

11. Go to the Properties Window and change the name of the Project and both Class Modules as follows:

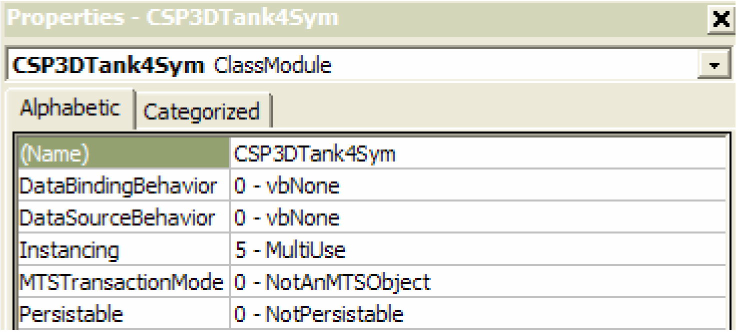


**SP3DTank4Asm** Project

| | |
|---|---|
| (Name) | SP3DTank4Asm |

**CSP3DTank4Def** ClassModule

| | |
|---|---|
| (Name) | CSP3DTank4Def |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |



| | |
|---|---|
| (Name) | CSP3DTank4Sym |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |

12. Go to the General Declarations section in CSP3DTank4Sym module and change the value of the *Constant Module variable* from *“CSP3DTemplateAsm:”* to *“CSP3DTank4Asm:”*

```
Private Const MODULE = "CSP3DTank4Asm:"  'Used for error messages
```

13. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
   Set m_oSymbolHelper = New SymbolServices
   m_oSymbolHelper.ProjectName = "SP3DTank4Asm"
   m_oSymbolHelper.ClassName = "CSP3DTank4Sym"
```

14. In this Class_Initialize() routine, add the following code to define the inputs and aspects definition for this symbol. Note: there is no outputs section.

```
' Inputs Section
   m_oSymbolHelper.NumInputs = 13
   m_oSymbolHelper.AddInputDef 1, "VesselDiameter", "VesselDiameter", 1
   m_oSymbolHelper.AddInputDef 2, "VesselTantoTan", "VesselTantoTan", 3
   m_oSymbolHelper.AddInputDef 3, "InsulationThickness", "InsulationThickness", 0.06

   m_oSymbolHelper.AddInputDef 4, "VesselLugOffset", "VesselLugOffset", 0.01
   m_oSymbolHelper.AddInputDef 5, "LugBasePlateWidth", "LugBasePlateWidth", 0.15
   m_oSymbolHelper.AddInputDef 6, "LugBasePlateLength", "LugBasePlateLength", 0.15
   m_oSymbolHelper.AddInputDef 7, "LugBasePlateThickness", "LugBasePlateThickness", 0.01
   m_oSymbolHelper.AddInputDef 8, "LugGussetHeight", "LugGussetHeight", 0.1
   m_oSymbolHelper.AddInputDef 9, "LugGussetWidth", "LugGussetWidth", 0.14
   m_oSymbolHelper.AddInputDef 10, "LugGussetThickness", "LugGussetThickness", 0.01
   m_oSymbolHelper.AddInputDef 11, "LugGussetSpacing", "LugGussetSpacing", 0.12
   m_oSymbolHelper.AddInputDef 12, "LugBoltSlotEccentricity", "LugBoltSlotEccentricity", 0.115
   m_oSymbolHelper.AddInputDef 13, "LugBoltDiameter", "LugBoltDiameter", 0.02

' Outputs Section

' Aspects Section
   m_oSymbolHelper.NumAspects = 1
   m_oSymbolHelper.AddAspectDef 1, "SimplePhysical", "SimplePhysical", 1
```

15. Go to CSimplePhysical Class module/Run subroutine and make sure there is no code to get the inputs

16. Go to the General Declarations section in CSP3DTank4Def module and change the value of the *Constant Module variable* from “*"SP3DTemplateAsm:CSP3DTemplateDef"* to *"SP3DTank4Asm:CSP3DTank4Def"*

17. Go to the top of the CSP3DTank4Def module and declare the following variables

```
Private Const MODULE = "SP3DTank4Asm:CSP3DTank4Def"

Private Const IID_IJDATTRIBUTES = "{B25FD387-CFEB-11D1-850B-080036DE8E03}"
Private Const IID_IJDGEOMETRY = "{A1732CBF-5136-11D1-9770-080036754203}"

Private m_oEquipCADHelper As IJEquipCADHelper
Private m_oEditErrors As IJEditErrors

Private m_avSymbolArrayOfInputs()   As Variant

'VDrum
Private m_VesselDiameter As Double
Private m_VesselTantoTan As Double
Private m_dInsulationThickness As Double

'Lugs
Private m_VesselLugOffset As Double
Private m_LugBasePlateWidth As Double
Private m_LugBasePlateLength As Double
Private m_LugBasePlateThickness As Double
Private m_LugGussetHeight As Double
Private m_LugGussetWidth As Double
Private m_LugGussetThickness As Double
Private m_LugGussetSpacing As Double
Private m_LugBoltSlotEccentricity As Double
Private m_LugBoltDiameter As Double
```

18. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
 m_oEquipCADHelper.ProjectName = "SP3DTank4Asm"
 m_oEquipCADHelper.ClassName = "CSP3DTank4Def"
```

19. Go to CSP3DTank4Def Class module. Declare the appropriate custom methods to manage the drum and the lug equipment components as follows:

```
'Add your code here for the declaration of the Public Custom Methods used to manage new members
'Add new member(VDrum) to the definition

Set oMemberDescription = Nothing
Set oMemberDescription = oMemberDescriptions.AddMember("VDrum", 1, _
                                        "CMConstructVDrum", imsCOOKIE_ID_USS_LIB)
oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsVDrum"
oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructVDrum"
oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalVDrum"
oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseVDrum"

Set oPropertyDescriptions = Nothing
```

```
    Set oPropertyDescriptions = oMemberDescription
    oPropertyDescriptions.AddProperty "VDrumProperties", 1, _
                                      IID_IJDATTRIBUTES, "CMEvaluateVDrum", imsCOOKIE_ID_USS_LIB
    oPropertyDescriptions.AddProperty "VDrumGeometryProperties", 2, IID_IJDGEOMETRY, _
                                       "CMEvaluateGeometryVDrum", imsCOOKIE_ID_USS_LIB
    'Add new member (Lug) to the definition

    Set oMemberDescription = Nothing
    Set oMemberDescription = oMemberDescriptions.AddMember("Lugs", 2, _
                                      "CMConstructLugs", imsCOOKIE_ID_USS_LIB)
    oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsLugs"
    oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructLugs"
    oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalLugs"
    oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseLugs"

    Set oPropertyDescriptions = Nothing
    Set oPropertyDescriptions = oMemberDescription
    oPropertyDescriptions.AddProperty "LugsProperties", 1, IID_IJDATTRIBUTES, _
                                      "CMEvaluateLugs", imsCOOKIE_ID_USS_LIB
    oPropertyDescriptions.AddProperty "LugsGeometryProperties", 2, IID_IJDGEOMETRY, _
                                       "CMEvaluateGeometryLugs", imsCOOKIE_ID_USS_LIB
```

20. Go to IJEquipUserAttrMgmt_OnPreLoad function and add the following code. Use the IJEquipAttrDescriptor interface to set the equipment component properties read only. If you want the support lug dimensions changeable, then you can comments the Case statement for the properties located in IJUAVESSELSUPPORTLUG so that these attributes can be changed by the user.

```
Private Function IJEquipUserAttrMgmt_OnPreLoad(ByVal pIJDAttrs As IJDAttributes, ByVal
CollAllDisplayedValues As Object) As String
    Const METHOD = "IJEquipUserAttrMgmt_OnPreLoad"
    On Error GoTo ErrorHandler

        Dim oMemberDescription As IJDMemberDescription

        Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)
        Dim oAttrCollection As Collection
        Dim oAttributeDescriptor As IJEquipAttrDescriptor
        Dim m As Long

'Set dimension and deletable attributes to read only.

        Set oAttrCollection = CollAllDisplayedValues
        Select Case oMemberDescription.Name
        Case "VDrum"
            For m = 1 To oAttrCollection.Count
                Set oAttributeDescriptor = oAttrCollection.Item(m)
                Select Case UCase(oAttributeDescriptor.InterfaceName)
                    Case "IJUAVESSELDIAMETER"
                       oAttributeDescriptor.AttrState = adsReadOnly
                    Case "IJUAVESSELTANTOTAN"
                       oAttributeDescriptor.AttrState = adsReadOnly
                    Case "IJDELETABLEMEMBER"
                       oAttributeDescriptor.AttrState = adsReadOnly
                    Case Else
                    '
                End Select
            Next
        Case "Lugs"
            For m = 1 To oAttrCollection.Count
                Set oAttributeDescriptor = oAttrCollection.Item(m)
                Select Case UCase(oAttributeDescriptor.InterfaceName)
                    Case "IJUAVESSELSUPPORTLUG"
                       oAttributeDescriptor.AttrState = adsReadOnly
                    Case "IJUAVESSELDIAMETER"
                       oAttributeDescriptor.AttrState = adsReadOnly
                    Case "IJDELETABLEMEMBER"
                       oAttributeDescriptor.AttrState = adsReadOnly
                    Case Else
                       '
                End Select
            Next
        Case Else
            '
        End Select

    Set oAttrCollection = Nothing
    Set oAttributeDescriptor = Nothing
    Set oMemberDescription = Nothing

    IJEquipUserAttrMgmt_OnPreLoad = ""

    Exit Function
ErrorHandler:
    IJEquipUserAttrMgmt_OnPreLoad = "ERROR"
    HandleError MODULE, METHOD
End Function
```

21. Go to the end of CSP3DTank4Def Class module. Add the custom methods to manage the drum and the lug equipment components as follows:

    **Custom Method Construct**:
    This method is in charge of the creation of the CAO member object (VDrum 01-EC). Use EquipCADHelper CreateEquipmentComponent () method to create the member given the equipment component part number. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructVDrum(ByVal pMemberDescription As IJDMemberDescription, _
                            ByVal pResourceManager As IUnknown, _
                            ByRef pObject As Object)
    Const METHOD = "CMConstructVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Dim oEquipment As IJEquipment
    Set oEquipment = pMemberDescription.CAO
    GetDimensionsFromSymbolArray oEquipment

    'Create Equipment Component
```

```
   Set pObject = m_oEquipCADHelper.CreateEquipmentComponent(pMemberDescription, _
                                                            pResourceManager, "VDrum 01-EC", "VDrum")
   'create name for the member
   SetObjNameRule pObject, "CPEquipmentComponent"
   Set oEquipment = Nothing

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom Method Final:**
There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructVDrum(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMFinalConstructVDrum"
   LogCalls METHOD
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method Inputs:**
There is no need to add any code for this custom method.

```
Public Sub CMSetInputsVDrum(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMSetInputsVDrum"
   LogCalls METHOD
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method Evaluate:**
This method is in charge of setting the attribute values to the member object. Use the MakeMemberDeletable method to set the member deletable.

```
Public Sub CMEvaluateVDrum(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
   Const METHOD = "CMEvaluateVDrum"
   LogCalls METHOD
   On Error GoTo ErrorHandler

   GetDimensionsFromSymbolArray oPropertyDescription.CAO

   Dim oAttribs As IJDAttributes
   Dim oSmartOcc As IJSmartOccurrence

   Set oSmartOcc = oPropertyDescription.Object
   Set oAttribs = oSmartOcc '.ItemObject

   oAttribs.CollectionOfAttributes("IJUAVesselDiameter").Item("VesselDiameter").Value = m_VesselDiameter
```

```
    oAttribs.CollectionOfAttributes("IJUAVesselTantoTan").Item("VesselTantoTan").Value = m_VesselTantoTan
    oAttribs.CollectionOfAttributes("IJInsulationThickness").Item("InsulationThickness").Value = m_dInsulationThickness
    oAttribs.CollectionOfAttributes("IJDeletableMember").Item("CanBeDeleted").Value = True

' set member deletable

    Dim oMemberDescription As IJDMemberDescription
    Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(oAttribs)
    m_oEquipCADHelper.MakeMemberDeletable oMemberDescription, oAttribs, True

    Set oAttribs = Nothing
    Set oSmartOcc = Nothing
    Set oMemberDescription = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom method GeometryEvaluate:**

This method is in charge of setting the transformation matrix to move the member object relative to the equipment. Use the TransformFromECStoGCS function to maintain the VDrum member coordinate system relative to the Equipment coordinate system.

```
Public Sub CMEvaluateGeometryVDrum(ByVal oPropertyDescription As IJDPropertyDescription, pObject As
Object)
    Const METHOD = "CMEvaluateGeometryVDrum"
    On Error GoTo ErrorHandler
    LogCalls METHOD

'Add the following code to avoid the eqp component to move alone

    Dim oEqpComp As IJEquipmentComponent
    Dim oEqpCompMatrix As IJEquipment
    Dim oEquipment As IJEquipment

    Set oEqpComp = oPropertyDescription.Object
    oEqpComp.GetParent oEquipment
    Set oEqpCompMatrix = oEqpComp

    GetDimensionsFromSymbolArray oEquipment

    Dim otransform As IngrGeom3D.IJDT4x4
    Set otransform = New DT4x4
    Dim iVector As IJDVector
    Set iVector = New DVector

'set translation vector
    otransform.LoadIdentity
    iVector.x = 0
    iVector.y = 0
    iVector.z = 0
    otransform.Translate iVector
'Set eqp comp position to 0,0,0
```

```
    oEqpCompMatrix.SetMatrix otransform

    TransformFromECStoGCS oEquipment, oEqpComp

    Set oEqpCompMatrix = Nothing
    Set oEqpComp = Nothing
    Set oEquipment = Nothing

    Set iVector = Nothing
    Set otransform = Nothing

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Conditional:

This method checks if the member is conditional based on the CanBeDeleted flag. Remember, we added code in the CMEvaluate to make the member deletable.

```
Public Sub CMConditionalVDrum(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As
Boolean)
    Const METHOD = "CMConditionalVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Release:

There is no need to add any code for this custom method

```
Public Sub CMReleaseVDrum(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMReleaseVDrum"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### **Custom Method Construct**:

This method is in charge of the creation of the CAO member object (LiftingLugwithBackingPlate 01-EC). Use EquipCADHelper CreateEquipmentComponent () method to create the member given the equipment component part number. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructLugs(ByVal pMemberDescription As IJDMemberDescription, _
                        ByVal pResourceManager As IUnknown, _
```

```
                              ByRef pObject As Object)
    Const METHOD = "CMConstructLugs"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Dim oEquipment As IJEquipment
    Set oEquipment = pMemberDescription.CAO
    GetDimensionsFromSymbolArray oEquipment

    'Create Equipment Component
    Set pObject = m_oEquipCADHelper.CreateEquipmentComponent(pMemberDescription, _
                                              pResourceManager, "LiftingLugwithBackingPlate 01-EC", "Lugs")
    SetObjNameRule pObject, "CPEquipmentComponent"
    Set oEquipment = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Final:**
There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructLugs(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMFinalConstructLugs"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Inputs:**
There is no need to add any code for this custom method.

```
Public Sub CMSetInputsLugs(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMSetInputsLugs"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Evaluate:**
This method is in charge of setting the attribute values to the member object. Use the MakeMemberDeletable method to set the member deletable.

```
Public Sub CMEvaluateLugs(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
    Const METHOD = "CMEvaluateLugs"
    LogCalls METHOD
    On Error GoTo ErrorHandler
```

```
   GetDimensionsFromSymbolArray oPropertyDescription.CAO

   Dim oAttribs As IJDAttributes
   Dim oSmartOcc As IJSmartOccurrence

   Set oSmartOcc = oPropertyDescription.Object
   Set oAttribs = oSmartOcc '.ItemObject

   oAttribs.CollectionOfAttributes("IJUAVesselDiameter").Item("VesselDiameter").Value = m_VesselDiameter
   oAttribs.CollectionOfAttributes("IJUAVesselSupportLug").Item("VesselLugOffset").Value = m_VesselLugOffset
   oAttribs.CollectionOfAttributes("IJUAVesselSupportLug").Item("LugBasePlateWidth").Value = m_LugBasePlateWidth
   oAttribs.CollectionOfAttributes("IJUAVesselSupportLug").Item("LugBasePlateLength").Value = m_LugBasePlateLength
   oAttribs.CollectionOfAttributes("IJUAVesselSupportLug").Item("LugBasePlateThickness").Value =
                                                                                   m_LugBasePlateThickness
   oAttribs.CollectionOfAttributes("IJUAVesselSupportLug").Item("LugGussetHeight").Value = m_LugGussetHeight
   oAttribs.CollectionOfAttributes("IJUAVesselSupportLug").Item("LugGussetWidth").Value = m_LugGussetWidth
   oAttribs.CollectionOfAttributes("IJUAVesselSupportLug").Item("LugGussetThickness").Value = m_LugGussetThickness
   oAttribs.CollectionOfAttributes("IJUAVesselSupportLug").Item("LugGussetSpacing").Value = m_LugGussetSpacing
   oAttribs.CollectionOfAttributes("IJUAVesselSupportLug").Item("LugBoltSlotEccentricity").Value =
                                                                                   m_LugBoltSlotEccentricity
   oAttribs.CollectionOfAttributes("IJUAVesselSupportLug").Item("LugBoltDiameter").Value = m_LugBoltDiameter

   oAttribs.CollectionOfAttributes("IJDeletableMember").Item("CanBeDeleted").Value = True

' set member deletable

   Dim oMemberDescription As IJDMemberDescription
   Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(oAttribs)
   m_oEquipCADHelper.MakeMemberDeletable oMemberDescription, oAttribs, True

   Set oAttribs = Nothing
   Set oSmartOcc = Nothing
   Set oMemberDescription = Nothing

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

### Custom Method GeometryEvaluate:

This method is in charge of setting the transformation matrix to move the member object relative to the equipment. Use the TransformFromECStoGCS function to maintain the Lug member coordinate system relative to the Equipment coordinate system.

```
Public Sub CMEvaluateGeometryLugs(ByVal oPropertyDescription As IJDPropertyDescription, pObject As
Object)
   Const METHOD = "CMEvaluateGeometryLugs"
   On Error GoTo ErrorHandler
   LogCalls METHOD

   Dim oEqpComp As IJEquipmentComponent
   Dim oEqpCompMatrix As IJEquipment
   Dim oEquipment As IJEquipment

   Set oEqpComp = oPropertyDescription.Object
```

```
    oEqpComp.GetParent oEquipment
    Set oEqpCompMatrix = oEqpComp

    GetDimensionsFromSymbolArray oEquipment

    Dim otransform As IngrGeom3D.IJDT4x4
    Set otransform = New DT4x4
    Dim iVector As IJDVector
    Set iVector = New DVector

'set translation matrix
    otransform.LoadIdentity
    iVector.x = 0
    iVector.y = 0
    iVector.z = m_VesselLugOffset

' set/display the lug position value in the properties page
    Dim Attrcol As IJDAttributesCol
    Dim oAttrs As IJDAttributes
    Dim oAttr As IJDAttribute

    Set oAttrs = oEquipment
    Set Attrcol = oAttrs.CollectionOfAttributes("IJUAVesselSupportLug")
    Set oAttr = Attrcol.Item("VesselLugOffset")
    oAttr.Value = m_VesselLugOffset

    otransform.Translate iVector
    oEqpCompMatrix.SetMatrix otransform

    TransformFromECStoGCS oEquipment, oEqpComp

    Set oEquipment = Nothing
    Set oEqpComp = Nothing
    Set oEqpCompMatrix = Nothing

    Set iVector = Nothing
    Set otransform = Nothing
    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

## Custom method Conditional:

This method checks if the member is conditional based on the CanBeDeleted flag. Remember, we added code in the CMEvaluate to make the member deletable.

```
Public Sub CMConditionalLugs(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As Boolean)
    Const METHOD = "CMConditionalLugs"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)

    Exit Sub
```

```
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom Method Release:**
There is no need to add any code for this custom method.

```
Public Sub CMReleaseLugs(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMReleaseLugs"
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

22. Add the following subroutine to convert the array of inputs in a set of global variables.

```
Private Sub GetDimensionsFromSymbolArray(SmartOccurrence As IJSmartOccurrence)
   Const METHOD = "GetDimensionsFromSymbolArray"
   On Error GoTo ErrorHandler

   m_avSymbolArrayOfInputs = m_oEquipCADHelper.GetSymbolArrayOfInputs(SmartOccurrence)

   'Inputs,  from equipment symbol code
   'Set m_oPartFclt = m_avSymbolArrayOfInputs(1)
   m_VesselDiameter = m_avSymbolArrayOfInputs(2)
   m_VesselTantoTan = m_avSymbolArrayOfInputs(3)
   m_dInsulationThickness = m_avSymbolArrayOfInputs(4)

   m_VesselLugOffset = m_avSymbolArrayOfInputs(5)
   m_LugBasePlateWidth = m_avSymbolArrayOfInputs(6)
   m_LugBasePlateLength = m_avSymbolArrayOfInputs(7)
   m_LugBasePlateThickness = m_avSymbolArrayOfInputs(8)
   m_LugGussetHeight = m_avSymbolArrayOfInputs(9)
   m_LugGussetWidth = m_avSymbolArrayOfInputs(10)
   m_LugGussetThickness = m_avSymbolArrayOfInputs(11)
   m_LugGussetSpacing = m_avSymbolArrayOfInputs(12)
   m_LugBoltSlotEccentricity = m_avSymbolArrayOfInputs(13)
   m_LugBoltDiameter = m_avSymbolArrayOfInputs(14)

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

23. Add the following subroutine to position and orient the member with respect to the equipment.

```
Private Sub TransformFromECStoGCS(Equipment As IJEquipment, Object As Object)
   Const METHOD = "TransformFromECStoGCS"
   LogCalls METHOD
   On Error GoTo ErrorHandler
   Dim oEqpMatrix As IJDT4x4
   Dim oShapeMatrix As IJDT4x4
```

```
    Dim otransform As IJDGeometry
    Dim oShape As IJShape

    If Not Object Is Nothing Then
        If TypeOf Object Is IJDGeometry Then
            Equipment.GetMatrix oEqpMatrix
            Set otransform = Object
            otransform.DTransform oEqpMatrix
            Set otransform = Nothing
            Set oEqpMatrix = Nothing
        End If
    End If

    Set otransform = Nothing
    Set oEqpMatrix = Nothing
    Set oShape = Nothing
    Set oShapeMatrix = Nothing

    Exit Sub
ErrorHandler:
    Set otransform = Nothing
    Set oEqpMatrix = Nothing
    Set oShape = Nothing
    Set oShapeMatrix = Nothing
    HandleError MODULE, METHOD
End Sub
```

24. Add the following subroutine to set the naming relation and generate a name based on the default naming rule.

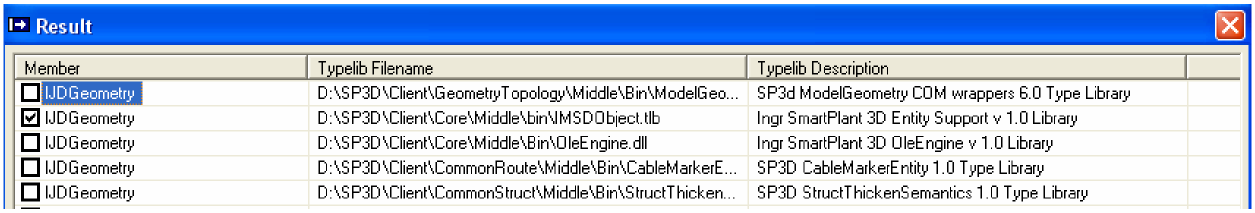
```
Public Sub SetObjNameRule(ByRef obj As Object, ByRef CLASSNAME As String)
' Apply the namerule using the IJDNamingRulesHelper helper interface
Const METHOD = "SetNameRule"
On Error GoTo ErrorHandler

    Dim NameRule As String
    Dim NamingRules As IJElements

    Dim oNameRuleHlpr As GSCADNameRuleSemantics.IJDNamingRulesHelper

'Returns a collection of the naming rules available in the catalog database
'for the given object
    Set oNameRuleHlpr = New GSCADNameRuleHlpr.NamingRulesHelper
    Call oNameRuleHlpr.GetEntityNamingRulesGivenName(CLASSNAME, NamingRules)
'get the first namerule from the collection
    Dim oNameRuleHolder As GSCADGenericNamingRulesFacelets.IJDNameRuleHolder
    Set oNameRuleHolder = NamingRules.Item(1)
'Create relations "NamedEntity" and "EntityNamingRule" and obj
    Dim oNameRuleAE As GSCADGenNameRuleAE.IJNameRuleAE
    Call oNameRuleHlpr.AddNamingRelations(obj, oNameRuleHolder, oNameRuleAE)

    GoTo CleanObjects
ErrorHandler:
    HandleError MODULE, METHOD
```

```
CleanObjects:
    Set oNameRuleHlpr = Nothing
    Set oNameRuleHolder = Nothing
    Set oNameRuleAE = Nothing
    Set NamingRules = Nothing
End Sub
```

25. Add the following subroutine to log any error.

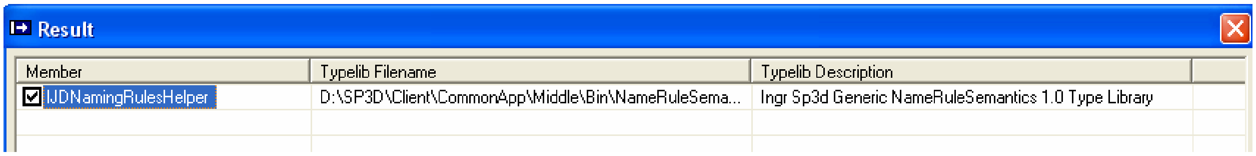
```
Private Sub LogCalls(sMethod As String)

    If Not m_oEditErrors Is Nothing Then
        m_oEditErrors.Add 5000, m_oEquipCADHelper.ProjectName & "." &
m_oEquipCADHelper.CLASSNAME, "Entering " & sMethod
    End If

End Sub
```

26. Compile the Visual Basic project and save the dll as SP3DTank4Asm.dll in the c:\Train\lab4
Note: Use the SP3D Reference Tool to attach the missing reference libraries.



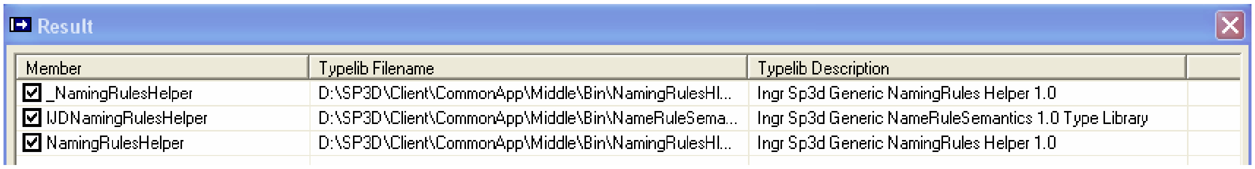
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☐ IJDGeometry | D:\SP3D\Client\GeometryTopology\Middle\Bin\ModelGeo... | SP3d ModelGeometry COM wrappers 6.0 Type Library |
| ☑ IJDGeometry | D:\SP3D\Client\Core\Middle\bin\IMSDObject.tlb | Ingr SmartPlant 3D Entity Support v 1.0 Library |
| ☐ IJDGeometry | D:\SP3D\Client\Core\Middle\Bin\OleEngine.dll | Ingr SmartPlant 3D OleEngine v 1.0 Library |
| ☐ IJDGeometry | D:\SP3D\Client\CommonRoute\Middle\Bin\CableMarkerE... | SP3D CableMarkerEntity 1.0 Type Library |
| ☐ IJDGeometry | D:\SP3D\Client\CommonStruct\Middle\Bin\StructThicken... | SP3D StructThickenSemantics 1.0 Type Library |



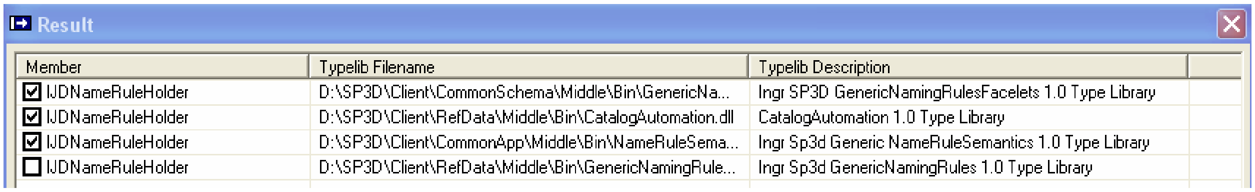
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDNamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |



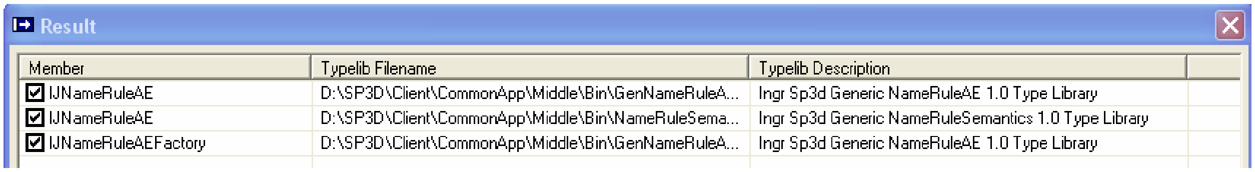
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ _NamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NamingRulesHl... | Ingr Sp3d Generic NamingRules Helper 1.0 |
| ☑ IJDNamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☑ NamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NamingRulesHl... | Ingr Sp3d Generic NamingRules Helper 1.0 |



Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDNameRuleHolder | D:\SP3D\Client\CommonSchema\Middle\Bin\GenericNa... | Ingr SP3D GenericNamingRulesFacelets 1.0 Type Library |
| ☑ IJDNameRuleHolder | D:\SP3D\Client\RefData\Middle\Bin\CatalogAutomation.dll | CatalogAutomation 1.0 Type Library |
| ☑ IJDNameRuleHolder | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☐ IJDNameRuleHolder | D:\SP3D\Client\RefData\Middle\Bin\GenericNamingRule... | Ingr Sp3d GenericNamingRules 1.0 Type Library |

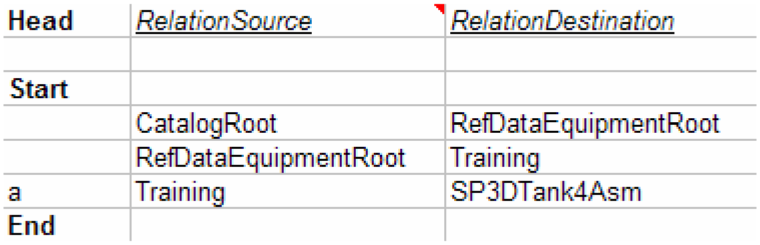


Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJNameRuleAE | D:\SP3D\Client\CommonApp\Middle\Bin\GenNameRuleA... | Ingr Sp3d Generic NameRuleAE 1.0 Type Library |
| ☑ IJNameRuleAE | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☑ IJNameRuleAEFactory | D:\SP3D\Client\CommonApp\Middle\Bin\GenNameRuleA... | Ingr Sp3d Generic NameRuleAE 1.0 Type Library |

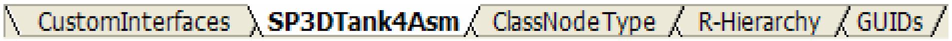
One of the most important steps in Visual Basic programming is to preserve the binary compatibility of your program. Save the final version of your dll file to be binary compatibility in order to preserve the CLSID.

27. Save the Visual Basic SP3DTank4Asm project.
28. Open the SP3DTemplate.xls workbook. Go the R-Hierarchy sheet and add the following entry.



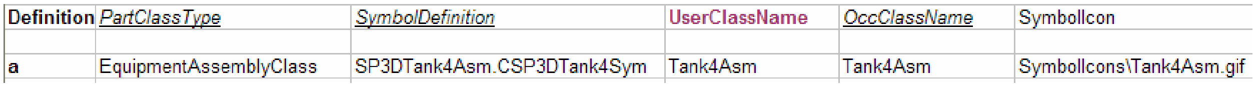
| Head | RelationSource | RelationDestination |
|---|---|---|
| | | |
| Start | | |
| | CatalogRoot | RefDataEquipmentRoot |
| | RefDataEquipmentRoot | Training |
| a | Training | SP3DTank4Asm |
| End | | |

29. Go to the SP3DTemplateAsm sheet and rename it as SP3DTank4Asm.



30. Go to the Class definition section and add/edit as follows:

31. In the Definition Section rows:

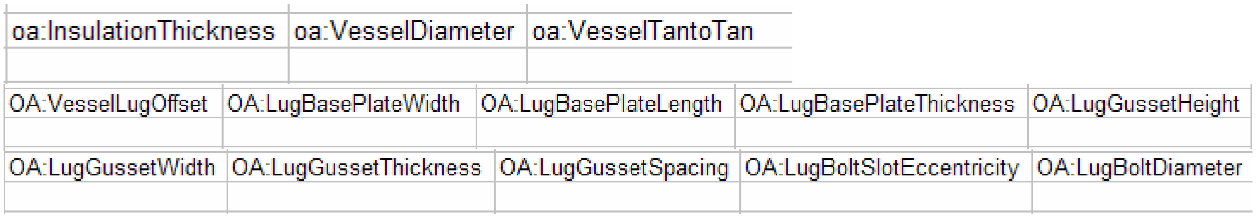


| Definition | PartClassType | SymbolDefinition | UserClassName | OccClassName | SymbolIcon |
|---|---|---|---|---|---|
| | | | | | |
| a | EquipmentAssemblyClass | SP3DTank4Asm.CSP3DTank4Sym | Tank4Asm | Tank4Asm | SymbolIcons\Tank4Asm.gif |

Note:

- Creating the bmp or gif file is optional. You can use Microsoft Paint to create the file and save it under your \\machine\symbols\SymbolIcons

Occurrence attributes:

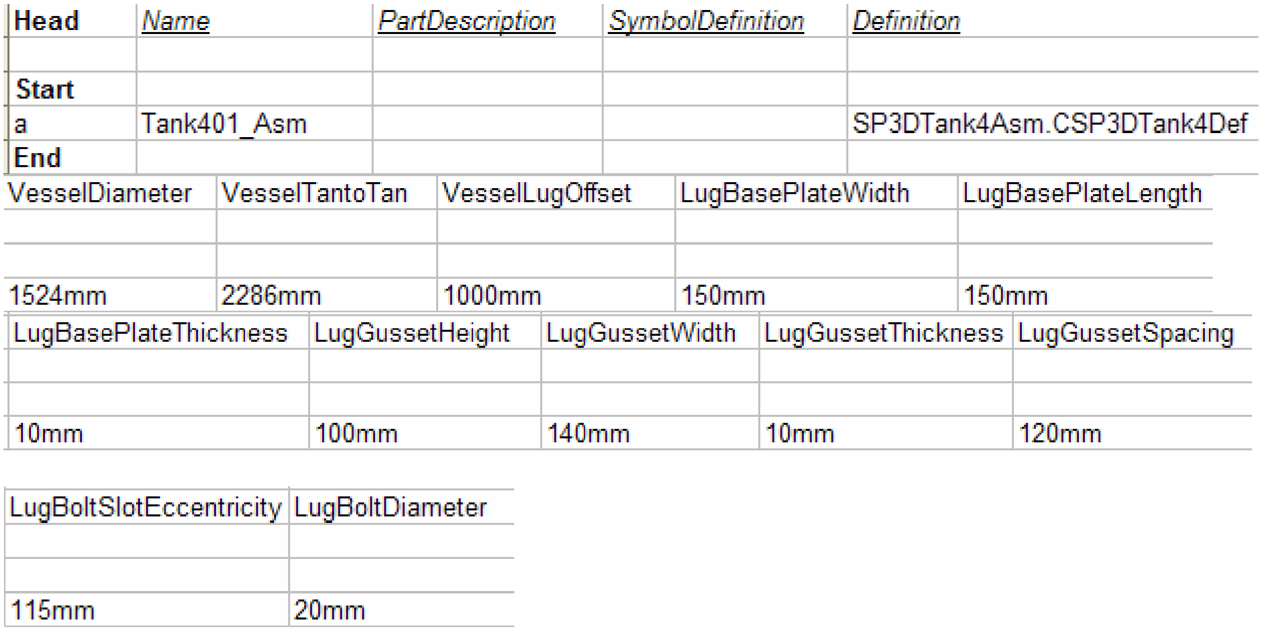


| oa:InsulationThickness | oa:VesselDiameter | oa:VesselTantoTan |
|---|---|---|
| | | |

| OA:VesselLugOffset | OA:LugBasePlateWidth | OA:LugBasePlateLength | OA:LugBasePlateThickness | OA:LugGussetHeight |
|---|---|---|---|---|
| | | | | |

| OA:LugGussetWidth | OA:LugGussetThickness | OA:LugGussetSpacing | OA:LugBoltSlotEccentricity | OA:LugBoltDiameter |
|---|---|---|---|---|
| | | | | |

32. In the Part Section rows:



| Head | Name | PartDescription | SymbolDefinition | Definition |
|---|---|---|---|---|
| | | | | |
| Start | | | | |
| a | Tank401_Asm | | | SP3DTank4Asm.CSP3DTank4Def |
| End | | | | |

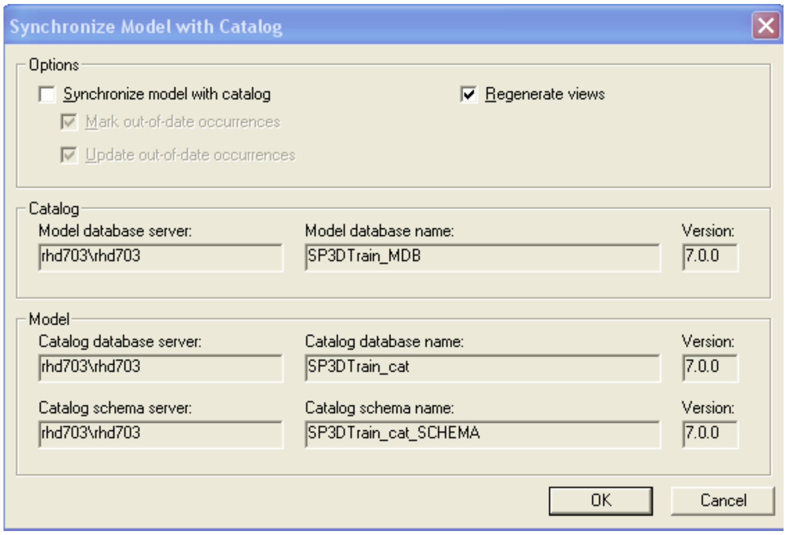
| VesselDiameter | VesselTantoTan | VesselLugOffset | LugBasePlateWidth | LugBasePlateLength |
|---|---|---|---|---|
| | | | | |
| | | | | |
| 1524mm | 2286mm | 1000mm | 150mm | 150mm |

| LugBasePlateThickness | LugGussetHeight | LugGussetWidth | LugGussetThickness | LugGussetSpacing |
|---|---|---|---|---|
| | | | | |
| | | | | |
| 10mm | 100mm | 140mm | 10mm | 120mm |

| LugBoltSlotEccentricity | LugBoltDiameter |
|---|---|
| | |
| | |
| 115mm | 20mm |

33. Save the Excel workbook as SP3DTank4Asm.xls in the c:\Train\lab4.
34. Optional step: Create the Tank4Asm.gif file and place it under \\<MachineName>\Symbols\SymbolIcons
35. Load the information into the catalog using the Add/Modify/Delete Mode. Once the bulkload process is completed, review the log file.
36. Run the Project Management Task. Select the Model in the hierarchy.
37. Select Tools -> Synchronize Model with the Catalog.
38. Uncheck the Synchronize Model with the Catalog option.

***Note**: You just need to update the views in the model.*



Synchronize Model with Catalog

Options

☐ Synchronize model with catalog

☑ Regenerate views

☑ Mark out-of-date occurrences

☑ Update out-of-date occurrences

Catalog

Model database server: rhd703\rhd703

Model database name: SP3DTrain_MDB

Version: 7.0.0

Model

Catalog database server: rhd703\rhd703

Catalog database name: SP3DTrain_cat

Version: 7.0.0

Catalog schema server: rhd703\rhd703

Catalog schema name: SP3DTrain_cat_SCHEMA

Version: 7.0.0

39. Hit "OK" Button.

40. Once the process is completed, Right click the training plant icon and select “Regenerate the Reports database” option to re-create the views in the report database.
41. Go to the Equipment Task and place the SP3DTank4Asm.

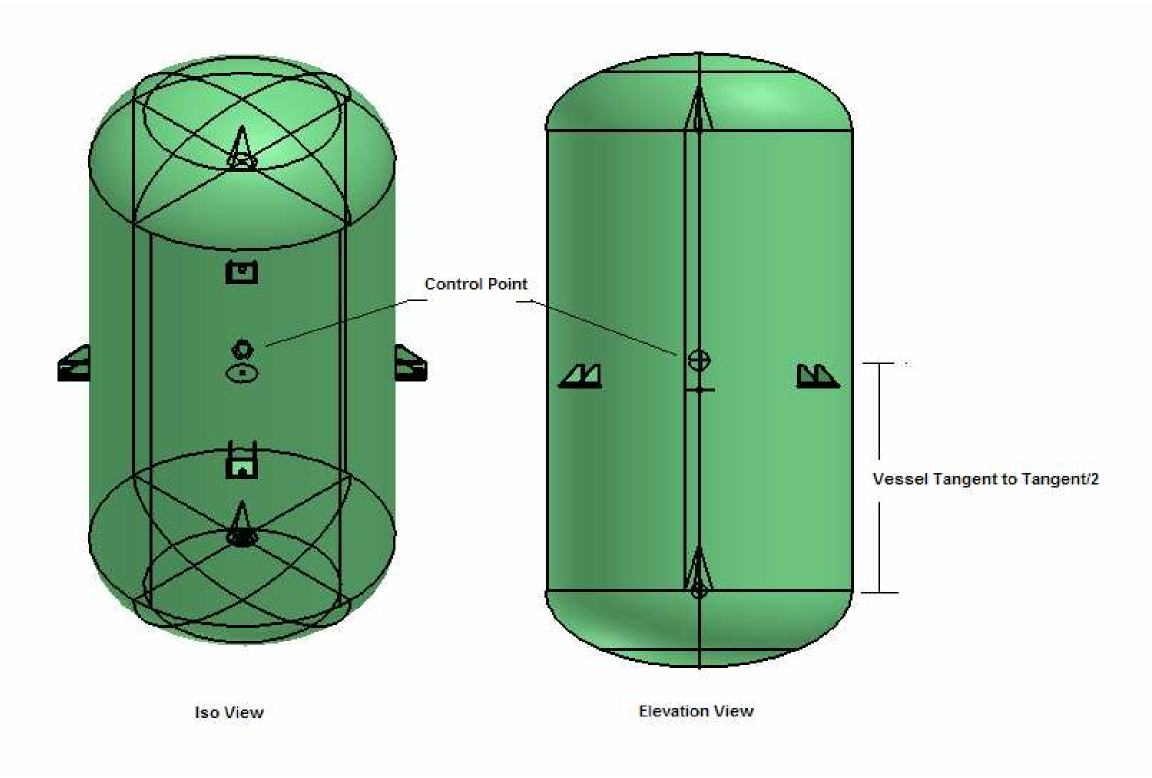
# Lab 6: Control Point as member of the Equipment

## Objectives

After completing this lab, you will be able to:

- Add control points as a member of the custom assembly
- Learn to use the IJGeneralBusinessObjectsFactory to create the control point object

Modify the Vertical Drum symbol (SP3DTank4Asm) by adding a control point as a member of the custom assembly occurrence.



1. Go to the top of the CSP3DTank4Def module and declare the following variable

*Private Const MODULE = "SP3DTank4Asm:CSP3DTank4Def"*

*Private Const IID_IJDATTRIBUTES = "{B25FD387-CFEB-11D1-850B-080036DE8E03}"*
*Private Const IID_IJDGEOMETRY = "{A1732CBF-5136-11D1-9770-080036754203}"*

*Private Const IID_IJControlPoint = "{F54DCDA8-2C24-47B9-A4ED-ACA4F3DF15D7}"*

2. Go to CSP3DTank4Def Class module. Declare the appropriate custom methods to manage the control point as follows:

```
'Add new member(Control Point) to the definition

Set oMemberDescription = Nothing
Set oMemberDescription = oMemberDescriptions.AddMember("ControlPoint", 3, _
                          "CMConstructControlPoint", imsCOOKIE_ID_USS_LIB)
oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputControlPoint"
oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructControlPoint"
oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalControlPoint"
oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseControlPoint"

Set oPropertyDescriptions = Nothing
Set oPropertyDescriptions = oMemberDescription
oPropertyDescriptions.AddProperty "ControlPointProperties", 1, IID_IJDATTRIBUTES, _
                                  "CMEvaluateControlPoint", imsCOOKIE_ID_USS_LIB
oPropertyDescriptions.AddProperty "ControlPointGeometryProperties", 2, IID_IJDGEOMETRY, _
                                  "CMEvaluateGeometryControlPoint", imsCOOKIE_ID_USS_LIB
```

3. Go to the end of CSP3DTank4Def Class module. Add the custom methods to manage the control point as follows:

   **Custom Method Construct**:
   This method is in charge of the creation of the control point object. Use *GeneralBusinessObjectsFactory object* to create the member. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructControlPoint(ByVal pMemberDescription As IJDMemberDescription, ByVal
pResourceManager As IUnknown, ByRef pObj As Object)
Const METHOD = "CMConstructControlPoint"
On Error GoTo ErrorHandler

   Dim oEquipment As IJEquipment
   Set oEquipment = pMemberDescription.CAO

   Dim oControlPoint As IJControlPoint
   Dim oGBSFactory  As IJGeneralBusinessObjectsFactory
   Set oGBSFactory = New GeneralBusinessObjectsFactory

   Set oControlPoint = oGBSFactory.CreateControlPoint(pResourceManager, 0#, 0#, 0#, 0.1, , False, True)

  ‘Set default properties
   oControlPoint.SubType = cpProcessEquipment
   oControlPoint.Type = cpControlPoint
   oControlPoint.IsAssociative = True
   oControlPoint.Diameter = 0.1

   Set pObj = oControlPoint
   SetObjNameRule oControlPoint, "CPControlPoint"

   Set oControlPoint = Nothing
   Set oEquipment = Nothing
```

```
    Set oGBSFactory = Nothing

Exit Sub
ErrorHandler:       HandleError MODULE, METHOD
End Sub
```

### Custom Method Final:

This method will call the CreateControlPointRelation method to create the relation.

```
Public Sub CMFinalConstructControlPoint(pMemberDesc As IJDMemberDescription)
Const METHOD = "CMFinalConstructControlPoint"
On Error GoTo ErrorHandler

     Call CreateControlPointRelation(pMemberDesc)

Exit Sub
ErrorHandler:       HandleError MODULE, METHOD
End Sub
```

### Custom method Inputs:

There is no need to add any code for this custom method.

```
Public Sub CMSetInputControlPoint(pMemberDesc As IJDMemberDescription)
Const METHOD = "CMSetInputControlPoint"
On Error GoTo ErrorHandler

Exit Sub
ErrorHandler: HandleError MODULE, METHOD
End Sub
```

### Custom method Evaluate:

This method is in charge of setting the properties and position of the control point relative to the equipment.

```
Public Sub CMEvaluateControlPoint(pPropertyDescriptions As IJDPropertyDescription, pObject As Object)
Const METHOD = "CMEvaluateControlPoint"
On Error GoTo ErrorHandler

    Dim oControlPoint As IJControlPoint
    Set oControlPoint = pPropertyDescriptions.Object

    oControlPoint.SubType = cpProcessEquipment
    oControlPoint.Type = cpControlPoint
    oControlPoint.IsAssociative = True
    oControlPoint.Diameter = 0.1

    Dim oEquipment As IJEquipment
    Set oEquipment = pPropertyDescriptions.CAO

    Dim x As Double, y As Double, z As Double

    oEquipment.GetPosition x, y, z

    Dim ControlPt As IJPoint
    Set ControlPt = New Point3d
    Set ControlPt = oControlPoint

    GetDimensionsFromSymbolArray oEquipment

    ControlPt.SetPoint x, y, z + m_VesselTantoTan / 2

    Set oControlPoint = Nothing
    Set ControlPt = Nothing
    Set oEquipment = Nothing

Exit Sub
ErrorHandler:	HandleError MODULE, METHOD
End Sub
```

### Custom method GeometryEvaluate:

This method will call the CMEvaluate.

```
Public Sub CMEvaluateGeometryControlPoint(pPropertyDescriptions As IJDPropertyDescription, pObject As Object)
Const METHOD = "CMEvaluateGeometryControlPoint"
On Error GoTo ErrorHandler

Call CMEvaluateControlPoint(pPropertyDescriptions, pObject)

Exit Sub
ErrorHandler:	HandleError MODULE, METHOD
End Sub
```

### Custom method Conditional:

There is no need to add any code for this custom method.

```
Public Sub CMConditionalControlPoint(ByVal pMemberDescription As IJDMemberDescription, ByRef IsNeeded As Boolean)
Const METHOD = "CMConditionalControlPoint"
On Error GoTo ErrorHandler

Exit Sub
ErrorHandler:	HandleError MODULE, METHOD
End Sub
```

### Custom method Release:

There is no need to add any code for this custom method.

```
Public Sub CMReleaseControlPoint(pMemberDesc As IJDMemberDescription)
Const METHOD = "CMReleaseControlPoint"
On Error GoTo ErrorHandler

Exit Sub
ErrorHandler:	HandleError MODULE, METHOD
End Sub
```

### Custom method Count:

There is no need to add any code for this custom method.

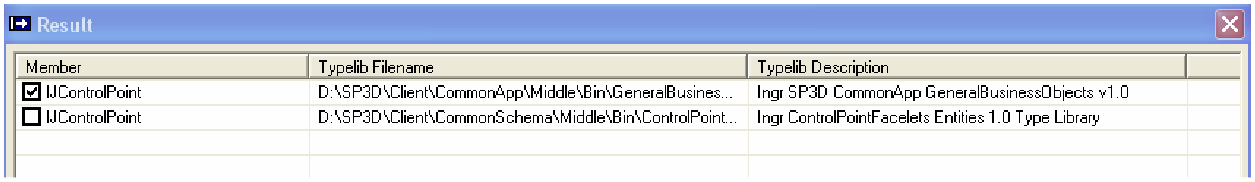
```
Public Sub CMCountControlPoint(ByVal pMemberDesc As IJDMemberDescription, ByRef Count As Long)
  Const METHOD = "CMCountControlPoint"
  On Error GoTo ErrorHandler

  Exit Sub
ErrorHandler:
  HandleError MODULE, METHOD
End Sub
```

4. Add the following subroutine *CreateControlPointRelation()* to set the equipment (Parent) associated with the control point (Child).

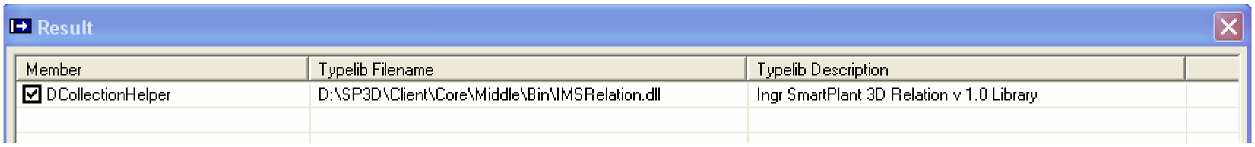
```
Public Sub CreateControlPointRelation(pMemberDesc As IJDMemberDescription)
Const METHOD = "CreateControlPointRelation"
On Error GoTo ErrorHandler

Dim CollHlpr As IMSRelation.DCollectionHelper
Dim TOColl As IJDTargetObjectCol
Dim AssocRel As IJDAssocRelation
Dim oRevision As IJRevision
Dim oRelationship As IJDRelationship

  Set AssocRel = pMemberDesc.Object
  Set CollHlpr = AssocRel.CollectionRelations(IID_IJControlPoint, "Parent")

  If Not CollHlpr Is Nothing Then
    Set TOColl = CollHlpr
    TOColl.Add pMemberDesc.CAO, vbNullString, oRelationship
    Set oRevision = New JRevision
    oRevision.AddRelationship oRelationship
  End If
  Set CollHlpr = Nothing
  Set oRevision = Nothing
  Set oRelationship = Nothing
  Set TOColl = Nothing
  Set AssocRel = Nothing
 Exit Sub
ErrorHandler:  HandleError MODULE, METHOD
End Sub
```

5. Open the property page of the Visual Basic project and increase the dll major version number.
6. Compile the Visula Basic project and save the dll in the c:\train\ *lab4*.

Note: Use the SP3D Reference Tool to attach the missing reference libraries.

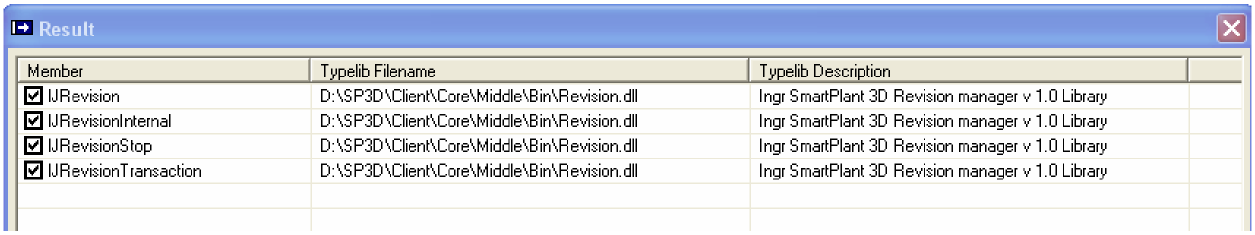


| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJControlPoint | D:\SP3D\Client\CommonApp\Middle\Bin\GeneralBusines... | Ingr SP3D CommonApp GeneralBusinessObjects v1.0 |
| □ IJControlPoint | D:\SP3D\Client\CommonSchema\Middle\Bin\ControlPoint... | Ingr ControlPointFacelets Entities 1.0 Type Library |



Result

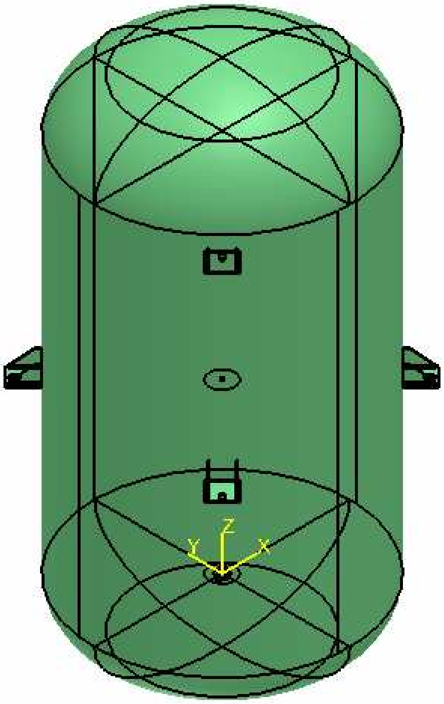
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ DCollectionHelper | D:\SP3D\Client\Core\Middle\Bin\IMSRelation.dll | Ingr SmartPlant 3D Relation v 1.0 Library |

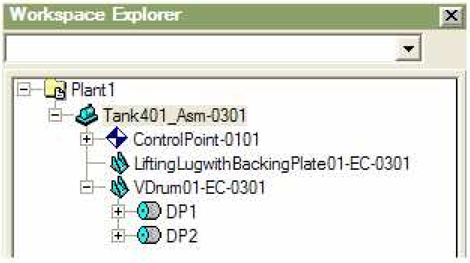


Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJRevision | D:\SP3D\Client\Core\Middle\Bin\Revision.dll | Ingr SmartPlant 3D Revision manager v 1.0 Library |
| ☑ IJRevisionInternal | D:\SP3D\Client\Core\Middle\Bin\Revision.dll | Ingr SmartPlant 3D Revision manager v 1.0 Library |
| ☑ IJRevisionStop | D:\SP3D\Client\Core\Middle\Bin\Revision.dll | Ingr SmartPlant 3D Revision manager v 1.0 Library |
| ☑ IJRevisionTransaction | D:\SP3D\Client\Core\Middle\Bin\Revision.dll | Ingr SmartPlant 3D Revision manager v 1.0 Library |

7. Save the Visual Basic SP3DTank4Asm project.
8. Open the SP3DTank4Asm.xls workbook. Go to the SP3DTank4Asm part class and add the letter M to the part.
9. Update the part class and part using the bulkload utility.
10. Go to Project Management task and run the synchronize model with the catalog database command.
11. Go to the Equipment Task and place the Tank401_Asm. Notice the control point is visible under the equipment occurrence in the system tab of the workspace explorer window.





Workspace Explorer

```
Plant1
  Tank401_Asm-0301
    ControlPoint-0101
    LiftingLugwithBackingPlate01-EC-0301
    VDrum01-EC-0301
      DP1
      DP2
```

# Lab 7: Managing variable members in the Equipment

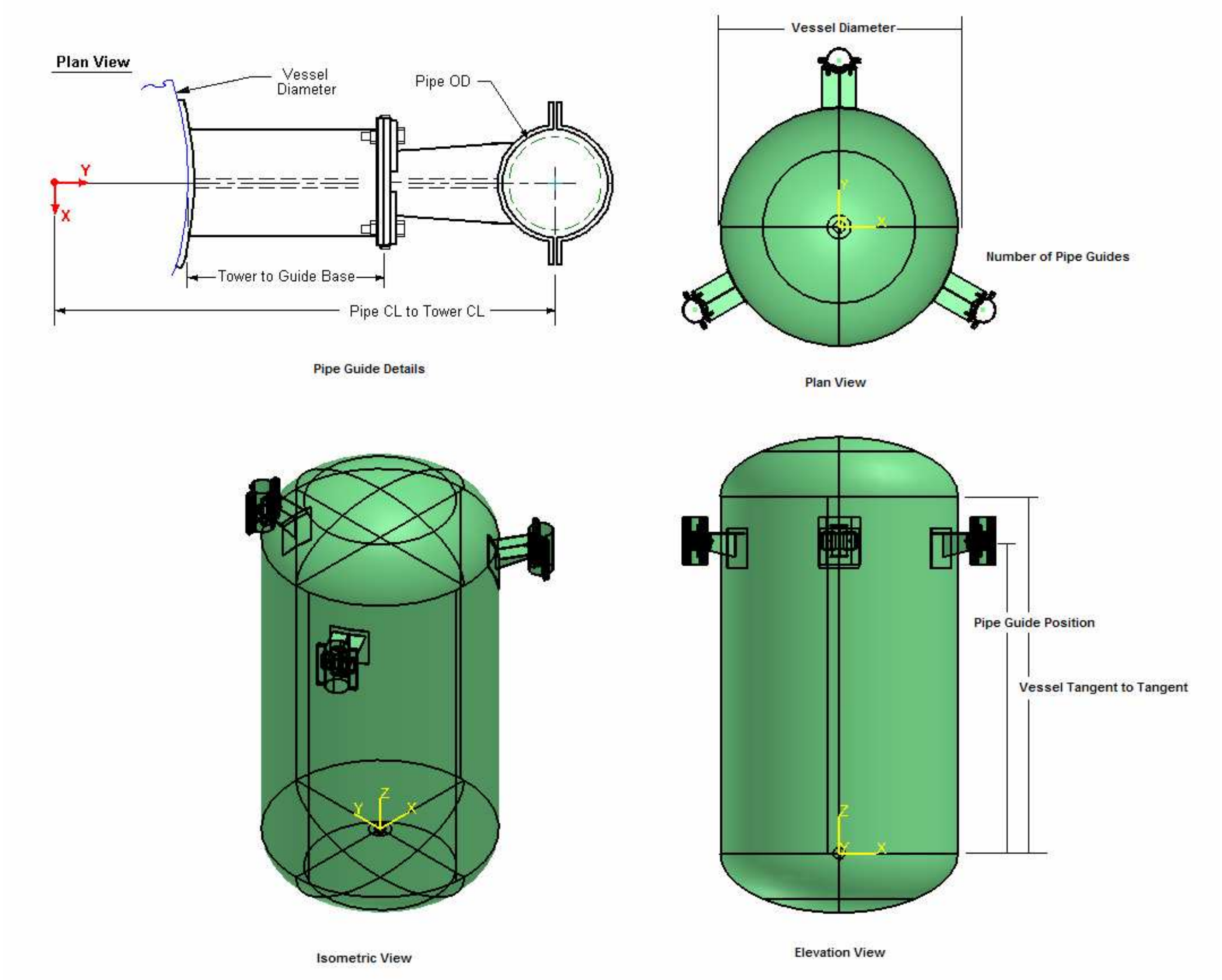
## Objectives

After completing this lab, you will be able to:

- Create a custom assembly occurrence symbol made of equipment components
- Learn to use the Symbol Helper service to create the symbol definition
- Use the Equipment CAD Helper to define the Custom Assembly Definition
- Learn to use the IJDAggregatorDescription , IJDMemberDescriptions, and IJDPropertyDescriptions to define the behaviors of the custom assembly occurrence (CAO)
- Use the CMCount to create variable members
- Use IJDNamingRulesHelper interface to create the naming relations between a naming rule and the object
- Use IJDAttributes interface to get a collection of attributes property
- Use IJDAttribute to get an object's attribute

*Part a: Equipment Component as variable member objects*

In this lab, you will create an equipment symbol as shown below. You start by using the SP3DEqpTemplateAsm template provided by the instructor to create the symbol. This symbol consists of two equipment components to define the symbol's output. Use the Equipment Custom Assembly Definition (CAD) Helper to create the members.



Pipe Guide Details

Plan View

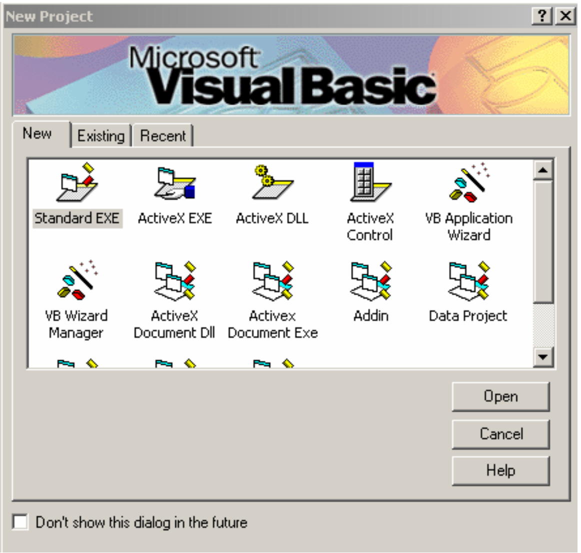
Isometric View

Elevation View

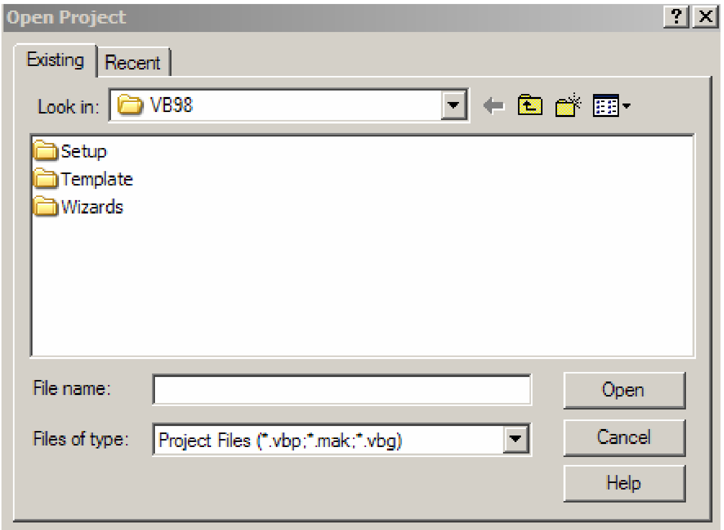
1. Create a directory called lab5 as follows:

   *c:\train\lab5*

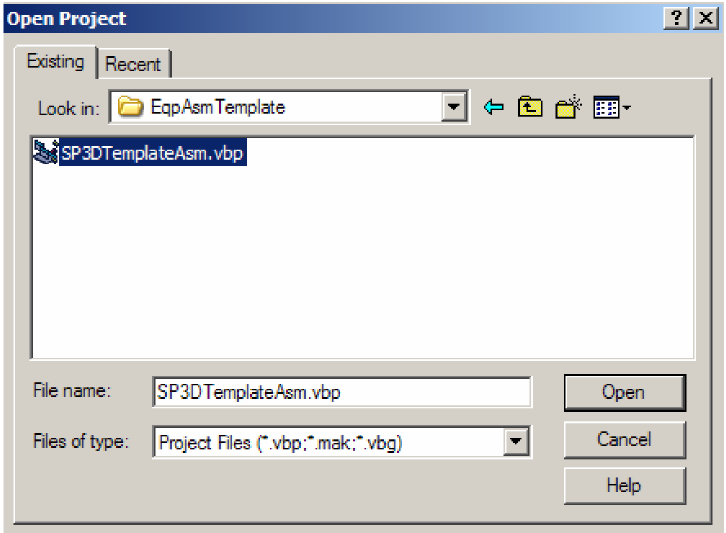
2. Run Microsoft Visual Basic 6.0
3. Close the Microsoft New Project dialog box.



4. Select *File* -> *Open Project* option to open the Open Project Dialog box



5. Navigate to c:\train\EqpAsmTemplate and open the SP3DTemplateAsm Template project



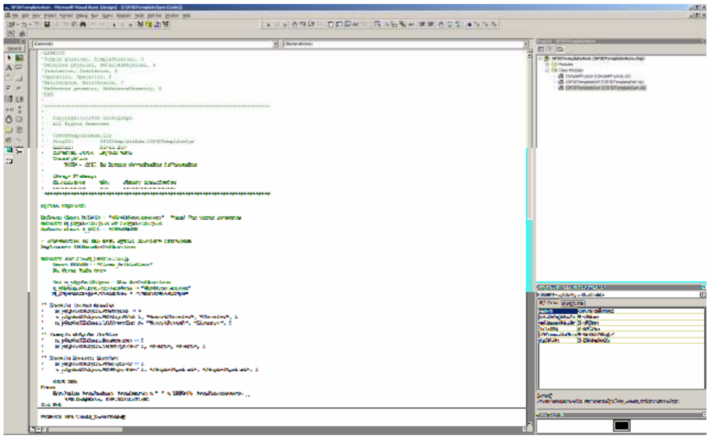
Look in: EqpAsmTemplate

SP3DTemplateAsm.vbp

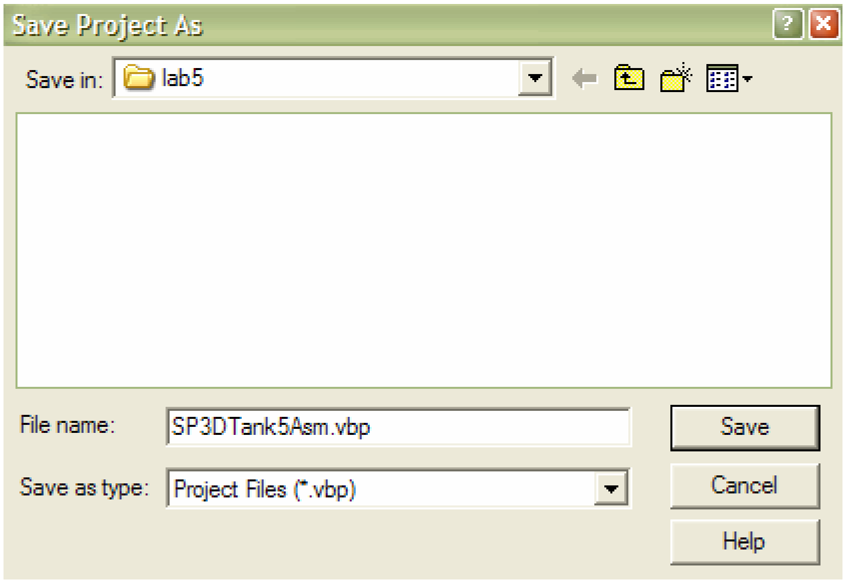
File name: SP3DTemplateAsm.vbp

Files of type: Project Files (*.vbp;*.mak;*.vbg)

6. Setup the Visual Basic Development Environment as shown below:



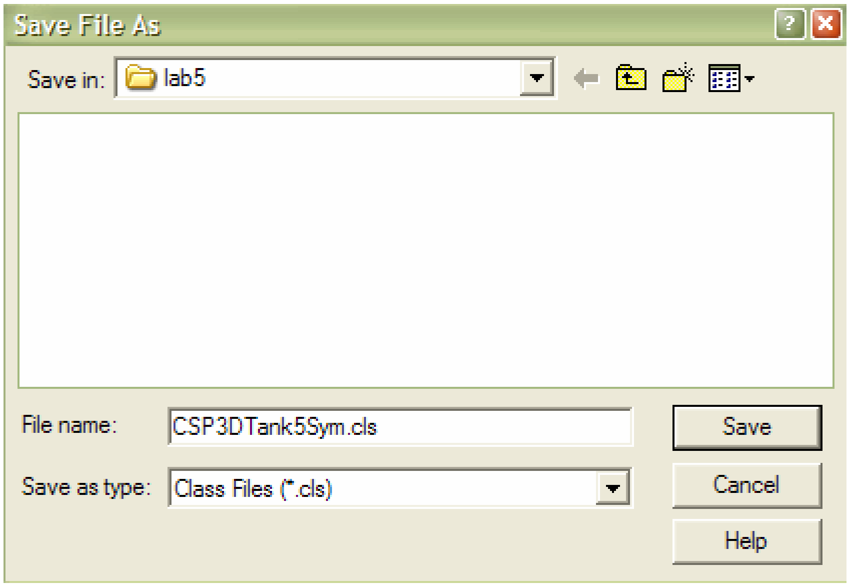
7. Go to the Visual Basic Explorer Window and select the Project node. Select *File* -> *Save Project As* option to save the project as SP3DTank5Asm.vbp under the lab5 directory



File name: SP3DTank5Asm.vbp

Save as type: Project Files (*.vbp)

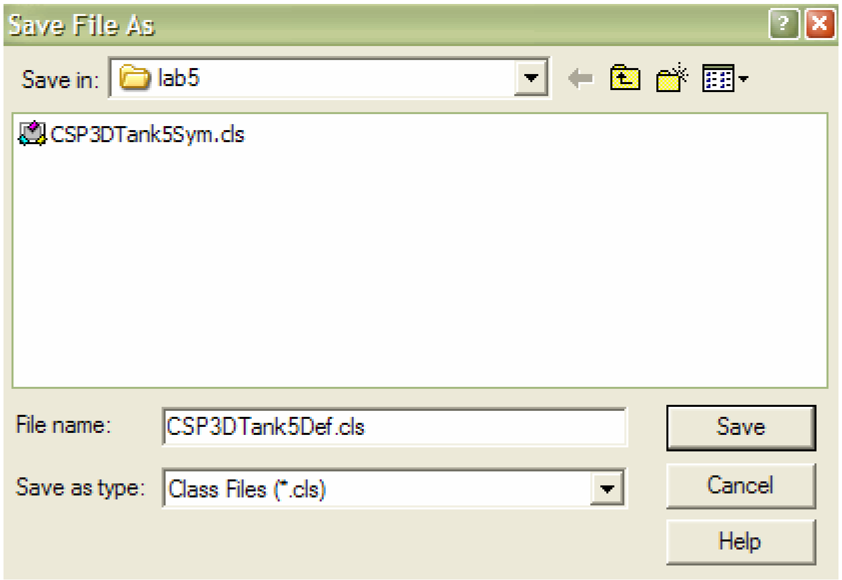
8. Go to the Visual Basic Explorer Window and select the CSP3DTemplateSym class node. Select *File* -> *Save CSP3DTemplateSym.cls As* option to save the class module as CSP3DTank5Sym.cls under lab5 directory



File name: CSP3DTank5Sym.cls

Save as type: Class Files (*.cls)

9. Go to the Visual Basic Explorer Window and select the CSP3DTemplateDef class node. Select *File* -> *Save CSP3DTemplateDef.cls As* option to save the class module as CSP3DTank5Def.cls under lab5 directory



Save in: lab5

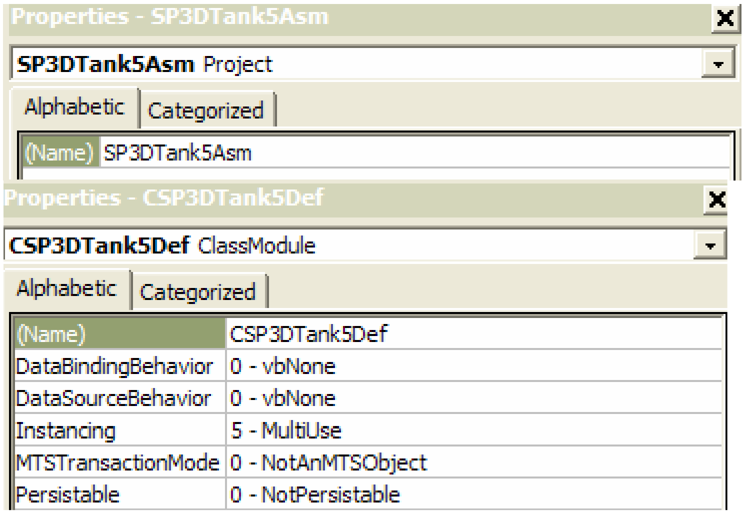
CSP3DTank5Sym.cls

File name: CSP3DTank5Def.cls

Save as type: Class Files (*.cls)

10. Go to the Visual Basic Explorer Window and select the CSimplePhysical class node. Select *File -> Save CSimplePhysical.cls As* option to save the class module as CSimplePhysical.cls under lab5 directory.

11. Go to the Properties Window and change the name of the Project and both Class Modules as follows:

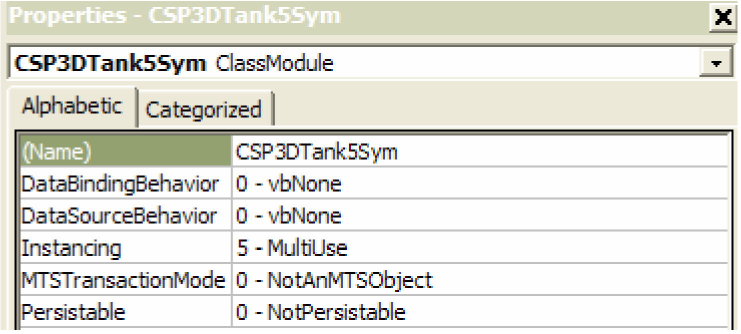


**SP3DTank5Asm** Project

| | |
|---|---|
| (Name) | SP3DTank5Asm |

**CSP3DTank5Def** ClassModule

| | |
|---|---|
| (Name) | CSP3DTank5Def |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |



Properties - CSP3DTank5Sym

**CSP3DTank5Sym** ClassModule

| | |
|---|---|
| (Name) | CSP3DTank5Sym |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |

12. Go to the General Declarations section in CSP3DTank5Sym module and change the value of the *Constant Module variable* from *“CSP3DTemplateAsm:”* to *“CSP3DTank5Asm:”*

```
Private Const MODULE = "CSP3DTank5Asm:"  'Used for error messages
```

13. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
    Set m_oSymbolHelper = New SymbolServices
    m_oSymbolHelper.ProjectName = "SP3DTank5Asm"
    m_oSymbolHelper.ClassName = "CSP3DTank5Sym"
```

14. In this Class_Initialize() routine, add the following code to define the inputs and aspects definition for this symbol. Note: there is no outputs section.

```
' Inputs Section
    m_oSymbolHelper.NumInputs = 5
    m_oSymbolHelper.AddInputDef 1, "VesselDiameter", "VesselDiameter", 1
    m_oSymbolHelper.AddInputDef 2, "VesselTantoTan", "VesselTantoTan", 3
    m_oSymbolHelper.AddInputDef 3, "InsulationThickness", "InsulationThickness", 0.06

    m_oSymbolHelper.AddInputDef 4, "PipeGuidePosition", "PipeGuidePosition", 0.4
    m_oSymbolHelper.AddInputDef 5, "NoOfPipeGuide", "NoOfPipeGuide", 3
' Outputs Section

' Aspects Section
    m_oSymbolHelper.NumAspects = 1
    m_oSymbolHelper.AddAspectDef 1, "SimplePhysical", "SimplePhysical", 1
```

15. Go to CSimplePhysical Class module/Run subroutine and make sure there is no code to get the inputs.

16. Go to the General Declarations section in CSP3DTank5Def module and change the value of the *Constant Module variable* from “*"SP3DTemplateAsm:CSP3DTemplateDef"* to *"SP3DTank5Asm:CSP3DTank5Def"*

17. Go to the top of the CSP3DTank5Def module and declare the following variables:

```
Private Const MODULE = "SP3DTank5Asm:CSP3DTank5Def"

Private Const IID_IJDATTRIBUTES = "{B25FD387-CFEB-11D1-850B-080036DE8E03}"
Private Const IID_IJDGEOMETRY = "{A1732CBF-5136-11D1-9770-080036754203}"

Private m_oEquipCADHelper As IJEquipCADHelper
Private m_oEditErrors As IJEditErrors

Private m_avSymbolArrayOfInputs()   As Variant

'VDrum
Private m_VesselDiameter As Double
Private m_VesselTantoTan As Double
Private m_dInsulationThickness As Double

'PipeGuide
Private m_PipeGuidePosition As Double
Private m_NoOfPipeGuide As Long

Private PI As Double
```

18. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
 m_oEquipCADHelper.ProjectName = "SP3DTank5Asm"
 m_oEquipCADHelper.ClassName = "CSP3DTank5Def"
```

19. In this Class_Initialize() routine, initialize the following variable:

```
PI = 4 * Atn(1)
```

20. Go to CSP3DTank5Def Class module. Declare the appropriate custom methods to manage the drum and the pipe guide equipment components as follows:

```
'Add your code here for the declaration of the Public Custom Methods used to manage new members
'Add new member(VDrum) to the definition

Set oMemberDescription = Nothing
Set oMemberDescription = oMemberDescriptions.AddMember("VDrum", 1, "CMConstructVDrum", _
                                                              imsCOOKIE_ID_USS_LIB)
oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsVDrum"
oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructVDrum"
oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalVDrum"
oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseVDrum"

Set oPropertyDescriptions = Nothing
Set oPropertyDescriptions = oMemberDescription
oPropertyDescriptions.AddProperty "VDrumProperties", 1, IID_IJDATTRIBUTES, "CMEvaluateVDrum", _
                                                              imsCOOKIE_ID_USS_LIB
oPropertyDescriptions.AddProperty "VDrumGeometryProperties", 2, IID_IJDGEOMETRY, _
                                                 "CMEvaluateGeometryVDrum", imsCOOKIE_ID_USS_LIB

'Add new member(Pipe Guide) to the definition

Set oMemberDescription = Nothing
Set oMemberDescription = oMemberDescriptions.AddMember("Pipeguide", 2, "CMConstructPipeguide", _
                                                              imsCOOKIE_ID_USS_LIB)
oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsPipeguide"
```

```
    oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructPipeguide"
    oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalPipeguide"
    oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleasePipeguide"
    oMemberDescription.SetCMCount imsCOOKIE_ID_USS_LIB, "CMCountPipeguide"

    Set oPropertyDescriptions = Nothing
    Set oPropertyDescriptions = oMemberDescription
    oPropertyDescriptions.AddProperty "PipeguideProperties", 1, IID_IJDATTRIBUTES, _
                                            "CMEvaluatePipeguide", imsCOOKIE_ID_USS_LIB
    oPropertyDescriptions.AddProperty "PipeguideGeometryProperties", 2, IID_IJDGEOMETRY, _
                                        "CMEvaluateGeometryPipeguide", imsCOOKIE_ID_USS_LIB
```

21. Go to IJEquipUserAttrMgmt_OnPreLoad function and add the following code. Use the IJEquipAttrDescriptor interface to set the VDrum properties read only.

```
Private Function IJEquipUserAttrMgmt_OnPreLoad(ByVal pIJDAttrs As IJDAttributes, ByVal
CollAllDisplayedValues As Object) As String
    Const METHOD = "IJEquipUserAttrMgmt_OnPreLoad"
    On Error GoTo ErrorHandler

        Dim oMemberDescription As IJDMemberDescription

        Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)
        Dim oAttrCollection As Collection
        Dim oAttributeDescriptor As IJEquipAttrDescriptor
        Dim m As Long

' set dimension and deletable attributes to read only.

        Set oAttrCollection = CollAllDisplayedValues
        Select Case oMemberDescription.Name
        Case "VDrum"
            For m = 1 To oAttrCollection.Count
                Set oAttributeDescriptor = oAttrCollection.Item(m)
                Select Case UCase(oAttributeDescriptor.InterfaceName)
                    Case "IJUAVESSELDIAMETER"
                      oAttributeDescriptor.AttrState = adsReadOnly
                    Case "IJUAVESSELTANTOTAN"
                      oAttributeDescriptor.AttrState = adsReadOnly
                    Case "IJDELETABLEMEMBER"
                      oAttributeDescriptor.AttrState = adsReadOnly
                    Case Else
                        '
                End Select
            Next
        Case Else
            '
        End Select

    Set oAttrCollection = Nothing
    Set oAttributeDescriptor = Nothing
    Set oMemberDescription = Nothing

    IJEquipUserAttrMgmt_OnPreLoad = ""
    Exit Function
ErrorHandler:
    IJEquipUserAttrMgmt_OnPreLoad = "ERROR"
    HandleError MODULE, METHOD
End Function
```

22. Go to the end of CSP3DTank5Def Class module. Add the custom methods to manage the drum and the pipe guide equipment components as follows:

    **Custom Method Construct**:
    This method is in charge of the creation of the CAO member object (VDrum 01-EC). Use EquipCADHelper CreateEquipmentComponent () method to create the member given the equipment component part number. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructVDrum(ByVal pMemberDescription As IJDMemberDescription, _
                            ByVal pResourceManager As IUnknown, _
                            ByRef pObject As Object)
    Const METHOD = "CMConstructVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Dim oEquipment As IJEquipment
    Set oEquipment = pMemberDescription.CAO
    GetDimensionsFromSymbolArray oEquipment

    'Create Equipment Component
    Set pObject = m_oEquipCADHelper.CreateEquipmentComponent(pMemberDescription,
pResourceManager, "VDrum 01-EC", "VDrum")

    'create name for the member
    SetObjNameRule pObject, "CPEquipmentComponent"

    Set oEquipment = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Final:**
There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructVDrum(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMFinalConstructVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom method Inputs:**

There is no need to add any code for this custom method.

```
Public Sub CMSetInputsVDrum(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMSetInputsVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Evaluate:

This method is in charge of setting the attribute values to the member object. Use the MakeMemberDeletable method to set the member deletable.

```
Public Sub CMEvaluateVDrum(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
    Const METHOD = "CMEvaluateVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    GetDimensionsFromSymbolArray oPropertyDescription.CAO

    Dim oAttribs As IJDAttributes
    Dim oSmartOcc As IJSmartOccurrence

    Set oSmartOcc = oPropertyDescription.Object
    Set oAttribs = oSmartOcc '.ItemObject

    oAttribs.CollectionOfAttributes("IJUAVesselDiameter").Item("VesselDiameter").Value = m_VesselDiameter
    oAttribs.CollectionOfAttributes("IJUAVesselTantoTan").Item("VesselTantoTan").Value =
m_VesselTantoTan
    oAttribs.CollectionOfAttributes("IJInsulationThickness").Item("InsulationThickness").Value =
m_dInsulationThickness
    oAttribs.CollectionOfAttributes("IJDeletableMember").Item("CanBeDeleted").Value = True

' set member deletable

    Dim oMemberDescription As IJDMemberDescription
    Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(oAttribs)
    m_oEquipCADHelper.MakeMemberDeletable oMemberDescription, oAttribs, True

    Set oAttribs = Nothing
    Set oSmartOcc = Nothing
    Set oMemberDescription = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method GeometryEvaluate:

This method is in charge of setting the transformation matrix to move the member object relative to the equipment. Use the TransformFromECStoGCS function to maintain the VDrum member coordinate system relative to the Equipment coordinate system.

```
Public Sub CMEvaluateGeometryVDrum(ByVal oPropertyDescription As IJDPropertyDescription, pObject As
Object)
    Const METHOD = "CMEvaluateGeometryVDrum"
    On Error GoTo ErrorHandler
    LogCalls METHOD

'Add the following code to avoid the eqp component to move alone

    Dim oEqpComp As IJEquipmentComponent
    Dim oEqpCompMatrix As IJEquipment
    Dim oEquipment As IJEquipment

    Set oEqpComp = oPropertyDescription.Object
    oEqpComp.GetParent oEquipment
    Set oEqpCompMatrix = oEqpComp

    GetDimensionsFromSymbolArray oEquipment

    Dim otransform As IngrGeom3D.IJDT4x4
    Set otransform = New DT4x4
    Dim iVector As IJDVector
    Set iVector = New DVector

    otransform.LoadIdentity
    iVector.x = 0
    iVector.y = 0
    iVector.z = 0
    otransform.Translate iVector
    oEqpCompMatrix.SetMatrix otransform

    TransformFromECStoGCS oEquipment, oEqpCompMatrix

    Set oEquipment = Nothing
    Set oEqpComp = Nothing
    Set oEqpCompMatrix = Nothing

    Set iVector = Nothing
    Set otransform = Nothing

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Conditional:

This method checks if the member is conditional based on the CanBeDeleted flag. Remember, we added code in the CMEvaluate to make the member deletable.

```
Public Sub CMConditionalVDrum(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As
Boolean)
```

```
    Const METHOD = "CMConditionalVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom method Release:**
There is no need to add any code for this custom method

```
Public Sub CMReleaseVDrum(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMReleaseVDrum"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Construct**:
This method is in charge of the creation of the CAO member object (PipeguideClamped 01-EC). Use EquipCADHelper CreateEquipmentComponent () method to create the member given the equipment component part number. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructPipeguide(ByVal pMemberDescription As IJDMemberDescription, _
                                ByVal pResourceManager As IUnknown, _
                                ByRef pObject As Object)
    Const METHOD = "CMConstructPipeguide"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Dim oEquipment As IJEquipment
    Set oEquipment = pMemberDescription.CAO
    GetDimensionsFromSymbolArray oEquipment

    'Create Equipment Component
    Set pObject = m_oEquipCADHelper.CreateEquipmentComponent(pMemberDescription,
pResourceManager, "PipeguideClamped 01-EC", "Pipeguide")
    SetObjNameRule pObject, "CPEquipmentComponent"
    Set oEquipment = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Final:**
There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructPipeguide(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMFinalConstructPipeguide"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Inputs:

There is no need to add any code for this custom method.

```
Public Sub CMSetInputsPipeguide(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMSetInputsPipeguide"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Evaluate:

This method is in charge of setting the attribute values to the member object.

```
Public Sub CMEvaluatePipeguide(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
    Const METHOD = "CMEvaluatePipeguide"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Dim oAttribs As IJDAttributes
    Dim oSmartOcc As IJSmartOccurrence

    Set oSmartOcc = oPropertyDescription.Object
    Set oAttribs = oSmartOcc '.ItemObject

    oAttribs.CollectionOfAttributes("IJUAVesselDiameter").Item("VesselDiameter").Value = m_VesselDiameter

    Set oAttribs = Nothing
    Set oSmartOcc = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method GeometryEvaluate:

This method is in charge of setting the transformation matrix to move the member object relative to the equipment. Use the TransformFromECStoGCS function to maintain the Pipe Guide member coordinate system relative to the Equipment coordinate system.

```
Public Sub CMEvaluateGeometryPipeguide(ByVal oPropertyDescription As IJDPropertyDescription, pObject
As Object)
```

```
    Const METHOD = "CMEvaluateGeometryPipeguide"
    On Error GoTo ErrorHandler
    LogCalls METHOD

        Dim oEqpComp As IJEquipmentComponent
    Dim oEqpCompMatrix As IJEquipment
    Dim oEquipment As IJEquipment

    Set oEqpComp = oPropertyDescription.Object
    oEqpComp.GetParent oEquipment
    Set oEqpCompMatrix = oEqpComp

    GetDimensionsFromSymbolArray oEquipment

    Dim otransform1 As IngrGeom3D.IJDT4x4
    Set otransform1 = New DT4x4
    Dim otransform2 As IngrGeom3D.IJDT4x4
    Set otransform2 = New DT4x4

    Dim iVector As IJDVector
    Set iVector = New DVector

    iVector.x = 0
    iVector.y = 0
    iVector.z = 1
    otransform1.LoadIdentity
    otransform2.LoadIdentity

'set the rotation matrix

    otransform2.Rotate oPropertyDescription.index * (2 * PI / m_NoOfPipeGuide), iVector
    otransform1.MultMatrix otransform2

'set the translation matrix
    iVector.x = 0
    iVector.y = 0
    iVector.z = m_PipeGuidePosition
    otransform1.Translate iVector

    oEqpCompMatrix.SetMatrix otransform1

    TransformFromECStoGCS oEquipment, oEqpComp

    Set oEquipment = Nothing
    Set oEqpComp = Nothing
    Set oEqpCompMatrix = Nothing
    Set otransform1 = Nothing
    Set otransform2 = Nothing

    Set iVector = Nothing

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Count:**
This custom method read the global variable m_NoOfPipeGuide to determine total number of members need to be created.

```
Public Sub CMCountPipeguide(ByVal pMemberDesc As IJDMemberDescription, ByRef Count As Long)
   Const METHOD = "CMCountPipeguide"
   On Error GoTo ErrorHandler

   GetDimensionsFromSymbolArray pMemberDesc.CAO
   Count = m_NoOfPipeGuide

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method Conditional:**
This method makes sure that the system will not construct the pipe guide if the count is less or equal to zero.

```
Public Sub CMConditionalPipeguide(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As
Boolean)
   Const METHOD = "CMConditionalPipeguide"
   LogCalls METHOD
   On Error GoTo ErrorHandler

   If pMemberDesc.index <= 0 Then
   IsNeeded = False
   Else
   IsNeeded = True
   End If
   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom Method Release:**
There is no need to add any code for this custom method.

```
Public Sub CMReleasePipeguide(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMReleasePipeguide"
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

23. Add the following subroutine to convert the array of inputs in a set of global variables.

```
Private Sub GetDimensionsFromSymbolArray(SmartOccurrence As IJSmartOccurrence)
   Const METHOD = "GetDimensionsFromSymbolArray"
   On Error GoTo ErrorHandler
```

```
        m_avSymbolArrayOfInputs = m_oEquipCADHelper.GetSymbolArrayOfInputs(SmartOccurrence)

        'Inputs,  from equipment symbol code
        'Set m_oPartFclt = m_avSymbolArrayOfInputs(1)
        m_VesselDiameter = m_avSymbolArrayOfInputs(2)
        m_VesselTantoTan = m_avSymbolArrayOfInputs(3)
        m_dInsulationThickness = m_avSymbolArrayOfInputs(4)

        m_PipeGuidePosition = m_avSymbolArrayOfInputs(5)
        m_NoOfPipeGuide = m_avSymbolArrayOfInputs(6)

        Exit Sub
    ErrorHandler:
        HandleError MODULE, METHOD
    End Sub
```

24. Add the following subroutine to position and orient the member with respect to the equipment.

```
    Private Sub TransformFromECStoGCS(Equipment As IJEquipment, Object As Object)
        Const METHOD = "TransformFromECStoGCS"
        LogCalls METHOD
        On Error GoTo ErrorHandler
        Dim oEqpMatrix As IJDT4x4
        Dim oShapeMatrix As IJDT4x4
        Dim otransform As IJDGeometry
        Dim oShape As IJShape

        If Not Object Is Nothing Then
            If TypeOf Object Is IJDGeometry Then
                Equipment.GetMatrix oEqpMatrix
                Set otransform = Object
                otransform.DTransform oEqpMatrix
                Set otransform = Nothing
                Set oEqpMatrix = Nothing
            End If
        End If

        Set otransform = Nothing
        Set oEqpMatrix = Nothing
        Set oShape = Nothing
        Set oShapeMatrix = Nothing

        Exit Sub
    ErrorHandler:
        Set otransform = Nothing
        Set oEqpMatrix = Nothing
        Set oShape = Nothing
        Set oShapeMatrix = Nothing
        HandleError MODULE, METHOD
    End Sub
```

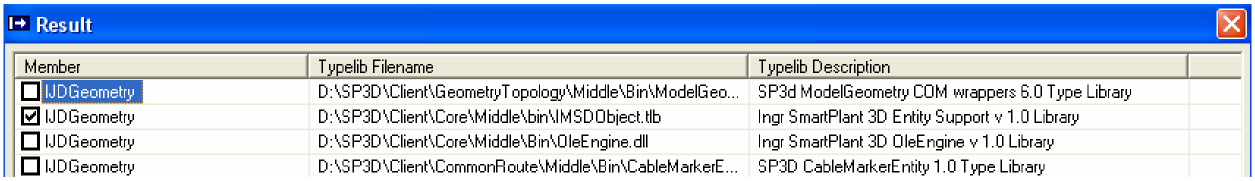
25. Add the following subroutine to set the naming relation and generate a name based on the default naming rule.

```
Public Sub SetObjNameRule(ByRef obj As Object, ByRef CLASSNAME As String)
' Apply the namerule using the IJDNamingRulesHelper helper interface
Const METHOD = "SetNameRule"
On Error GoTo ErrorHandler

    Dim NameRule As String
    Dim NamingRules As IJElements

    Dim oNameRuleHlpr As GSCADNameRuleSemantics.IJDNamingRulesHelper

'Returns a collection of the naming rules available in the catalog database
'for the given object
    Set oNameRuleHlpr = New GSCADNameRuleHlpr.NamingRulesHelper
    Call oNameRuleHlpr.GetEntityNamingRulesGivenName(CLASSNAME, NamingRules)
'get the first namerule from the collection
    Dim oNameRuleHolder As GSCADGenericNamingRulesFacelets.IJDNameRuleHolder
    Set oNameRuleHolder = NamingRules.Item(1)
'Create relations "NamedEntity" and "EntityNamingRule" and obj
    Dim oNameRuleAE As GSCADGenNameRuleAE.IJNameRuleAE
    Call oNameRuleHlpr.AddNamingRelations(obj, oNameRuleHolder, oNameRuleAE)

    GoTo CleanObjects
ErrorHandler:
    HandleError MODULE, METHOD

CleanObjects:
    Set oNameRuleHlpr = Nothing
    Set oNameRuleHolder = Nothing
    Set oNameRuleAE = Nothing
    Set NamingRules = Nothing
End Sub
```

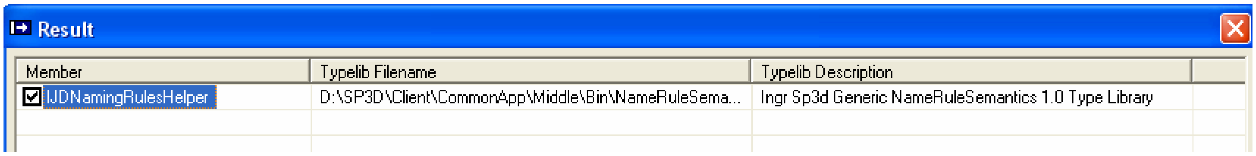
26. Add the following subroutine to log any error.

```
Private Sub LogCalls(sMethod As String)

    If Not m_oEditErrors Is Nothing Then
        m_oEditErrors.Add 5000, m_oEquipCADHelper.ProjectName & "." &
m_oEquipCADHelper.CLASSNAME, "Entering " & sMethod
    End If

End Sub
```

27. Compile the Visual Basic project and save the dll as SP3DTank5Asm.dll in the c:\Train\lab5 Note: Use the SP3D Reference Tool to attach the missing reference libraries.

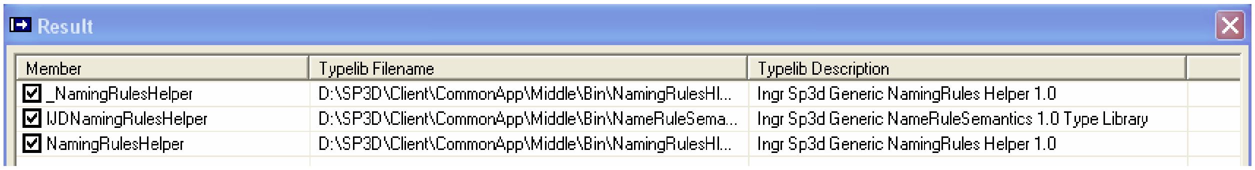


| Member | Typelib Filename | Typelib Description |
|---|---|---|
| □ IJDGeometry | D:\SP3D\Client\GeometryTopology\Middle\Bin\ModelGeo... | SP3d ModelGeometry COM wrappers 6.0 Type Library |
| ☑ IJDGeometry | D:\SP3D\Client\Core\Middle\bin\IMSDObject.tlb | Ingr SmartPlant 3D Entity Support v 1.0 Library |
| □ IJDGeometry | D:\SP3D\Client\Core\Middle\Bin\OleEngine.dll | Ingr SmartPlant 3D OleEngine v 1.0 Library |
| □ IJDGeometry | D:\SP3D\Client\CommonRoute\Middle\Bin\CableMarkerE... | SP3D CableMarkerEntity 1.0 Type Library |



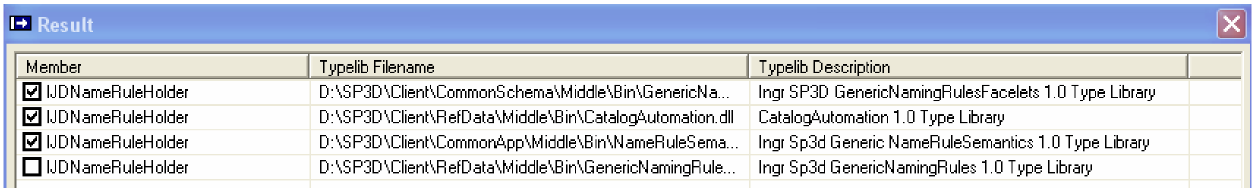
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDNamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |



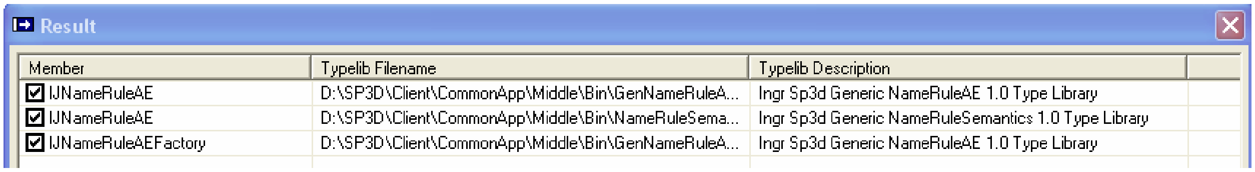
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ _NamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NamingRulesHl... | Ingr Sp3d Generic NamingRules Helper 1.0 |
| ☑ IJDNamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☑ NamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NamingRulesHl... | Ingr Sp3d Generic NamingRules Helper 1.0 |



Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDNameRuleHolder | D:\SP3D\Client\CommonSchema\Middle\Bin\GenericNa... | Ingr SP3D GenericNamingRulesFacelets 1.0 Type Library |
| ☑ IJDNameRuleHolder | D:\SP3D\Client\RefData\Middle\Bin\CatalogAutomation.dll | CatalogAutomation 1.0 Type Library |
| ☑ IJDNameRuleHolder | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| □ IJDNameRuleHolder | D:\SP3D\Client\RefData\Middle\Bin\GenericNamingRule... | Ingr Sp3d GenericNamingRules 1.0 Type Library |

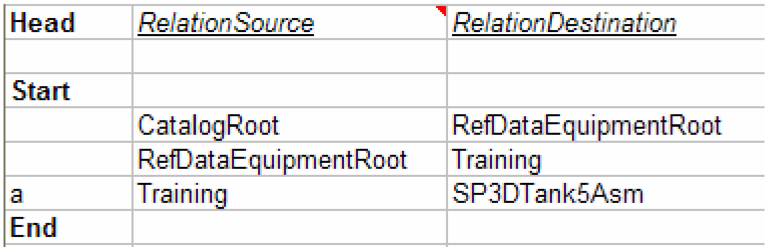


Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJNameRuleAE | D:\SP3D\Client\CommonApp\Middle\Bin\GenNameRuleA... | Ingr Sp3d Generic NameRuleAE 1.0 Type Library |
| ☑ IJNameRuleAE | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☑ IJNameRuleAEFactory | D:\SP3D\Client\CommonApp\Middle\Bin\GenNameRuleA... | Ingr Sp3d Generic NameRuleAE 1.0 Type Library |

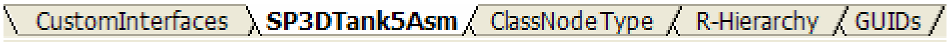
One of the most important steps in Visual Basic programming is to preserve the binary compatibility of your program. Save the final version of your dll file to be binary compatibility in order to preserve the CLSID.

28. Save the Visual Basic SP3DTank5Asm project.
29. Open the SP3DTemplate.xls workbook. Go the R-Hierarchy sheet and add the following entry.



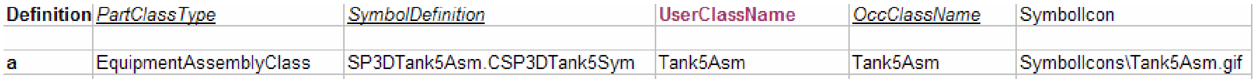
| Head | *RelationSource* | *RelationDestination* |
|---|---|---|
| | | |
| **Start** | | |
| | CatalogRoot | RefDataEquipmentRoot |
| | RefDataEquipmentRoot | Training |
| a | Training | SP3DTank5Asm |
| **End** | | |

30. Go to the SP3DTemplateAsm sheet and rename it as SP3DTank5Asm.



31. Go to the Class definition section and add/edit as follows:

In the Definition Section rows:

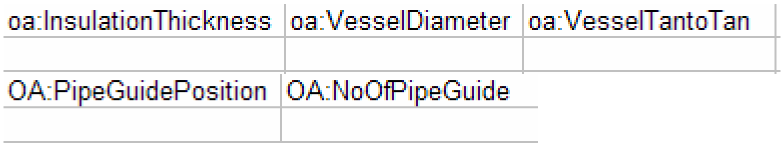


| Definition | PartClassType | SymbolDefinition | UserClassName | OccClassName | SymbolIcon |
|---|---|---|---|---|---|
| | | | | | |
| a | EquipmentAssemblyClass | SP3DTank5Asm.CSP3DTank5Sym | Tank5Asm | Tank5Asm | SymbolIcons\Tank5Asm.gif |

Note:

- Creating the bmp or gif file is optional. You can use Microsoft Paint to create the file and save it under your \\machine\symbols\SymbolIcons

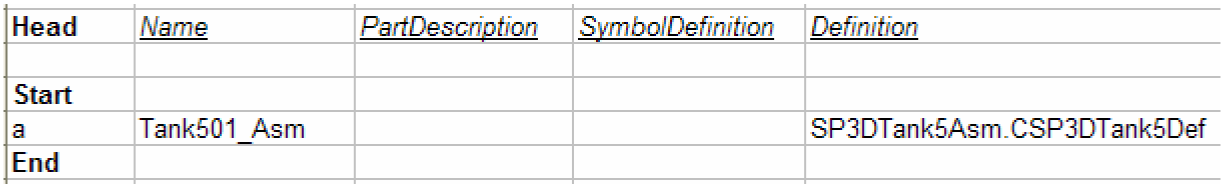
Occurrence attributes:



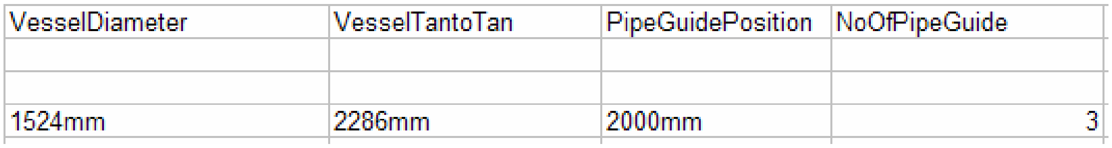
| oa:InsulationThickness | oa:VesselDiameter | oa:VesselTantoTan |
|---|---|---|
| | | |

| OA:PipeGuidePosition | OA:NoOfPipeGuide |
|---|---|
| | |

In the Part Section rows:

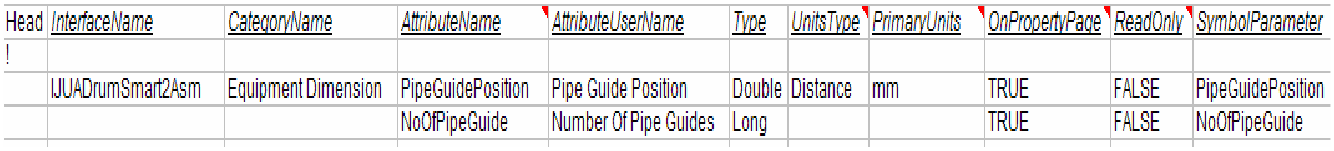


| Head | Name | PartDescription | SymbolDefinition | Definition |
|---|---|---|---|---|
| | | | | |
| Start | | | | |
| a | Tank501_Asm | | | SP3DTank5Asm.CSP3DTank5Def |
| End | | | | |



| VesselDiameter | VesselTantoTan | PipeGuidePosition | NoOfPipeGuide |
|---|---|---|---|
| | | | |
| | | | |
| 1524mm | 2286mm | 2000mm | 3 |

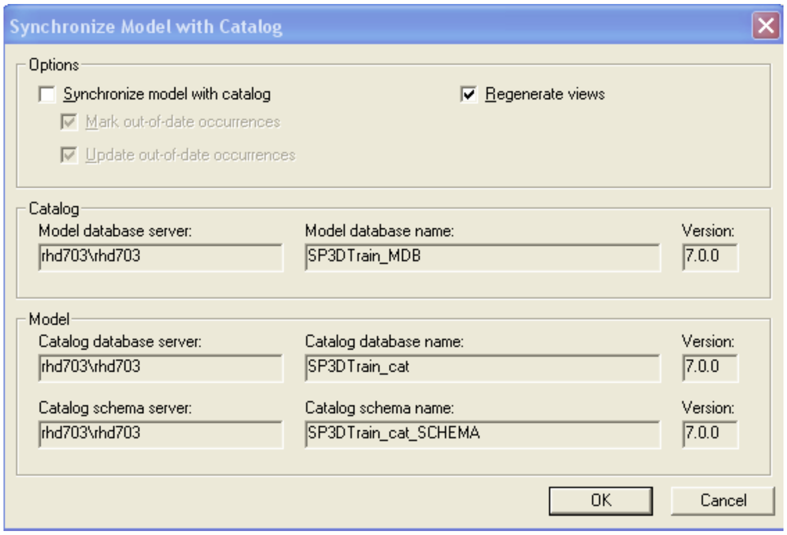
32. Create a new interface called IJUADrumSmart2Asm. Go to the Custom Interface sheet and add the following entries:



| Head | InterfaceName | CategoryName | AttributeName | AttributeUserName | Type | UnitsType | PrimaryUnits | OnPropertyPage | ReadOnly | SymbolParameter |
|---|---|---|---|---|---|---|---|---|---|---|
| ! | | | | | | | | | | |
| | IJUADrumSmart2Asm | Equipment Dimension | PipeGuidePosition | Pipe Guide Position | Double | Distance | mm | TRUE | FALSE | PipeGuidePosition |
| | | | NoOfPipeGuide | Number Of Pipe Guides | Long | | | TRUE | FALSE | NoOfPipeGuide |

33. Save the Excel workbook as SP3DTank5Asm.xls in the c:\Train\lab5.
34. Optional step: Create the Tank5Asm.gif file and place it under \\<MachineName>\Symbols\SymbolIcons
35. Load the information into the catalog using the Add/Modify/Delete Mode. Once the bulkload process is completed, review the log file.
36. Run the Project Management Task. Select the Model in the hierarchy.
37. Select Tools -> Synchronize Model with the Catalog.
38. Uncheck the Synchronize Model with the Catalog option.

***Note**: You just need to update the views in the model.*



Synchronize Model with Catalog

Options

☐ Synchronize model with catalog

☑ Mark out-of-date occurrences

☑ Update out-of-date occurrences

☑ Regenerate views

Catalog

| Model database server: | Model database name: | Version: |
|---|---|---|
| rhd703\rhd703 | SP3DTrain_MDB | 7.0.0 |

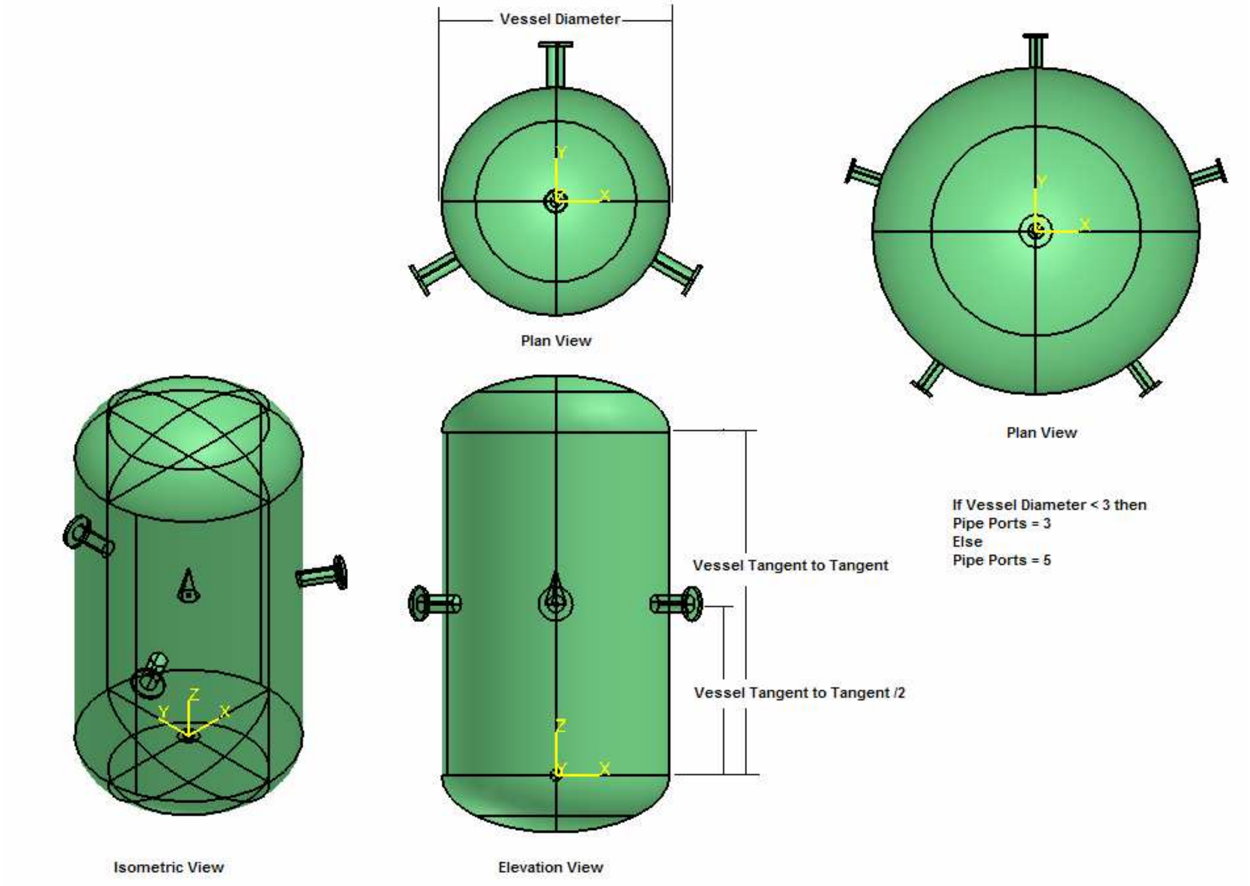
Model

| Catalog database server: | Catalog database name: | Version: |
|---|---|---|
| rhd703\rhd703 | SP3DTrain_cat | 7.0.0 |
| **Catalog schema server:** | **Catalog schema name:** | **Version:** |
| rhd703\rhd703 | SP3DTrain_cat_SCHEMA | 7.0.0 |

39. Hit “OK” Button.
40. Once the process is completed, Right click the training plant icon and select “Regenerate the Reports database” option to re-create the views in the report database.
41. Go to the Equipment Task and place the SP3DTank5Asm.

*Part b: Piping Nozzles as variable member objects*

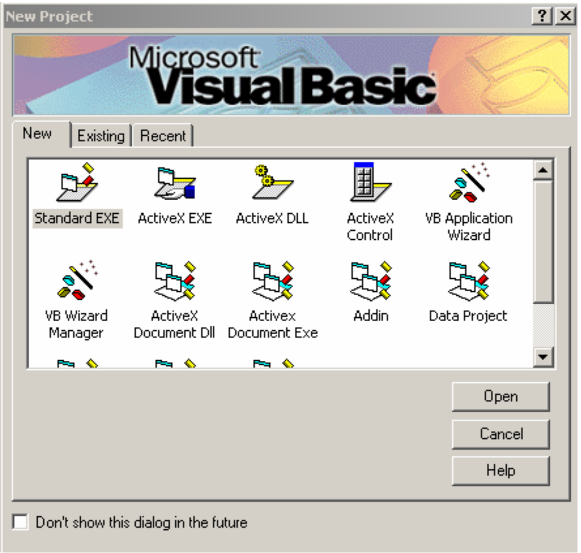
In this lab, you will create an equipment symbol as shown below. You start by using the SP3DEqpTemplateAsm template provided by the instructor to create the symbol. This symbol consists of one equipment component, datum shape and piping ports to define the symbol’s output. The number of pipe ports will depend on the Vessel Diameter. Use the Equipment Custom Assembly Definition (CAD) Helper to create the members.



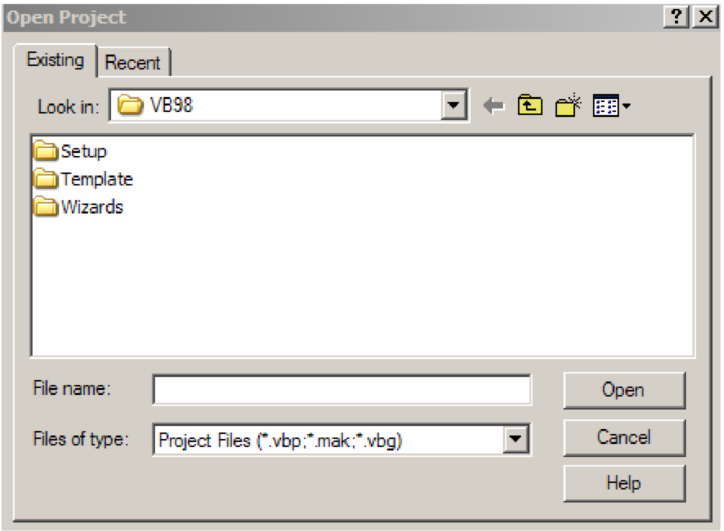
1. Create a directory called lab4 as follows:

   *c:\train\lab6*

2. Run Microsoft Visual Basic 6.0
3. Close the Microsoft New Project dialog box.



4. Select *File* -> *Open Project* option to open the Open Project Dialog box

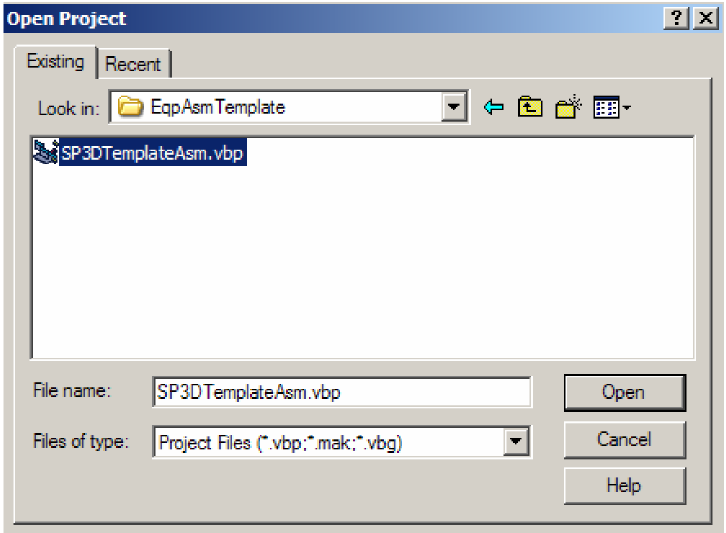


Look in: VB98

File name:

Files of type: Project Files (*.vbp;*.mak;*.vbg)

5. Navigate to c:\train\EqpAsmTemplate and open the SP3DTemplateAsm Template project



Open Project

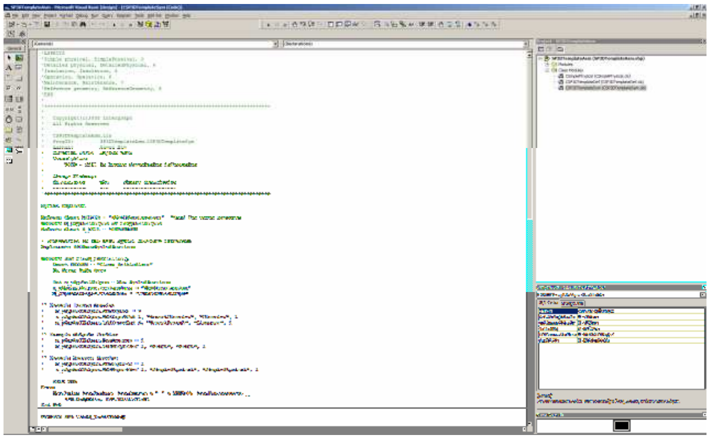
Look in: EqpAsmTemplate

SP3DTemplateAsm.vbp

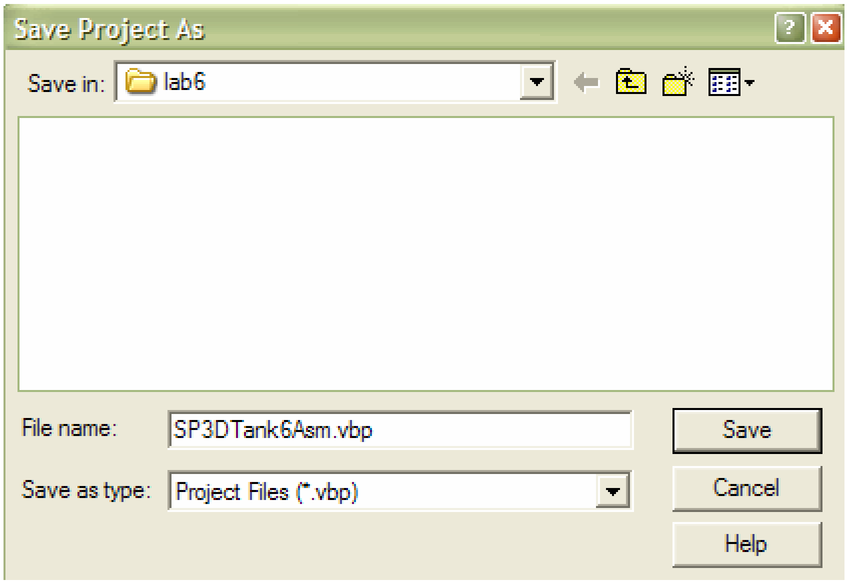
File name: SP3DTemplateAsm.vbp

Files of type: Project Files (*.vbp;*.mak;*.vbg)

6. Setup the Visual Basic Development Environment as shown below:



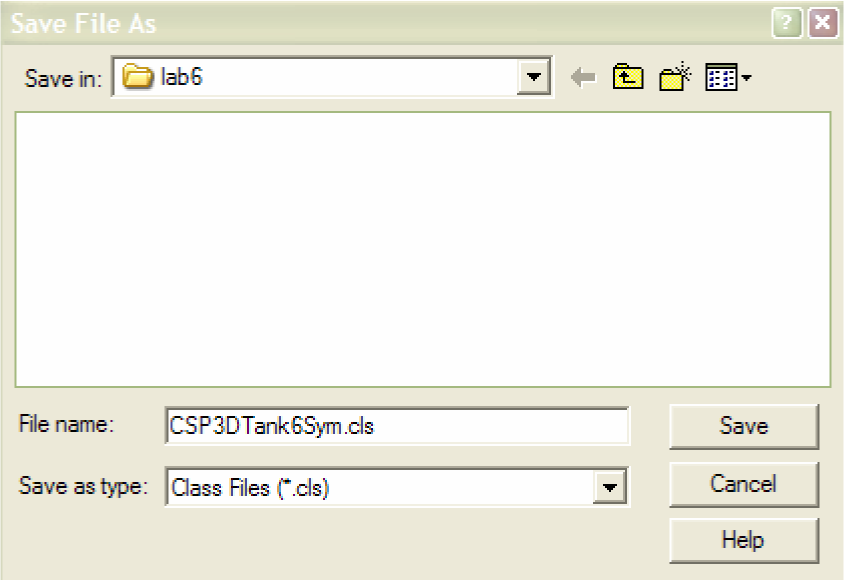
7. Go to the Visual Basic Explorer Window and select the Project node. Select *File* -> *Save Project As* option to save the project as SP3DTank6Asm.vbp under the lab6 directory



File name: SP3DTank6Asm.vbp

Save as type: Project Files (*.vbp)

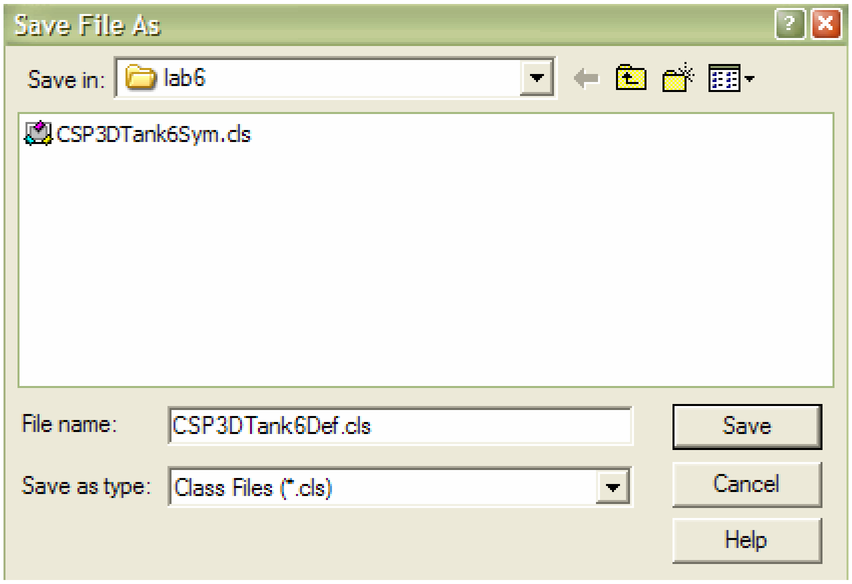
8. Go to the Visual Basic Explorer Window and select the CSP3DTemplateSym class node. Select *File -> Save CSP3DTemplateSym.cls As* option to save the class module as CSP3DTank6Sym.cls under lab6 directory



File name: CSP3DTank6Sym.cls

Save as type: Class Files (*.cls)

9. Go to the Visual Basic Explorer Window and select the CSP3DTemplateDef class node. Select *File -> Save CSP3DTemplateDef.cls As* option to save the class module as CSP3DTank6Def.cls under lab 6 directory



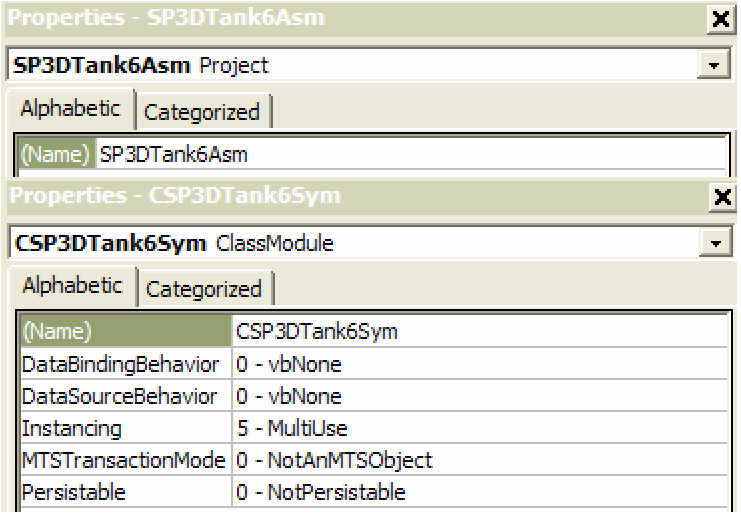
Save in: lab6

File name: CSP3DTank6Def.cls

Save as type: Class Files (*.cls)

10. Go to the Visual Basic Explorer Window and select the CSimplePhysical class node. Select *File -> Save CSimplePhysical.cls As* option to save the class module as CSimplePhysical.cls under lab6 directory.

11. Go to the Properties Window and change the name of the Project and both Class Modules as follows:

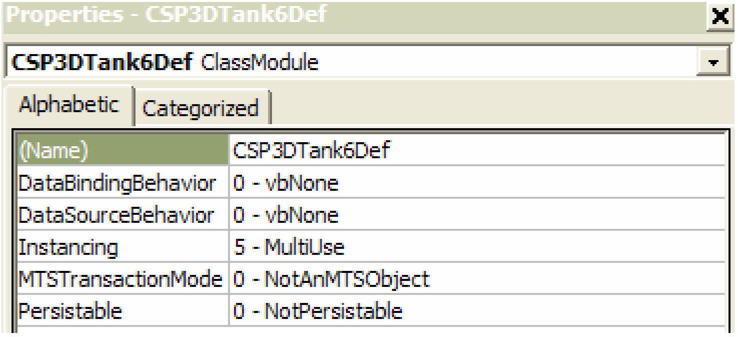


**SP3DTank6Asm** Project

(Name) SP3DTank6Asm

**CSP3DTank6Sym** ClassModule

| | |
|---|---|
| (Name) | CSP3DTank6Sym |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |



Properties - CSP3DTank6Def

**CSP3DTank6Def** ClassModule

Alphabetic | Categorized

| | |
|---|---|
| (Name) | CSP3DTank6Def |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |

12. Go to the General Declarations section in CSP3DTank6Sym module and change the value of the *Constant Module variable* from *“CSP3DTemplateAsm:”* to *“CSP3DTank6Asm:”*

```
Private Const MODULE = "CSP3DTank6Asm:"  'Used for error messages
```

13. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
    Set m_oSymbolHelper = New SymbolServices
    m_oSymbolHelper.ProjectName = "SP3DTank6Asm"
    m_oSymbolHelper.ClassName = "CSP3DTank6Sym"
```

14. In this Class_Initialize() routine, add the following code to define the inputs and aspects definition for this symbol. Note: there is no outputs section.

```
' Inputs Section
    m_oSymbolHelper.NumInputs = 3
    m_oSymbolHelper.AddInputDef 1, "VesselDiameter", "VesselDiameter", 1
    m_oSymbolHelper.AddInputDef 2, "VesselTantoTan", "VesselTantoTan", 3
    m_oSymbolHelper.AddInputDef 3, "InsulationThickness", "InsulationThickness", 0.06

'Aspects Section
    m_oSymbolHelper.NumAspects = 1
    m_oSymbolHelper.AddAspectDef 1, "SimplePhysical", "SimplePhysical", 1
```

15. Go to CSimplePhysical Class module/Run subroutine and make sure there is no code to get the inputs.

16. Go to the General Declarations section in CSP3DTank6Def module and change the value of the *Constant Module variable* from “*"SP3DTemplateAsm:CSP3DTemplateDef"* to *"SP3DTank6Asm:CSP3DTank6Def"*

17. Go to the top of the CSP3DTank6Def module and declare the following variables:

```
Private Const MODULE = "SP3DTemplateAsm:CSP3DTemplateDef"

Private Const IID_IJDATTRIBUTES = "{B25FD387-CFEB-11D1-850B-080036DE8E03}"
Private Const IID_IJDGEOMETRY = "{A1732CBF-5136-11D1-9770-080036754203}"

Private m_oEquipCADHelper As IJEquipCADHelper
Private m_oEditErrors As IJEditErrors
Private m_avSymbolArrayOfInputs()   As Variant

Private m_oNorth                As IJDVector
Private m_oEast                 As IJDVector
Private m_oElevation            As IJDVector

'VDrum
Private m_VesselDiameter As Double
Private m_VesselTantoTan As Double
Private m_dInsulationThickness As Double
Private m_NoOfPipePorts As Double

Private PI As Double
```

18. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
 m_oEquipCADHelper.ProjectName = "SP3DTank6Asm"
 m_oEquipCADHelper.ClassName = "CSP3DTank6Def"
```

19. In this Class_Initialize() routine, initialize the following variable:

```
   PI = 4 * Atn(1)
   Set m_oEast = New DVector
   m_oEast.x = 1
   m_oEast.y = 0
   m_oEast.z = 0

   Set m_oNorth = New DVector
   m_oNorth.x = 0
   m_oNorth.y = 1
   m_oNorth.z = 0

   Set m_oElevation = New DVector
   m_oElevation.x = 0
   m_oElevation.y = 0
   m_oElevation.z = 1
```

20. Go to the Class_Terminate() routine and use the Set statement to clear the references from all object variables.

```
Private Sub Class_Terminate()

   Set m_oNorth = Nothing
   Set m_oEast = Nothing
   Set m_oElevation = Nothing

   Set m_oEditErrors = Nothing
   Set m_oEquipCADHelper = Nothing
End Sub
```

21. Go to CSP3DTank6Def Class module. Declare the appropriate custom methods to manage the vertical drum, datum shape and the pipe ports as follows:

```
  'Add your code here for the declaration of the Public Custom Methods used to manage new members
  'Add new member(VDrum) to the definition

  Set oMemberDescription = Nothing
  Set oMemberDescription = oMemberDescriptions.AddMember("VDrum", 1, "CMConstructVDrum", _
                                          imsCOOKIE_ID_USS_LIB)
  oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsVDrum"
  oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructVDrum"
  oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalVDrum"
  oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseVDrum"

  Set oPropertyDescriptions = Nothing
  Set oPropertyDescriptions = oMemberDescription
  oPropertyDescriptions.AddProperty "VDrumProperties", 1, IID_IJDATTRIBUTES, "CMEvaluateVDrum", _
                                          imsCOOKIE_ID_USS_LIB
  oPropertyDescriptions.AddProperty "VDrumGeometryProperties", 2, IID_IJDGEOMETRY,_
                                          "CMEvaluateGeometryVDrum", imsCOOKIE_ID_USS_LIB

  'Add new member(DP1) to the definition

  Set oMemberDescription = Nothing
  Set oMemberDescription = oMemberDescriptions.AddMember("DP1", 2, "CMConstructDP1", _
                                          imsCOOKIE_ID_USS_LIB)
  oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsDP1"
  oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructDP1"
  oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalDP1"
  oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseDP1"

  'Add properties for DP1
  Set oPropertyDescriptions = Nothing
  Set oPropertyDescriptions = oMemberDescription
  oPropertyDescriptions.AddProperty "DP1Properties", 1, IID_IJDATTRIBUTES, "CMEvaluateDP1", _
                                         imsCOOKIE_ID_USS_LIB
  oPropertyDescriptions.AddProperty "DP1GeometryProperties", 2, IID_IJDGEOMETRY,_
                                  "CMEvaluateGeometryDP1", imsCOOKIE_ID_USS_LIB

  'Add new member(NozzleN1) to the definition

  Set oMemberDescription = Nothing
  Set oMemberDescription = oMemberDescriptions.AddMember("NozzleN1", 3, "CMConstructNozzleN1", _
                                          imsCOOKIE_ID_USS_LIB)
  oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsNozzleN1"
  oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructNozzleN1"
  oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalNozzleN1"
  oMemberDescription.SetCMCount imsCOOKIE_ID_USS_LIB, "CMCountNozzleN1"
  oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseNozzleN1"

 'Add properties for (NozzleN1)
  Set oPropertyDescriptions = Nothing
  Set oPropertyDescriptions = oMemberDescription
  oPropertyDescriptions.AddProperty "NozzleN1Properties", 1, IID_IJDATTRIBUTES, _
                                  "CMEvaluateNozzleN1", imsCOOKIE_ID_USS_LIB
  oPropertyDescriptions.AddProperty "NozzleN1GeometryProperties", 2, IID_IJDGEOMETRY, _
                                  "CMEvaluateGeometryNozzleN1", imsCOOKIE_ID_USS_LIB
```

22. Go to IJEquipUserAttrMgmt_OnPreLoad function and add the following code. Use the IJEquipAttrDescriptor interface to set the VDrum properties read only.

```
Private Function IJEquipUserAttrMgmt_OnPreLoad(ByVal pIJDAttrs As IJDAttributes, ByVal
CollAllDisplayedValues As Object) As String
   Const METHOD = "IJEquipUserAttrMgmt_OnPreLoad"
   On Error GoTo ErrorHandler

      Dim oMemberDescription As IJDMemberDescription

      Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)
      Dim oAttrCollection As Collection
      Dim oAttributeDescriptor As IJEquipAttrDescriptor
      Dim m As Long

' set dimension and deletable attributes to read only.

      Set oAttrCollection = CollAllDisplayedValues
      Select Case oMemberDescription.Name
      Case "VDrum"
         For m = 1 To oAttrCollection.Count
            Set oAttributeDescriptor = oAttrCollection.Item(m)
            Select Case UCase(oAttributeDescriptor.InterfaceName)
               Case "IJUAVESSELDIAMETER"
                 oAttributeDescriptor.AttrState = adsReadOnly
               Case "IJUAVESSELTANTOTAN"
                 oAttributeDescriptor.AttrState = adsReadOnly
               Case "IJDELETABLEMEMBER"
                 oAttributeDescriptor.AttrState = adsReadOnly
               Case Else
                  '
            End Select
         Next
      Case "DP1"
         For m = 1 To oAttrCollection.Count
            Set oAttributeDescriptor = oAttrCollection.Item(m)
            oAttributeDescriptor.AttrState = adsReadOnly
         Next
      Case Else
         '
      End Select

   Set oAttrCollection = Nothing
   Set oAttributeDescriptor = Nothing
   Set oMemberDescription = Nothing

   IJEquipUserAttrMgmt_OnPreLoad = ""

   Exit Function
ErrorHandler:
   IJEquipUserAttrMgmt_OnPreLoad = "ERROR"
   HandleError MODULE, METHOD
End Function
```

23. Go to the end of CSP3DTank6Def Class module. Add the custom methods to manage the drum, datum shape and the pipe port as follows:

    **Custom Method Construct**:
    This method is in charge of the creation of the CAO member object (VDrum 01-EC). Use EquipCADHelper CreateEquipmentComponent () method to create the member given the equipment component part number. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructVDrum(ByVal pMemberDescription As IJDMemberDescription, _
                        ByVal pResourceManager As IUnknown, _
                        ByRef pObject As Object)
   Const METHOD = "CMConstructVDrum"
   LogCalls METHOD
   On Error GoTo ErrorHandler

   Dim oEquipment As IJEquipment
   Set oEquipment = pMemberDescription.CAO
   GetDimensionsFromSymbolArray oEquipment

   'Create Equipment Component
   Set pObject = m_oEquipCADHelper.CreateEquipmentComponent(pMemberDescription, _
                                                   pResourceManager, "VDrum 01-EC", "VDrum")

   'create name for the member
   SetObjNameRule pObject, "CPEquipmentComponent"

   Set oEquipment = Nothing

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

    **Custom Method Final:**
    There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructVDrum(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMFinalConstructVDrum"
   LogCalls METHOD
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

    **Custom method Inputs:**
    There is no need to add any code for this custom method.

```
Public Sub CMSetInputsVDrum(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMSetInputsVDrum"
   LogCalls METHOD
   On Error GoTo ErrorHandler
   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method Evaluate:**

This method is in charge of setting the attribute values to the member object. Use the MakeMemberDeletable method to set the member deletable.

```
Public Sub CMEvaluateVDrum(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
   Const METHOD = "CMEvaluateVDrum"
   LogCalls METHOD
   On Error GoTo ErrorHandler

   GetDimensionsFromSymbolArray oPropertyDescription.CAO

   Dim oAttribs As IJDAttributes
   Dim oSmartOcc As IJSmartOccurrence

   Set oSmartOcc = oPropertyDescription.Object
   Set oAttribs = oSmartOcc '.ItemObject

   oAttribs.CollectionOfAttributes("IJUAVesselDiameter").Item("VesselDiameter").Value = m_VesselDiameter
   oAttribs.CollectionOfAttributes("IJUAVesselTantoTan").Item("VesselTantoTan").Value = m_VesselTantoTan
   oAttribs.CollectionOfAttributes("IJInsulationThickness").Item("InsulationThickness").Value = m_dInsulationThickness
   oAttribs.CollectionOfAttributes("IJDeletableMember").Item("CanBeDeleted").Value = True

' set member deletable

   Dim oMemberDescription As IJDMemberDescription
   Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(oAttribs)
   m_oEquipCADHelper.MakeMemberDeletable oMemberDescription, oAttribs, True

   Set oAttribs = Nothing
   Set oSmartOcc = Nothing
   Set oMemberDescription = Nothing

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method GeometryEvaluate:**

This method is in charge of setting the transformation matrix to move the member object relative to the equipment. Use the TransformFromECStoGCS function to maintain the VDrum member coordinate system relative to the Equipment coordinate system.

```
Public Sub CMEvaluateGeometryVDrum(ByVal oPropertyDescription As IJDPropertyDescription, pObject As
Object)
   Const METHOD = "CMEvaluateGeometryVDrum"
   On Error GoTo ErrorHandler
   LogCalls METHOD
```

```
'Add the following code to avoid the eqp component to move alone

    Dim oEqpComp As IJEquipmentComponent
    Dim oEqpCompMatrix As IJEquipment
    Dim oEquipment As IJEquipment

    Set oEqpComp = oPropertyDescription.Object
    oEqpComp.GetParent oEquipment
    Set oEqpCompMatrix = oEqpComp

    GetDimensionsFromSymbolArray oEquipment

    Dim otransform As IngrGeom3D.IJDT4x4
    Set otransform = New DT4x4
    Dim iVector As IJDVector
    Set iVector = New DVector

    otransform.LoadIdentity
    iVector.x = 0
    iVector.y = 0
    iVector.z = 0
    otransform.Translate iVector
    oEqpCompMatrix.SetMatrix otransform

    TransformFromECStoGCS oEquipment, oEqpCompMatrix

    Set oEquipment = Nothing
    Set oEqpComp = Nothing
    Set oEqpCompMatrix = Nothing

    Set iVector = Nothing
    Set otransform = Nothing

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

## Custom method Conditional:

This method checks if the member is conditional based on the CanBeDeleted flag. Remember, we added code in the CMEvaluate to make the member deletable.

```
Public Sub CMConditionalVDrum(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As
Boolean)
    Const METHOD = "CMConditionalVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Release:

There is no need to add any code for this custom method

```
Public Sub CMReleaseVDrum(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMReleaseVDrum"
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

### Custom Method Construct:

This method is in charge of the creation of the CAO member object (datum shape). Use EquipCADHelper CreateShape () method to create the member given the shape part number.

```
Public Sub CMConstructDP1(ByVal pMemberDescription As IJDMemberDescription, _
                         ByVal pResourceManager As IUnknown, _
                         ByRef pObject As Object)
   Const METHOD = "CMConstructDP1"
   LogCalls METHOD
   On Error GoTo ErrorHandler
   Dim oDatumShape As IJShape
   Dim oDesignEquipment As IJDesignEquipment

   'Create Datum Shape DP1
   Set oDatumShape = m_oEquipCADHelper.CreateShape(pMemberDescription, pResourceManager,
"DatumShape 001", "DP1")

   If Not oDatumShape Is Nothing Then
      Set pObject = oDatumShape
      oDatumShape.RepresentationId = ReferenceGeometry
      Set oDesignEquipment = pMemberDescription.CAO
      oDesignEquipment.AddShape oDatumShape
      GetDimensionsFromSymbolArray oDesignEquipment
      PositionAndOrientDP1 oDesignEquipment, oDatumShape
   End If

   Set oDesignEquipment = Nothing
   Set oDatumShape = Nothing

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

### Custom Method Final:

There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructDP1(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMFinalConstructDP1"
   LogCalls METHOD
   On Error GoTo ErrorHandler
```

```
    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Inputs:

There is no need to add any code for this custom method.

```
Public Sub CMSetInputsDP1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMSetInputsDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Evaluate:

There is no need to add any code for this custom method.

```
Public Sub CMEvaluateDP1(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
    Const METHOD = "CMEvaluateDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method GeometryEvaluate:

This method is in charge of moving the datum shape relative to the equipment. Use the PositionAndOrientDP1 function to maintain the datum shape position relative to the Equipment.

```
Public Sub CMEvaluateGeometryDP1(ByVal oPropertyDescription As IJDPropertyDescription, pObject As
Object)
    Const METHOD = "CMEvaluateGeometryDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler
    Dim oEquipment As IJEquipment
    Dim oDatumShape As IJShape

    Set oDatumShape = oPropertyDescription.Object

        Set oEquipment = oPropertyDescription.CAO
        GetDimensionsFromSymbolArray oEquipment
        PositionAndOrientDP1 oEquipment, oDatumShape

    Set oDatumShape = Nothing
    Set oEquipment = Nothing
    Exit Sub

ErrorHandler:
    Set oDatumShape = Nothing
    Set oEquipment = Nothing
    HandleError MODULE, METHOD
End Sub
```

### Custom method Conditional:

There is no need to add any code for this custom method.

```
Public Sub CMConditionalDP1(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As Boolean)
    Const METHOD = "CMConditionalDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Release:

There is no need to add any code for this custom method.

```
Public Sub CMReleaseDP1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMReleaseDP1"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Construct:

This method is in charge of the creation of the CAO member object (pipe nozzle). Use CreateNozzleGivenIndex() method to create the pipe nozzle given the nozzle index. The nozzle data are retrieved from the part. Use the CreateOrientationAndSetRelations() function to set the nozzle default position and nozzle default orientation

```
Public Sub CMConstructNozzleN1(ByVal pMemberDescription As IJDMemberDescription, _
                               ByVal pResourceManager As IUnknown, _
                               ByRef pObject As Object)
    Const METHOD = "CMConstructNozzleN1"
    On Error GoTo ErrorHandler

    Dim oOrientation As IJNozzleOrientation
    Dim oNozzle As IJDNozzle

    GetDimensionsFromSymbolArray pMemberDescription.CAO

    'Create Nozzle
    m_oEquipCADHelper.CreateNozzleGivenIndex pMemberDescription, pMemberDescription.index,
pResourceManager, DistribPortType_PIPE, pObject, False
```

```
    Set oNozzle = pObject
    oNozzle.Length = 0.3
    Set oOrientation = m_oEquipCADHelper.CreateOrientationAndSetRelations(Nothing, oNozzle)

    'Set the default values
    oOrientation.PlacementType = Radial
    oOrientation.N1 = 0
    oOrientation.N2 = m_VesselDiameter / 2 + oNozzle.Length
    oOrientation.OR1 = pMemberDescription.index * (2 * PI / m_NoOfPipePorts)

    Set oNozzle = Nothing
    Set oOrientation = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Final:

There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructNozzleN1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMFinalConstructNozzleN1"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Inputs:

This method is used to set the input arguments to the member object. Use IJDMemberDescription to get the appropriate shape (Datum shape) in order to set the relations with the pipe nozzle.

```
Public Sub CMSetInputsNozzleN1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMSetInputsNozzleN1"
    On Error GoTo ErrorHandler

        Dim l As Long
        Dim oShape As IJShape
        Dim oSmartOcc As IJSmartOccurrence
        Dim oMemberobjects As IJDMemberObjects
        Dim oMemberDesc As IJDMemberDescription
        Dim oOrientation As IJNozzleOrientation
        Dim oNozzle As IJDNozzle

        Set oSmartOcc = pMemberDesc.CAO

        Set oMemberobjects = oSmartOcc
        For l = 1 To oMemberobjects.Count
            Set oMemberDesc = oMemberobjects.MemberDescriptions.Item(l)
            If oMemberDesc.Name = "DP1" Then
                Set oShape = oMemberobjects.Item(l)
            Exit For
          End If
        Next l

    ' Set the nozzle orientation
        Set oOrientation = m_oEquipCADHelper.CreateOrientationAndSetRelations(oShape,
pMemberDesc.Object)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Evaluate:**
There is no need to add any code for this custom method.

```
Public Sub CMEvaluateNozzleN1(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
    Const METHOD = "CMEvaluateNozzleN1"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method GeometryEvaluate:**
This method is in charge of reposition the pipe ports relative to the equipment. Use the Relation helper service to access the nozzle orientation attributes of the pipe port.

```
Public Sub CMEvaluateGeometryNozzleN1(ByVal oPropertyDescription As IJDPropertyDescription, pObject
As Object)
    Const METHOD = "CMEvaluateGeometryNozzleN1"
    On Error GoTo ErrorHandler

        Dim oOrientation As IJNozzleOrientation
        Dim oNozzle As IJDNozzle
        Dim oObject As IJDObject
        Dim oAssocRelation As IJDAssocRelation
        Dim oTargetObjCol As IJDTargetObjectCol

        Set oNozzle = oPropertyDescription.Object
        Set oObject = oNozzle
        Set oAssocRelation = oNozzle
        Set oTargetObjCol = oAssocRelation.CollectionRelations("IJPort", "NozzleOrientation")
        Set oOrientation = oTargetObjCol.Item(1)

        GetDimensionsFromSymbolArray oPropertyDescription.CAO

        oNozzle.Length = 0.3
        oOrientation.PlacementType = Radial
        oOrientation.N1 = 0
        oOrientation.N2 = m_VesselDiameter / 2 + oNozzle.Length
        oOrientation.OR1 = oPropertyDescription.index * (2 * PI / m_NoOfPipePorts)

        Set oOrientation = Nothing
        Set oNozzle = Nothing
        Set oObject = Nothing
        Set oAssocRelation = Nothing
        Set oTargetObjCol = Nothing


    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Count:

This custom method read the global variable m_VesselDiameter to determine total number of members to be created.

```
Public Sub CMCountNozzleN1(ByVal pMemberDesc As IJDMemberDescription, ByRef Count As Long)
    Const METHOD = "CMCountNozzleN1"
    On Error GoTo ErrorHandler

    GetDimensionsFromSymbolArray pMemberDesc.CAO

    If m_VesselDiameter < 3 Then
        m_NoOfPipePorts = 3
    Else
        m_NoOfPipePorts = 5
    End If

    Count = m_NoOfPipePorts

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Conditional:

There is no need to add any code for this custom method.

```
Public Sub CMConditionalNozzleN1(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As
Boolean)
    Const METHOD = "CMConditionalNozzleN1"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Release:

There is no need to add any code for this custom method.

```
Public Sub CMReleaseNozzleN1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMReleaseNozzleN1"
    On Error GoTo ErrorHandler
    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

24. Add the following subroutine to convert the array of inputs in a set of global variables.

```
Private Sub GetDimensionsFromSymbolArray(SmartOccurrence As IJSmartOccurrence)
    Const METHOD = "GetDimensionsFromSymbolArray"
    On Error GoTo ErrorHandler

    m_avSymbolArrayOfInputs = m_oEquipCADHelper.GetSymbolArrayOfInputs(SmartOccurrence)

    'Inputs,  from equipment symbol code
    'Set m_oPartFclt = m_avSymbolArrayOfInputs(1)
    m_VesselDiameter = m_avSymbolArrayOfInputs(2)
    m_VesselTantoTan = m_avSymbolArrayOfInputs(3)
    m_dInsulationThickness = m_avSymbolArrayOfInputs(4)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

25. Add the following subroutine to position and orient the shape with respect to the equipment.

```
Private Sub PositionAndOrientDP1(Equipment As IJEquipment, Shape As IJShape)

    Dim oPosition As IJDPosition
    Set oPosition = New DPosition

    oPosition.Set 0, 0, m_VesselTantoTan / 2
    m_oEquipCADHelper.PositionAndOrientShape Equipment, Shape, oPosition, m_oElevation, m_oNorth
    Set oPosition = Nothing

End Sub
```

26. Add the following subroutine to position and orient the member with respect to the equipment.

```
Private Sub TransformFromECStoGCS(Equipment As IJEquipment, Object As Object)
    Const METHOD = "TransformFromECStoGCS"
    LogCalls METHOD
    On Error GoTo ErrorHandler
    Dim oEqpMatrix As IJDT4x4
    Dim oShapeMatrix As IJDT4x4
    Dim otransform As IJDGeometry
    Dim oShape As IJShape

    If Not Object Is Nothing Then
        If TypeOf Object Is IJDGeometry Then
            Equipment.GetMatrix oEqpMatrix
            Set otransform = Object
            otransform.DTransform oEqpMatrix
            Set otransform = Nothing
            Set oEqpMatrix = Nothing
        End If
    End If

    Set otransform = Nothing
    Set oEqpMatrix = Nothing
    Set oShape = Nothing
    Set oShapeMatrix = Nothing

    Exit Sub
ErrorHandler:
    Set otransform = Nothing
    Set oEqpMatrix = Nothing
    Set oShape = Nothing
    Set oShapeMatrix = Nothing
    HandleError MODULE, METHOD
End Sub
```

27. Add the following subroutine to set the naming relation and generate a name based on the default naming rule.

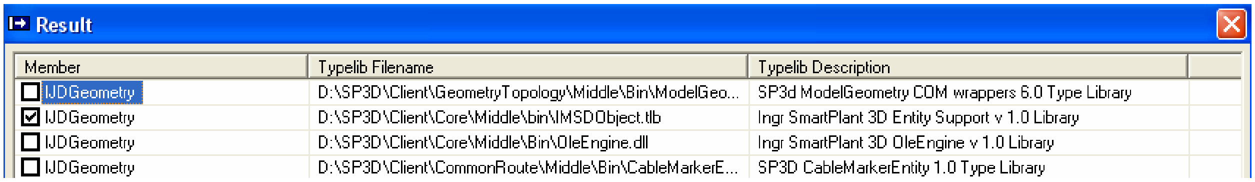
```
Public Sub SetObjNameRule(ByRef obj As Object, ByRef CLASSNAME As String)
' Apply the namerule using the IJDNamingRulesHelper helper interface
Const METHOD = "SetNameRule"
On Error GoTo ErrorHandler

    Dim NameRule As String
    Dim NamingRules As IJElements

    Dim oNameRuleHlpr As GSCADNameRuleSemantics.IJDNamingRulesHelper

'Returns a collection of the naming rules available in the catalog database
'for the given object
    Set oNameRuleHlpr = New GSCADNameRuleHlpr.NamingRulesHelper
    Call oNameRuleHlpr.GetEntityNamingRulesGivenName(CLASSNAME, NamingRules)
'get the first namerule from the collection
    Dim oNameRuleHolder As GSCADGenericNamingRulesFacelets.IJDNameRuleHolder
    Set oNameRuleHolder = NamingRules.Item(1)
'Create relations "NamedEntity" and "EntityNamingRule" and obj
    Dim oNameRuleAE As GSCADGenNameRuleAE.IJNameRuleAE
    Call oNameRuleHlpr.AddNamingRelations(obj, oNameRuleHolder, oNameRuleAE)

    GoTo CleanObjects
ErrorHandler:
    HandleError MODULE, METHOD

CleanObjects:
    Set oNameRuleHlpr = Nothing
    Set oNameRuleHolder = Nothing
    Set oNameRuleAE = Nothing
    Set NamingRules = Nothing
End Sub
```

28. Add the following subroutine to log any error.

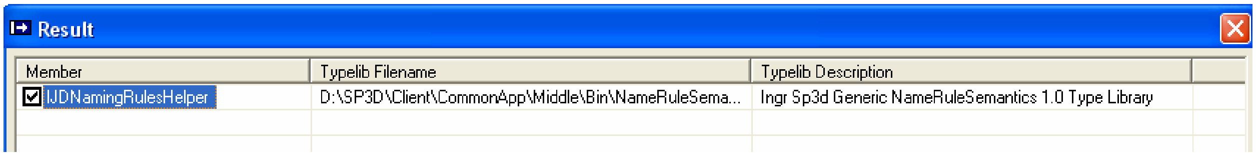
```
Private Sub LogCalls(sMethod As String)

   If Not m_oEditErrors Is Nothing Then
      m_oEditErrors.Add 5000, m_oEquipCADHelper.ProjectName & "." &
m_oEquipCADHelper.CLASSNAME, "Entering " & sMethod
   End If

End Sub
```

29. Compile the Visual Basic project and save the dll as SP3DTank6Asm.dll in the c:\Train\lab6

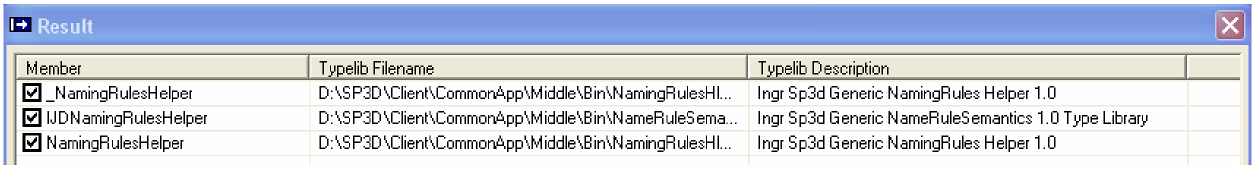
Note: Use the SP3D Reference Tool to attach the missing reference libraries.



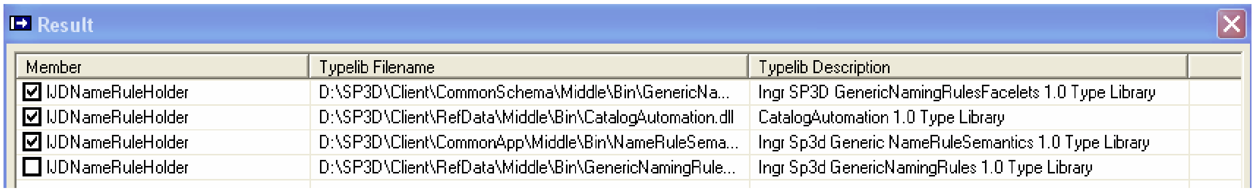
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☐ IJDGeometry | D:\SP3D\Client\GeometryTopology\Middle\Bin\ModelGeo... | SP3d ModelGeometry COM wrappers 6.0 Type Library |
| ☑ IJDGeometry | D:\SP3D\Client\Core\Middle\bin\IMSDObject.tlb | Ingr SmartPlant 3D Entity Support v 1.0 Library |
| ☐ IJDGeometry | D:\SP3D\Client\Core\Middle\Bin\OleEngine.dll | Ingr SmartPlant 3D OleEngine v 1.0 Library |
| ☐ IJDGeometry | D:\SP3D\Client\CommonRoute\Middle\Bin\CableMarkerE... | SP3D CableMarkerEntity 1.0 Type Library |



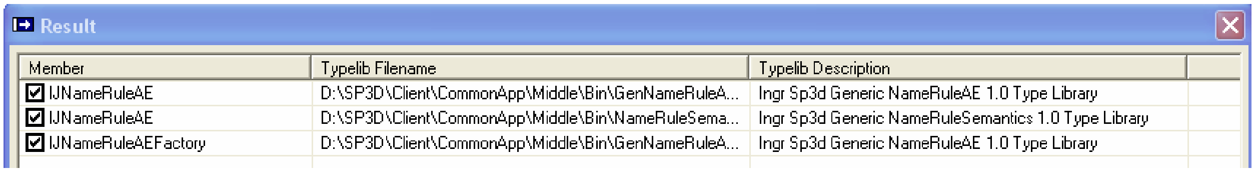
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDNamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |



| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ _NamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NamingRulesHl... | Ingr Sp3d Generic NamingRules Helper 1.0 |
| ☑ IJDNamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☑ NamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NamingRulesHl... | Ingr Sp3d Generic NamingRules Helper 1.0 |



| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDNameRuleHolder | D:\SP3D\Client\CommonSchema\Middle\Bin\GenericNa... | Ingr SP3D GenericNamingRulesFacelets 1.0 Type Library |
| ☑ IJDNameRuleHolder | D:\SP3D\Client\RefData\Middle\Bin\CatalogAutomation.dll | CatalogAutomation 1.0 Type Library |
| ☑ IJDNameRuleHolder | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☐ IJDNameRuleHolder | D:\SP3D\Client\RefData\Middle\Bin\GenericNamingRule... | Ingr Sp3d GenericNamingRules 1.0 Type Library |

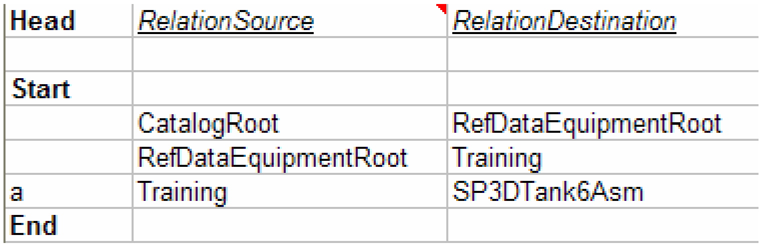


| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJNameRuleAE | D:\SP3D\Client\CommonApp\Middle\Bin\GenNameRuleA... | Ingr Sp3d Generic NameRuleAE 1.0 Type Library |
| ☑ IJNameRuleAE | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☑ IJNameRuleAEFactory | D:\SP3D\Client\CommonApp\Middle\Bin\GenNameRuleA... | Ingr Sp3d Generic NameRuleAE 1.0 Type Library |

One of the most important steps in Visual Basic programming is to preserve the binary compatibility of your program. Save the final version of your dll file to be binary compatibility in order to preserve the CLSID.

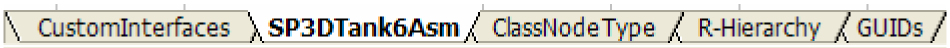
30. Save the Visual Basic SP3DTank6Asm project.

31. Open the SP3DTemplate.xls workbook. Go the R-Hierarchy sheet and add the following row.



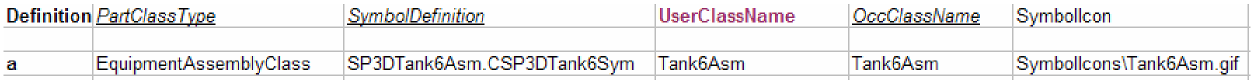
| Head | RelationSource | RelationDestination |
|---|---|---|
| | | |
| Start | | |
| | CatalogRoot | RefDataEquipmentRoot |
| | RefDataEquipmentRoot | Training |
| a | Training | SP3DTank6Asm |
| End | | |

32. Go to the SP3DTemplateAsm sheet and rename it as SP3DTank5Asm.



33. Go to the Class definition section and add/edit as follows:

In the Definition Section rows:

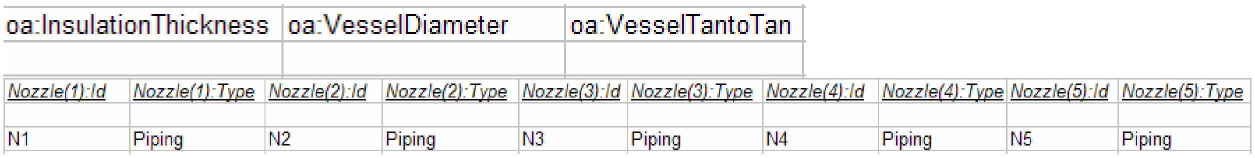


| Definition | PartClassType | SymbolDefinition | UserClassName | OccClassName | SymbolIcon |
|---|---|---|---|---|---|
| | | | | | |
| a | EquipmentAssemblyClass | SP3DTank6Asm.CSP3DTank6Sym | Tank6Asm | Tank6Asm | SymbolIcons\Tank6Asm.gif |

Note:

- Creating the bmp or gif file is optional. You can use Microsoft Paint to create the file and save it under your \\machine\symbols\SymbolIcons

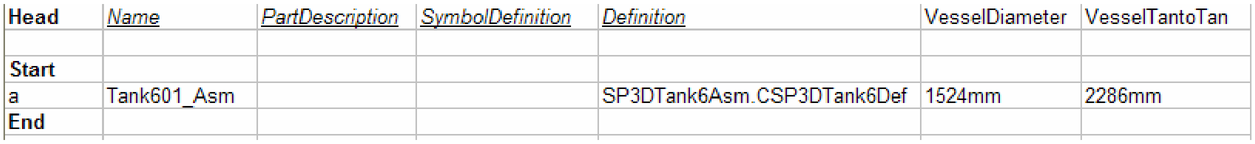
Occurrence attributes:



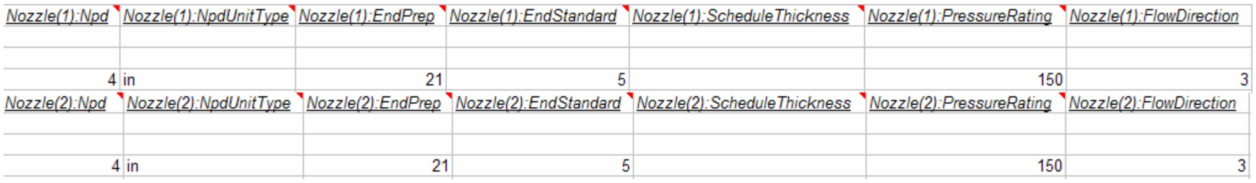
| oa:InsulationThickness | oa:VesselDiameter | oa:VesselTantoTan |
|---|---|---|
| | | |

| Nozzle(1):Id | Nozzle(1):Type | Nozzle(2):Id | Nozzle(2):Type | Nozzle(3):Id | Nozzle(3):Type | Nozzle(4):Id | Nozzle(4):Type | Nozzle(5):Id | Nozzle(5):Type |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| N1 | Piping | N2 | Piping | N3 | Piping | N4 | Piping | N5 | Piping |

In the Part Section rows:

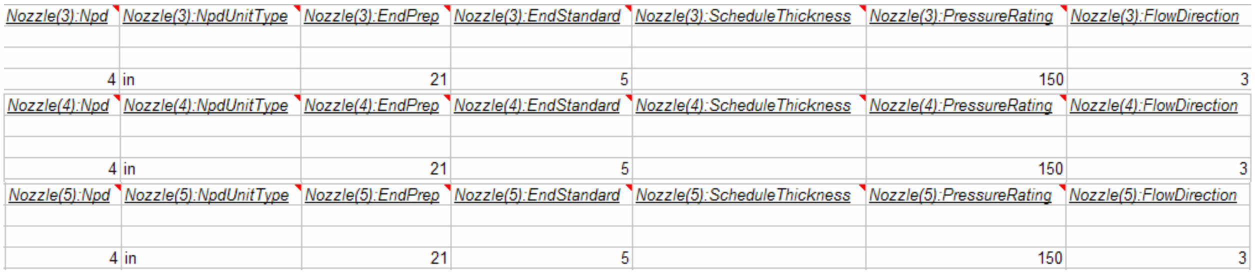


| Head | Name | PartDescription | SymbolDefinition | Definition | VesselDiameter | VesselTantoTan |
|---|---|---|---|---|---|---|
| | | | | | | |
| Start | | | | | | |
| a | Tank601_Asm | | | SP3DTank6Asm.CSP3DTank6Def | 1524mm | 2286mm |
| End | | | | | | |



| Nozzle(1):Npd | Nozzle(1):NpdUnitType | Nozzle(1):EndPrep | Nozzle(1):EndStandard | Nozzle(1):ScheduleThickness | Nozzle(1):PressureRating | Nozzle(1):FlowDirection |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| 4 | in | 21 | 5 | | 150 | 3 |

| Nozzle(2):Npd | Nozzle(2):NpdUnitType | Nozzle(2):EndPrep | Nozzle(2):EndStandard | Nozzle(2):ScheduleThickness | Nozzle(2):PressureRating | Nozzle(2):FlowDirection |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| 4 | in | 21 | 5 | | 150 | 3 |



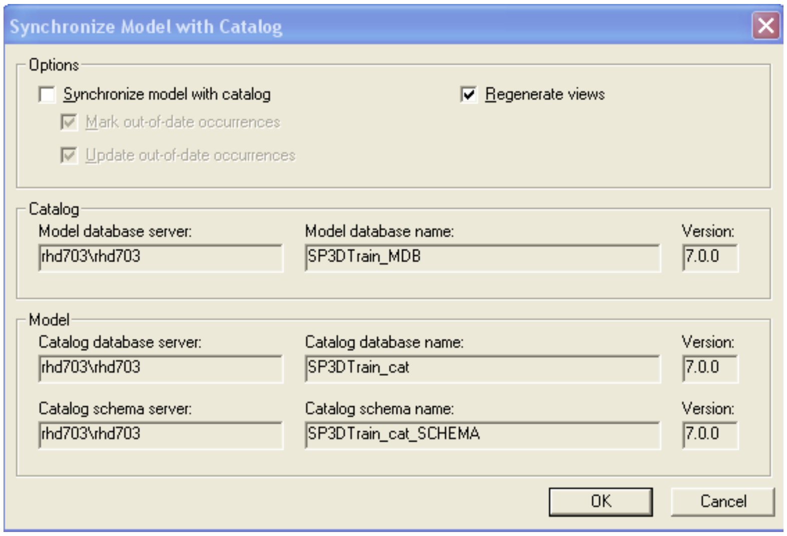
| Nozzle(3):Npd | Nozzle(3):NpdUnitType | Nozzle(3):EndPrep | Nozzle(3):EndStandard | Nozzle(3):ScheduleThickness | Nozzle(3):PressureRating | Nozzle(3):FlowDirection |
|---|---|---|---|---|---|---|
| 4 | in | 21 | 5 | | 150 | 3 |

| Nozzle(4):Npd | Nozzle(4):NpdUnitType | Nozzle(4):EndPrep | Nozzle(4):EndStandard | Nozzle(4):ScheduleThickness | Nozzle(4):PressureRating | Nozzle(4):FlowDirection |
|---|---|---|---|---|---|---|
| 4 | in | 21 | 5 | | 150 | 3 |

| Nozzle(5):Npd | Nozzle(5):NpdUnitType | Nozzle(5):EndPrep | Nozzle(5):EndStandard | Nozzle(5):ScheduleThickness | Nozzle(5):PressureRating | Nozzle(5):FlowDirection |
|---|---|---|---|---|---|---|
| 4 | in | 21 | 5 | | 150 | 3 |

34. Save the Excel workbook as SP3DTank6Asm.xls in the c:\Train\lab6.
35. Optional step: Create the Tank6Asm.gif file and place it under \\<MachineName>\Symbols\SymbolIcons
36. Load the information into the catalog using the Add/Modify/Delete Mode. Once the bulkload process is completed, review the log file.
37. Run the Project Management Task. Select the Model in the hierarchy.
38. Select Tools -> Synchronize Model with the Catalog.
39. Uncheck the Synchronize Model with the Catalog option.

***Note**: You just need to update the views in the model.*



Synchronize Model with Catalog

Options

☐ Synchronize model with catalog

☑ Mark out-of-date occurrences

☑ Update out-of-date occurrences

☑ Regenerate views

Catalog

Model database server: rhd703\rhd703

Model database name: SP3DTrain_MDB

Version: 7.0.0

Model

Catalog database server: rhd703\rhd703

Catalog database name: SP3DTrain_cat

Version: 7.0.0

Catalog schema server: rhd703\rhd703

Catalog schema name: SP3DTrain_cat_SCHEMA

Version: 7.0.0

40. Hit “OK” Button.
41. Once the process is completed, Right click the training plant icon and select “Regenerate the Reports database” option to re-create the views in the report database.
42. Go to the Equipment Task and place the SP3DTank6Asm

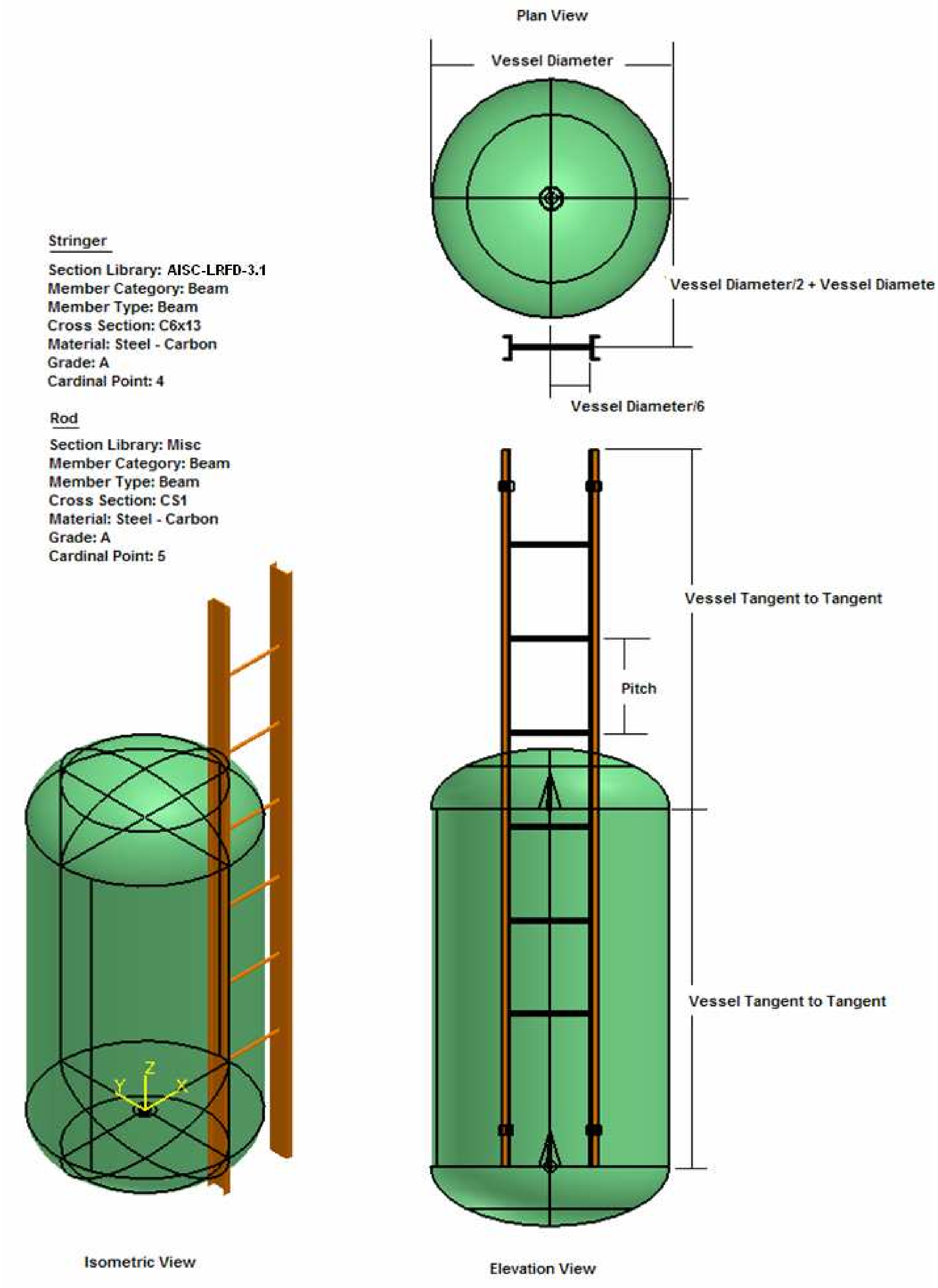
# Lab 8: Structure objects as members of the Equipment (Optional)

## Objectives

After completing this lab, you will be able to:

- Create a catalog equipment symbol made of equipment components and structure objects
- Learn to use the Symbol Helper service to create the symbol definition
- Use the Equipment CAD Helper to define the Custom Assembly Definition
- Learn to use the IJDAggregatorDescription , IJDMemberDescriptions, and IJDPropertyDescriptions to define the behaviors of the custom assembly occurrence (CAO)
- Use the IJEquipUserAttrMgmt Interface to show the equipment component attributes as read only in the property page.
- Use IJDNamingRulesHelper interface to create the naming relations between a naming rule and the object
- Use IJDAttributes interface to get a collection of attributes property
- Use IJDAttribute to get an object's attribute
- Use SPSMemberFactory to create the structure member systems
- Use GetStructureCrossSectionDefinition service to get the cross section
- Use the CMCount to create variable members

In this lab, you will create an equipment symbol as shown below. You start by using the SP3DEqpTemplateAsm template provided by the instructor to create the symbol. This symbol consists of one equipment component and structure members to define the symbol’s output. Use the Equipment Custom Assembly Definition (CAD) Helper to create the members.



Plan View
Vessel Diameter
Vessel Diameter/2 + Vessel Diamete
Vessel Diameter/6
Vessel Tangent to Tangent
Pitch
Vessel Tangent to Tangent
Z
Y
X
Isometric View
Elevation View

Stringer

Section Library: AISC-LRFD-3.1
Member Category: Beam
Member Type: Beam
Cross Section: C6x13
Material: Steel - Carbon
Grade: A
Cardinal Point: 4

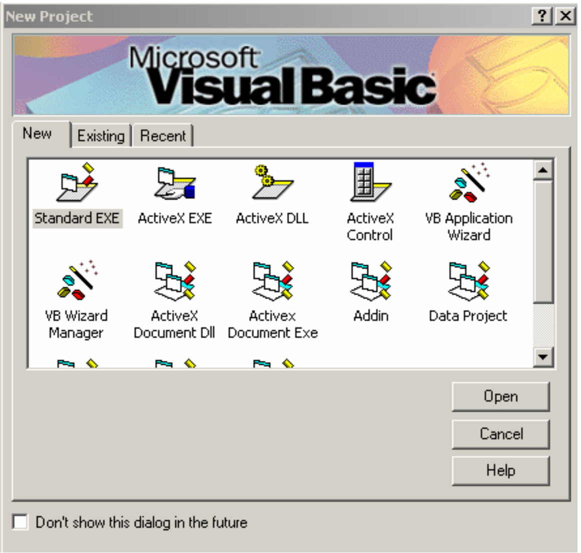
Rod

Section Library: Misc
Member Category: Beam
Member Type: Beam
Cross Section: CS1
Material: Steel - Carbon
Grade: A
Cardinal Point: 5

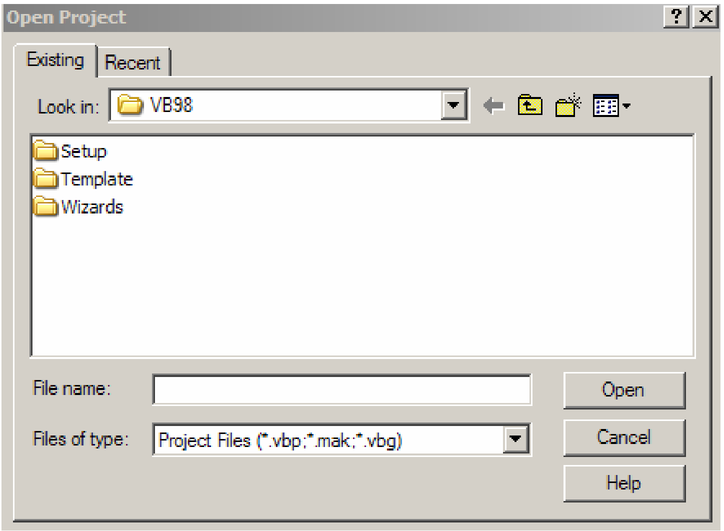
1. Create a directory called lab4 as follows:

   *c:\train\lab7*

2. Run Microsoft Visual Basic 6.0
3. Close the Microsoft New Project dialog box.



4. Select *File* -> *Open Project* option to open the Open Project Dialog box

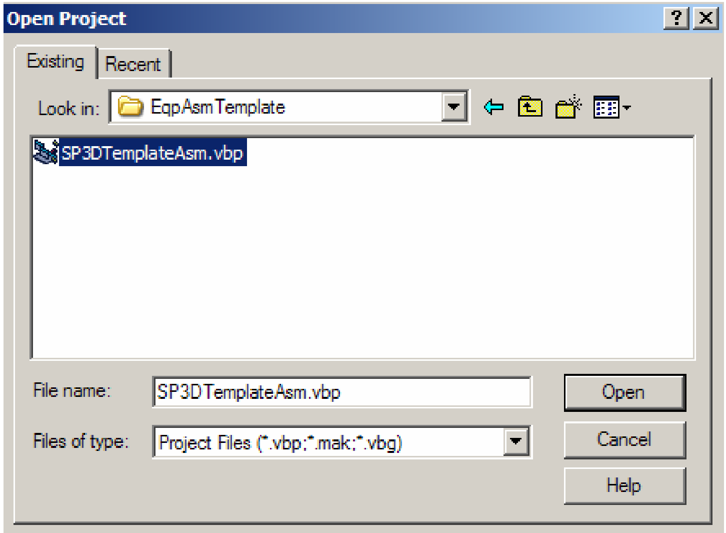


Look in: VB98

File name:

Files of type: Project Files (*.vbp;*.mak;*.vbg)

5. Navigate to c:\train\EqpAsmTemplate and open the SP3DTemplateAsm Template project



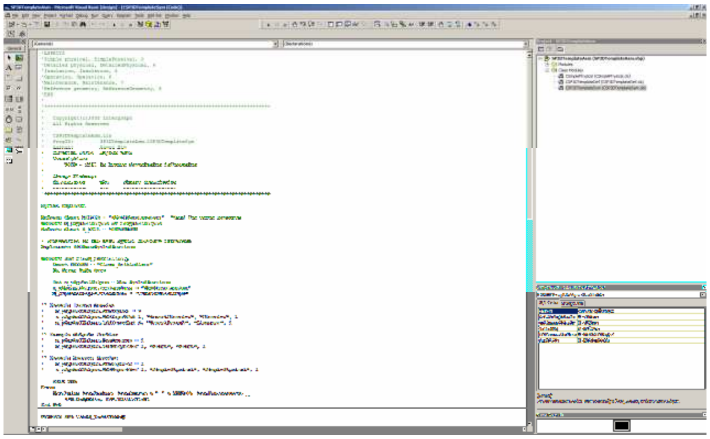
Look in: EqpAsmTemplate

SP3DTemplateAsm.vbp

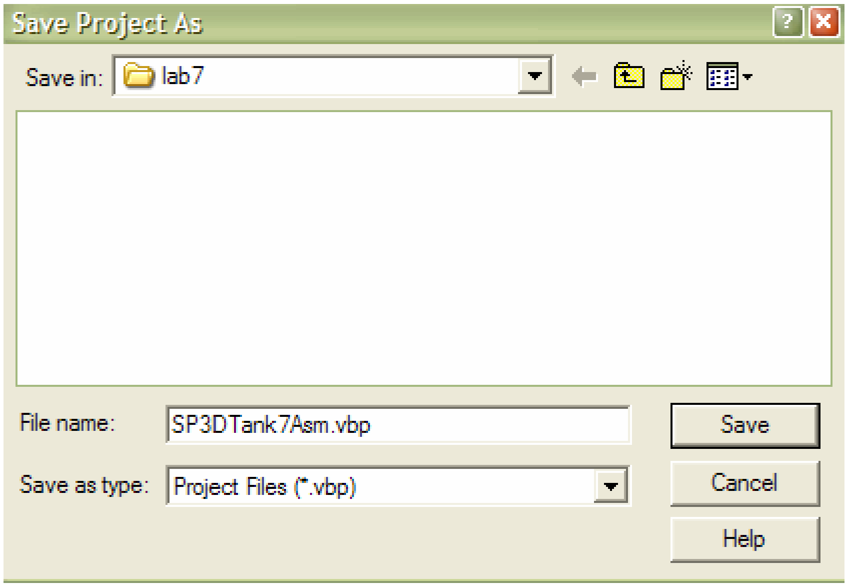
File name: SP3DTemplateAsm.vbp

Files of type: Project Files (*.vbp;*.mak;*.vbg)

6. Setup the Visual Basic Development Environment as shown below:



7. Go to the Visual Basic Explorer Window and select the Project node. Select *File -> Save Project As* option to save the project as SP3DTank7Asm.vbp under the lab7 directory

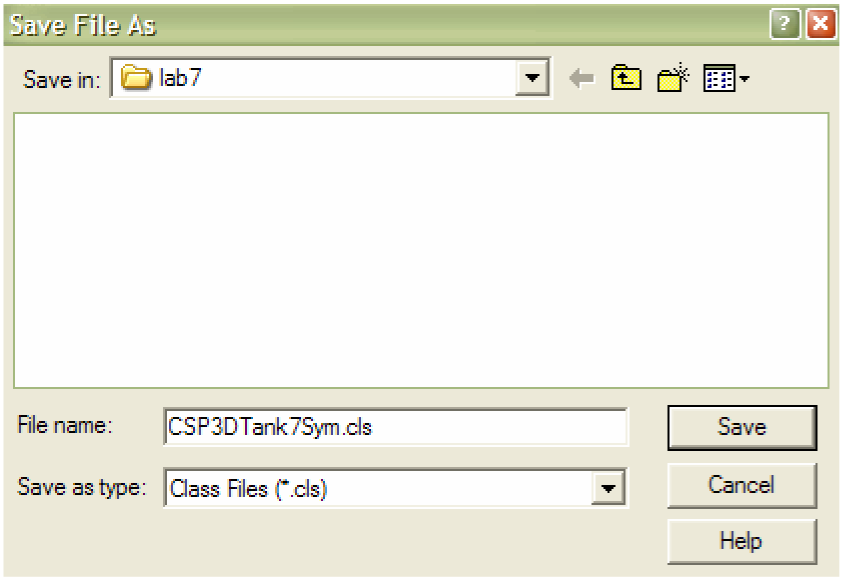


Save in: lab7

File name: SP3DTank7Asm.vbp

Save as type: Project Files (*.vbp)

8. Go to the Visual Basic Explorer Window and select the CSP3DTemplateSym class node. Select *File -> Save CSP3DTemplateSym.cls As* option to save the class module as CSP3DTank7Sym.cls under lab7 directory

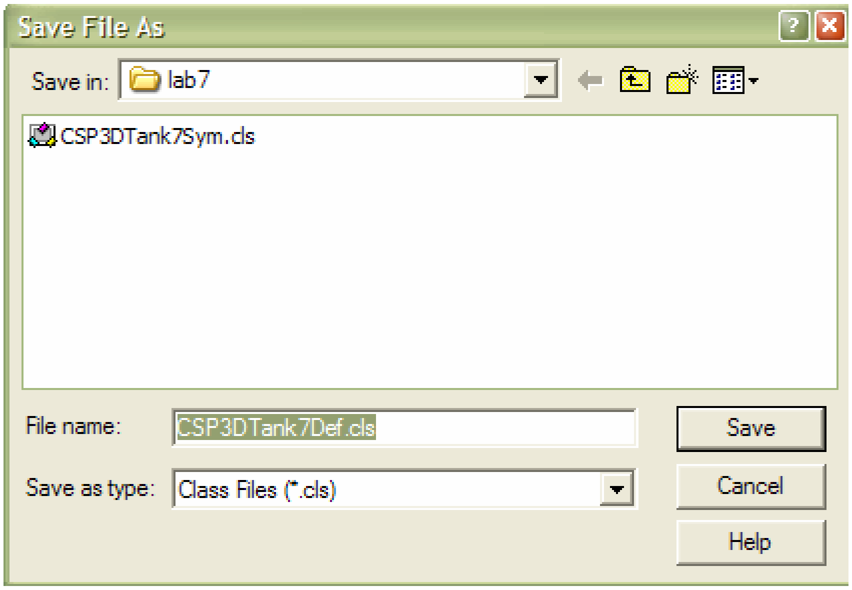


Save in: lab7

File name: CSP3DTank7Sym.cls

Save as type: Class Files (*.cls)

9. Go to the Visual Basic Explorer Window and select the CSP3DTemplateDef class node. Select *File -> Save CSP3DTemplateDef.cls As* option to save the class module as CSP3DTank7Def.cls under lab7 directory



Save in: lab7

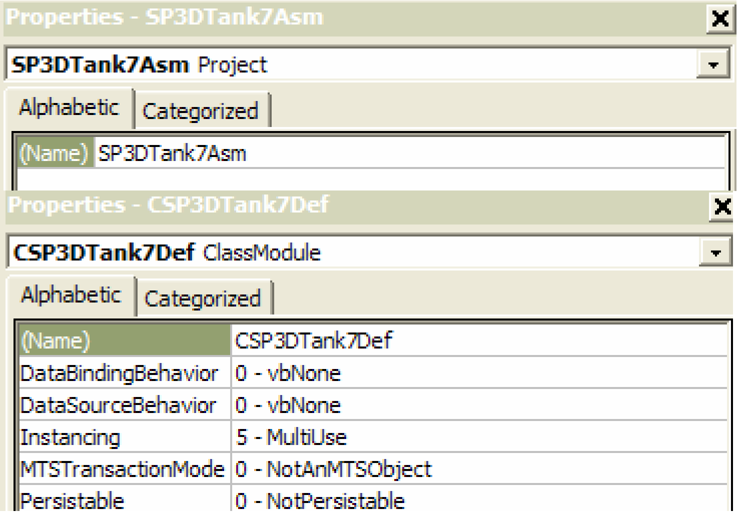
CSP3DTank7Sym.cls

File name: CSP3DTank7Def.cls

Save as type: Class Files (*.cls)

10. Go to the Visual Basic Explorer Window and select the CSimplePhysical class file in the tree. Select *File -> Save CSimplePhysical.cls As* option to save the class module as CSimplePhysical.cls under lab7 directory. Repeat the procedure for the two bas modules.

11. Go to the Properties Window and change the name of the Project and both Class Modules as follows:

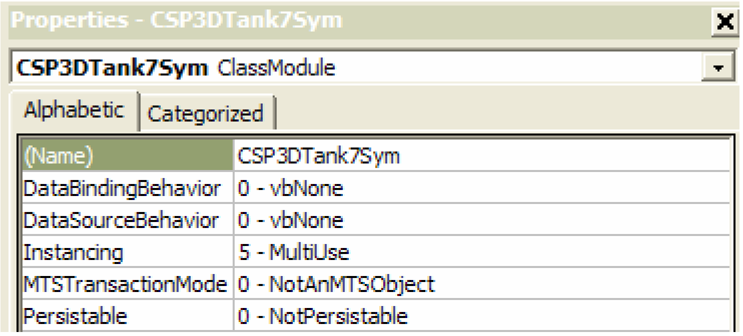


**SP3DTank7Asm** Project

| | |
|---|---|
| (Name) | SP3DTank7Asm |

**CSP3DTank7Def** ClassModule

| | |
|---|---|
| (Name) | CSP3DTank7Def |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |



Properties - CSP3DTank7Sym

**CSP3DTank7Sym** ClassModule

| | |
|---|---|
| (Name) | CSP3DTank7Sym |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |

12. Go to the General Declarations section in CSP3DTank7Sym module and change the value of the *Constant Module variable* from *“CSP3DTemplateAsm:”* to *“CSP3DTank7Asm:”*

```
Private Const MODULE = "CSP3DTank7Asm:"  'Used for error messages
```

13. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
  Set m_oSymbolHelper = New SymbolServices
  m_oSymbolHelper.ProjectName = "SP3DTank7Asm"
  m_oSymbolHelper.ClassName = "CSP3DTank7Sym"
```

14. In this Class_Initialize() routine, add the following code to define the inputs and aspects definition for this symbol. Note: there is no outputs section.

```
' Inputs Section
  m_oSymbolHelper.NumInputs = 4
  m_oSymbolHelper.AddInputDef 1, "VesselDiameter", "VesselDiameter", 1
  m_oSymbolHelper.AddInputDef 2, "VesselTantoTan", "VesselTantoTan", 3
  m_oSymbolHelper.AddInputDef 3, "InsulationThickness", "InsulationThickness", 0.06

  m_oSymbolHelper.AddInputDef 4, "Pitch", "Pitch", 0.1

' Aspects Section
  m_oSymbolHelper.NumAspects = 1
  m_oSymbolHelper.AddAspectDef 1, "SimplePhysical", "SimplePhysical", 1
```

15. Go to CSimplePhysical Class module/Run subroutine and make sure there is no code to get the inputs.

16. Go to the General Declarations section in CSP3DTank7Def module and change the value of the *Constant Module variable* from “*"SP3DTemplateAsm:CSP3DTemplateDef"* to *"SP3DTank7Asm:CSP3DTank7Def"*

17. Go to the top of the CSP3DTank7Def module and declare the following variables

```
Private Const MODULE = "SP3DTank7Asm:CSP3DTank7Def"

Private Const IID_IJDATTRIBUTES = "{B25FD387-CFEB-11D1-850B-080036DE8E03}"
Private Const IID_IJDGEOMETRY = "{A1732CBF-5136-11D1-9770-080036754203}"
Private m_oEquipCADHelper As IJEquipCADHelper
Private m_oEditErrors As IJEditErrors

Private m_avSymbolArrayOfInputs()   As Variant

'VDrum
Private m_VesselDiameter As Double
Private m_VesselTantoTan As Double
Private m_dInsulationThickness As Double
Private m_Pitch As Double
```

18. Declare the CLSID for the Assemblymembers1Relationship

```
Private Const AssemblyMembers1RelationshipCLSID = "{45E4020F-F8D8-47A1-9B00-C9570C1E0B17}"
```

19. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
 m_oEquipCADHelper.ProjectName = "SP3DTank7Asm"
 m_oEquipCADHelper.ClassName = "CSP3DTank7Def"
```

20. Go to CSP3DTank7Def Class module. Declare the appropriate custom methods to manage the drum and the structure objects as follows:

```
'Add your code here for the declaration of the Public Custom Methods used to manage new members
'Add new member(VDrum) to the definition

   Set oMemberDescription = Nothing
   Set oMemberDescription = oMemberDescriptions.AddMember("VDrum", 1, "CMConstructVDrum",
imsCOOKIE_ID_USS_LIB)
   oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsVDrum"
   oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructVDrum"
   oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalVDrum"
   oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseVDrum"

   Set oPropertyDescriptions = Nothing
   Set oPropertyDescriptions = oMemberDescription
   oPropertyDescriptions.AddProperty "VDrumProperties", 1, IID_IJDATTRIBUTES,
"CMEvaluateVDrum", imsCOOKIE_ID_USS_LIB
   oPropertyDescriptions.AddProperty "VDrumGeometryProperties", 2, IID_IJDGEOMETRY,
"CMEvaluateGeometryVDrum", imsCOOKIE_ID_USS_LIB

   'Add new member(SPSSTR1MemSys) to the definition

   Set oMemberDescription = Nothing
   Set oMemberDescription = oMemberDescriptions.AddMember("SPSSTRMemSys1", 2,
"CMConstructSPSSTR1", imsCOOKIE_ID_USS_LIB)
   oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputSPSSTR1"
   oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructSPSSTR1"
   oMemberDescription.RelationshipClsid = AssemblyMembers1RelationshipCLSID

   Set oPropertyDescriptions = Nothing
   Set oPropertyDescriptions = oMemberDescription
```

```
    oPropertyDescriptions.AddProperty "SPSSTR1Properties", 1, IID_IJDATTRIBUTES,
"CMEvaluateSPSSTR1", imsCOOKIE_ID_USS_LIB
    oPropertyDescriptions.AddProperty "SPSSTR1ModifiedbyMemberSys", 2, IID_IJDGEOMETRY,
"CMEvaluateGeometrySPSSTR1", imsCOOKIE_ID_USS_LIB


    'Add new member(SPSSTR2MemSys) to the definition

    Set oMemberDescription = Nothing
    Set oMemberDescription = oMemberDescriptions.AddMember("SPSSTRMemSys2", 3,
"CMConstructSPSSTR2", imsCOOKIE_ID_USS_LIB)
    oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputSPSSTR2"
    oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructSPSSTR2"
    oMemberDescription.RelationshipClsid = AssemblyMembers1RelationshipCLSID

    Set oPropertyDescriptions = Nothing
    Set oPropertyDescriptions = oMemberDescription
    oPropertyDescriptions.AddProperty "SPSSTR2Properties", 1, IID_IJDATTRIBUTES,
"CMEvaluateSPSSTR2", imsCOOKIE_ID_USS_LIB
    oPropertyDescriptions.AddProperty "SPSSTR2ModifiedbyMemberSys", 2, IID_IJDGEOMETRY,
"CMEvaluateGeometrySPSSTR2", imsCOOKIE_ID_USS_LIB

    'Add new member(Rod) to the definition

    Set oMemberDescription = Nothing
    Set oMemberDescription = oMemberDescriptions.AddMember("SPSRod", 4, "CMConstructSPSRod",
imsCOOKIE_ID_USS_LIB)
    oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputSPSRod"
    oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructSPSRod"
    oMemberDescription.SetCMCount imsCOOKIE_ID_USS_LIB, "CMCountSPSRod"
    oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalRod"
    oMemberDescription.RelationshipClsid = AssemblyMembers1RelationshipCLSID

    Set oPropertyDescriptions = Nothing
    Set oPropertyDescriptions = oMemberDescription
    oPropertyDescriptions.AddProperty "SPSSTRRodProperties", 1, IID_IJDATTRIBUTES,
"CMEvaluateSPSRod", imsCOOKIE_ID_USS_LIB
    oPropertyDescriptions.AddProperty "SPSSTRRodModified", 2, IID_IJDGEOMETRY,
"CMEvaluateGeometrySPSRod", imsCOOKIE_ID_USS_LIB
```

21. Go to IJEquipUserAttrMgmt_OnPreLoad function and add the following code. Use the IJEquipAttrDescriptor interface to set the equipment component properties read only.

```
Private Function IJEquipUserAttrMgmt_OnPreLoad(ByVal pIJDAttrs As IJDAttributes, ByVal
CollAllDisplayedValues As Object) As String
    Const METHOD = "IJEquipUserAttrMgmt_OnPreLoad"
    On Error GoTo ErrorHandler

        Dim oMemberDescription As IJDMemberDescription

        Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)
        Dim oAttrCollection As Collection
        Dim oAttributeDescriptor As IJEquipAttrDescriptor
        Dim m As Long
```

```
' set dimension and deletable attributes to read only.

        Set oAttrCollection = CollAllDisplayedValues
        Select Case oMemberDescription.Name
        Case "VDrum"
            For m = 1 To oAttrCollection.Count
                Set oAttributeDescriptor = oAttrCollection.Item(m)
                Select Case UCase(oAttributeDescriptor.InterfaceName)
                    Case "IJUAVESSELDIAMETER"
                       oAttributeDescriptor.AttrState = adsReadOnly
                    Case "IJUAVESSELTANTOTAN"
                       oAttributeDescriptor.AttrState = adsReadOnly
                    Case "IJDELETABLEMEMBER"
                       oAttributeDescriptor.AttrState = adsReadOnly
                    Case Else
                        '
                End Select
            Next
        Case Else
            '
        End Select

    Set oAttrCollection = Nothing
    Set oAttributeDescriptor = Nothing
    Set oMemberDescription = Nothing

    IJEquipUserAttrMgmt_OnPreLoad = ""

    Exit Function
ErrorHandler:
    IJEquipUserAttrMgmt_OnPreLoad = "ERROR"
    HandleError MODULE, METHOD
End Function
```

22. Go to the end of CSP3DTank7Def Class module. Add the custom methods to manage the drum and the structure objects as follows:

    **Custom Method Construct**:
    This method is in charge of the creation of the CAO member object (VDrum 01-EC). Use EquipCADHelper CreateEquipmentComponent () method to create the member given the equipment component part number. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructVDrum(ByVal pMemberDescription As IJDMemberDescription, _
                            ByVal pResourceManager As IUnknown, _
                            ByRef pObject As Object)
    Const METHOD = "CMConstructVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Dim oEquipment As IJEquipment
    Set oEquipment = pMemberDescription.CAO
    GetDimensionsFromSymbolArray oEquipment
    'Create Equipment Component
    Set pObject = m_oEquipCADHelper.CreateEquipmentComponent(pMemberDescription,
pResourceManager, "VDrum 01-EC", "VDrum")
    'create name for the member
    SetObjNameRule pObject, "CPEquipmentComponent"

    Set oEquipment = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

## Custom Method Final:

There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructVDrum(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMFinalConstructVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

## Custom method Inputs:

There is no need to add any code for this custom method.

```
Public Sub CMSetInputsVDrum(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMSetInputsVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

## Custom method Evaluate:

This method is in charge of setting the attribute values to the member object.

```
Public Sub CMEvaluateVDrum(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
    Const METHOD = "CMEvaluateVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    GetDimensionsFromSymbolArray oPropertyDescription.CAO

    Dim oAttribs As IJDAttributes
    Dim oSmartOcc As IJSmartOccurrence

    Set oSmartOcc = oPropertyDescription.Object
    Set oAttribs = oSmartOcc '.ItemObject
```

```
   oAttribs.CollectionOfAttributes("IJUAVesselDiameter").Item("VesselDiameter").Value = m_VesselDiameter
   oAttribs.CollectionOfAttributes("IJUAVesselTantoTan").Item("VesselTantoTan").Value = m_VesselTantoTan
   oAttribs.CollectionOfAttributes("IJInsulationThickness").Item("InsulationThickness").Value = m_dInsulationThickness
'
   Set oAttribs = Nothing
   Set oSmartOcc = Nothing

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method GeometryEvaluate:**

This method is in charge of setting the transformation matrix to move the member object relative to the equipment. Use the TransformFromECStoGCS function to maintain the VDrum member coordinate system relative to the Equipment coordinate system.

```
Public Sub CMEvaluateGeometryVDrum(ByVal oPropertyDescription As IJDPropertyDescription, pObject As
Object)
   Const METHOD = "CMEvaluateGeometryVDrum"
   On Error GoTo ErrorHandler
   LogCalls METHOD

'Add the following code to avoid the eqp component to move alone

   Dim oEqpComp As IJEquipmentComponent
   Dim oEqpCompMatrix As IJEquipment
   Dim oEquipment As IJEquipment

   Set oEqpComp = oPropertyDescription.Object
   oEqpComp.GetParent oEquipment
   Set oEqpCompMatrix = oEqpComp

   GetDimensionsFromSymbolArray oEquipment

   Dim otransform As IngrGeom3D.IJDT4x4
   Set otransform = New DT4x4
   Dim iVector As IJDVector
   Set iVector = New DVector

   otransform.LoadIdentity
   iVector.x = 0
   iVector.y = 0
   iVector.z = 0
   otransform.Translate iVector
   oEqpCompMatrix.SetMatrix otransform

   TransformFromECStoGCS oEquipment, oEqpCompMatrix

   Set oEquipment = Nothing
   Set oEqpComp = Nothing
   Set oEqpCompMatrix = Nothing

   Set iVector = Nothing
   Set otransform = Nothing
```

```
    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Conditional:

This method checks if the member is conditional based on the CanBeDeleted flag. Remember, we added code in the CMEvaluate to make the member deletable.

```
Public Sub CMConditionalVDrum(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As
Boolean)
    Const METHOD = "CMConditionalVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Release:

There is no need to add any code for this custom method

```
Public Sub CMReleaseVDrum(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMReleaseVDrum"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Construct:

This method is in charge of the creation of the CAO structure member system 1. Use *SPSMembers.SPSMemberFactory* to create the member. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructSPSSTR1(ByVal pMemberDescription As IJDMemberDescription, _
                              ByVal pResourceManager As IUnknown, _
                              ByRef pObj As Object)
Const METHOD = "CMConstructSPSSTR1"
On Error GoTo ErrorHandler

    'retrieve the inputs of the custom assembly occurrence

    Dim oDesignParent As IJDesignParent
    Set oDesignParent = pMemberDescription.CAO
    Dim oEquipment As IJEquipment
    Set oEquipment = pMemberDescription.CAO

    'get cross section and its ref standard
```

```
Dim LegSecStandard As String, LegSectionName As String
LegSecStandard = "AISC-LRFD-3.1"
LegSectionName = "C6x13"

'get cross section object
Dim oCrossSection As IJCrossSection
Set oCrossSection = GetCrossSection(LegSecStandard, LegSectionName)
Dim strMaterial As String, strGrade As String
strMaterial = "Steel - Carbon"
strGrade = "A"
Dim oMaterialDefinition As Object
Set oMaterialDefinition = GetMaterialObject(strMaterial, strGrade)

' get the equipment positional coordinates
Dim eqPos As AutoMath.DPosition
Set eqPos = New AutoMath.DPosition

GetDimensionsFromSymbolArray oEquipment
eqPos.Set m_VesselDiameter / 6, -m_VesselDiameter / 2 - m_VesselDiameter / 8, 0

Dim oMemberFactory  As SPSMembers.SPSMemberFactory
Dim oMemberSystem   As SPSMembers.ISPSMemberSystem
Dim oMemberPart     As SPSMembers.ISPSMemberPartPrismatic

Set oMemberFactory = New SPSMembers.SPSMemberFactory
Set oMemberSystem = oMemberFactory.CreateMemberSystemPrismaticLinear(pResourceManager)
Set oMemberPart = oMemberSystem.DesignPartAtEnd(SPSMemberAxisEnd)

Set oMemberPart.CrossSection.Definition = oCrossSection

oMemberPart.CrossSection.CardinalPoint = 4
oMemberPart.MemberType.TypeCategory = 1
oMemberPart.MemberType.Type = 100

Set oMemberPart.MaterialDefinition = oMaterialDefinition

oMemberSystem.LogicalAxis.SetLogicalStartPoint eqPos.x, eqPos.y, eqPos.z
oMemberSystem.LogicalAxis.SetLogicalEndPoint eqPos.x, eqPos.y, eqPos.z + m_VesselTantoTan * 2

oDesignParent.AddChild oMemberSystem

Dim oFrmConnE As ISPSFrameConnection
Set oFrmConnE = oMemberSystem.FrameConnectionAtEnd(SPSMemberAxisStart)
SetObjNameRule oFrmConnE, "CSPSFrameConnection"

Set oFrmConnE = oMemberSystem.FrameConnectionAtEnd(SPSMemberAxisEnd)
SetObjNameRule oFrmConnE, "CSPSFrameConnection"
Set oFrmConnE = Nothing

'create name for the member
SetObjNameRule oMemberPart, "CSPSMemberPartPrismatic"
SetObjNameRule oMemberSystem, "CSPSMemberSystemLinear"

Set pObj = oMemberSystem 'return the created structure member object to the custom assembly

Set oMemberSystem = Nothing
```

```
    Set oMemberPart = Nothing
    Set oMemberFactory = Nothing
    Set oCrossSection = Nothing
    Set oMaterialDefinition = Nothing
    Set oDesignParent = Nothing
    Set oEquipment = Nothing

    Set eqPos = Nothing

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Final:**
There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructSPSSTR1(pMemberDesc As IJDMemberDescription)
Const METHOD = "CMFinalConstructSPSSTR1"
On Error GoTo ErrorHandler

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Inputs:**
There is no need to add any code for this custom method.

```
Public Sub CMSetInputSPSSTR1(pMemberDesc As IJDMemberDescription)
Const METHOD = "CMSetInputSPSSTR1"
On Error GoTo ErrorHandler

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Evaluate:**
This method is in charge of setting the attribute values to the member object.

```
Public Sub CMEvaluateSPSSTR1(pPropertyDescriptions As IJDPropertyDescription, _
                              pObject As Object)
Const METHOD = "CMEvaluateSPSSTR1"
On Error GoTo ErrorHandler

    Dim oEquipment As IJEquipment
    Set oEquipment = pPropertyDescriptions.CAO

    ' get the equipment positional coordinates
    Dim eqPos As AutoMath.DPosition
    Set eqPos = New AutoMath.DPosition
    GetDimensionsFromSymbolArray oEquipment

    eqPos.Set m_VesselDiameter / 6, -m_VesselDiameter / 2 - m_VesselDiameter / 8, 0

    Dim oMemberPart As ISPSMemberPartPrismatic
    Dim oMemberSystem As ISPSMemberSystem
    Set oMemberSystem = pObject
    Set oMemberPart = oMemberSystem.DesignPartAtEnd(SPSMemberAxisEnd)

    oMemberSystem.LogicalAxis.SetLogicalStartPoint eqPos.x, eqPos.y, eqPos.z
    oMemberSystem.LogicalAxis.SetLogicalEndPoint eqPos.x, eqPos.y, eqPos.z + m_VesselTantoTan * 2

    oMemberSystem.Rotation.Mirror = True
    oMemberSystem.Rotation.SetOrientationVector 0, 1, 0

    TransformFromECStoGCS oEquipment, pObject

    Set oMemberSystem = Nothing
    Set oMemberPart = Nothing
    Set oEquipment = Nothing
    Set eqPos = Nothing

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method GeometryEvaluate:**

This method is in charge of maintaining the structure member system relative to the equipment.

```
Public Sub CMEvaluateGeometrySPSSTR1(pPropertyDescriptions As IJDPropertyDescription, _
                                    pObject As Object)
Const METHOD = "CMEvaluateGeometrySPSSTR1"
On Error GoTo ErrorHandler

    Call CMEvaluateSPSSTR1(pPropertyDescriptions, pObject)

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Construct**:

This method is in charge of the creation of the CAO structure member system 2. Use *SPSMembers.SPSMemberFactory* to create the member. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructSPSSTR2(ByVal pMemberDescription As IJDMemberDescription, _
                              ByVal pResourceManager As IUnknown, _
                              ByRef pObj As Object)
```

```
Const METHOD = "CMConstructSPSSTR2"
On Error GoTo ErrorHandler

    'retrieve the inputs of the custom assembly occurrence

    Dim oDesignParent As IJDesignParent
    Set oDesignParent = pMemberDescription.CAO
    Dim oEquipment As IJEquipment
    Set oEquipment = pMemberDescription.CAO

    'get cross section and its ref standard
    Dim LegSecStandard As String, LegSectionName As String
    LegSecStandard = "AISC-LRFD-3.1"
    LegSectionName = "C6x13"

    'get cross section object
    Dim oCrossSection As IJCrossSection
    Set oCrossSection = GetCrossSection(LegSecStandard, LegSectionName)
    Dim strMaterial As String, strGrade As String
    strMaterial = "Steel - Carbon"
    strGrade = "A"
    Dim oMaterialDefinition As Object
    Set oMaterialDefinition = GetMaterialObject(strMaterial, strGrade)

    ' get the equipment positional coordinates
    Dim eqPos As AutoMath.DPosition

    Set eqPos = New AutoMath.DPosition

    GetDimensionsFromSymbolArray oEquipment

    eqPos.Set -m_VesselDiameter / 6, -m_VesselDiameter / 2 - m_VesselDiameter / 8, 0

    Dim oMemberFactory  As SPSMembers.SPSMemberFactory
    Dim oMemberSystem   As SPSMembers.ISPSMemberSystem
    Dim oMemberPart     As SPSMembers.ISPSMemberPartPrismatic

    Set oMemberFactory = New SPSMembers.SPSMemberFactory
    Set oMemberSystem = oMemberFactory.CreateMemberSystemPrismaticLinear(pResourceManager)
    Set oMemberPart = oMemberSystem.DesignPartAtEnd(SPSMemberAxisEnd)

    Set oMemberPart.CrossSection.Definition = oCrossSection

    oMemberPart.CrossSection.CardinalPoint = 4
    oMemberPart.MemberType.TypeCategory = 1
    oMemberPart.MemberType.Type = 100

    Set oMemberPart.MaterialDefinition = oMaterialDefinition

    oMemberSystem.LogicalAxis.SetLogicalStartPoint eqPos.x, eqPos.y, eqPos.z
    oMemberSystem.LogicalAxis.SetLogicalEndPoint eqPos.x, eqPos.y, eqPos.z + m_VesselTantoTan * 2

    oDesignParent.AddChild oMemberSystem

    Dim oFrmConnE As ISPSFrameConnection
    Set oFrmConnE = oMemberSystem.FrameConnectionAtEnd(SPSMemberAxisStart)
    SetObjNameRule oFrmConnE, "CSPSFrameConnection"

    Set oFrmConnE = oMemberSystem.FrameConnectionAtEnd(SPSMemberAxisEnd)
    SetObjNameRule oFrmConnE, "CSPSFrameConnection"
    Set oFrmConnE = Nothing

    'create name for the member
    SetObjNameRule oMemberPart, "CSPSMemberPartPrismatic"
    SetObjNameRule oMemberSystem, "CSPSMemberSystemLinear"

    Set pObj = oMemberSystem 'return the created structure member object to the custom assembly

    Set oMemberSystem = Nothing
    Set oMemberPart = Nothing
    Set oMemberFactory = Nothing
    Set oCrossSection = Nothing
    Set oMaterialDefinition = Nothing
    Set oDesignParent = Nothing
    Set oEquipment = Nothing
    Set eqPos = Nothing

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Final:**

There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructSPSSTR2(pMemberDesc As IJDMemberDescription)
Const METHOD = "CMFinalConstructSPSSTR2"
On Error GoTo ErrorHandler

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Inputs:**

There is no need to add any code for this custom method.

```
Public Sub CMSetInputSPSSTR2(pMemberDesc As IJDMemberDescription)
Const METHOD = "CMSetInputSPSSTR2"
On Error GoTo ErrorHandler

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Evaluate:**

This method is in charge of setting the attribute values to the member object.

```
Public Sub CMEvaluateSPSSTR2(pPropertyDescriptions As IJDPropertyDescription, _
                              pObject As Object)
Const METHOD = "CMEvaluateSPSSTR2"
On Error GoTo ErrorHandler

   Dim oEquipment As IJEquipment
   Set oEquipment = pPropertyDescriptions.CAO

   ' get the equipment positional coordinates
   Dim eqPos As AutoMath.DPosition
   Set eqPos = New AutoMath.DPosition

   GetDimensionsFromSymbolArray oEquipment
   eqPos.Set -m_VesselDiameter / 6, -m_VesselDiameter / 2 - m_VesselDiameter / 8, 0

   Dim oMemberPart As ISPSMemberPartPrismatic
   Dim oMemberSystem As ISPSMemberSystem
   Set oMemberSystem = pObject
   Set oMemberPart = oMemberSystem.DesignPartAtEnd(SPSMemberAxisEnd)

   oMemberSystem.LogicalAxis.SetLogicalStartPoint eqPos.x, eqPos.y, eqPos.z
   oMemberSystem.LogicalAxis.SetLogicalEndPoint eqPos.x, eqPos.y, eqPos.z + m_VesselTantoTan * 2

   oMemberSystem.Rotation.Mirror = False
   oMemberSystem.Rotation.SetOrientationVector 0, 1, 0

   TransformFromECStoGCS oEquipment, pObject

   Set oMemberSystem = Nothing
   Set oMemberPart = Nothing
   Set oEquipment = Nothing
   Set eqPos = Nothing

   Exit Sub

ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

### Custom Method GeometryEvaluate:

This method is in charge of maintaining the structure member system relative to the equipment.

```
Public Sub CMEvaluateGeometrySPSSTR2(pPropertyDescriptions As IJDPropertyDescription, _
                                      pObject As Object)
Const METHOD = "CMEvaluateGeometrySPSSTR2"
On Error GoTo ErrorHandler

   Call CMEvaluateSPSSTR2(pPropertyDescriptions, pObject)
   Exit Sub

ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom Method Construct**:
This method is in charge of the creation of all the rods. Use *SPSMembers.SPSMemberFactory* to create the members. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructSPSRod(ByVal pMemberDescription As IJDMemberDescription, _
                              ByVal pResourceManager As IUnknown, _
                              ByRef pObj As Object)
Const METHOD = "CMConstructSPSRod"
On Error GoTo ErrorHandler

    'retrieve the inputs of the custom assembly occurrence

    Dim oDesignParent As IJDesignParent
    Set oDesignParent = pMemberDescription.CAO
    Dim oEquipment As IJEquipment
    Set oEquipment = pMemberDescription.CAO

    'get cross section and its ref standard
    Dim LegSecStandard As String, LegSectionName As String
    LegSecStandard = "Misc"
    LegSectionName = "CS1"

    'get cross section object
    Dim oCrossSection As IJCrossSection
    Set oCrossSection = GetCrossSection(LegSecStandard, LegSectionName)
    Dim strMaterial As String, strGrade As String
    strMaterial = "Steel - Carbon"
    strGrade = "A"
    Dim oMaterialDefinition As Object
    Set oMaterialDefinition = GetMaterialObject(strMaterial, strGrade)

    ' get the equipment positional coordinates
    Dim eqPos As AutoMath.DPosition
    Set eqPos = New AutoMath.DPosition

    GetDimensionsFromSymbolArray oEquipment

    eqPos.Set m_VesselDiameter / 6, -m_VesselDiameter / 2 - m_VesselDiameter / 8, m_VesselTantoTan * 2 -
m_Pitch * pMemberDescription.index

    Dim oMemberFactory  As SPSMembers.SPSMemberFactory
    Dim oMemberSystem   As SPSMembers.ISPSMemberSystem
    Dim oMemberPart     As SPSMembers.ISPSMemberPartPrismatic

    Set oMemberFactory = New SPSMembers.SPSMemberFactory
    Set oMemberSystem = oMemberFactory.CreateMemberSystemPrismaticLinear(pResourceManager)
    Set oMemberPart = oMemberSystem.DesignPartAtEnd(SPSMemberAxisEnd)

    Set oMemberPart.CrossSection.Definition = oCrossSection

    oMemberPart.CrossSection.CardinalPoint = 5
    oMemberPart.MemberType.TypeCategory = 1
    oMemberPart.MemberType.Type = 100
    Set oMemberPart.MaterialDefinition = oMaterialDefinition

    oMemberSystem.LogicalAxis.SetLogicalStartPoint eqPos.x, eqPos.y, eqPos.z
    oMemberSystem.LogicalAxis.SetLogicalEndPoint eqPos.x - m_VesselDiameter / 3, eqPos.y, eqPos.z

    oDesignParent.AddChild oMemberSystem

    Dim oFrmConnE As ISPSFrameConnection
    Set oFrmConnE = oMemberSystem.FrameConnectionAtEnd(SPSMemberAxisStart)
    SetObjNameRule oFrmConnE, "CSPSFrameConnection"

    Set oFrmConnE = oMemberSystem.FrameConnectionAtEnd(SPSMemberAxisEnd)
    SetObjNameRule oFrmConnE, "CSPSFrameConnection"
    Set oFrmConnE = Nothing

    'create name for the member
    SetObjNameRule oMemberPart, "CSPSMemberPartPrismatic"
    SetObjNameRule oMemberSystem, "CSPSMemberSystemLinear"

    Set pObj = oMemberSystem 'return the created structure member object to the custom assembly

    Set oMemberSystem = Nothing
    Set oMemberPart = Nothing
    Set oMemberFactory = Nothing
    Set oCrossSection = Nothing
    Set oMaterialDefinition = Nothing
    Set oDesignParent = Nothing
    Set oEquipment = Nothing
    Set eqPos = Nothing

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Final:

There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructSPSRod(pMemberDesc As IJDMemberDescription)
Const METHOD = "CMFinalConstructSPSRod"
On Error GoTo ErrorHandler

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Inputs:

There is no need to add any code for this custom method.

```
Public Sub CMSetInputSPSRod(pMemberDesc As IJDMemberDescription)
Const METHOD = "CMSetInputSPSRod"
On Error GoTo ErrorHandler
```

```
    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Evaluate:

This method is in charge of setting the attribute values to the member object.

```
Public Sub CMEvaluateSPSRod(pPropertyDescriptions As IJDPropertyDescription, _
                              pObject As Object)
Const METHOD = "CMEvaluateSPSRod"
On Error GoTo ErrorHandler

    Dim oEquipment As IJEquipment
    Set oEquipment = pPropertyDescriptions.CAO

    ' get the equipment positional coordinates
    Dim eqPos As AutoMath.DPosition
    Set eqPos = New AutoMath.DPosition

    GetDimensionsFromSymbolArray oEquipment

    eqPos.Set m_VesselDiameter / 6, -m_VesselDiameter / 2 - m_VesselDiameter / 8, m_VesselTantoTan * 2 -
m_Pitch * pPropertyDescriptions.index

    Dim oMemberPart As ISPSMemberPartPrismatic
    Dim oMemberSystem As ISPSMemberSystem
    Set oMemberSystem = pObject
    Set oMemberPart = oMemberSystem.DesignPartAtEnd(SPSMemberAxisEnd)

    oMemberSystem.LogicalAxis.SetLogicalStartPoint eqPos.x, eqPos.y, eqPos.z
    oMemberSystem.LogicalAxis.SetLogicalEndPoint eqPos.x - m_VesselDiameter / 3, eqPos.y, eqPos.z

    oMemberSystem.Rotation.Mirror = False
    oMemberSystem.Rotation.SetOrientationVector 0,1,0

    TransformFromECStoGCS oEquipment, pObject

    Set oMemberSystem = Nothing
    Set oMemberPart = Nothing
    Set oEquipment = Nothing
    Set eqPos = Nothing

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method GeometryEvaluate:

This method is in charge of maintaining the structure member system relative to the equipment.

```
Public Sub CMEvaluateGeometrySPSRod(pPropertyDescriptions As IJDPropertyDescription, _
                                    pObject As Object)
Const METHOD = "CMEvaluateGeometrySPSRod"
On Error GoTo ErrorHandler

    Call CMEvaluateSPSRod(pPropertyDescriptions, pObject)

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom Method Count:

This custom method read the global variable m_Pitch and m_vesselTantoTan to determine total number of rods need to be created.

```
Public Sub CMCountSPSRod(ByVal pMemberDesc As IJDMemberDescription, ByRef Count As Long)
    Const METHOD = "CMCountSPSRod"
    On Error GoTo ErrorHandler

    GetDimensionsFromSymbolArray pMemberDesc.CAO
    Count = Int(m_VesselTantoTan * 2 / m_Pitch) - 1

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Conditional:

This method makes sure that the system will not construct the rod if the count is less or equal to zero.

```
Public Sub CMConditionalRod(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As Boolean)
    Const METHOD = "CMConditionalRod"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    If pMemberDesc.index <= 0 Then
    IsNeeded = False
    Else
    IsNeeded = True
    End If

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

23. Add the following subroutine to get the cross section object given the library and section name.

```
Public Function GetCrossSection(ByVal SectionStandard As String, _
                                ByVal SectionName As String) As Object
```

```
Const METHOD = "GetCrossSection"
On Error GoTo ErrorHandler

   Dim oCrossSec As Object
   Dim oStructServices As New RefDataMiddleServices.StructCrossSectionServices
   oStructServices.GetStructureCrossSectionDefinition GetCatalogResourceManager, SectionStandard, "", 
SectionName, oCrossSec

   Set GetCrossSection = oCrossSec

   Set oCrossSec = Nothing
   Set oStructServices = Nothing

   Exit Function
ErrorHandler: HandleError MODULE, METHOD
End Function
```

24. Add the following subroutine to convert the array of inputs in a set of global variables.

```
Private Sub GetDimensionsFromSymbolArray(SmartOccurrence As IJSmartOccurrence)
   Const METHOD = "GetDimensionsFromSymbolArray"
   On Error GoTo ErrorHandler

   m_avSymbolArrayOfInputs = m_oEquipCADHelper.GetSymbolArrayOfInputs(SmartOccurrence)

   'Inputs,  from equipment symbol code
   'Set m_oPartFclt = m_avSymbolArrayOfInputs(1)
   m_VesselDiameter = m_avSymbolArrayOfInputs(2)
   m_VesselTantoTan = m_avSymbolArrayOfInputs(3)
   m_dInsulationThickness = m_avSymbolArrayOfInputs(4)
   m_Pitch = m_avSymbolArrayOfInputs(5)

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

25. Add the following subroutine to position and orient the member with respect to the equipment.

```
Private Sub TransformFromECStoGCS(Equipment As IJEquipment, Object As Object)
   Const METHOD = "TransformFromECStoGCS"
   LogCalls METHOD
   On Error GoTo ErrorHandler
   Dim oEqpMatrix As IJDT4x4
   Dim oShapeMatrix As IJDT4x4
   Dim otransform As IJDGeometry
   Dim oShape As IJShape

   If Not Object Is Nothing Then
      If TypeOf Object Is IJDGeometry Then
         Equipment.GetMatrix oEqpMatrix
         Set otransform = Object
         otransform.DTransform oEqpMatrix
         Set otransform = Nothing
```

```
            Set oEqpMatrix = Nothing
        End If
    End If

    Set otransform = Nothing
    Set oEqpMatrix = Nothing
    Set oShape = Nothing
    Set oShapeMatrix = Nothing

    Exit Sub
ErrorHandler:
    Set otransform = Nothing
    Set oEqpMatrix = Nothing
    Set oShape = Nothing
    Set oShapeMatrix = Nothing
    HandleError MODULE, METHOD
End Sub
```

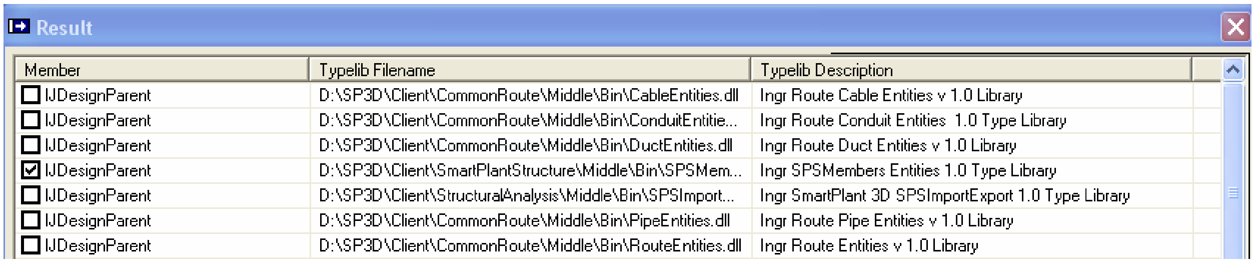
26. Add the following subroutine to set the naming relation and generate a name based on the default naming rule.

```
Public Sub SetObjNameRule(ByRef obj As Object, ByRef CLASSNAME As String)
' Apply the namerule using the IJDNamingRulesHelper helper interface
Const METHOD = "SetNameRule"
On Error GoTo ErrorHandler

    Dim NameRule As String
    Dim NamingRules As IJElements

    Dim oNameRuleHlpr As GSCADNameRuleSemantics.IJDNamingRulesHelper

'Returns a collection of the naming rules available in the catalog database
'for the given object
    Set oNameRuleHlpr = New GSCADNameRuleHlpr.NamingRulesHelper
    Call oNameRuleHlpr.GetEntityNamingRulesGivenName(CLASSNAME, NamingRules)
'get the first namerule from the collection
    Dim oNameRuleHolder As GSCADGenericNamingRulesFacelets.IJDNameRuleHolder
    Set oNameRuleHolder = NamingRules.Item(1)
'Create relations "NamedEntity" and "EntityNamingRule" and obj
    Dim oNameRuleAE As GSCADGenNameRuleAE.IJNameRuleAE
    Call oNameRuleHlpr.AddNamingRelations(obj, oNameRuleHolder, oNameRuleAE)

    GoTo CleanObjects
ErrorHandler:
    HandleError MODULE, METHOD

CleanObjects:
    Set oNameRuleHlpr = Nothing
    Set oNameRuleHolder = Nothing
    Set oNameRuleAE = Nothing
    Set NamingRules = Nothing
End Sub
```

27. Add the following subroutine to log any error.

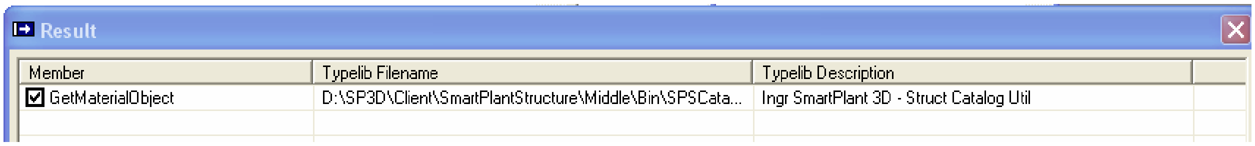
```
Private Sub LogCalls(sMethod As String)

   If Not m_oEditErrors Is Nothing Then
      m_oEditErrors.Add 5000, m_oEquipCADHelper.ProjectName & "." &
m_oEquipCADHelper.CLASSNAME, "Entering " & sMethod
   End If

End Sub
```

28. Compile the Visual Basic project and save the dll as SP3DTank7Asm.dll in the c:\Train\lab7
29. Use the SP3D Reference Tool to attach the missing reference libraries.



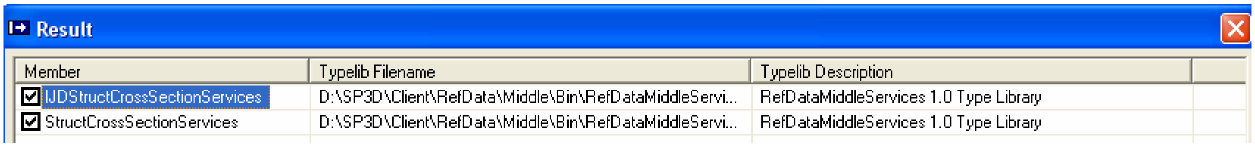
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| □ IJDesignParent | D:\SP3D\Client\CommonRoute\Middle\Bin\CableEntities.dll | Ingr Route Cable Entities v 1.0 Library |
| □ IJDesignParent | D:\SP3D\Client\CommonRoute\Middle\Bin\ConduitEntitie... | Ingr Route Conduit Entities  1.0 Type Library |
| □ IJDesignParent | D:\SP3D\Client\CommonRoute\Middle\Bin\DuctEntities.dll | Ingr Route Duct Entities v 1.0 Library |
| ☑ IJDesignParent | D:\SP3D\Client\SmartPlantStructure\Middle\Bin\SPSMem... | Ingr SPSMembers Entities 1.0 Type Library |
| □ IJDesignParent | D:\SP3D\Client\StructuralAnalysis\Middle\Bin\SPSImport... | Ingr SmartPlant 3D SPSImportExport 1.0 Type Library |
| □ IJDesignParent | D:\SP3D\Client\CommonRoute\Middle\Bin\PipeEntities.dll | Ingr Route Pipe Entities v 1.0 Library |
| □ IJDesignParent | D:\SP3D\Client\CommonRoute\Middle\Bin\RouteEntities.dll | Ingr Route Entities v 1.0 Library |



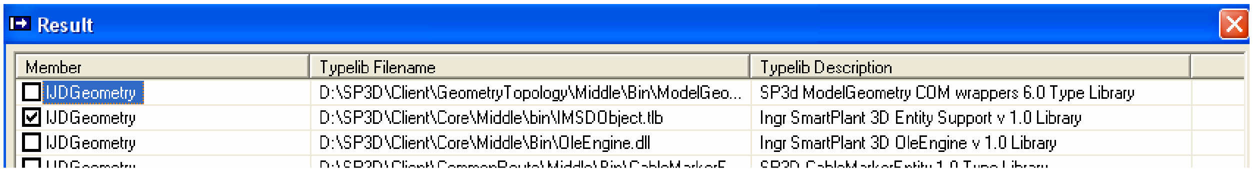
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ GetMaterialObject | D:\SP3D\Client\SmartPlantStructure\Middle\Bin\SPSCata... | Ingr SmartPlant 3D - Struct Catalog Util |



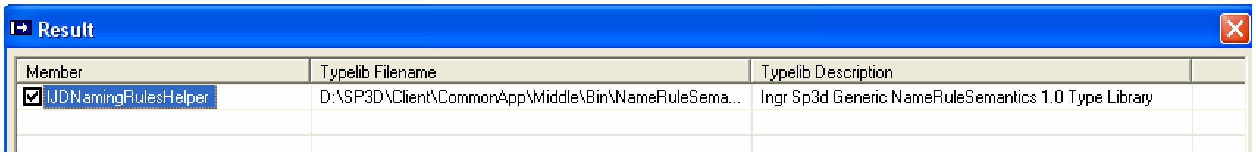
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDStructCrossSectionServices | D:\SP3D\Client\RefData\Middle\Bin\RefDataMiddleServi... | RefDataMiddleServices 1.0 Type Library |
| ☑ StructCrossSectionServices | D:\SP3D\Client\RefData\Middle\Bin\RefDataMiddleServi... | RefDataMiddleServices 1.0 Type Library |



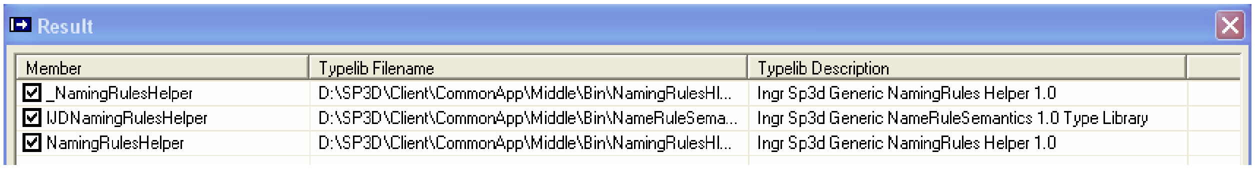
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| □ IJDGeometry | D:\SP3D\Client\GeometryTopology\Middle\Bin\ModelGeo... | SP3d ModelGeometry COM wrappers 6.0 Type Library |
| ☑ IJDGeometry | D:\SP3D\Client\Core\Middle\bin\IMSDObject.tlb | Ingr SmartPlant 3D Entity Support v 1.0 Library |
| □ IJDGeometry | D:\SP3D\Client\Core\Middle\Bin\OleEngine.dll | Ingr SmartPlant 3D OleEngine v 1.0 Library |
| □ IJDGeometry | D:\SP3D\Client\CommonRoute\Middle\Bin\CableMarkerE... | SP3D CableMarkerEntity 1.0 Type Library |



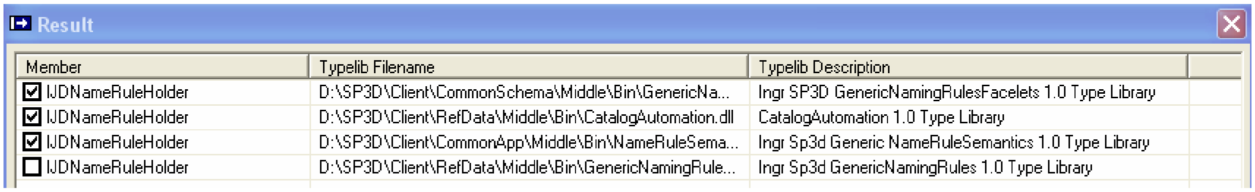
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDNamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |



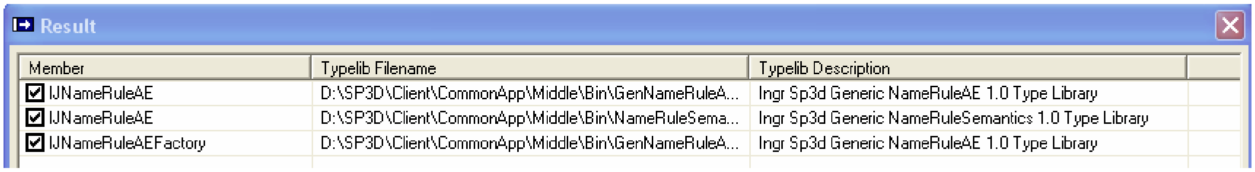
Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ _NamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NamingRulesHl... | Ingr Sp3d Generic NamingRules Helper 1.0 |
| ☑ IJDNamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☑ NamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NamingRulesHl... | Ingr Sp3d Generic NamingRules Helper 1.0 |



Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDNameRuleHolder | D:\SP3D\Client\CommonSchema\Middle\Bin\GenericNa... | Ingr SP3D GenericNamingRulesFacelets 1.0 Type Library |
| ☑ IJDNameRuleHolder | D:\SP3D\Client\RefData\Middle\Bin\CatalogAutomation.dll | CatalogAutomation 1.0 Type Library |
| ☑ IJDNameRuleHolder | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| □ IJDNameRuleHolder | D:\SP3D\Client\RefData\Middle\Bin\GenericNamingRule... | Ingr Sp3d GenericNamingRules 1.0 Type Library |

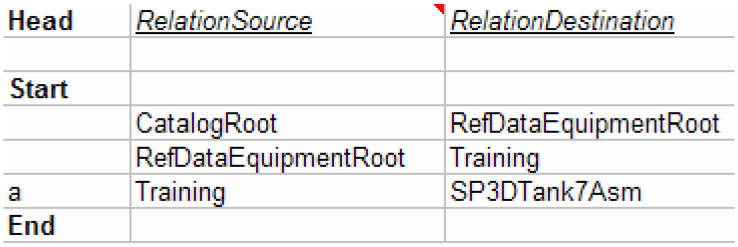


Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJNameRuleAE | D:\SP3D\Client\CommonApp\Middle\Bin\GenNameRuleA... | Ingr Sp3d Generic NameRuleAE 1.0 Type Library |
| ☑ IJNameRuleAE | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☑ IJNameRuleAEFactory | D:\SP3D\Client\CommonApp\Middle\Bin\GenNameRuleA... | Ingr Sp3d Generic NameRuleAE 1.0 Type Library |

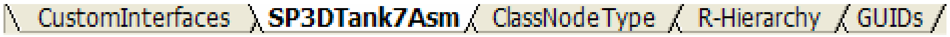
One of the most important steps in Visual Basic programming is to preserve the binary compatibility of your program. Save the final version of your dll file to be binary compatibility in order to preserve the CLSID.

30. Save the Visual Basic SP3DTank7Asm project.
31. Open the SP3DTemplate.xls workbook. Go the R-Hierarchy sheet and add the following entry.



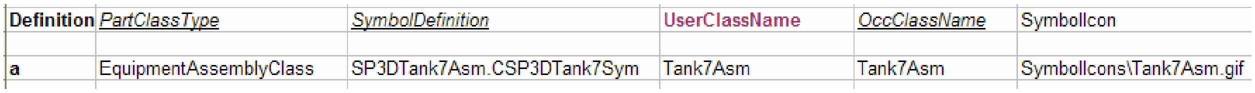
| Head | RelationSource | RelationDestination |
|---|---|---|
| | | |
| Start | | |
| | CatalogRoot | RefDataEquipmentRoot |
| | RefDataEquipmentRoot | Training |
| a | Training | SP3DTank7Asm |
| End | | |

32. Go to the SP3DTemplateAsm sheet and rename it as SP3DTank7Asm.



33. Go to the Class definition section and add/edit as follows:

34. In the Definition Section rows:

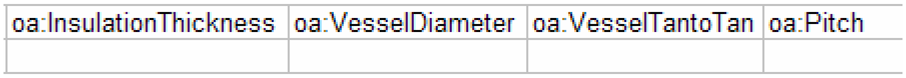


| Definition | PartClassType | SymbolDefinition | UserClassName | OccClassName | SymbolIcon |
|---|---|---|---|---|---|
| a | EquipmentAssemblyClass | SP3DTank7Asm.CSP3DTank7Sym | Tank7Asm | Tank7Asm | SymbolIcons\Tank7Asm.gif |

Note:

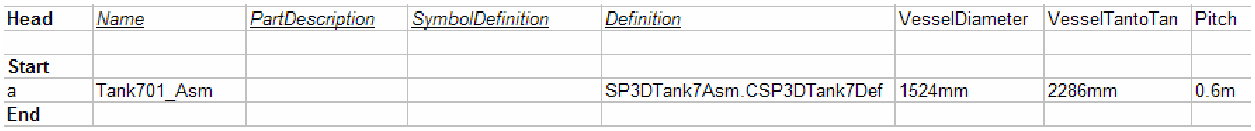
- Creating the bmp or gif file is optional. You can use Microsoft Paint to create the file and save it under your \\machine\symbols\SymbolIcons

Occurrence attributes:



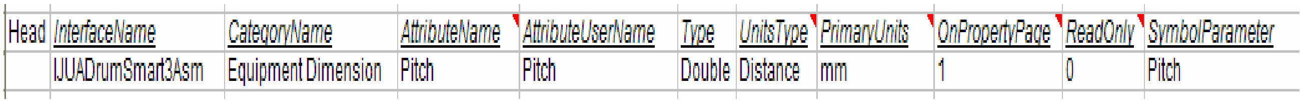
| oa:InsulationThickness | oa:VesselDiameter | oa:VesselTantoTan | oa:Pitch |
|---|---|---|---|
| | | | |

35. In the Part Section rows:



| Head | Name | PartDescription | SymbolDefinition | Definition | VesselDiameter | VesselTantoTan | Pitch |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| Start | | | | | | | |
| a | Tank701_Asm | | | SP3DTank7Asm.CSP3DTank7Def | 1524mm | 2286mm | 0.6m |
| End | | | | | | | |

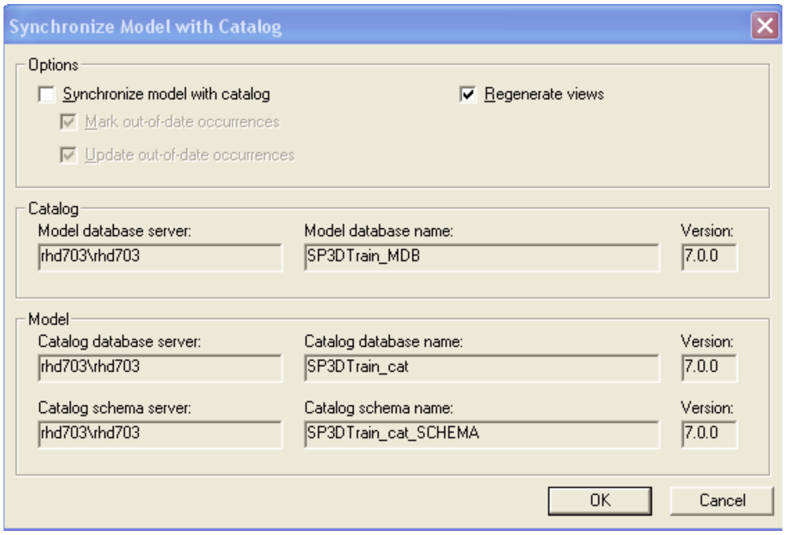
36. Create a new interface called IJUADrumSmart3Asm. Go to the Custom Interface sheet and add the following entries:



| Head | InterfaceName | CategoryName | AttributeName | AttributeUserName | Type | UnitsType | PrimaryUnits | OnPropertyPage | ReadOnly | SymbolParameter |
|---|---|---|---|---|---|---|---|---|---|---|
| | IJUADrumSmart3Asm | Equipment Dimension | Pitch | Pitch | Double | Distance | mm | 1 | 0 | Pitch |

37. Save the Excel workbook as SP3DTank7Asm.xls in the c:\Train\lab7.
38. Optional step: Create the Tank7Asm.gif file and place it under \\<MachineName>\Symbols\SymbolIcons
39. Load the information into the catalog using the Add/Modify/Delete Mode. Once the bulkload process is completed, review the log file.
40. Run the Project Management Task. Select the Model in the hierarchy.
41. Select Tools -> Synchronize Model with the Catalog.
42. Uncheck the Synchronize Model with the Catalog option.

***Note**: You just need to update the views in the model.*



Synchronize Model with Catalog

Options

□ Synchronize model with catalog ☑ Regenerate views

☑ Mark out-of-date occurrences

☑ Update out-of-date occurrences

Catalog

Model database server: rhd703\rhd703

Model database name: SP3DTrain_MDB

Version: 7.0.0

Model

Catalog database server: rhd703\rhd703

Catalog database name: SP3DTrain_cat

Version: 7.0.0

Catalog schema server: rhd703\rhd703

Catalog schema name: SP3DTrain_cat_SCHEMA

Version: 7.0.0

43. Hit “OK” Button.
44. Once the process is completed, Right click the training plant icon and select “Regenerate the Reports database” option to re-create the views in the report database.
45. Go to the Equipment Task and place the SP3DTank7Asm.

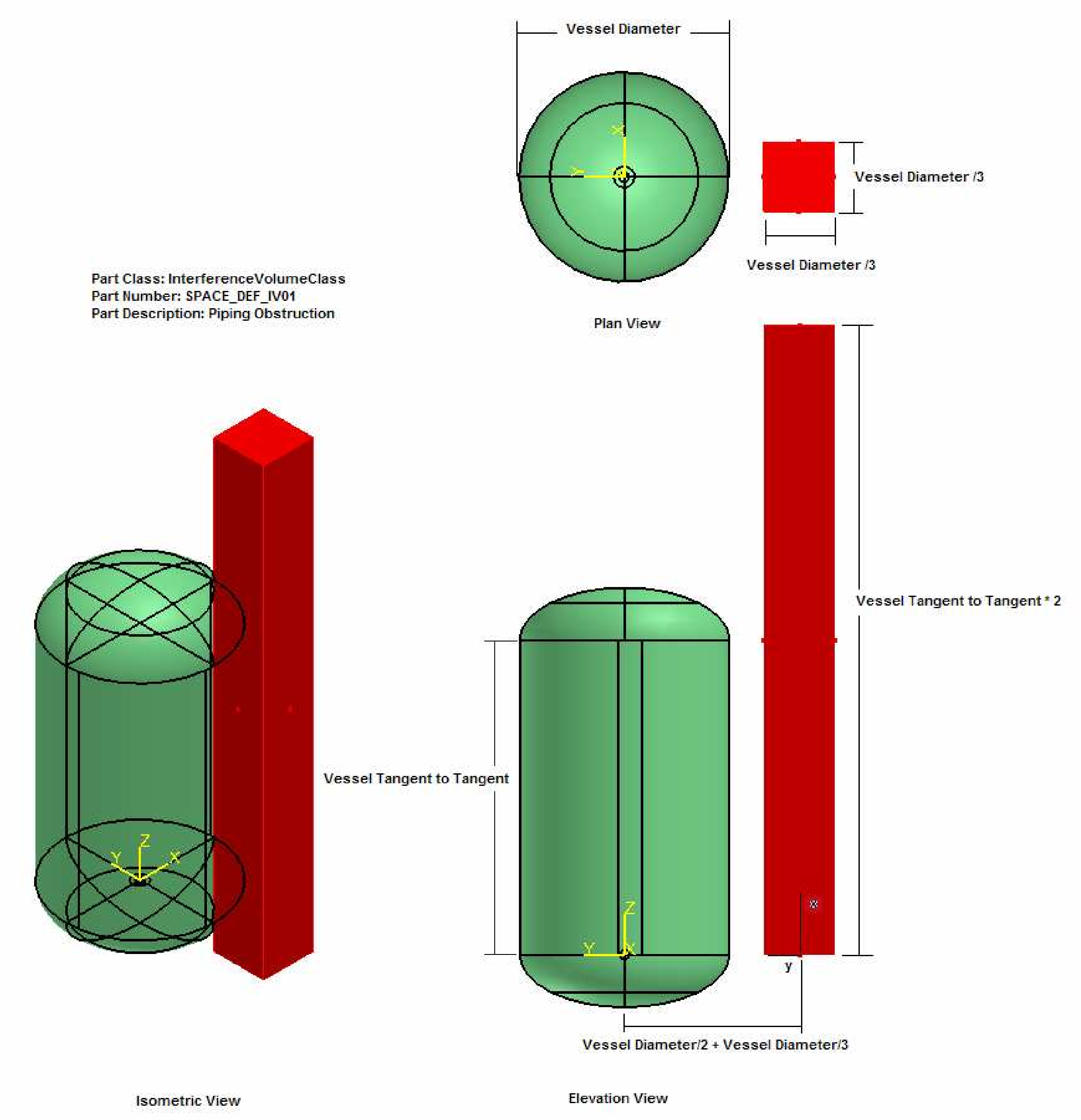
# Lab 9: Spatial object as member of the Equipment (Optional)

## Objectives

After completing this lab, you will be able to:

- Create a catalog equipment symbol made of equipment components and a spatial object
- Learn to use the Symbol Helper service to create the symbol definition
- Use the Equipment CAD Helper to define the Custom Assembly Definition
- Learn to use the IJDAggregatorDescription , IJDMemberDescriptions, and IJDPropertyDescriptions to define the behaviors of the custom assembly occurrence (CAO)
- Use the MakeMemberDeletable method to make the member deletable
- Use the IJEquipUserAttrMgmt Interface to show the equipment component attributes as read only in the property page.
- Use IJDNamingRulesHelper interface to create the naming relations between a naming rule and the object
- Use IJDAttributes interface to get a collection of attributes property
- Use IJDAttribute to get an object's attribute
- Use the IJDSpaceFactory to create the space object

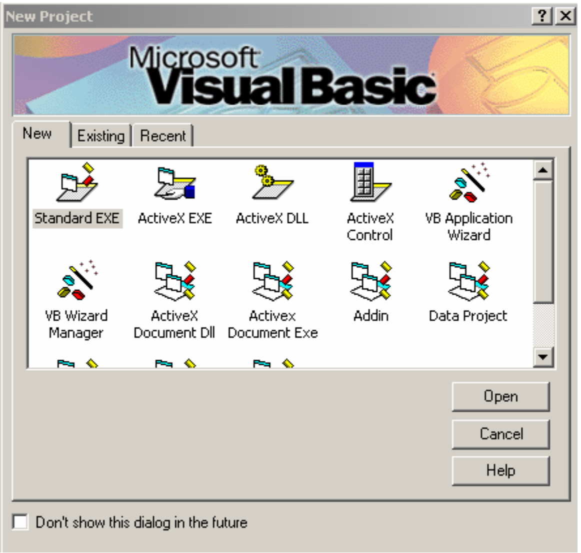
In this lab, you will create an equipment symbol as shown below. You start by using the SP3DEqpTemplateAsm template provided by the instructor to create the symbol. This symbol consists of one equipment component and a space object to define the symbol’s output. Use the Equipment Custom Assembly Definition (CAD) Helper to create the members.


Vessel Diameter
Vessel Diameter /3
Vessel Diameter /3
Part Class: InterferenceVolumeClass
Part Number: SPACE_DEF_IV01
Part Description: Piping Obstruction
Plan View
Vessel Tangent to Tangent * 2
Vessel Tangent to Tangent
y
Vessel Diameter/2 + Vessel Diameter/3
Isometric View
Elevation View

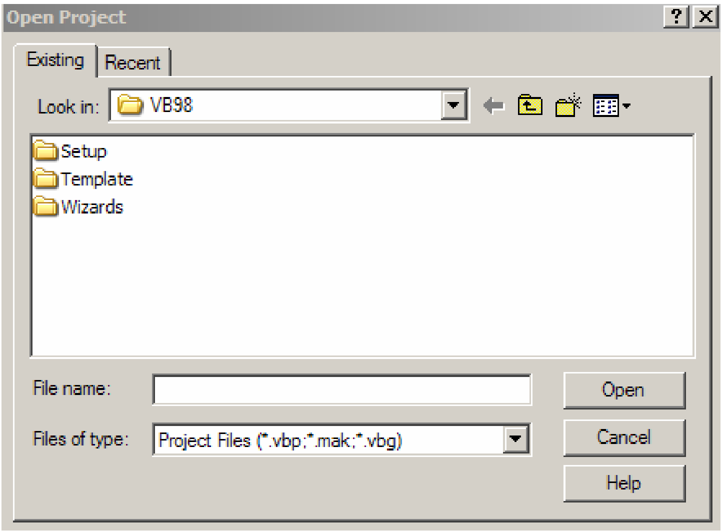
1. Create a directory called lab8 as follows:

   *c:\train\lab8*

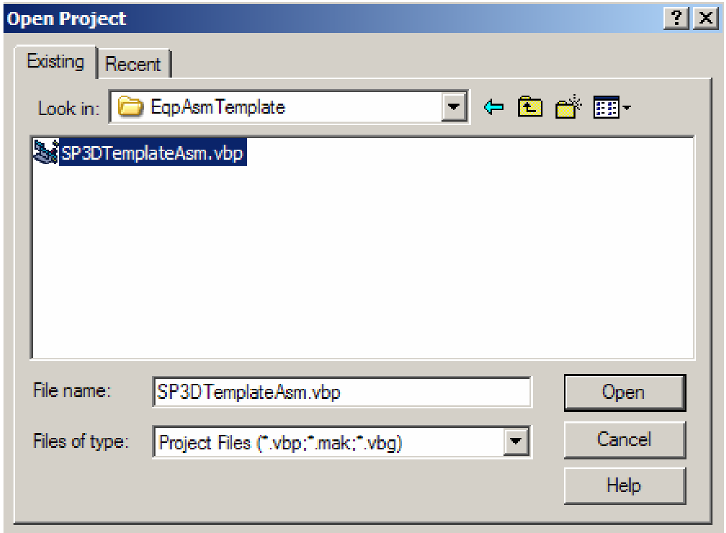
2. Run Microsoft Visual Basic 6.0
3. Close the Microsoft New Project dialog box.



4. Select *File* -> *Open Project* option to open the Open Project Dialog box



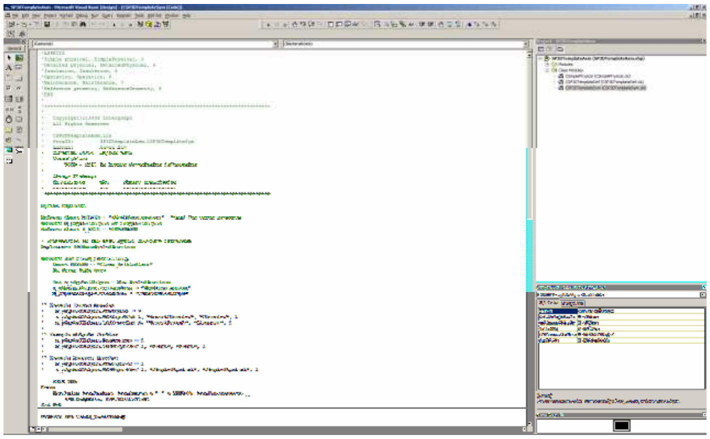
5. Navigate to c:\train\EqpAsmTemplate and open the SP3DTemplateAsm Template project



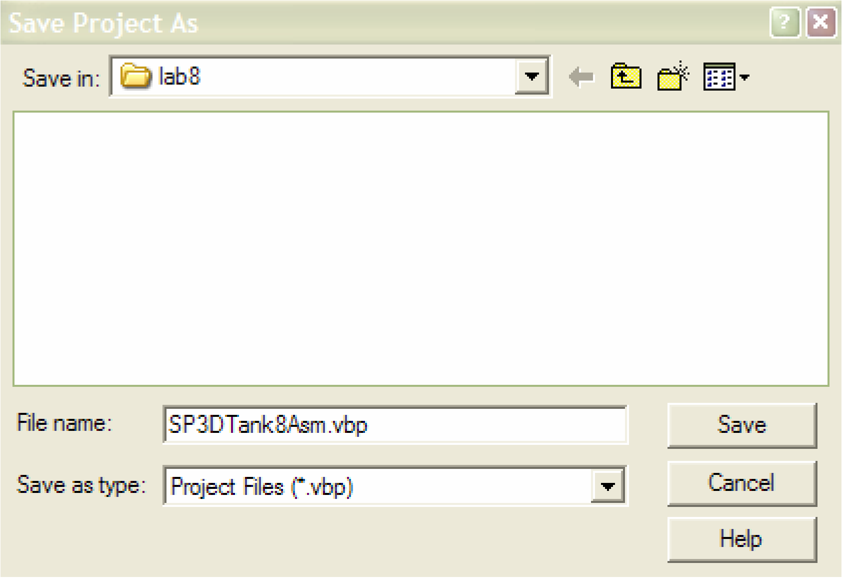
File name: SP3DTemplateAsm.vbp

Files of type: Project Files (*.vbp;*.mak;*.vbg)

6. Setup the Visual Basic Development Environment as shown below:



7. Go to the Visual Basic Explorer Window and select the Project node. Select *File* -> *Save Project As* option to save the project as SP3DTank8Asm.vbp under the lab8 directory

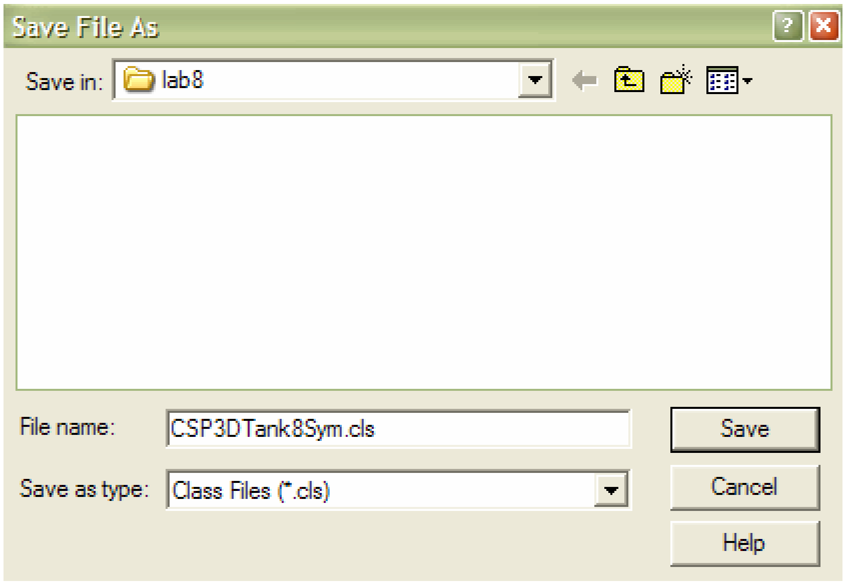


Save in: lab8

File name: SP3DTank8Asm.vbp

Save as type: Project Files (*.vbp)

8. Go to the Visual Basic Explorer Window and select the CSP3DTemplateSym class node. Select *File -> Save CSP3DTemplateSym.cls As* option to save the class module as CSP3DTank8Sym.cls under lab8 directory

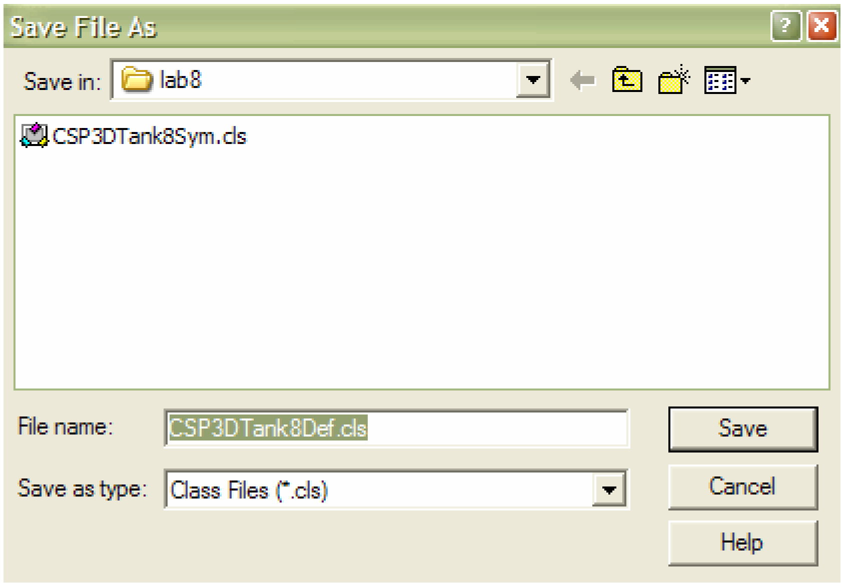


Save in: lab8

File name: CSP3DTank8Sym.cls

Save as type: Class Files (*.cls)

9. Go to the Visual Basic Explorer Window and select the CSP3DTemplateDef class node. Select *File -> Save CSP3DTemplateDef.cls As* option to save the class module as CSP3DTank8Def.cls under lab8 directory



Save File As

Save in: lab8

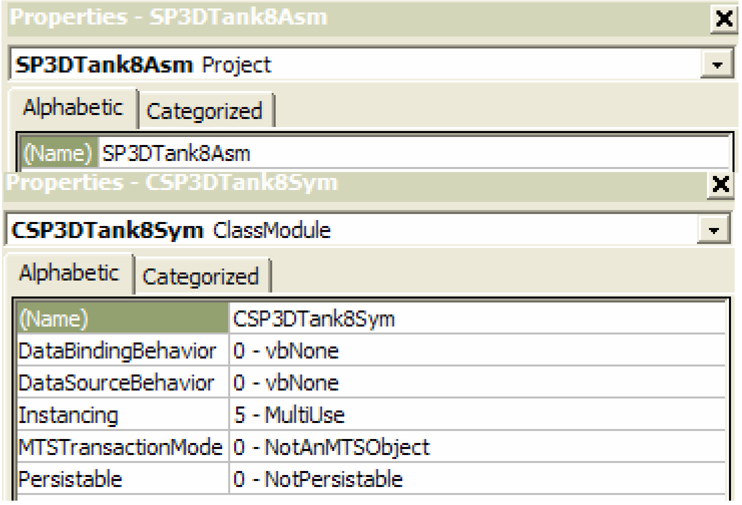
CSP3DTank8Sym.cls

File name: CSP3DTank8Def.cls

Save as type: Class Files (*.cls)

10. Go to the Visual Basic Explorer Window and select the CSimplePhysical class node. Select *File -> Save CSimplePhysical.cls As* option to save the class module as CSimplePhysical.cls under lab8 directory.

11. Go to the Properties Window and change the name of the Project and both Class Modules as follows:



Properties - SP3DTank8Asm

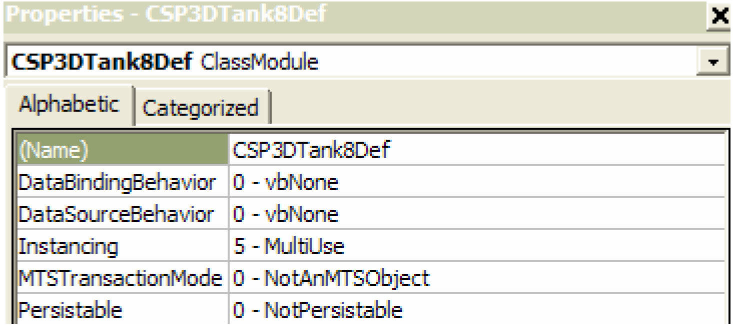
SP3DTank8Asm Project

|  |  |
|---|---|
| (Name) | SP3DTank8Asm |

Properties - CSP3DTank8Sym

CSP3DTank8Sym ClassModule

|  |  |
|---|---|
| (Name) | CSP3DTank8Sym |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |



| | |
|---|---|
| (Name) | CSP3DTank8Def |
| DataBindingBehavior | 0 - vbNone |
| DataSourceBehavior | 0 - vbNone |
| Instancing | 5 - MultiUse |
| MTSTransactionMode | 0 - NotAnMTSObject |
| Persistable | 0 - NotPersistable |

12. Go to the General Declarations section in CSP3DTank8Sym module and change the value of the *Constant Module variable* from *“CSP3DTemplateAsm:”* to *“CSP3DTank8Asm:”*

```
Private Const MODULE = "CSP3DTank8Asm:"  'Used for error messages
```

13. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
    Set m_oSymbolHelper = New SymbolServices
    m_oSymbolHelper.ProjectName = "SP3DTank8Asm"
    m_oSymbolHelper.ClassName = "CSP3DTank8Sym"
```

14. In this Class_Initialize() routine, add the following code to define the inputs and aspects definition for this symbol. Note: there is no outputs section.

```
' Inputs Section
    m_oSymbolHelper.NumInputs = 3
    m_oSymbolHelper.AddInputDef 1, "VesselDiameter", "VesselDiameter", 1
    m_oSymbolHelper.AddInputDef 2, "VesselTantoTan", "VesselTantoTan", 3
    m_oSymbolHelper.AddInputDef 3, "InsulationThickness", "InsulationThickness", 0.06

' Aspects Section
    m_oSymbolHelper.NumAspects = 1
    m_oSymbolHelper.AddAspectDef 1, "SimplePhysical", "SimplePhysical", 1
```

15. Go to CSimplePhysical Class module/Run subroutine and make sure there is no code to get the inputs.

16. Go to the General Declarations section in CSP3DTank8Def module and change the value of the *Constant Module variable* from “*"SP3DTemplateAsm:CSP3DTemplateDef"* to *"SP3DTank8Asm:CSP3DTank8Def"*

17. Go to the top of the CSP3DTank8Def module and declare the following variables

```
Private Const MODULE = "SP3DTemplateAsm:CSP3DTemplateDef"

Private Const IID_IJDATTRIBUTES = "{B25FD387-CFEB-11D1-850B-080036DE8E03}"
Private Const IID_IJDGEOMETRY = "{A1732CBF-5136-11D1-9770-080036754203}"

Private m_oEquipCADHelper As IJEquipCADHelper
Private m_oEditErrors As IJEditErrors
Private m_avSymbolArrayOfInputs()   As Variant

'VDrum
Private m_VesselDiameter As Double
Private m_VesselTantoTan As Double
Private m_dInsulationThickness As Double

Private m_oNorth                  As IJDVector
Private m_oEast                   As IJDVector
Private m_oElevation              As IJDVector
Private m_FolderName As String
```

18. Declare the BYPRIMITIVEPROGID

```
Private Const BYPRIMITIVEPROGID = "SpaceMgmtSemantics.SpaceByPrimitiveAE.1"
```

19. Go to the Class_Initialize() routine, rename the project name and class name as shown below:

```
 m_oEquipCADHelper.ProjectName = "SP3DTank8Asm"
 m_oEquipCADHelper.ClassName = "CSP3DTank8Def"
```

20. In this Class_Initialize() routine, initialize the following variable:

```
   Set m_oEast = New DVector
   m_oEast.x = 1
   m_oEast.y = 0
   m_oEast.z = 0

   Set m_oNorth = New DVector
   m_oNorth.x = 0
   m_oNorth.y = 1
   m_oNorth.z = 0

   Set m_oElevation = New DVector
   m_oElevation.x = 0
   m_oElevation.y = 0
   m_oElevation.z = 1
```

21. Go to the Class_Terminate() routine and use the Set statement to clear the references from all object variables.

```
Private Sub Class_Terminate()

   Set m_oNorth = Nothing
   Set m_oEast = Nothing
   Set m_oElevation = Nothing

   Set m_oEditErrors = Nothing
   Set m_oEquipCADHelper = Nothing
End Sub
```

22. Go to CSP3DTank8Def Class module. Declare the appropriate custom methods to manage the drum and the space objects as follows:

```
    'Add your code here for the declaration of the Public Custom Methods used to manage new members
    'Add new member(VDrum) to the definition

    Set oMemberDescription = Nothing
    Set oMemberDescription = oMemberDescriptions.AddMember("VDrum", 1, "CMConstructVDrum",
imsCOOKIE_ID_USS_LIB)
    oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsVDrum"
    oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructVDrum"
    oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalVDrum"
    oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseVDrum"

    Set oPropertyDescriptions = Nothing
    Set oPropertyDescriptions = oMemberDescription
    oPropertyDescriptions.AddProperty "VDrumProperties", 1, IID_IJDATTRIBUTES, "CMEvaluateVDrum",
imsCOOKIE_ID_USS_LIB
    oPropertyDescriptions.AddProperty "VDrumGeometryProperties", 2, IID_IJDGEOMETRY,
"CMEvaluateGeometryVDrum", imsCOOKIE_ID_USS_LIB

'    'Add new member(Volume1) to the definition

    Set oMemberDescription = Nothing
    Set oMemberDescription = oMemberDescriptions.AddMember("Volume1", 2, "CMConstructVolume1",
imsCOOKIE_ID_USS_LIB)
    oMemberDescription.SetCMSetInputs imsCOOKIE_ID_USS_LIB, "CMSetInputsVolume1"
    oMemberDescription.SetCMFinalConstruct imsCOOKIE_ID_USS_LIB, "CMFinalConstructVolume1"
    oMemberDescription.SetCMConditional imsCOOKIE_ID_USS_LIB, "CMConditionalVolume1"
    oMemberDescription.SetCMRelease imsCOOKIE_ID_USS_LIB, "CMReleaseVolume1"

    Set oPropertyDescriptions = Nothing
    Set oPropertyDescriptions = oMemberDescription
    oPropertyDescriptions.AddProperty "Volume1Properties", 1, IID_IJDATTRIBUTES,
"CMEvaluateVolume1", imsCOOKIE_ID_USS_LIB
    oPropertyDescriptions.AddProperty "Volume1GeometryProperties", 2, IID_IJDGEOMETRY,
"CMEvaluateGeometryVolume1", imsCOOKIE_ID_USS_LIB
```

23. Go to IJEquipUserAttrMgmt_OnPreLoad function and add the following code. Use the IJEquipAttrDescriptor interface to set the equipment component properties read only.

```
Private Function IJEquipUserAttrMgmt_OnPreLoad(ByVal pIJDAttrs As IJDAttributes, ByVal
CollAllDisplayedValues As Object) As String
    Const METHOD = "IJEquipUserAttrMgmt_OnPreLoad"
    On Error GoTo ErrorHandler

        Dim oMemberDescription As IJDMemberDescription

        Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)
        Dim oAttrCollection As Collection
        Dim oAttributeDescriptor As IJEquipAttrDescriptor
        Dim m As Long

' set dimension and deletable attributes to read only.
```

```
        Set oAttrCollection = CollAllDisplayedValues
        Select Case oMemberDescription.Name
        Case "VDrum"
           For m = 1 To oAttrCollection.Count
              Set oAttributeDescriptor = oAttrCollection.Item(m)
              Select Case UCase(oAttributeDescriptor.InterfaceName)
                 Case "IJUAVESSELDIAMETER"
                   oAttributeDescriptor.AttrState = adsReadOnly
                 Case "IJUAVESSELTANTOTAN"
                   oAttributeDescriptor.AttrState = adsReadOnly
                 Case "IJDELETABLEMEMBER"
                   oAttributeDescriptor.AttrState = adsReadOnly
                 Case Else
                   '
              End Select
           Next
        Case Else
           '
        End Select

     Set oAttrCollection = Nothing
     Set oAttributeDescriptor = Nothing
     Set oMemberDescription = Nothing

     IJEquipUserAttrMgmt_OnPreLoad = ""

     Exit Function
  ErrorHandler:
     IJEquipUserAttrMgmt_OnPreLoad = "ERROR"
     HandleError MODULE, METHOD
  End Function
```

24. Go to the end of CSP3DTank8Def Class module. Add the custom methods to manage the drum and the space objects as follows:

    **Custom Method Construct**:
    This method is in charge of the creation of the CAO member object (VDrum 01-EC). Use EquipCADHelper CreateEquipmentComponent () method to create the member given the equipment component part number. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
  Public Sub CMConstructVDrum(ByVal pMemberDescription As IJDMemberDescription, _
                              ByVal pResourceManager As IUnknown, _
                              ByRef pObject As Object)
     Const METHOD = "CMConstructVDrum"
     LogCalls METHOD
     On Error GoTo ErrorHandler

     Dim oEquipment As IJEquipment
     Set oEquipment = pMemberDescription.CAO
     GetDimensionsFromSymbolArray oEquipment

     'Create Equipment Component
```

```
    Set pObject = m_oEquipCADHelper.CreateEquipmentComponent(pMemberDescription,
pResourceManager, "VDrum 01-EC", "VDrum")
    'create name for the member
    SetObjNameRule pObject, "CPEquipmentComponent"

    Set oEquipment = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Final:**
There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructVDrum(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMFinalConstructVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom method Inputs:**
There is no need to add any code for this custom method.

```
Public Sub CMSetInputsVDrum(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMSetInputsVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom method Evaluate:**
This method is in charge of setting the attribute values to the member object.

```
Public Sub CMEvaluateVDrum(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
    Const METHOD = "CMEvaluateVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    GetDimensionsFromSymbolArray oPropertyDescription.CAO

    Dim oAttribs As IJDAttributes
    Dim oSmartOcc As IJSmartOccurrence

    Set oSmartOcc = oPropertyDescription.Object
    Set oAttribs = oSmartOcc '.ItemObject

    oAttribs.CollectionOfAttributes("IJUAVesselDiameter").Item("VesselDiameter").Value = m_VesselDiameter
```

```
   oAttribs.CollectionOfAttributes("IJUAVesselTantoTan").Item("VesselTantoTan").Value =
m_VesselTantoTan
   oAttribs.CollectionOfAttributes("IJInsulationThickness").Item("InsulationThickness").Value =
m_dInsulationThickness
   oAttribs.CollectionOfAttributes("IJDeletableMember").Item("CanBeDeleted").Value = True

' set member deletable

   Dim oMemberDescription As IJDMemberDescription
   Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(oAttribs)
   m_oEquipCADHelper.MakeMemberDeletable oMemberDescription, oAttribs, True

   Set oAttribs = Nothing
   Set oSmartOcc = Nothing
   Set oMemberDescription = Nothing

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method GeometryEvaluate:**

This method is in charge of setting the transformation matrix to move the member object relative to the equipment. Use the TransformFromECStoGCS function to maintain the VDrum member coordinate system relative to the Equipment coordinate system.

```
Public Sub CMEvaluateGeometryVDrum(ByVal oPropertyDescription As IJDPropertyDescription, pObject As
Object)
   Const METHOD = "CMEvaluateGeometryVDrum"
   On Error GoTo ErrorHandler
   LogCalls METHOD

'Add the following code to avoid the eqp component to move alone

   Dim oEqpComp As IJEquipmentComponent
   Dim oEqpCompMatrix As IJEquipment
   Dim oEquipment As IJEquipment

   Set oEqpComp = oPropertyDescription.Object
   oEqpComp.GetParent oEquipment
   Set oEqpCompMatrix = oEqpComp

   GetDimensionsFromSymbolArray oEquipment

   Dim otransform As IngrGeom3D.IJDT4x4
   Set otransform = New DT4x4
   Dim iVector As IJDVector
   Set iVector = New DVector

   otransform.LoadIdentity
   iVector.x = 0
   iVector.y = 0
   iVector.z = 0
   otransform.Translate iVector
   oEqpCompMatrix.SetMatrix otransform
```

```
    TransformFromECStoGCS oEquipment, oEqpCompMatrix

    Set oEquipment = Nothing
    Set oEqpComp = Nothing
    Set oEqpCompMatrix = Nothing

    Set iVector = Nothing
    Set otransform = Nothing

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Conditional:

This method checks if the member is conditional based on the CanBeDeleted flag. Remember, we added code in the CMEvaluate to make the member deletable.

```
Public Sub CMConditionalVDrum(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As
Boolean)
    Const METHOD = "CMConditionalVDrum"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### Custom method Release:

There is no need to add any code for this custom method

```
Public Sub CMReleaseVDrum(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMReleaseVDrum"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

### **Custom Method Construct**:

This method is in charge of the creation of the CAO IFC space object. Use *IJDSpaceFactory* to create the member. Use the *SetObjNameRule* function to get a name from the default naming rule.

```
Public Sub CMConstructVolume1(ByVal pMemberDescription As IJDMemberDescription, _
                        ByVal pResourceManager As IUnknown, _
                        ByRef pObject As Object)
    Const METHOD = "CMConstructVolume1"
```

```
  On Error GoTo ErrorHandler
  Dim oSpacePrimitiveFactory  As IJDSpacePrimitiveFactory
  Dim oSpaceNodeService    As IJSpaceNodeService
  Dim oSpaceFactory      As IJDSpaceFactory
  Dim oSpacePrimitive     As IJDSpacePrimitive
  Dim oSpaceCreation      As IJSpaceCreation
  Dim oSpacePart        As IJDPart
  Dim oSpace          As IJSpaceParent
  Dim oShapePart        As IJDPart
  Dim oShape          As IJShape
  Dim oEquipment        As IJEquipment
  Dim oParent          As Object
  Dim oCollElements      As IJElements
  Dim oSmartOccurrence     As IJSmartOccurrence
  Dim oCatalogResourceMgr   As IUnknown
  Dim oAttributes       As IJDAttributes
  Dim dblDummy()        As Double
  Dim sPartDesc         As String
  Dim i             As Integer

  'Create Volume
  Set oSmartOccurrence = pMemberDescription.CAO
  Set oEquipment = pMemberDescription.CAO

  'Identify the parent (by default the plant)
  Set oSpaceNodeService = New SpatialFunctions '****Client tier****
  Set oParent = oSpaceNodeService.GetConfigShipClass() '****Client tier****

  Dim oSpaceFolderFactory As IJDSpaceFolderFactory
  Dim oSpaceFolder  As IJDSpaceFolder
  Dim NodeTypeNone  As SpaceNodeType
  Dim strNodeName  As String
  strNodeName = "EqpVolumeFolder"

  Set oSpaceFolderFactory = New CSpaceFolderFactory
  Set oSpaceFolder = oSpaceFolderFactory.CreateEntity(SpaceFactorySystem, NodeTypeNone, strNodeName,
oParent, pResourceManager)

  'Access Catalog
  Set oCatalogResourceMgr = oSmartOccurrence.CatalogResourceMgr
  'Choose a space part
  Set oSpacePart = GetPartFromPartNumber(oCatalogResourceMgr, "SPACE_DEF_IV01",
pResourceManager)

  'Choose a shape part
  Set oShapePart = GetPartFromPartNumber(oCatalogResourceMgr, "RectangularSolid 001",
pResourceManager)

  'Create space object. Possible values for progid are: "AreaEntity.Area.1", "ZoneEntity.Zone.1",
"InterferenceVolumeEntity.InterferenceVolume.1"
  Set oSpaceFactory = New SpaceFactory
  Set oSpace = oSpaceFactory.CreateEntity("InterferenceVolumeEntity.InterferenceVolume.1", oSpaceFolder,
oSpacePart)

  Set pObject = oSpace
```

```
    Set oParent = Nothing
    Set oSpacePart = Nothing

    'Assign a name to the space object
    Set oSpaceCreation = oSpace
    SetObjNameRule oSpace, "CPInterferenceVolume"
    Set oSpace = Nothing

    Set oSpacePrimitiveFactory = New SpacePrimitiveFactory

    'Create SpacePrimitive object
    Set oSpacePrimitive = oSpacePrimitiveFactory.CreateSpacePrimitive(oShapePart, pResourceManager)
    Set oShapePart = Nothing

    If oSpacePrimitive Is Nothing Then Exit Sub

    Set oShape = oSpacePrimitive

    'Set dimension of the shape. use interface IJUARectSolid dim A along X, B along Y symetrical centered, C along Z symetrical centered
    Set oAttributes = oShape
    oAttributes.CollectionOfAttributes("IJUARectSolid").Item("A").Value = m_VesselTantoTan * 2
    oAttributes.CollectionOfAttributes("IJUARectSolid").Item("B").Value = m_VesselDiameter / 3
    oAttributes.CollectionOfAttributes("IJUARectSolid").Item("C").Value = m_VesselDiameter / 3
    Set oAttributes = Nothing

    'Set the position of the shape
    PositionAndOrientVolume1 oEquipment, oShape
    Set oShape = Nothing

    'Add spaceprimitive object to the collection
    Set oCollElements = New JObjectCollection '***Client Tier***
    oCollElements.Add oSpacePrimitive
    oSpaceCreation.SetInputs BYPRIMITIVEPROGID, 1, oCollElements, dblDummy

    Set oSpacePrimitive = Nothing
    Set oCollElements = Nothing

    'Create the relationship SpaceAssociation to allow the transform of the Space with the equipment.
    CreateSpaceOrientationRelation oSpaceCreation, oEquipment, pResourceManager

    Set oEquipment = Nothing
    Set oSpaceCreation = Nothing
    Set oSpaceNodeService = Nothing
    Set oSmartOccurrence = Nothing
    Set oCatalogResourceMgr = Nothing
    Set oParent = Nothing
    Set oSpacePart = Nothing
    Set oSpace = Nothing
    Set oShapePart = Nothing
    Set oSpaceFactory = Nothing
    Set oSpaceCreation = Nothing
    Set oSpacePrimitiveFactory = Nothing

    Set oSpaceFolderFactory = Nothing
    Set oSpaceFolder = Nothing
    Exit Sub
ErrorHandler:
    Set oEquipment = Nothing
    Set oSpaceCreation = Nothing
    Set oSpaceNodeService = Nothing
    Set oSmartOccurrence = Nothing
    Set oCatalogResourceMgr = Nothing
    Set oParent = Nothing
    Set oSpacePart = Nothing
    Set oShape = Nothing
    Set oSpace = Nothing
    Set oShapePart = Nothing
    Set oSpaceFactory = Nothing
    Set oSpaceCreation = Nothing
    Set oSpacePrimitiveFactory = Nothing
    Set oSpacePrimitive = Nothing
    Set oCollElements = Nothing
    Set oAttributes = Nothing

    Set oSpaceFolderFactory = Nothing
    Set oSpaceFolder = Nothing

    HandleError MODULE, METHOD
End Sub
```

**Custom Method Final:**
There is no need to add any code for this custom method.

```
Public Sub CMFinalConstructVolume1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMFinalConstructVolume1"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Inputs:**
There is no need to add any code for this custom method.

```
Public Sub CMSetInputsVolume1(ByVal pMemberDesc As IJDMemberDescription)
    Const METHOD = "CMSetInputsVolume1"
    On Error GoTo ErrorHandler

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method Evaluate:**
This method is in charge of setting the attribute values to the member object.

```
Public Sub CMEvaluateVolume1(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
    Const METHOD = "CMEvaluateVolume1"
    On Error GoTo ErrorHandler

    Dim oEquipment             As IJEquipment
    Dim oSpace                 As IJSpaceCreation
    Dim oVolumeShape           As IJShape
    Dim oParentCollElements    As IJElements
    Dim varDummy()             As Variant
    Dim oAttributes            As IJDAttributes

    Set oSpace = oPropertyDescription.Object
    Set oEquipment = oPropertyDescription.CAO

    If Not oSpace Is Nothing Then
        Set oParentCollElements = New IMSElements.DynElements
        oSpace.GetInputs oParentCollElements, varDummy
        Set oVolumeShape = oParentCollElements.Item(1)      'If the space is not created by Primitives, it would
fail here.

        GetDimensionsFromSymbolArray oEquipment

        Set oAttributes = oVolumeShape
        oAttributes.CollectionOfAttributes("IJUARectSolid").Item("A").Value = m_VesselTantoTan * 2
        oAttributes.CollectionOfAttributes("IJUARectSolid").Item("B").Value = m_VesselDiameter / 3
        oAttributes.CollectionOfAttributes("IJUARectSolid").Item("C").Value = m_VesselDiameter / 3
    End If


    Set oSpace = Nothing
    Set oVolumeShape = Nothing
    Set oEquipment = Nothing
    Set oParentCollElements = Nothing
    Set oAttributes = Nothing

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom Method GeometryEvaluate:**

This method is in charge of maintaining the space object relative to the equipment by calling the Evaluate custom method.

```
Public Sub CMEvaluateGeometryVolume1(ByVal oPropertyDescription As IJDPropertyDescription, pObject
As Object)
    Const METHOD = "CMEvaluateGeometryVolume1"
    LogCalls METHOD
    On Error GoTo ErrorHandler

    Call CMEvaluateVolume1(oPropertyDescription, pObject)

    Exit Sub

ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

**Custom method Conditional:**
There is no need to add any code for this custom method

```
Public Sub CMConditionalVolume1(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As
Boolean)
   Const METHOD = "CMConditionalVolume1"
   On Error GoTo ErrorHandler


   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

**Custom method Release:**
There is no need to add any code for this custom method

```
Public Sub CMReleaseVolume1(ByVal pMemberDesc As IJDMemberDescription)
   Const METHOD = "CMReleaseVolume1"
   On Error GoTo ErrorHandler

   Exit Sub
ErrorHandler:
   HandleError MODULE, METHOD
End Sub
```

25. Add the following subroutine to get the part from the catalog given the part number. Use the NamingContextObject object to get the object moniker of the part in the catalog. Then, use the Persistent Object Manager interface (POM) to retrieve the proxy (part occurrence) from the model.

```
Private Function GetPartFromPartNumber(ByVal CatalogResourceMgr As Object, ByVal PartNumber As
String, ByVal ModelResourceMgr As Object) As Object
   Const METHOD = "GetPartFromPartNumber"
   On Error GoTo ErrHandler

   Dim oPart As IJDPart
   Dim oModelPOM As IJDPOM
   Dim oNamingCntxObject As IJDNamingContextObject

   Set oNamingCntxObject = New NamingContextObject

   If Trim(PartNumber) <> vbNullString Then
      Set oPart = oNamingCntxObject.ObjectMoniker(CatalogResourceMgr, PartNumber)

      ' Get the proxy of the part in model
      Set oModelPOM = ModelResourceMgr
      Set GetPartFromPartNumber = oModelPOM.GetProxy(oPart, True)
   End If

   Set oPart = Nothing
   Set oModelPOM = Nothing
   Set oNamingCntxObject = Nothing
```

```
    Exit Function

ErrHandler:
    Set oPart = Nothing
    Set oModelPOM = Nothing
    Set oNamingCntxObject = Nothing
    HandleError MODULE, METHOD
End Function
```

26. Add the following subroutine to define the position and orientation of the volume object with respect to the equipment.

```
Private Sub PositionAndOrientVolume1(Equipment As IJEquipment, Shape As Object)
    Dim oVolumeShapeLCS As IJLocalCoordinateSystem
    Dim oAttributes As IJDAttributes

    'We want:
    'the X (primary) of the shape on the Z or Elevation of the equipment (ECS)
    'the Y (secondary) of the shape on the Y or North of the equipment (ECS)
    Set oVolumeShapeLCS = Shape
    oVolumeShapeLCS.XAxis = m_oElevation
    oVolumeShapeLCS.YAxis = m_oNorth

    Set oAttributes = Shape
    Shape.SetOrigin 0, -m_VesselDiameter / 2 - m_VesselDiameter / 3, 0

    'The shape is persisted in GCS (actually we transform the space)
    TransformFromECStoGCS Equipment, Shape

    Set oVolumeShapeLCS = Nothing
    Set oAttributes = Nothing
End Sub
```

27. Add the following subroutine to associate the space object with the equipment (graphic entity). Use *SpaceAssociationAEFactory* object to create the Active Entity and then associates the space with the Active Entity.

```
Private Sub CreateSpaceOrientationRelation(ByRef Space As Object, ByRef Equipment As Object,
ResourceManager As Object)
    Const METHOD = "CreateSpaceOrientationRelation"
    On Error GoTo ErrorHandler

    Dim oCoordinateSys          As IJLocalCoordinateSystem
    Dim oSpaceAssocFactory      As SpaceAssociationAEFactory
    Dim oSpaceAssocAE           As SpaceAssociationAE
    Dim oDT4x4                  As IJDT4x4

    Set oCoordinateSys = Equipment

    Set oSpaceAssocFactory = New SpaceAssociationAEFactory
    'Get or create the AE
    Set oSpaceAssocAE = oSpaceAssocFactory.ActiveEntity(ResourceManager, oCoordinateSys)

    oSpaceAssocAE.ObjectMatrix = BuildMatrix(oCoordinateSys)
    oSpaceAssocAE.AddAssociatedSpace Space

    'Clear everything
    Set oSpaceAssocAE = Nothing
    Set oSpaceAssocFactory = Nothing
    Set oCoordinateSys = Nothing
    Set oDT4x4 = Nothing


    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

28. Add the following subroutine to build the object matrix.

```
Private Function BuildMatrix(oLCS As IJLocalCoordinateSystem) As IJDT4x4
    Const METHOD = "BuildMatrix"
    On Error GoTo ErrorHandler

    Dim oXVect                As IJDVector
    Dim oYVect                As IJDVector
    Dim oZVect                As IJDVector
    Dim dblMat(0 To 15)           As Double
    Dim oPosition              As IJDPosition
    Dim dX                  As Double
    Dim dY                  As Double
    Dim dZ                  As Double
    Dim oCS                  As IJLocalCoordinateSystem
    Dim oDT4x4                 As IJDT4x4
    Dim index                As Integer

    Set oCS = oLCS

    'Get X,Y,Z vectors from the coordinate system
    Set oXVect = oCS.XAxis
    Set oYVect = oCS.YAxis
    Set oZVect = oCS.ZAxis

    'Get the position from the Coordinate system
    Set oPosition = oCS.Position

    For index = 1 To 14
        dblMat(index) = 0#
    Next index

    dblMat(0) = 1#
    dblMat(5) = 1#
    dblMat(10) = 1#
    dblMat(15) = 1#

    'Build the matrix using X,Y,Z vectors and the position
    oXVect.Get dblMat(0), dblMat(1), dblMat(2)
    oYVect.Get dblMat(4), dblMat(5), dblMat(6)
    oZVect.Get dblMat(8), dblMat(9), dblMat(10)
    oPosition.Get dX, dY, dZ

    dblMat(12) = dX
    dblMat(13) = dY
    dblMat(14) = dZ

    Set oDT4x4 = New DT4x4

    'Load the matrix as identity matrix
    oDT4x4.LoadIdentity

    oDT4x4.Set dblMat(0)
    Set BuildMatrix = oDT4x4

    Set oDT4x4 = Nothing
    Set oXVect = Nothing
    Set oYVect = Nothing
    Set oZVect = Nothing
    Set oPosition = Nothing
    Set oCS = Nothing

    Exit Function
ErrorHandler:
    HandleError MODULE, METHOD
End Function
```

29. Add the following subroutine to convert the array of inputs in a set of global variables.

```
Private Sub GetDimensionsFromSymbolArray(SmartOccurrence As IJSmartOccurrence)
    Const METHOD = "GetDimensionsFromSymbolArray"
    On Error GoTo ErrorHandler

    m_avSymbolArrayOfInputs = m_oEquipCADHelper.GetSymbolArrayOfInputs(SmartOccurrence)

    'Inputs,  from equipment symbol code
    'Set m_oPartFclt = m_avSymbolArrayOfInputs(1)
    m_VesselDiameter = m_avSymbolArrayOfInputs(2)
    m_VesselTantoTan = m_avSymbolArrayOfInputs(3)
    m_dInsulationThickness = m_avSymbolArrayOfInputs(4)

    Exit Sub
ErrorHandler:
    HandleError MODULE, METHOD
End Sub
```

30. Add the following subroutine to position and orient the member with respect to the equipment.

```
Private Sub TransformFromECStoGCS(Equipment As IJEquipment, Object As Object)
    Const METHOD = "TransformFromECStoGCS"
    LogCalls METHOD
    On Error GoTo ErrorHandler
    Dim oEqpMatrix As IJDT4x4
    Dim oShapeMatrix As IJDT4x4
    Dim otransform As IJDGeometry
    Dim oShape As IJShape

    If Not Object Is Nothing Then
        If TypeOf Object Is IJDGeometry Then
            Equipment.GetMatrix oEqpMatrix
            Set otransform = Object
            otransform.DTransform oEqpMatrix
            Set otransform = Nothing
            Set oEqpMatrix = Nothing
        End If
    End If

    Set otransform = Nothing
    Set oEqpMatrix = Nothing
    Set oShape = Nothing
    Set oShapeMatrix = Nothing

    Exit Sub
ErrorHandler:
    Set otransform = Nothing
    Set oEqpMatrix = Nothing
    Set oShape = Nothing
    Set oShapeMatrix = Nothing
    HandleError MODULE, METHOD
End Sub
```

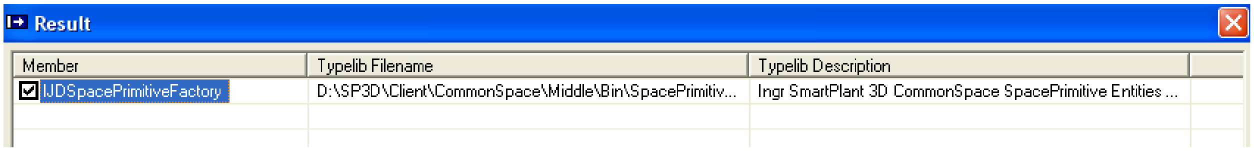
31. Add the following subroutine to set the naming relation and generate a name based on the default naming rule.

```
Public Sub SetObjNameRule(ByRef obj As Object, ByRef CLASSNAME As String)
' Apply the namerule using the IJDNamingRulesHelper helper interface
Const METHOD = "SetNameRule"
On Error GoTo ErrorHandler

    Dim NameRule As String
    Dim NamingRules As IJElements

    Dim oNameRuleHlpr As GSCADNameRuleSemantics.IJDNamingRulesHelper

'Returns a collection of the naming rules available in the catalog database
'for the given object
    Set oNameRuleHlpr = New GSCADNameRuleHlpr.NamingRulesHelper
    Call oNameRuleHlpr.GetEntityNamingRulesGivenName(CLASSNAME, NamingRules)
'get the first namerule from the collection
    Dim oNameRuleHolder As GSCADGenericNamingRulesFacelets.IJDNameRuleHolder
    Set oNameRuleHolder = NamingRules.Item(1)
'Create relations "NamedEntity" and "EntityNamingRule" and obj
    Dim oNameRuleAE As GSCADGenNameRuleAE.IJNameRuleAE
    Call oNameRuleHlpr.AddNamingRelations(obj, oNameRuleHolder, oNameRuleAE)

    GoTo CleanObjects
ErrorHandler:
    HandleError MODULE, METHOD

CleanObjects:
    Set oNameRuleHlpr = Nothing
    Set oNameRuleHolder = Nothing
    Set oNameRuleAE = Nothing
    Set NamingRules = Nothing
End Sub
```

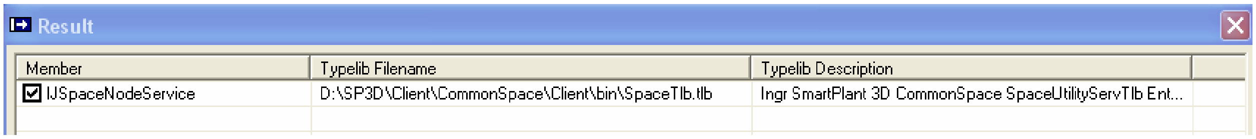
32. Add the following subroutine to log any error.

```
Private Sub LogCalls(sMethod As String)

    If Not m_oEditErrors Is Nothing Then
        m_oEditErrors.Add 5000, m_oEquipCADHelper.ProjectName & "." &
m_oEquipCADHelper.CLASSNAME, "Entering " & sMethod
    End If

End Sub
```

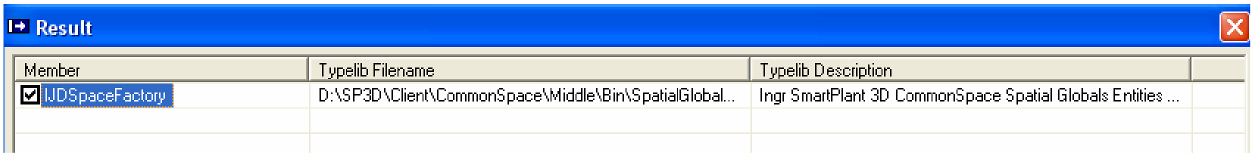
33. Compile the Visual Basic project and save the dll as SP3DTank8Asm.dll in the c:\Train\lab8
34. Note: Use the SP3D Reference Tool to attach the missing reference libraries.



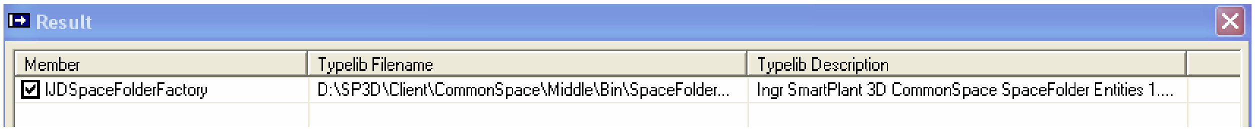
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDSpacePrimitiveFactory | D:\SP3D\Client\CommonSpace\Middle\Bin\SpacePrimitiv... | Ingr SmartPlant 3D CommonSpace SpacePrimitive Entities ... |



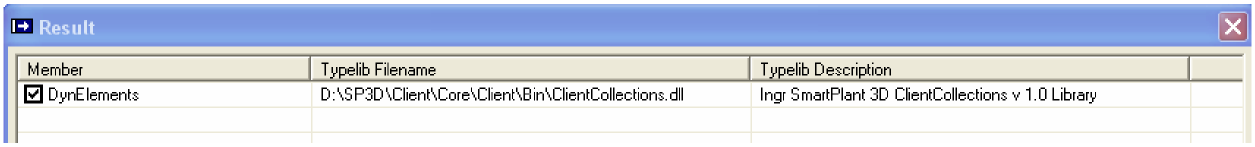
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJSpaceNodeService | D:\SP3D\Client\CommonSpace\Client\bin\SpaceTlb.tlb | Ingr SmartPlant 3D CommonSpace SpaceUtilityServTlb Ent... |



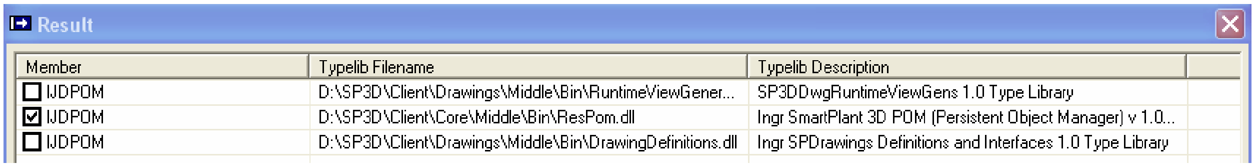
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDSpaceFactory | D:\SP3D\Client\CommonSpace\Middle\Bin\SpatialGlobal... | Ingr SmartPlant 3D CommonSpace Spatial Globals Entities ... |



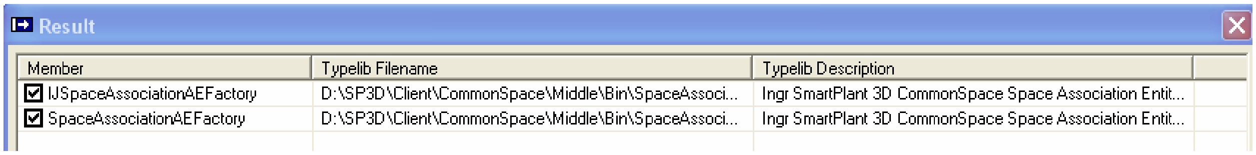
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDSpaceFolderFactory | D:\SP3D\Client\CommonSpace\Middle\Bin\SpaceFolder... | Ingr SmartPlant 3D CommonSpace SpaceFolder Entities 1.... |



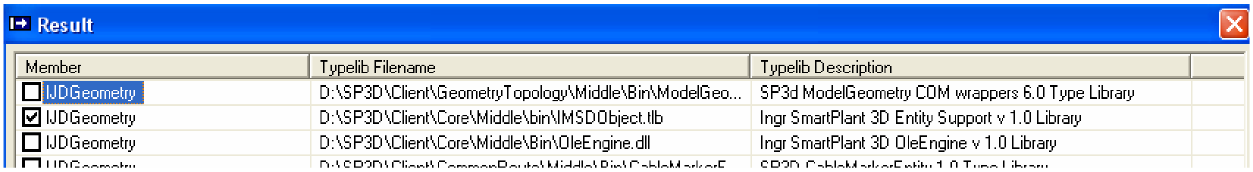
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ DynElements | D:\SP3D\Client\Core\Client\Bin\ClientCollections.dll | Ingr SmartPlant 3D ClientCollections v 1.0 Library |



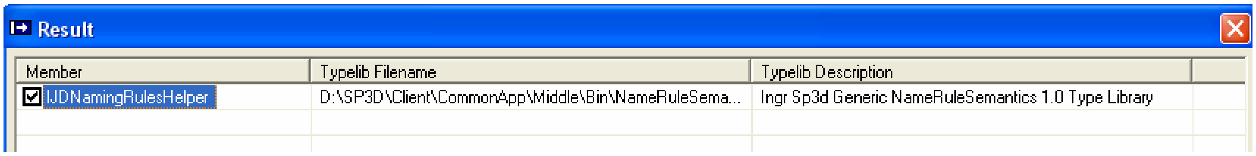
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☐ IJDPOM | D:\SP3D\Client\Drawings\Middle\Bin\RuntimeViewGener... | SP3DDwgRuntimeViewGens 1.0 Type Library |
| ☑ IJDPOM | D:\SP3D\Client\Core\Middle\Bin\ResPom.dll | Ingr SmartPlant 3D POM (Persistent Object Manager) v 1.0... |
| ☐ IJDPOM | D:\SP3D\Client\Drawings\Middle\Bin\DrawingDefinitions.dll | Ingr SPDrawings Definitions and Interfaces 1.0 Type Library |


Result

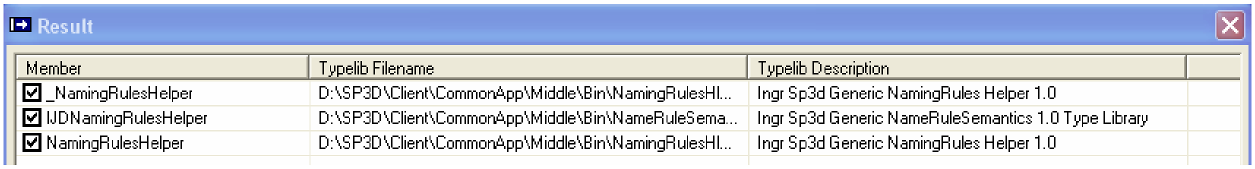
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJSpaceAssociationAEFactory | D:\SP3D\Client\CommonSpace\Middle\Bin\SpaceAssoci... | Ingr SmartPlant 3D CommonSpace Space Association Entit... |
| ☑ SpaceAssociationAEFactory | D:\SP3D\Client\CommonSpace\Middle\Bin\SpaceAssoci... | Ingr SmartPlant 3D CommonSpace Space Association Entit... |


Result

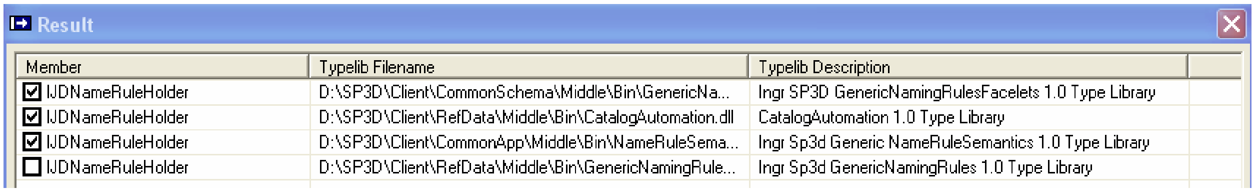
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☐ IJDGeometry | D:\SP3D\Client\GeometryTopology\Middle\Bin\ModelGeo... | SP3d ModelGeometry COM wrappers 6.0 Type Library |
| ☑ IJDGeometry | D:\SP3D\Client\Core\Middle\bin\IMSDObject.tlb | Ingr SmartPlant 3D Entity Support v 1.0 Library |
| ☐ IJDGeometry | D:\SP3D\Client\Core\Middle\Bin\OleEngine.dll | Ingr SmartPlant 3D OleEngine v 1.0 Library |
| ☐ IJDGeometry | D:\SP3D\Client\CommonRoute\Middle\Bin\CableMarkerE... | SP3D CableMarkerEntity 1.0 Type Library |


Result

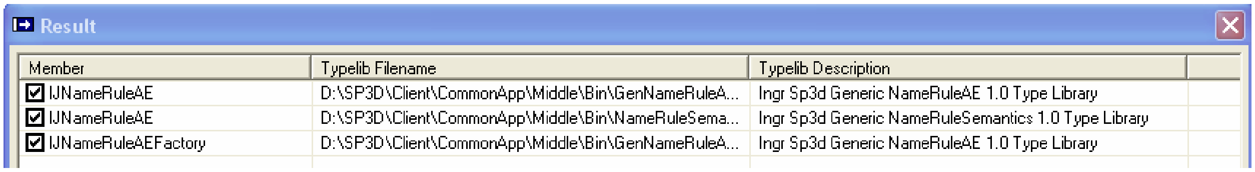
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDNamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |


Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ _NamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NamingRulesHl... | Ingr Sp3d Generic NamingRules Helper 1.0 |
| ☑ IJDNamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☑ NamingRulesHelper | D:\SP3D\Client\CommonApp\Middle\Bin\NamingRulesHl... | Ingr Sp3d Generic NamingRules Helper 1.0 |


Result

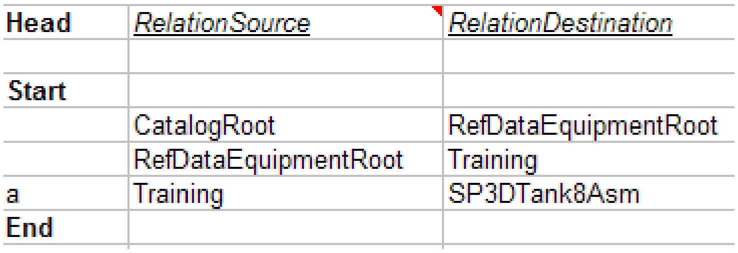
| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDNameRuleHolder | D:\SP3D\Client\CommonSchema\Middle\Bin\GenericNa... | Ingr SP3D GenericNamingRulesFacelets 1.0 Type Library |
| ☑ IJDNameRuleHolder | D:\SP3D\Client\RefData\Middle\Bin\CatalogAutomation.dll | CatalogAutomation 1.0 Type Library |
| ☑ IJDNameRuleHolder | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☐ IJDNameRuleHolder | D:\SP3D\Client\RefData\Middle\Bin\GenericNamingRule... | Ingr Sp3d GenericNamingRules 1.0 Type Library |


Result

| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJNameRuleAE | D:\SP3D\Client\CommonApp\Middle\Bin\GenNameRuleA... | Ingr Sp3d Generic NameRuleAE 1.0 Type Library |
| ☑ IJNameRuleAE | D:\SP3D\Client\CommonApp\Middle\Bin\NameRuleSema... | Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library |
| ☑ IJNameRuleAEFactory | D:\SP3D\Client\CommonApp\Middle\Bin\GenNameRuleA... | Ingr Sp3d Generic NameRuleAE 1.0 Type Library |

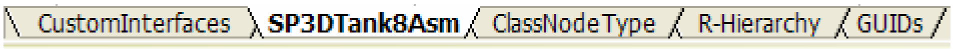
One of the most important steps in Visual Basic programming is to preserve the binary compatibility of your program. Save the final version of your dll file to be binary compatibility in order to preserve the CLSID.

35. Save the Visual Basic SP3DTank8Asm project.
36. Open the SP3DTemplate.xls workbook. Go the R-Hierarchy sheet and add the following entry.



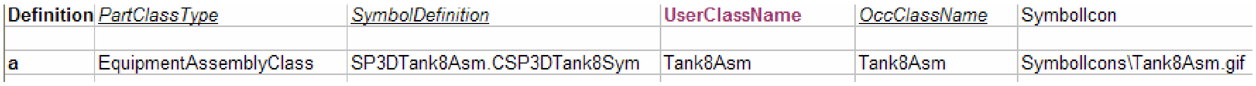
| Head | RelationSource | RelationDestination |
|---|---|---|
| | | |
| Start | | |
| | CatalogRoot | RefDataEquipmentRoot |
| | RefDataEquipmentRoot | Training |
| a | Training | SP3DTank8Asm |
| End | | |

37. Go to the SP3DTemplateAsm sheet and rename it as SP3DTank8Asm.



38. Go to the Class definition section and add/edit as follows:

39. In the Definition Section rows:

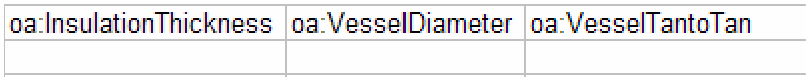


| Definition | PartClassType | SymbolDefinition | UserClassName | OccClassName | SymbolIcon |
|---|---|---|---|---|---|
| | | | | | |
| a | EquipmentAssemblyClass | SP3DTank8Asm.CSP3DTank8Sym | Tank8Asm | Tank8Asm | SymbolIcons\Tank8Asm.gif |

Note:

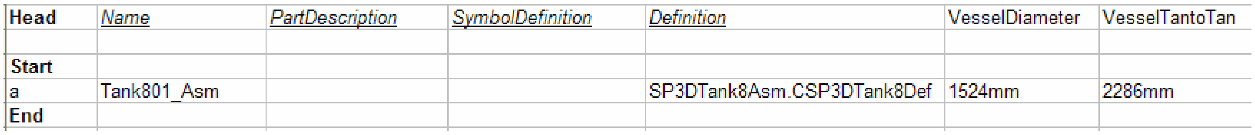
- You can use Microsoft Paint to create the file and save it under your \\machine\symbols\SymbolIcons

Occurrence attributes:



| oa:InsulationThickness | oa:VesselDiameter | oa:VesselTantoTan |
|---|---|---|
| | | |

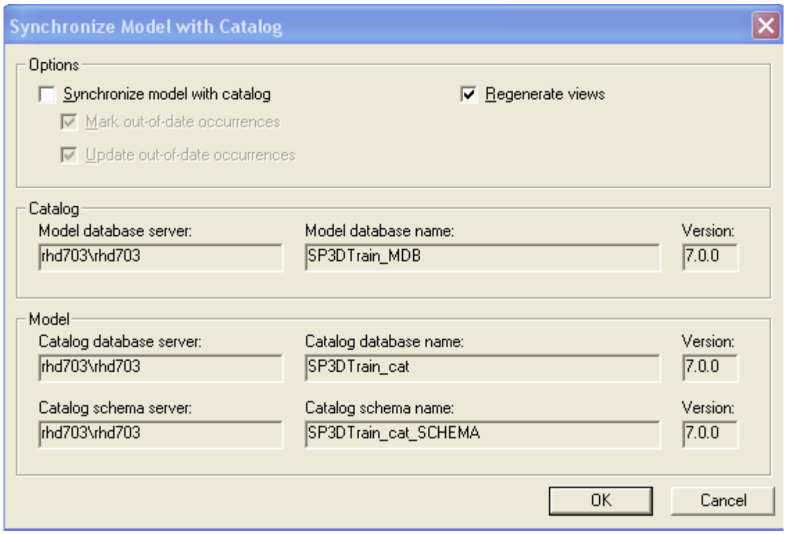
40. In the Part Section rows:



| Head | Name | PartDescription | SymbolDefinition | Definition | VesselDiameter | VesselTantoTan |
|---|---|---|---|---|---|---|
| | | | | | | |
| Start | | | | | | |
| a | Tank801_Asm | | | SP3DTank8Asm.CSP3DTank8Def | 1524mm | 2286mm |
| End | | | | | | |

40. Save the Excel workbook as SP3DTank8Asm.xls in the c:\Train\lab8.
41. Optional steps: Create the Tank8Asm.gif file and place it under \\<MachineName>\Symbols\SymbolIcons
42. Load the information into the catalog using the Add/Modify/Delete Mode. Once the bulkload process is completed, review the log file.
43. Run the Project Management Task. Select the Model in the hierarchy.
44. Select Tools -> Synchronize Model with the Catalog.
45. Uncheck the Synchronize Model with the Catalog option.

***Note****: You just need to update the views in the model.*


Synchronize Model with Catalog

Options

☐ Synchronize model with catalog

☑ Mark out-of-date occurrences

☑ Update out-of-date occurrences

☑ Regenerate views

Catalog

Model database server: rhd703\rhd703

Model database name: SP3DTrain_MDB

Version: 7.0.0

Model

Catalog database server: rhd703\rhd703

Catalog database name: SP3DTrain_cat

Version: 7.0.0

Catalog schema server: rhd703\rhd703

Catalog schema name: SP3DTrain_cat_SCHEMA

Version: 7.0.0

46. Hit “OK” Button.
47. Once the process is completed, Right click the training plant icon and select “Regenerate the Reports database” option to re-create the views in the report database.
48. Go to the Equipment Task and place the SP3DTank8Asm.

# Appendix

## Symbol Helper Reference

The Symbol Helper Reference provides documentation for symbol math functions and properties.

### IJSymbolHelper

This interface provides methods to help in creating the definition of a VB symbol. It provides the implementation of the IJDUserSymbolServices interface as well as provides support for declaring the inputs and outputs of the symbol. Call this interface when you want to:

- Instantiate a symbol definition in a datastore.
- Update an existing symbol definition.
- Compute the symbol using a function.
- Edit the symbol occurrence.

### Methods

| AddInputDef(Count As Integer, Name As String, Description As String, DefaultValue As Double) | |
|---|---|
| Description: | Adds the input definition to the collection of inputs defined for the symbol |
| Parameters: | |
| [in] count | Index for the input parameter |
| [in] Name | Name of the input parameter |
| [in] Description | Description of the input parameter |
| [in] DefaultValue | Default value for the input parameter |

| AddOutputDef(Count As Integer, Name As String, Description As String, aspect as integer) | |
|---|---|
| Description: | Adds the output definition to the collection of outputs defined for the symbol |
| Parameters: | |
| [in] count | Index for the output parameter |
| [in] Name | Name of the output parameter |
| [in] Description | Description of the output parameter |
| [in] aspect | Aspect number for the output |

| AddAspectDef (Count As Integer, Name As String, Description As String, aspect as integer) | |
|---|---|
| Description: | Adds the aspect definition to the symbol |
| Parameters: | |
| [in] count | Index for the aspect |
| [in] Name | Name of the aspect |
| [in] Description | Description of the aspect |
| [in] aspect | Aspect number for the output |

| InstanciateDefinition (ByVal CodeBase As String, ByVal defParameters As Variant, ByVal ActiveConnection As Object) | |
|---|---|
| Description: | This method will create a symbol definition entity and initialize it. It will also set the progid and the code base values on the definition. It will take the same set of parameters as the method on the interface 'IJDUserSymbolServices'. |
| Parameters: | |
| [in] CodeBase | Specifies the URL (or UNC) of the .cab file that can provides the dll associated to the symbol definition object (ActiveX control packaging). |
| [in] defParameters | Definition parameters. |
| [in] ActiveConnection | Resource manager to which the symbol definition will be connected |

| InitializeSymbolDefinition(ByRef pSymbolDefinition As IJDSymbolDefinition) | |
|---|---|
| Description: | This method will define the inputs for the symbol definition, define the required number of representations and add the outputs defined to the correct representation. The input collection as well as the output collection can be made a 'VARIABLECOLLECTION' if required. |
| Parameters: | |
| pSymbolDefinition | Symbol definition passed by reference that will be initialized in this method. |

| InvokeRepresentation(ByVal sblOcc As Object, ByVal repName As String, ByVal outputcoll As Object, ByRef arrayOfInputs()) | |
|---|---|
| Description: | This method will create the object that contains the implementation details for the required representation. The wizard follows a specific convention like so:<br>ProjectName.<RepresentationName>.<br>So the helper function can obtain the progid given this rule and create the object and then call the method 'Run' on the IDispatch interface of this object. This method will also take all the parameters in addition to an array of strings that contain the names of outputs belonging to that representation. |
| Parameters: | |
| [in] sblOcc | Symbol occurrence that calls the method. |
| [in] repName | Name of the representation requested on the symbol. |
| [in] outputcoll | Collection object to which the generated outputs will be attached. |
| [in] arrayOfInputs | A safearray of inputs defined as VARIANT. |

## Properties

| NumInputs as Integer | |
|---|---|
| Description: | Number of inputs for the symbol |
| Modifiability: | Read/write |

| NumOutputs as Integer | |
|---|---|
| Description: | Number of outputs for the symbol. |
| Modifiability: | Read/write |

| NumAspects as Integer | |
|---|---|
| Description: | Number of aspects defined for the symbol |
| Modifiability: | Read/write |

| ProjectName as String | |
|---|---|
| Description: | Project Name for the symbol |
| Modifiability: | Read/write |

| ClassName as String | |
|---|---|
| Description: | Class name for the symbol |
| Modifiability: | Read/write |

## IJSymbolGeometryHelper

This interface provides methods to help in creating simple geometric primitives like Cylinder (given center, radius and length), Cone (given the 4 points), Sphere (center and radius), Torus (center, major radius, minor radius). The other geometric primitives are not yet implemented.

### Methods

| AddGeometry(Output As String, Aspect As Long, Geometry As Object) | |
|---|---|
| Description: | Adds the Geometry Object to the Output Collection. |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] Aspect | Required long value |
| [in] Geometry | Required Object Geometry |

| CreateChildPartOcc(Output As String, ChildPart As Object, Position As IJDPosition, VecX As IJDVector, VecY As IJDVector, VecZ As IJDVector) As Object | |
|---|---|
| Description: | |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] ChildPart | Required Object ChildPart |
| [in] Position | Required IJDPosition Position |
| [in] VecX | Required IJDVector VecX |
| [in] VecY | Required IJDVector VecY |
| [in] VecZ | Required IJDVector VecZ |

| CreateCone( Output As String, PosStart As IJDPosition, PosEnd As IJDPosition, diameterStart As Double, diameterEnd As Double, Optional Offset As Double = 0#) As Object | |
|---|---|
| Description: | Creates the Cone Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] PosStart | Required IJDPosition Start |
| [in] PosEnd | Required IJDPosition End |
| [in] diameterStart | Required double value |
| [in] diameterEnd | Required double value |
| [in, defaultvalue(0)] Offset | Optional double value – is an optional parameter |

| CreateCylinder( Output As String, PosStart As IJDPosition, PosEnd As IJDPosition, Diameter As Double) As Object | |
|---|---|
| Description: | Creates the Cylinder Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] PosStart | Required IJDPosition Start |
| [in] PosEnd | Required IJDPosition End |
| [in] Diameter | Required double value – diameter of the Cylinder |

| CreateMiteredTorus( Output As String, Origin As IJDPosition, NormalAxis As IJDVector, MajorAxis As IJDVector, Radius As Double, Angle As Double, Diameter As Double, NumberOfCuts As Long) As Object | |
|---|---|
| Description: | Creates the CreateMiteredTorus Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] Origin | Required IJDPosition Origin |
| [in] NormalAxis | Required IJDVector NormalAxis |
| [in] MajorAxis | Required IJDVector MajorAxis |
| [in] Radius | Required double value |
| [in] Angle | Required double value |
| [in] Diameter | Required double value |
| [in] NumberOfCuts | Required long value |

| CreatePolygon( Output As String, NumberOfSides As Long, SideLength As Double, Depth As Double, Object As Object) | |
|---|---|
| Description: | Creates the CreatePolygon Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] NumberOfSides | Required long value |
| [in] SideLength | Required double value |
| [in] Depth | Required double value |

| CreatePrism( Output As String, Width As Double, Depth As Double, Length As Double, Width2 As Double, Depth2 As Double, Optional Offset As Double = 0#) As Object | |
|---|---|
| Description: | Creates the CreatePrism Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] Width | Required double value |
| [in] Depth | Required double value |
| [in] Length | Required double value |
| [in] Width2 | Required double value |
| [in] Depth2, | Required double value |
| [in, defaultvalue(0)] Offset | Optional double value |

| CreateProjectedRectangle( Output As String, PosStart As IJDPosition, PosEnd As IJDPosition, Axis As IJDVector, Width As Double, Depth As Double) As Object | |
|---|---|
| Description: | Creates the CreateProjectedRectangle Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] PosStart | Required IJDPosition Start |
| [in] PosEnd | Required IJDPosition End |
| [in] Axis | Required IJDVector Axis |

| | |
|---|---|
| [in] Width | Required double value |
| [in] Depth | Required double value |

| CreateProjectedShape( Output As String, Length As Double, Curve As Object) As Object | |
|---|---|
| Description: | Creates the CreateProjectedShape Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] Length | Required double value |
| [in] Curve | Required object curve |

| CreateProjectedShapeByPoints( Output As String, NumberOfPoints As Long, Length As Double, Points As IJElements) As Object | |
|---|---|
| Description: | Creates the CreateProjectedShapeByPoints Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] NumberOfPoints | Required long value |
| [in] Length | Required double value |
| [in] Points | Required point objects as IJElements collection |

| CreateProjectedTriangle( Output As String, PosStart As IJDPosition, PosEnd As IJDPosition, Axis As IJDVector, Width As Double, Depth As Double) As Object | |
|---|---|
| Description: | Creates the CreateProjectedTriangle Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] PosStart | Required IJDPosition start |
| [in] PosEnd | Required IJDPostion end |
| [in] Axis | Required IJDVector Axis |
| [in] Width | Required double value |
| [in] Depth | Required double value |

| CreateRectangularTorus( Output As String, Radius As Double, SweepAngle As Double, Width As Double, Depth As Double) As Object | |
|---|---|
| Description: | Creates the CreateRectangularTorus Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] Radius | Required double value |
| [in] SweepAngle | Required double value |
| [in] Width | Required double value |
| [in] Depth | Required double value |

| CreateSemiEllipsoid( Output As String, Origin As IJDPosition, NormalAxis As IJDVector, MajorAxis As IJDVector, AxisDiameter As Double, MinorAxisRadius As Long) As Object | |
|---|---|
| Description: | Creates the CreateSemiEllipsoid Object and adds it to the output collection |
| Parameters: | |

| | |
|---|---|
| [in] Output | Required Output as string |
| [in] Origin | Required IJDPosition Origin |
| [in] NormalAxis | Required IJDVector NormalAxis |
| [in] MajorAxis | Required IJDVector MajorAxis |
| [in] AxisDiameter | Required double value |
| [in] MinorAxisRadius | Required long value |

| CreateSphere( Output As String, Origin As IJDPosition, Radius As Double) As Object | |
|---|---|
| Description: | Creates the CreateSphere Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] Origin | Required IJDPosition Origin |
| [in] Radius | Required double value |

| CreateTorus( Output As String, Origin As IJDPosition, NormalAxis As IJDVector, MajorAxis As IJDVector, Radius As Double, Angle As Double, Diameter As Double) As Object | |
|---|---|
| Description: | Creates the CreateTorus Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] Origin | Required IJDPosition Origin |
| [in] NormalAxis | Required IJDVector NormalAxis |
| [in] MajorAxis | Required IJDVector MajorAxis |
| [in] Radius | Required double value |
| [in] Angle | Required double value |
| [in] Diameter | Required double value |

| CreateTransitionalElement( Output As String, Width As Double, Depth As Double, Length As Double, Radius As Double, Offset As Double) As Object | |
|---|---|
| Description: | Creates the CreateTransitionalElement Object and adds it to the output collection |
| Parameters: | |
| [in] Output | Required Output as string |
| [in] Width | Required double value |
| [in] Depth | Required double value |
| [in] Length | Required double value |
| [in] Radius | Required double value |
| [in] Offset | Required double value |

## Properties

| AutoTransformUpdate() As Boolean | |
|---|---|
| Description: | Adding or getting the AutoTransformUpdate boolean value |
| Modifiability: | Read/write |

| OutputCollection() As IJDOutputCollection | |
|---|---|
| Description: | Adding or getting created output objects in the output collection |

| | |
|---|---|
| Modifiability: | Read/write |

| Transform() As IJDT4x4 | |
|---|---|
| Description: | Adding or getting the transformation matrix IJDT4x4 |
| Modifiability: | Read/write |

# Geometry Factory Programming Reference

The Geometry Factory Programming Reference provides documentation of Geom3d.dll, which includes the objects, methods, and properties for the geometry factory.

Description

The GeometryFactory object is the class factory for the creation of geometry entities. The factory implements properties that return "collection-like" interfaces for each of the geometry types. These interfaces have creation methods that the application programmer can use to create, initialize, and optionally specify a persistent database connection for the object.

If the objects are created with a NULL database connection, the object is created as a "transient." Transient objects can be displayed and added to the highlight system, but they do not participate in transactions or relationships.

### IJGeometryFactory

Use this interface when you want to create transient or persistent geometry objects

### Properties

| Points3d ( ) as IPoints3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IPoints3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| Lines3d ( ) as ILines3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the ILines3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| Arcs3d ( ) as IArcs3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IArcs3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| Circles3d ( ) as ICircles3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the ICircles3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| Ellipses3d ( ) as IEllipses3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IEllipses3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| EllipticalArcs3d ( ) as IEllipticalArcs3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IEllipticalArcs3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| LineStrings3d ( ) as ILineStrings3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the ILineStrings3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| BSplineCurves3d ( ) as IBSplineCurves3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IBSplineCurves3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| ComplexStrings3d ( ) as IComplexStrings3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IComplexStrings3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| Planes3d ( ) as IPlanes3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IPlanes3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| Cones3d ( ) as ICones3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the ICones3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| Projections3d ( ) as IProjections3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IProjections3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| Revolutions3d ( ) as IRevolutions3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IRevolutions3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| RuledSurfaces3d ( ) as IRuledSurfaces3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IRuledSurfaces3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| Spheres3d ( ) as ISpheres3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the ISpheres3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| Tori3d ( ) as ITori3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the ITori3d interface of the first element in the collection. |
| Modifiability: | Read Only |

| BSplineSurfaces3d ( ) as IBSplineSurfaces3d | |
|---|---|
| Description: | Returns a pointer (pVal) to the IBSplineSurfaces3d interface of the first element in the collection. |
| Modifiability: | Read Only |

## Methods:

### CreateBSplineSurfaceByParametersWCaps Method

**Description**

The CreateBSplineSurfaceByParametersWCaps method creates and returns a BSplineSurface3d object based on a desired order, a set of poles, and optional caps. Weights and knots are optional and are set to NULL, or an empty array. The output will be the surface, then the caps.

If the order is equal to the number of poles, the curve evolves into the control polygon of a Bezier curve.

B-spline weights can be considered a gravitational type force with the magnitude of the weight equal to the pulling force. The weights are always normalized. If no weights are present, the curve is considered to be non-rational and may be NULL. Non-rational curves have weights with a value of 1.

The B-spline knots define the parameterization of the curve, and they may be periodic. Knots, also known as knot vectors, must be monotonic and strictly increasing. Monotonic refers to the successive terms as non-decreasing or non-increasing.

The Order property determines the relative accuracy of the poles with regard to the points that are entered to create the curve. The order returned evaluates as a polynomial degree plus one. For example, an order of 4 defines cubic. Since it is more efficient to use even-order b-spline curves, the number of poles (and knots) are maximized by increasing the order to the next even number.

**Syntax**

object.CreateBSplineSurfaceByParametersWCaps*(pConnection, uNumPoles, vNumPoles, Poles, Weights, uOrder, vOrder, uKnots, vKnots, uPeriodic, vPeriodic, ReverseNor, Solid, WCaps, numCaps)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| uNumPoles | long | Required. This argument is the number of poles in the u-direction. The type is long. |
| vNumPoles | long | Required. This argument is the number of poles in the v-direction. The type is long. |
| Poles | double | Required. This argument is a SAFEARRAY of poles. The type is double. |
| Weights | double | Required. This argument is a SAFEARRAY of weights. The type is double. |
| uOrder | long | Required. This argument is the order in the u-direction. The type is long. |
| vOrder | long | Required. This argument is the order in the v-direction. The type is long. |
| uKnots | double | Required. This argument is a SAFEARRAY of knots. The type is double. |
| vKnots | double | Required. This argument is a SAFEARRAY of Knots. The type is double. |
| uPeriodic | Boolean | Required. This argument is a Boolean flag that specifies whether the surface is periodic in u. |
| vPeriodic | Boolean | Required. This argument is a Boolean flag that specifies whether or not the surface is periodic in v. |
| ReverseNor | Boolean | Required. This argument specifies the outward normal. It is False when the outward normal is U X V. It is True when the outward normal is U (curve) cross V (proj vector). The type is Boolean. |
| Solid | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object is solid. Possible values are: 0 - Set normal as hollow; 1 - Set normal as solid; 2 - Set normal according to input point; 3 - Just toggle the outward normal (no checks). |
| WCaps | Boolean | Required. This argument specifies whether or not the object has caps. If the value is False, the surface does not have caps; if the value is True, the surface has caps. |
| numCaps | Int | Required. This argument is the number of caps. The type is integer. |

### CreateBy2Points Method

**Description**
The CreateBy2Points method creates and returns a Line3d object defined by two points.

**Syntax**
object.CreateBy2Points*(pConnection, StartX, StartY, StartZ, EndX, EndY, EndZ)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| StartX | double | Required. This argument is the X-coordinate for the starting point. The type is double. |
| StartY | double | Required. This argument is the Y-coordinate for the starting point. The type is double. |
| StartZ | double | Required. This argument is the Z-coordinate for the starting point. The type is double. |
| EndX | double | Required. This argument is the X-coordinate for the ending point. The type is double. |
| EndY | double | Required. This argument is the Y-coordinate for the ending point. The type is double. |
| EndZ | double | Required. This argument is the Z-coordinate for the ending point. The type is double. |

### CreateBy3Points Method (IArcs3d)

**Description**
The CreateBy3Points method creates and returns an Arc3d object given three non-colinear points along the arc.

**Syntax**
object.CreateBy3Points*(pConnection, StartX, StartY, StartZ, AlongX, AlongY, AlongZ, EndX, EndY, EndZ)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| StartX | double | Required. This argument is the X-coordinate for the starting point on the arc. The type is double. |
| StartY | double | Required. This argument is the Y-coordinate for the starting point on the arc. The type is double. |
| StartZ | double | Required. This argument is the Z-coordinate for the starting point on the arc. The type is double. |
| AlongX | double | Required. This argument is the X-coordinate for the middle point on the arc. The type is double. |
| AlongY | double | Required. This argument is the Y-coordinate for the middle point on the arc. The type is double. |
| AlongZ | double | Required. This argument is the Z-coordinate for the middle point on the arc. The type is double. |
| EndX | double | Required. This argument is the X-coordinate for the ending point on the arc. The type is double. |
| EndY | double | Required. This argument is the Y-coordinate for the ending point on the arc. The type is double. |
| EndZ | double | Required. This argument is the Z-coordinate for the ending point on the arc. The type is double. |

### CreateBy3Points Method (ICircles3d)

**Description**
The CreateBy3Points method creates and returns a pointer (ppObj) to the IJCircle interface of a Circle3d object. This method uses three inscribed non-colinear points to create the circle.

**Syntax**
object.CreateBy3Points*(pConnection, X1, Y1, Z1, X2, Y2, Z2, X3, Y3, Z3)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| X1 | double | Required. This argument is the first X-coordinate value. The type is double. |
| Y1 | double | Required. This argument is the first Y-coordinate value. The type is double. |
| Z1 | double | Required. This argument is the first Z-coordinate value. The type is double. |
| X2 | double | Required. This argument is the second X-coordinate value. The type is double. |
| Y2 | double | Required. This argument is the second Y-coordinate value. The type is double. |
| Z2 | double | Required. This argument is the second Z-coordinate value. The type is double. |
| X3 | double | Required. This argument is the third X-coordinate value. The type is double. |
| Y3 | double | Required. This argument is the third Y-coordinate value. The type is double. |
| Z3 | double | Required. This argument is the third Z-coordinate value. The type is double. |

### CreateBy4Pts Method

**Description**
The CreateBy4Pts method creates and returns a pointer (ppObj) to the IJCone interface of a full bounded Cone3d. This method takes as input a base center point, a top center point, a base starting point, and a top starting point. The axis runs through the top center point and base center point, and the cone follows the right-hand rule about the axis.
The base ellipse must not be degenerate, so the base center point cannot be the same as the base starting point.
To create a point cone, set the top center point to the top starting point.

**Syntax**
object.CreateBy4Pts*(pConnection, CenterBx, CenterBy, CenterBz, CenterTx, CenterTy, CenterTz, StartBx, StartBy, StartBz, StartTx, StartTy, StartTz, Solid)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CenterBx | double | Required. This argument is the X-coordinate of the base center point. The type is double. |
| CenterBy | double | Required. This argument is the Y-coordinate of the base center point. The type is double. |
| CenterBz | double | Required. This argument is the Z-coordinate of the base center point. The type is double. |
| CenterTx | double | Required. This argument is the X-coordinate of the top center point. The type is double. |
| CenterTy | double | Required. This argument is the Y-coordinate of the top center point. The type is double. |

| | | |
|---|---|---|
| CenterTz | double | Required. This argument is the Z-coordinate of the top center point. The type is double. |
| StartBx | double | Required. This argument is the X-coordinate of the base starting point. The type is double. |
| StartBy | double | Required. This argument is the Y-coordinate of the base starting point. The type is double. |
| StartBz | double | Required. This argument is the Z-coordinate of the base starting point. The type is double. |
| StartTx | double | Required. This argument is the X-coordinate of the top starting point. The type is double. |
| StartTy | double | Required. This argument is the Y-coordinate of the top starting point. The type is double. |
| StartTz | double | Required. This argument is the Z-coordinate of the top starting point. The type is double. |
| Solid | Boolean | Required. This argument is a Boolean flag indicating whether the cone is solid or not. |

**CreateByAxisMajorMinorRadius Method**

**Description**
The CreateByAxisMajorMinor method creates and returns a pointer (ppObj) to the IJTorus interface of a Torus3d object. This method defines a torus by a point on the axis at the center of the torus, an axis vector, a vector toward the center of a minor circle (determining the origin of UV space), a major radius, and a minor radius. Set major radius = -major radius if the center of the torus is on the left-hand side of the axis, indicating the torus is a lemon shape.

**Syntax**
object.CreateByAxisMajorMinorRadius*(pConnection, AxisCenterX, AxisCenterY, AxisCenterZ, AxisVecX, AxisVecY, AxisVecZ, OriginDirX, OriginDirY, OriginDirZ, MajorRadius, MinorRadius, Solid)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| AxisCenterX | double | Required. This argument is the X-coordinate of the point on the center axis. The type is double. |
| AxisCenterY | double | Required. This argument is the Y-coordinate of the point on the center axis. The type is double. |
| AxisCenterZ | double | Required. This argument is the Z-coordinate of the point on the center axis. The type is double. |
| AxisVecX | double | Required. This argument is the X-coordinate of a point along the axis vector. The type is double. |
| AxisVecY | double | Required. This argument is the Y-coordinate of a point along the axis vector. The type is double. |
| AxisVecZ | double | Required. This argument is the Z-coordinate of a point along the axis vector. The type is double. |
| OriginDirX | double | Required. This argument is the X-coordinate of a point along the vector toward the center of the minor circle. The type is double. |
| OriginDirY | double | Required. This argument is the Y-coordinate of a point along the vector toward the center of the minor circle. The type is double. |
| OriginDirZ | double | Required. This argument is the Z-coordinate of a point along the vector toward the center of the minor circle. The type is double. |
| MajorRadius | double | Required. This argument is the length of the major radius. The type is double. |
| MinorRadius | double | Required. This argument is the length of the minor radius. The type is double. |
| Solid | Boolean | Required. This argument is a Boolean flag indicating whether or not the torus is solid. |

### CreateByAxisMajorMinorRadiusSweep Method

**Description**

The CreateByAxisMajorMinorRadiusSweep method creates and returns a pointer (ppObj) to the IJTorus interface of a Torus3d object. This method defines a partial torus by a point on the axis at the center of the torus, an axis vector, a vector toward the center of the minor circle (determining the origin of UV space), a major radius, a minor radius, and a sweep angle. Set the major radius = -major radius if the center of the torus is on the left-hand side of the axis, indicating the torus is a lemon shape.

**Syntax**

object.CreateByAxisMajorMinorRadiusSweep*(pConnection, AxisCenterX, AxisCenterY, AxisCenterZ, AxisVecX, AxisVecY, AxisVecZ, OriginDirX, OriginDirY, OriginDirZ, MajorRadius, MinorRadius, SwAngle, Solid)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| AxisCenterX | double | Required. This argument is the X-coordinate of a point on the center axis.The type is double. |
| AxisCenterY | double | Required. This argument is the Y-coordinate of a point on the center axis.The type is double. |
| AxisCenterZ | double | Required. This argument is the Z-coordinate of a point on the center axis.The type is double. |
| AxisVecX | double | Required. This argument is the X-coordinate of a point along the axis vector. The type is double. |
| AxisVecY | double | Required. This argument is the Y-coordinate of a point along the axis vector. The type is double. |
| AxisVecZ | double | Required. This argument is the Z-coordinate of a point along the axis vector. The type is double. |
| OriginDirX | double | Required. This argument is the X-coordinate of a point along the vector toward the center of the minor circle. The type is double. |
| OriginDirY | double | Required. This argument is the Y-coordinate of a point along the vector toward the center of the minor circle. The type is double. |
| OriginDirZ | double | Required. This argument is the Z-coordinate of a point along the vector toward the center of the minor circle. The type is double. |
| MajorRadius | double | Required. This argument is the length of the major radius. The type is double. |
| MinorRadius | double | Required. This argument is the length of the minor radius. The type is double. |
| SwAngle | double | Required. This argument is the sweep angle in radians. The type is double. |
| Solid | Boolean | Required. This argument is a Boolean flag indicating whether or not the torus is a solid. |

### CreateByCenterAxisRadEnds Method

**Description**

The CreateByCenterAxisRadEnds method creates and returns a pointer (ppObj) to the IJCone interface of a bounded partial Cone3d. This method takes as input a base center point, axis, base starting point, base ending point, and a top radius.

The cone follows the right-hand rule about the axis.

The axis vector must contain the height of the cylinder.

The base ellipse must not be degenerate, so the base center point cannot be the same as the base starting point.

To create a point cone, set the top radius length to zero.

**Syntax**

object.CreateByCenterAxisRadEnds*(pConnection, CenterBx, CenterBy, CenterBz, AxisVx, AxisVy, AxisVz, RadiusT, StartBx, StartBy, StartBz, EndBx, EndBy, EndBz, Solid)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CenterBx | double | Required. This argument is the X-coordinate of the base center point. The type is double. |
| CenterBy | double | Required. This argument is the Y-coordinate of the base center point. The type is double. |
| CenterBz | double | Required. This argument is the Z-coordinate of the base center point. The type is double. |
| AxisVx | double | Required. This argument is the X-coordinate of a point on the axis vector. The type is double. |
| AxisVy | double | Required. This argument is the Y-coordinate of a point on the axis vector. The type is double. |
| AxisVz | double | Required. This argument is the Z-coordinate of a point on the axis vector. The type is double. |
| RadiusT | double | Required. This argument is the top radius value. The type is double. |
| StartBx | double | Required. This argument is the X-coordinate of the base starting point. The type is double. |
| StartBy | double | Required. This argument is the Y-coordinate of the base starting point. The type is double. |
| StartBz | double | Required. This argument is the Z-coordinate of the base starting point. The type is double. |
| EndBx | double | Required. This argument is the X-coordinate of the base ending point. The type is double. |
| EndBy | double | Required. This argument is the Y-coordinate of the base ending point. The type is double. |
| EndBz | double | Required. This argument is the Z-coordinate of the base ending point. The type is double. |
| Solid | Boolean | Required. This argument is a Boolean flag indicating whether the cone is solid or not. |

### CreateByCenterNormalMajAxisRatioAngle Method

**Description**
The CreateByCenterNormalMajAxisRatioAngle method creates and returns an EllipticalArc3d object given a center point, normal axis, major axis containing length, minor/major ratio, start angle, and sweep angle (angles in radians).

**Syntax**
object.CreateByCenterNormalMajAxisRatioAngle*(pConnection, CenterX, CenterY, CenterZ, NormalX, NormalY, NormalZ, MajorX, MajorY, MajorZ, MMRatio, StartAngle, SwAngle)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CenterX | double | Required. This argument is the X-coordinate of the center point. The type is double. |
| CenterY | double | Required. This argument is the Y-coordinate of the center point. The type is double. |
| CenterZ | double | Required. This argument is the Z-coordinate of the center point. The type is double. |
| NormalX | double | Required. This argument is the X-coordinate of a point on the normal vector. The type is double. |
| NormalY | double | Required. This argument is the Y-coordinate of a point on the normal vector. The type is double. |
| NormalZ | double | Required. This argument is the Z-coordinate of a point on the normal vector. The type is double. |

| | | |
|---|---|---|
| MajorX | double | Required. This argument is the X-coordinate of a point on the major axis vector. The type is double. |
| MajorY | double | Required. This argument is the Y-coordinate of a point on the major axis vector. The type is double. |
| MajorZ | double | Required. This argument is the Z-coordinate of a point on the major axis vector. The type is double. |
| MMRatio | double | Required. This argument is the minor axis to major axis ratio. The type is double. |
| StartAngle | double | Required. This argument is the start angle in radians. The type is double. |
| SwAngle | double | Required. This argument is the sweep angle in radians. The type is double. |

**CreateByCenterNormalRadius Method**

**Description**
The CreateByCenterNormalRadius method creates and returns a pointer (ppObj) to an IJCircle interface of a Circle3d object, given the center, normal unit vector, and radius.

**Syntax**
object.CreateByCenterNormalRadius*(pConnection, CenterX, CenterY, CenterZ, NormalX, NormalY, NormalZ, Radius)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CenterX | double | Required. This argument is the X-coordinate of the center of the circle. The type is double. |
| CenterY | double | Required. This argument is the Y-coordinate of the center of the circle. The type is double. |
| CenterZ | double | Required. This argument is the Z-coordinate of the center of the circle. The type is double. |
| NormalX | double | Required. This argument is the X-coordinate of a point on the normal vector. The type is double. |
| NormalY | double | Required. This argument is the Y-coordinate of a point on the normal vector. The type is double. |
| NormalZ | double | Required. This argument is the Z-coordinate of a point on the normal vector. The type is double. |
| Radius | double | Required. This argument is the radius of the circle. The type is double. |

**CreateByCenterNormMajAxisRatio Method**

**Description**
The CreateByCenterNormMajAxisRatio method creates and returns a pointer (ppObj) to the IJEllipse interface of an Ellipse3d object, given a center point, normal axis, major axis containing length, and minor/major ratio.

**Syntax**
object.CreateByCenterNormMajAxisRatio*(pConnection, CenterX, CenterY, CenterZ, NormalX, NormalY, NormalZ, MajorX, MajorY, MajorZ, MMRatio)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CenterX | double | Required. This argument is the X-coordinate of the center point. The type is double. |
| CenterY | double | Required. This argument is the Y-coordinate of the center point. The type is double. |

| | | |
|---|---|---|
| CenterZ | double | Required. This argument is the Z-coordinate of the center point. The type is double. |
| NormalX | double | Required. This argument is the X-coordinate of a point on the normal vector. The type is double. |
| NormalY | double | Required. This argument is the Y-coordinate of a point on the normal vector. The type is double. |
| NormalZ | double | Required. This argument is the Z-coordinate of a point on the normal vector. The type is double. |
| MajorX | double | Required. This argument is the X-coordinate of a point on the major axis vector. The type is double. |
| MajorY | double | Required. This argument is the Y-coordinate of a point on the major axis vector. The type is double. |
| MajorZ | double | Required. This argument is the Z-coordinate of a point on the major axis vector. The type is double. |
| MMRatio | double | Required. This argument is the minor axis to major axis ratio. The type is double. |

### CreateByCenterRadius Method

**Description**
The CreateByCenterRadius method creates and returns a pointer (ppObj) to the IJSphere interface of a Sphere3d object, based on a center point and a radius.

**Syntax**
object.CreateByCenterRadius*(pConnection, CenterX, CenterY, CenterZ, Radius, Solid)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CenterX | double | Required. This argument is the X-coordinate of the center point. The type is double. |
| CenterY | double | Required. This argument is the Y-coordinate of the center point. The type is double. |
| CenterZ | double | Required. This argument is the Z-coordinate of the center point. The type is double. |
| Radius | double | Required. This argument is the length of the radius. The type is double. |
| Solid | Boolean | Required. This argument is a Boolean flag indicating whether or not the sphere is solid. |

### CreateByCenterStartEnd Method

**Description**
The CreateByCenterStartEnd method creates an Arc3d object according to the specified inputs.
The center and start coordinates define the radius. A non-colinear ending point defines the sweep angle and plane (this returns an arc between 0 and P1).

**Syntax**
object.CreateByCenterStartEnd*(pConnection, CenterX, CenterY, CenterZ, StartX, StartY, StartZ, EndX, EndY, EndZ)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CenterX | double | Required. This argument is the X-coordinate for the center point on the arc. The type is double. |
| CenterY | double | Required. This argument is the Y-coordinate for the center point on the arc. The type is double. |

| | | |
|---|---|---|
| CenterZ | double | Required. This argument is the Z-coordinate for the center point on the arc. The type is double. |
| StartX | double | Required. This argument is the X-coordinate for the starting point on the arc. The type is double. |
| StartY | double | Required. This argument is the Y-coordinate for the starting point on the arc. The type is double. |
| StartZ | double | Required. This argument is the X-coordinate for the starting point on the arc. The type is double. |
| EndX | double | Required. This argument is the X-coordinate for the ending point on the arc. The type is double. |
| EndY | double | Required. This argument is the Y-coordinate for the ending point on the arc. The type is double. |
| EndZ | double | Required. This argument is the Z-coordinate for the ending point on the arc. The type is double. |

### CreateByComplexString Method

**Description**
The CreateByComplexString method creates and returns a pointer (ppObject) to the interface of a BSplineCurve3d object. This method works by converting an input complex string.

**Syntax**
object.CreateByComplexString*(pConnection, pCS)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| pCS | IJComplexString | Required. This argument is a pointer to IJComplexString. |

### CreateByCtrNormStartEnd Method

**Description**
The CreateByCtrNormStartEnd method creates and returns an Arc3d object given the center, normal vector, start and end points, radius, and direction.

**Syntax**
object.CreateByCtrNormStartEnd*(pConnection, CenterX, CenterY, CenterZ, NormalX, NormalY, NormalZ, StartX, StartY, StartZ, EndX, EndY, EndZ)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CenterX | double | Required. This argument is the X-coordinate for the center point of the arc. The type is double. |
| CenterY | double | Required. This argument is the Y-coordinate for the center point of the arc. The type is double. |
| CenterZ | double | Required. This argument is the Z-coordinate for the center point of the arc. The type is double. |
| NormalX | double | Required. This argument is the X-coordinate for a point on the normal vector. The type is double. |
| NormalY | double | Required. This argument is the Y-coordinate for a point on the normal vector. The type is double. |
| NormalZ | double | Required. This argument is the Z-coordinate for a point on the normal vector. The type is double. |
| StartX | double | Required. This argument is the X-coordinate for the starting point on the arc.The type is double. |
| StartY | double | Required. This argument is the Y-coordinate for the starting point on the arc. The type is double. |

|  |  |  |
|---|---|---|
| StartZ | double | Required. This argument is the Z-coordinate for the starting point on the arc. The type is double. |
| EndX | double | Required. This argument is the X-coordinate for the ending point on the arc. The type is double. |
| EndY | double | Required. This argument is the Y-coordinate for the ending point on the arc. The type is double. |
| EndZ | double | Required. This argument is the Z-coordinate for the ending point on the arc. The type is double. |

### CreateByCurve Method (IProjections3d)

**Description**
The CreateByCurve method creates and returns a pointer (ppObj) to the IJProjection interface of a Projection3d object based on a planar curve, direction, and length.

**Syntax**
object.CreateByCurve*(pConnection, CurveObject, uvX, uvY, uvZ, Length, Capped)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CurveObject | Object | Required. This argument is the IDispatch interface of the planar curve. |
| uvX | double | Required. This argument is the X-coordinate of the point along the curve in the plane. The type is double. |
| uvY | double | Required. This argument is the Y-coordinate of the point along the curve in the plane. The type is double. |
| uvZ | double | Required. This argument is the Z-coordinate of the point along the curve in the plane. The type is double. |
| Length | double | Required. This argument is the length of the projection in the direction of the point. The type is double. |
| Capped | Boolean | Required. This argument is a Boolean flag indicating whether or not the object is capped. |

### CreateByCurve Method (IRevolutions3d)

**Description**
The CreateByCurve method creates and returns a pointer (ppObj) to the IJRevolution interface of a Revolution3d object based on a curve to revolve, an axis vector, and a point on the axis.

**Syntax**
object.CreateByCurve*(pConnection, CurveObject, AxisX, AxisY, AxisZ, CenterX, CenterY, CenterZ, SwAngle, Capped)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CurveObject | Object | Required. This argument is the IDispatch interface of the planar curve. |
| AxisX | double | Required. This argument is the X-coordinate of a point on the axis vector. The type is double. |
| AxisY | double | Required. This argument is the Y-coordinate of a point on the axis vector. The type is double. |
| AxisZ | double | Required. This argument is the Z-coordinate of a point on the axis vector. The type is double. |

| | | |
|---|---|---|
| CenterX | double | Required. This argument is the X-coordinate of the center point on the axis. The type is double. |
| CenterY | double | Required. This argument is the Y-coordinate of the center point on the axis. The type is double. |
| CenterZ | double | Required. This argument is the Z-coordinate of the center point on the axis. The type is double. |
| SwAngle | double | Required. This argument is the sweep angle in radians. The type is double. |
| Capped | Boolean | Required. This argument is a Boolean flag indicating whether or not the object is capped. If capped, then the result is either a closed planar curve revolved partially or an open planar curve revolved fully. |

### CreateByCurves Method (IComplexStrings3d)

**Description**
The CreateByCurves method creates and returns a pointer (ppObj) to the IJComplexString interface of a ComplexString3d object. The input to this method is an array of Curves. Allowable open curve types include Line3d, Arc3d, EllipticalArc3d, LineString3d, ComplexString3d, and BsplineCurve3d.

**Syntax**
object.CreateByCurves*(pConnection, pIJCurveElements)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| pIJCurveElements | IJElements | Required. This argument is a pointer to the first element in an array of Curves. |

### CreateByCurves Method (IRuledSurfaces3d)

**Description**
The CreateByCurves method creates and returns a pointer (ppObj) to the IJRuled interface of a RuledSurface3d object based on a base curve and a top curve.

**Syntax**
object.CreateByCurves*(pConnection, CurveObjectBase, CurveObjectTop, Capped)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CurveObjectBase | Object | Required. This argument is the IDispatch interface of the base planar curve. |
| CurveObjectTop | Object | Required. This argument is the IDispatch interface of the top planar curve. The type is double. |
| Capped | Boolean | Required. This argument is a Boolean flag indicating whether or not the object is capped. If capped, then the result is either two closed planar curves or one degenerate and the other closed and planar. |

### CreateByFitCurve Method

**Description**
The CreateByFitCurve method creates and returns a pointer (ppObj) to the interface of a BSplineCurve3d object. This method works by direct fitting a set of points.
The start and end tangent constraints are optional. These constraints should be set to 0.0 if they are not needed.
The Order property determines the relative accuracy of the poles with regard to the points that are entered to create the curve. The order returned evaluates as a polynomial degree plus one. For example, an order of 4 defines cubic.

Since it is more efficient to use even-order b-spline curves, the number of poles (and knots) are maximized by increasing the order to the next even number.

**Syntax**
object.CreateByFitCurve*(pConnection, Order, PointCount, Points, Start_vX, Start_vY, Start_vZ, End_vX, End_vY, End_vZ, Closed, periodic)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| Order | long | Required. This argument is the order of the curve. The type is long. |
| PointCount | long | Required. This argument is the number of points along the curve. The type is long. |
| Points | double | Required. This argument is a SAFEARRAY of points along the curve. The type is double. |
| Start_vX | double | Required. This argument is the X-coordinate for the starting point of the curve. The type is double. |
| Start_vY | double | Required. This argument is the Y-coordinate for the starting point of the curve. The type is double. |
| Start_vZ | double | Required. This argument is the Z-coordinate for the starting point of the curve. The type is double. |
| End_vX | double | Required. This argument is the X-coordinate for the ending point of the curve. The type is double. |
| End_vY | double | Required. This argument is the Y-coordinate for the ending point of the curve. The type is double. |
| End_vZ | double | Required. This argument is the Z-coordinate for the ending point of the curve. The type is double. |
| Closed | Boolean | Required. This argument is a Boolean flag that specifies whether or not the curve is closed. |
| periodic | Boolean | Required. This argument is a Boolean flag that specifies whether or not the curve is periodic. |

**CreateByFitSurface Method**

**Description**
The CreateByFitSurface method creates and returns a pointer (ppObj) to an interface for a BSplineSurface3d object. This method does a direct fit of a B-spline surface through a set of points. The points are ordered (as surface poles are) in the u-direction by v-direction.

**Syntax**
object.CreateByFitSurface*(pConnection, vNumPoints, uNumPoints, Points, uOrder, vOrder, uClosedForm, vClosedForm)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| vNumPoints | long | Required. This argument is a SAFEARRAY of the v-number of points along the surface. The type is double. |
| uNumPoints | double | Required. This argument is a SAFEARRAY of the u-number of points along the surface. The type is double. |
| Points | double | Required. This argument is a SAFEARRAY of points along the surface. The type is double. |
| uOrder | long | Required. This argument is the u order of the surface, which must be greater than 1. The type is long. |

| | | |
|---|---|---|
| vOrder | long | Required. This argument is the v-order of the surface, which must be greater than 1. The type is long. |
| uClosedForm | long | Required. This argument specifies the smoothness at the start and end of a closed B-spline surface in the u-direction. The type is long. If 0: no smoothness requirements, 1: closed with tangent continuity (no tangents input) (this value is not currently supported), 2: closed and periodic. |
| vClosedForm | long | Required. This argument specifies the smoothness at the start and end of a closed B-spline surface in the v-direction. The type is long. If 0: no smoothness requirements, 1: closed with tangent continuity (no tangents input) (this value is not currently supported), 2: closed and periodic. |

### CreateByLeastSquareFitCurve Method

**Description**
The CreateByLeastSquareFitCurve method creates and returns a pointer (ppObj) to the interface of a BSplineCurve3d object. This method fits a set of points using least squares.
The start and end tangent constraints are optional. You should set these constraints to 0.0 if they are not needed.

**Syntax**
object.CreateByLeastSquareFitCurve*(pConnection, Order, PointCount, Points, Start_vX, Start_vY, Start_vZ, End_vX, End_vY, End_vZ, Closed, periodic, opt, nseg, tol)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| Order | long | Required. This argument specifies the order of the curve. The type is long. |
| PointCount | long | Required. This argument is the number of points along the curve. The type is long. |
| Points | double | Required. This argument is a SAFEARRAY of points along the curve. The type is double. |
| Start_vX | double | Required. This argument is the X-coordinate for the starting point of the curve. The type is double. |
| Start_vY | double | Required. This argument is the Y-coordinate for the starting point of the curve. The type is double. |
| Start_vZ | double | Required. This argument is the Z-coordinate for the starting point of the curve. The type is double. |
| End_vX | double | Required. This argument is the X-coordinate for the ending point of the curve. The type is double. |
| End_vY | double | Required. This argument is the Y-coordinate for the ending point of the curve. The type is double. |
| End_vZ | double | Required. This argument is the Z-coordinate for the ending point of the curve. The type is double. |
| Closed | Boolean | Required. This argument is a Boolean flag that specifies whether or not the curve is closed. |
| periodic | Boolean | Required. This argument is a Boolean flag that specifies whether or not the curve is periodic. |
| opt | Boolean | Required. This argument is an option that specifies the fit of the curve. Its type is Boolean. If this option is 0, it means fit within the given tolerance; if it is 1, it means fit with the given number of segments. |
| nseg | long | Required. This argument is the number of segments used in the fitting, if opt=1. The type is long. |
| tol | double | Required. This argument is the tolerance used in the fitting, if opt = 0. The type is double. |

### CreateByLeastSquareFitSurface Method

**Description**
The CreateByLeastSquareFitSurface method creates and returns a pointer (ppObj) to an interface for a a BSplineSurface3d object. This method does a least square fit of a B-spline surface through a set of points. The points are ordered (as surface poles are) in the u-direction by v-direction.

**Syntax**
object.CreateByLeastSquareFitSurface*(pConnection, vNumPoints, uNumPoints, Points, uOrder, vOrder, uPeriodic, vPeriodic, uNseg, vNseg)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| vNumPoints | long | Required. This argument is a SAFEARRAY of the v-number of points along the surface. The type is double. |
| uNumPoints | double | Required. This argument is a SAFEARRAY of the u-number of points along the surface. The type is double. |
| Points | double | Required. This argument is a SAFEARRAY of points along the surface. The type is double. |
| uOrder | long | Required. This argument is the u-order of the surface, which must be greater than 1. The type is long. |
| vOrder | long | Required. This argument is the v-order of the surface, which must be greater than 1. The type is long. |
| uPeriodic | Boolean | Required. This argument is a Boolean flag that specifies whether or not the surface is periodic in u. |
| vPeriodic | Boolean | Required. This argument is a Boolean flag that specifies whether the surface is periodic in v. |
| uNseg | long | Required. This argument is the number of segments in u. The type is long. |
| vNseg | long | Required. This argument is the number of segments in v. The type is long. |

### CreateByOffset Method

**Description**
The CreateByOffset method creates and returns an offset curve.

**Syntax**
object.CreateByOffset*(pConnection, Obj, DPtx, DPty, DPtz, OffsetDist, code)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| Obj | Object | Required. This argument is the curve to offset. The type is Object. |
| DPtx | double | Required. This argument is the vector component in the X-direction. The type is double. |
| DPty | double | Required. This argument is the vector component in the Y-direction. The type is double. |
| DPtz | double | Required. This argument is the vector component in the Z-direction. The type is double. |
| OffsetDist | double | Required. This argument is the distance for the offset. The type is double. |

| | | |
|---|---|---|
| code | Int | Required. This argument is an integer that describes the offset curve. Possible values are: 0 - extend; 1 - fillet. |

**CreateByOuterBdry Method**

**Description**

The CreateByOuterBdry method creates and returns a pointer (ppObj) to the IJPlane interface of an infinite Plane3d object, based on a point and a normal.

**Syntax**

object.CreateByOuterBdry*(pConnection, CurveObject)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CurveObject | Object | Required. This argument is the IDispatch interface of the planar curve. |

**CreateByParameters Method (IBSplineCurves3d)**

**Description**

The CreateByParameters method creates and returns a pointer (ppObj) to the interface of a BSplineCurve3d object. This method uses order, poles, weights, and knots. The weights and knots are optional; they should be set to NULL if not needed.

If the order is equal to the number of poles, the curve evolves into the control polygon of a Bezier curve.

B-spline weights can be considered a gravitational type force with the magnitude of the weight equal to the pulling force. The weights are always normalized. If no weights are present, the curve is considered to be non-rational and may be NULL. Non-rational curves have weights with a value of 1.

The B-spline knots define the parameterization of the curve, and they may be periodic. Knots, also known as knot vectors, must be monotonic and strictly increasing. Monotonic refers to the successive terms as non-decreasing or non-increasing.

The Order property determines the relative accuracy of the poles with regard to the points that are entered to create the curve. The order returned evaluates as a polynomial degree plus one. For example, an order of 4 defines cubic. Since it is more efficient to use even-order b-spline curves, the number of poles (and knots) are maximized by increasing the order to the next even number.

**Syntax**

object.CreateByParameters*(pConnection, Order, PoleCount, Poles, Weights, Knots, periodic)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| Order | long | Required. This argument specifies the order of the curve. The type is long. |
| PoleCount | long | Required. This argument is the number of poles. The type is long. |
| Poles | double | Required. This argument is a SAFEARRAY of poles. The type is double. |
| Weights | double | Required. This argument is a SAFEARRAY of weights. The type is double. |
| Knots | double | Required. This argument is a SAFEARRAY of knots. The type is double. Generally, this value is the number of poles plus the order value. |
| periodic | Boolean | Required. This argument is a Boolean flag that specifies whether or not the curve is periodic. |

**CreateByParameters Method (IBSplineSurfaces3d)**

**Description**
The CreateByParameters method creates and returns a pointer (ppObj) to an interface for a BSplineSurface3d object based on the desired order and a set of poles (weights and knots are optional).
If periodic knots are passed in, but periodic is set to False, the knots will be converted to multiple end knots.
The outward normal is generally U cross V, but if the reverse normal is desired, set ReverseNor to True.
The poles are ordered in the u-direction by v-direction. Weights and knots are optional. The number of poles (u or v) must be greater than or equal to the order in that direction.

**Syntax**
object.CreateByParameters*(pConnection, uNumPoles, vNumPoles, Poles, Weights, uOrder, vOrder, uKnots, vKnots, uPeriodic, vPeriodic, ReverseNor)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| uNumPoles | long | Required. This argument is the number of poles in the u-direction. The type is long. |
| vNumPoles | long | Required. This argument is the number of poles in the v-direction. The type is long. |
| Poles | double | Required. This argument is a SAFEARRAY of poles. The type is double. |
| Weights | double | Required. This argument is a SAFEARRAY of weights. The type is double. |
| uOrder | long | Required. This argument is the u-order of the surface, which must be greater than 1. The type is long. |
| vOrder | long | Required. This argument is the v-order of the surface, which must be greater than 1. The type is long. |
| uKnots | double | Required. This argument is a SAFEARRAY of knots. The type is double. |
| vKnots | double | Required. This argument is a SAFEARRAY of knots. The type is double. |
| uPeriodic | Boolean | Required. This argument is a Boolean flag that specifies whether the surface is periodic in u. |
| vPeriodic | Boolean | Required. This argument is a Boolean flag that specifies whether the surface is periodic in v. |
| ReverseNor | Boolean | Required. This argument is a Boolean flag that specifies whether or not the direction of the normal is reversed. |

**CreateByPartOfCurve Method**

**Description**
The CreateByPartOfCurve method creates and returns a part of the input curve.
Note: It is possible to cross the seam.

**Syntax**
object.CreateByPartOfCurve*(pConnection, Obj, startPar, dirPar, endPar)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| Obj | Object | Required. This argument is the IDispatch interface of the top planar curve. |
| startPar | double | Required. This argument is the start of the part of the curve. |

| | | |
|---|---|---|
| dirPar | double | Required. This argument is a point as the direction of the part of the curve that is returned. |
| endPar | double | Required. This argument is the end of the part of the curve. |

### CreateByPoint Method

**Description**
The CreateByPoint method creates and returns an interface for a Point3d object, given X-, Y- and Z-coordinates.

**Syntax**
object.CreateByPoint*(pConnection, x, y, z)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| x | double | Required. This argument is the X-coordinate. The type is double. |
| y | double | Required. This argument is the Y-coordinate. The type is double. |
| z | double | Required. This argument is the Z-coordinate. The type is double. |

### CreateByPointNormal Method

**Description**
The CreateByPointNormal method creates and returns a pointer (ppObj) to the IJPlane interface of an infinite Plane3d object, based on a point and a normal.

**Syntax**
object.CreateByPointNormal*(pConnection, PointX, PointY, PointZ, NormalX, NormalY, NormalZ)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| PointX | double | Required. This argument is the X-coordinate of the point. The type is double. |
| PointY | double | Required. This argument is the Y-coordinate of the point. The type is double. |
| PointZ | double | Required. This argument is the Z-coordinate of the point. The type is double. |
| NormalX | double | Required. This argument is the X-coordinate of a point on the normal. The type is double. |
| NormalY | double | Required. This argument is the Y-coordinate of a point on the normal. The type is double. |
| NormalZ | double | Required. This argument is the Z-coordinate of a point on the normal. The type is double. |

### CreateByPoints Method

**Description**
The CreateByPoints method creates and returns a pointer (ppObj) to the interface of a LineString3d object. This method takes as input an array of points. The array is a one-dimensional array of doubles containing the X-, Y-, and Z-coordinates of the vertex points.

**Syntax**
object.CreateByPoints*(pConnection, PointCount, Points)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| PointCount | long | Required. This argument is the number of points in the array. The type is long. |
| Points | double | Required. This argument is a SAFEARRAY of points. The type is double. |

### CreateByPtVectLength Method

**Description**
The CreateByPtVectLength method creates and returns a Line3d object, given the starting point, direction vector, and length.

**Syntax**
object.CreateByPtVectLength*(pConnection, StartX, StartY, StartZ, uvX, uvY, uvZ, Length)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| StartX | double | Required. This argument is the X-coordinate for the starting point. The type is double. |
| StartY | double | Required. This argument is the Y-coordinate for the starting point. The type is double. |
| StartZ | double | Required. This argument is the Z-coordinate for the starting point. The type is double. |
| uvX | double | Required. This argument is the X-coordinate for the ending point. The type is double. |
| uvY | double | Required. This argument is the Y-coordinate for the ending point. The type is double. |
| uvZ | double | Required. This argument is the Z-coordinate for the ending point. The type is double. |
| Length | double | Required. This argument is the length of the line from the starting point. The type is double. |

### CreateBySingleSweepWCaps Method

**Description**
The CreateBySingleSweepWCaps method creates a collection of swept surfaces with the option of caps. The output is surfaces, and then caps.

**Syntax**
object.CreateBySingleSweepWCaps*(pConnection, TrObj, CsObj, cornerOpt, BrkCv, StartOpt, StNorm, EdNorm, WCaps, numCaps)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| TrObj | Object | Required. This argument is the trace curve. The type is Object. |
| CsObj | Object | Required. This object is the cross section curve or curve to sweep. It can be one curve, or it can be a plane object that contains boundary curves, where the boundary curves are each swept to make a separate surface; the first boundary of the plane is always the region, and any following boundaries are holes. The type for CsObj is Object. |
| cornerOpt | SkinningCornerOptions | Required. This argument is an option on how to handle trace curves that are line strings. If the value is 0, the method averages the left/right tangent to get the plane for |

| | | |
|---|---|---|
| | | placing the cross section. If the value is 1, the method turns around the trace cusp with an arc. |
| BrkCv | SkinningBreakOptions | Required. This argument specifies whether or not the curves have breaks. Possible values include: 0 - No breaks. 1 - If the cross is a GComplexString, then break and create separate surfaces. 2 - If the trace is a GComplexString, then break and create separate surfaces. 3 - Break cross and trace. |
| StartOpt | SkinningCrossSectionStart | Required. This argument is the starting option. Possible values are: 0 - No breaks; 1 - If the cross is a GComplexString, then break and create separate surfaces; 2 - If the trace is a GComplexString, then break and create separate surfaces; 3 - Break cross and trace. |
| StNorm | double | Required. This argument specifies the starting normal. It is a SAFEARRAY of type double. |
| EdNorm | double | Required. This argument specifies the ending normal. It is a SAFEARRAY of type double. |
| WCaps | Boolean | Required. This argument is a Boolean flag that specifies whether or not the surfaces have caps. If the value is False, there are no caps; if the value is True, there are caps. |
| numCaps | Int | Required. This argument is the number of caps. The type is integer. |

**CreateBySkinning Method**

**Description**
The CreateBySkinning method creates a skinned surface with the option of caps. The output is caps and the skin surface.

**Syntax**
object.CreateBySkinning*(pConnection, pTrElements, pCsElements, WCaps)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| pTrElements | IJElements | Required. This argument is a pointer to the trace curves (can be more than 1). If there is one trace only, the trace curve does not have to touch the cross section, but must cross the plane containing the cross section. If there is more than one trace, then the trace curves must touch the cross sections. |
| pCsElements | IJElements | Required. This argument is a pointer to the cross section curves The value can be more than 1. Cross sections are placed exactly how they are to be skinned. |
| WCaps | Int | Required. This argument is a Boolean flag that specifies whether or not the object has caps. If the value is False, there are no caps; if True, there are caps. |

**CreateConeBy4PtsWCaps Method**

**Description**
The CreateConeBy4PtsWCaps method creates and returns a bounded Cone3d object based on four points **-** base center point, top center point, base starting point, and top starting point. Caps are optional. The output is the surface, and then caps.

**Syntax**
object.CreateConeBy4PtsWCaps*(pConnection, CenterBx, CenterBy, CenterBz, CenterTx, CenterTy, CenterTz, StartBx, StartBy, StartBz, StartTx, StartTy, StartTz, Solid, WCaps, numCaps)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CenterBx | double | Required. This argument is the X-coordinate for the base ellipse center point. The type is double. |
| CenterBy | double | Required. This argument is the Y-coordinate for the base ellipse center point. The type is double. |
| CenterBz | double | Required. This argument is the Z-coordinate for the base ellipse center point. The type is double. |
| CenterTx | double | Required. This argument is the X-coordinate for the top ellipse center point. The type is double. |
| CenterTy | double | Required. This argument is the Y-coordinate for the top ellipse center point. The type is double. |
| CenterTz | double | Required. This argument is the Z-coordinate for the top ellipse center point. The type is double. |
| StartBx | double | Required. This argument is the X-coordinate for the base ellipse starting point. The type is double. |
| StartBy | double | Required. This argument is the Y-coordinate for the top ellipse starting point. The type is double. |
| StartBz | double | Required. This argument is the Z-coordinate for the base ellipse starting point. The type is double. |
| StartTx | double | Required. This argument is the X-coordinate for the top ellipse starting point. The type is double. |
| StartTy | double | Required. This argument is the Y-coordinate for the top ellipse starting point. The type is double. |
| StartTz | double | Required. This argument is the Z-coordinate for the top ellipse starting point. The type is double. |
| Solid | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object is solid. Possible values are: 0 - Set normal as hollow; 1 - Set normal as solid; 2 - Set normal according to input point; 3 - Toggle the outward normal (no checks). |
| WCaps | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object has caps. If the value is False, there are no caps; if True, there are caps. |
| numCaps | Int | Required. This argument is the number of caps. The type is integer. |

### CreateProjectionByCurveWCaps Method

**Description**
The CreateProjectionByCurveWCaps method creates a Projection3d object from a curve, direction, and length. Valid curves are Line, Arc, Circle, Ellipse, EllipticalArc, LineString, ComplexString, and BSplineCurve. Caps are optional. The output is the surface, and then the caps.

**Syntax**
object.CreateProjectionByCurveWCaps*(pConnection, CurveObject, uvX, uvY, uvZ, Length, Solid, WCaps, numCaps)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CurveObject | Object | Required. This argument is the curve to project. The type is Object. |
| uvX | double | Required. This argument is the X-coordinate of the point that specifies the vector. The type is double. |
| uvY | double | Required. This argument is the Y-coordinate of the point that specifies the vector. The type is double. |

| | | |
|---|---|---|
| uvZ | double | Required. This argument is the Z-coordinate of the point that specifies the vector. The type is double. |
| Length | double | Required. This argument is the projection distance. The type is double. |
| Solid | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object is solid. Possible values are: 0 - Set normal as hollow; 1 - Set normal as solid; 2 - Set normal according to input point; 3 - Toggle the outward normal (no checks). |
| WCaps | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object has caps. If the value is False, there are no caps; if True, there are caps. |
| numCaps | Int | Required. This argument is the number of caps. The type is integer. |

### CreateRevolutionByCurveWCaps Method

**Description**
The CreateRevolutionByCurveWCaps method creates a Revolution3d object from a curve, axis vector, point on axis, and sweep angle (radians). Valid curves are Line, Arc, Circle, Ellipse, EllipticalArc, LineString, ComplexString, and BSplineCurve. Caps are optional. Output is the surface, and then the caps.

**Syntax**
object.CreateRevolutionByCurveWCaps*(pConnection, CurveObject, AxisX, AxisY, AxisZ, CenterX, CenterY, CenterZ, SwAngle, Solid, WCaps, numCaps)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CurveObject | Object | Required. This argument is the curve from which to create the revolution. The type is Object. |
| AxisX | double | Required. This argument is the X-coordinate of the point that specifies the axis direction. The type is double. |
| AxisY | double | Required. This argument is the Y-coordinate of the point that specifies the axis direction. The type is double. |
| AxisZ | double | Required. This argument is the Z-coordinate of the point that specifies the axis direction. The type is double. |
| CenterX | double | Required. This argument is the X-coordinate of the center point. The type is double. |
| CenterY | double | Required. This argument is the Y-coordinate of the center point. The type is double. |
| CenterZ | double | Required. This argument is the Z-coordinate of the center point. The type is double. |
| SwAngle | double | Required. This argument is the sweep angle. The type is double. |
| Solid | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object is solid. Possible values are: 0 - Set normal as hollow; 1 - Set normal as solid; 2 - Set normal according to input point; 3 - Toggle the outward normal (no checks). |
| WCaps | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object has caps. If the value is False, there are no caps; if True, there are caps. |
| numCaps | Int | Required. This argument is the number of caps. The type is integer. |

**CreateRuledByCurvesWCaps Method**

**Description**
The CreateRuledByCurvesWCaps method creates a RuledSurface3d object from a base curve and a top curve. Valid curves are Line, Arc, Circle, Ellipse, EllipticalArc, LineString, ComplexString, and BSplineCurve. Caps are optional. The output is the surface, and then the caps.

**Syntax**
object.CreateRuledByCurvesWCaps*(pConnection, CurveObjectBase, CurveObjectTop, Solid, WCaps, numCaps)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| CurveObjectBase | Object | Required. This argument is the base curve. |
| CurveObjectTop | Object | Required. This argument is the top curve. |
| Solid | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object is solid. Possible values are: 0 - Set normal as hollow; 1 - Set normal as solid; 2 - Set normal according to input point; 3 - Toggle the outward normal (no checks). |
| WCaps | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object has caps. If the value is False, there are no caps; if True, there are caps. |
| numCaps | Int | Required. This argument is the number of caps. The type is integer. |

**CreateTorusByAxisMajorMinorRadiusSweepWCaps Method**

**Description**
The CreateTorusByAxisMajorMinorRadiusSweepWCaps method creates and returns a Tori3d (torus) object based on an axis, a center point on the axis, the direction to the origin in UV space (orthogonal to the axis), a major radius, and a minor radius. Caps are optional. The output is the surface, and then the caps.

**Syntax**
object.CreateTorusByAxisMajorMinorRadiusSweepWCaps*(pConnection, AxisCenterX, AxisCenterY, AxisCenterZ, AxisVecX, AxisVecY, AxisVecZ, OriginDirX, OriginDirY, OriginDirZ, MajorRadius, MinorRadius, SwAngle, Solid, WCaps, numCaps)*

| Parameter | Data Type | Description |
|---|---|---|
| pConnection | Unknown | Required. This argument is a pointer to IUnknown. It creates a transient object. |
| AxisCenterX | double | Required. This argument is the X-coordinate of the axis center point. The type is double. |
| AxisCenterY | double | Required. This argument is the Y-coordinate of the axis center point. The type is double. |
| AxisCenterZ | double | Required. This argument is the Z-coordinate of the axis center point. The type is double. |
| AxisVecX | double | Required. This argument is the X-coordinate of the point that specifies the axis direction. The type is double. |
| AxisVecY | double | Required. This argument is the Y-coordinate of the point that specifies the axis direction. The type is double. |
| AxisVecZ | double | Required. This argument is the Z-coordinate of the point that specifies the axis direction. The type is double. |
| OriginDirX | double | Required. This argument is the X-coordinate of the point that specifies the origin direction. The |

| | | |
|---|---|---|
| | | type is double. |
| OriginDirY | double | Required. This argument is the Y-coordinate of the point that specifies the origin direction. The type is double. |
| OriginDirZ | double | Required. This argument is the Z-coordinate of the point that specifies the origin direction. The type is double. |
| MajorRadius | double | Required. This argument is the major radius for the torus. The type is double. |
| MinorRadius | double | Required. This argument is the minor radius for the torus. The type is double. |
| SwAngle | double | Required. This argument is the sweep angle. The type is double. |
| Solid | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object is solid. Possible values are: 0 - Set normal as hollow; 1 - Set normal as solid; 2 - Set normal according to input point; 3 - Toggle the outward normal (no checks). |
| WCaps | Boolean | Required. This argument is a Boolean flag that specifies whether or not the object has caps. If the value is False, there are no caps; if True, there are caps. |
| numCaps | Int | Required. This argument is the number of caps. The type is integer. |

The following section shows some examples on how to create some geometry components:

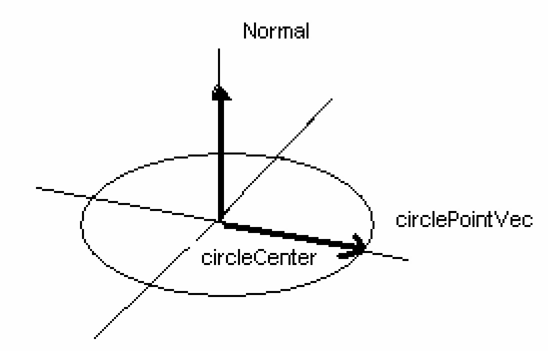
## GeometryFactory.Ellipses3dCreateByCenterNormMajAxisRatio

Creates/returns an Ellipse given the center point, normal axis, major axis containing length, and minor/major ratio.

*Function Ellipses3d.CreateByCenterNormMajAxisRatio(pConnection As Unknown, CenterX As Double, CenterY As Double, CenterZ As Double, NormalX As Double, NormalY As Double, NormalZ As Double, MajorX As Double, MajorY As Double, MajorZ As Double, MMRatio As Double) As Ellipse3d*

| | |
|---|---|
| Define the collection item: | m_outputColl.ResourceManager |
| Define the center of the ellipse: | CenterX, CenterY, CenterZ |
| Define the normal vector: | NormalX, NormalY, NormalZ |
| Define the major axis vector: | MajorPointVecX, MajorPointVecY, MajorPointVecZ |
| Define the axis ratio: | MMRatio |

Example:



*Dim ellipse As IngrGeom3D.Ellipse3d*
*Dim circlePointVecX As Double, circlePointVecY As Double, circlePointVecZ As Double*

```
Dim circleNormalX As Double, circleNormalY As Double, circleNormalZ As Double
Dim projVecX As Double, projVecY As Double, projVecZ As Double

   circleCenterX = 0#
   circleCenterY = 0#
   circleCenterZ = 0#

   circleNormalX = 0#
   circleNormalY = 0#
   circleNormalZ = 1#

   circlePointVecX = 0#
   circlePointVecY = diameter * 0.5
   circlePointVecZ = 0#
  axesRation 1.0

Set ellipse = geomFactory.Ellipses3d.CreateByCenterNormMajAxisRatio(m_outputColl.ResourceManager, _
                        circleCenterX, circleCenterY, circleCenterZ, _
                        circleNormalX, circleNormalY, circleNormalZ, _
                        circlePointVecX, circlePointVecY, circlePointVecZ, _
                        axesRatio)
```

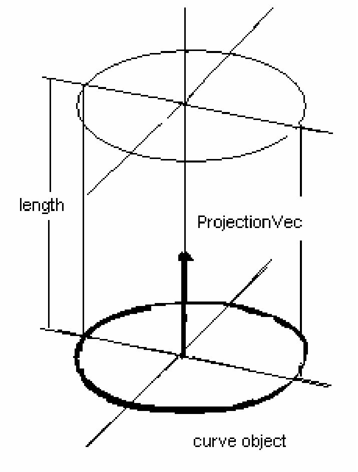
### GeomFactory.Projections3d.CreateByCurve

Creates and returns a Projection3d based on a curve, direction and length. Valid curve objects are Line, Arc, Circle, Ellipse, EllipticalArc, LineString, ComplexString, and BSplineCurve.

*Function Projections3d.CreateByCurve(pConnection As Unknown, CurveObject As Object, projvecX As Double, projvecY As Double, projvecZ As Double, Length As Double, Capped As Boolean) As Projection3d*

| | |
|---|---|
| Define the collection item: | m_outputColl.ResourceManager |
| Define the CurveObject to be projected: | CurveObject As Object |
| Define the projection vector: | projVecX, projVecY, projVecZ As Double |
| Define the projection sidtance: | Length As Double |
| Set the ends to be capped true or false: | Capped As Boolean |

Example:



```
Dim projection As IngrGeom3D.Projection3d
Dim projVecX As Double, projVecY As Double, projVecZ As Double
Dim length As Double
```

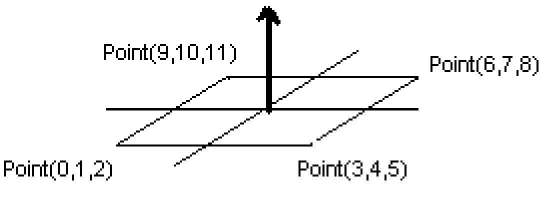
```
  projVecX = 0#
  projVecY = 0#
  projVecZ = 1#

Set projection = geomFactory.Projections3d.CreateByCurve(m_outputColl.ResourceManager, ellipse, _
                        projVecX, projVecY, projVecZ, length, True)
```

### GeomFactory.Planes3d.CreateByPoints

Creates and returns a bounded Plane3d based on 3 or more non-linear, coplanar points. The points must be oriented such that the orientation of the points defines the normal(follows the right hand rule).

*Function Planes3d.CreateByPoints(pConnection As Unknown, PointCount As Long, Points() As Double) As Plane3d*

| | |
|---|---|
| Define the collection item: | m_outputColl.ResourceManager |
| Define the numbe of point in the collection: | PointCount As Long |
| Input an array of Points(): | Points as Double |



Example:

```
Dim plane As IngrGeom3D.Plane3d
Dim Points(0 To 11) As Double

   Points(0) = MinX
   Points(1) = MinY
   Points(2) = 0#
   Points(3) = MaxX
   Points(4) = MinY
   Points(5) = 0#
   Points(6) = MaxX
   Points(7) = MaxY
   Points(8) = 0#
   Points(9) = MinX
   Points(10) = MaxY1
   Points(11) = 0#

Set plane = geomFactory.Planes3d.CreateByPoints(m_outputColl.ResourceManager, 4, Points)
```

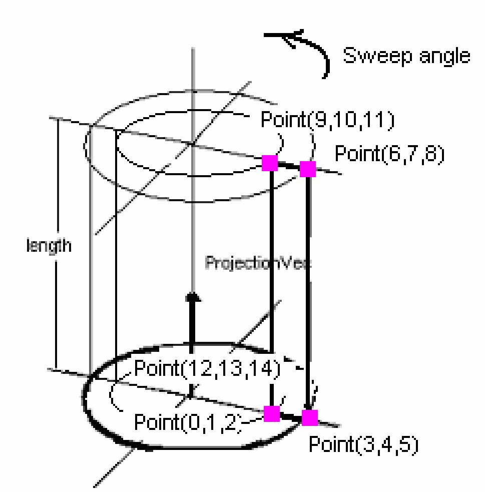
### GeomFactory.Revolutions3d.CreateByCurve

Creates and returns a Revolution3d based on a curve to revolve, axis vector, point on axis and sweep angle(radians).Valid curve objects are Line, Arc, Circle, Ellipse, EllipticalArc, LineString, ComplexString, and BSplineCurve.

*Function Revolutions3d .CreateByCurve(pConnection As Unknown, CurveObject As Object, AxisX As Double, AxisY As Double, AxisZ As Double, CenterX As Double, CenterY As Double, CenterZ As Double, SwAngle As Double, Capped As Boolean) As Revolution3d*

Define the Projection vector to be revolved: AxisX, AxisY, AxisZ as Double
Define the point on axis: CenterX, CenterY, CenterZ as Double
Define sweep angle as Double
Set the ends to be capped true or false: Capped As Boolean

*Example:*



```
Dim axisCenterX As Double, axisCenterY As Double, axisCenterZ As Double
Dim axisVecX As Double, axisVecY As Double, axisVecZ As Double
Dim oRevolution As IngrGeom3D.Revolution3d

   axisCenterX = 0#
   axisCenterY = 0#
   axisCenterZ = 0#
   axisVecX = 0#
   axisVecY = 0#
   axisVecZ = 1#

Dim oLineString As IngrGeom3D.LineString3d
Dim planePoints(0 To 14) As Double

   planePoints(0) = diameter / 2
   planePoints(1) = 0
   planePoints(2) = 0
   planePoints(3) = diameter / 2 + dInsulationThickness
   planePoints(4) = 0
   planePoints(5) = 0
   planePoints(6) = diameter / 2 + dInsulationThickness
   planePoints(7) = 0
   planePoints(8) = length
   planePoints(9) = diameter / 2
   planePoints(10) = 0
   planePoints(11) = length
   planePoints(12) = diameter / 2
   planePoints(13) = 0
   planePoints(14) = 0

 Set oLineString = geomFactory.LineStrings3d.CreateByPoints(m_outputColl.ResourceManager, 5,_
                   planePoints)

Set oRevolution = geomFactory.Revolutions3d.CreateByCurve(m_outputColl.ResourceManager, _
                  oLineString,  axisVecX, axisVecY, axisVecZ, axisCenterX, axisCenterY, axisCenterZ, _
                  2 * PI, False)
```

# Equipment Symbol CAD Helper Reference

The Equipment Symbol CAD Helper Reference provides documentation for the interfaces, methods, and properties of IJEquipCADHelper.dll.

**IJEquipCADHelper**

**Description**

The IJEquipCADHelper interface is the helper for the Custom Assembly Definition (a class that implements IJDUserSymbolServices) for Equipment.

**Remarks**

The CADServices class of the SP3DEquipCADHelper Component provides the default implementation of the methods and the properties defined in the interface IJEquipCADHelper.

CADServices is a member of SP3DEqpCADHelper, which is unique implementation class of the interface IJEquipCADHelper.

SP3DEqpCADHelper library, SP3DEqpCADHelper.dll, is named the Ingr SmartPlant 3D Equipment Custom Assembly Definition Helper v 1.0.

**Properties**

**ClassName Property**

**Description**
The ClassName property sets or returns the Custom Assembly Definition module class name. This class name, as part of a ProgID, must be referenced in the column 'Definition' for each part in the bulkloaded spreadsheets.

**Syntax**
object.ClassName

**Remarks**
The length of ClassName concatenated to ProjectName must not exceed 39 characters.

```
' /////////////////////////
' Use case for let

Private Sub Class_Initialize()
    . . .
    Set m_oEquipCADHelper = New CADServices
    m_oEquipCADHelper.ProjectName = "MyEqpProjName"
    m_oEquipCADHelper.ClassName = "CMyPartClassDef"
    m_oEquipCADHelper.OccurrenceRootClass = orcEquipment
    . . .
End Sub

' ////////////////////////////////
' Use case for get:
Private Function IJDUserSymbolServices_GetDefinitionName(ByVal definitionParameters As Variant) As String
```

```
    IJDUserSymbolServices_GetDefinitionName = m_oEquipCADHelper.ProjectName & "." &
m_oEquipCADHelper.ClassName

End Function
```

**OccurrenceRootClass Property**

**Description**
The OccurrenceRootClass property sets or returns the type of Custom Assembly Occurrence root class.
It must be in sync with the part class type used in the bulkloaded spreadsheets.

**Syntax**
object.OccurrenceRootClass

```
' //////////////////////////////////
' Use case for let

Private Sub Class_Initialize()
. . .
Set m_oEquipCADHelper = New CADServices
m_oEquipCADHelper.ProjectName = "MyEqpProjName"
m_oEquipCADHelper.ClassName = "CMyPartClassDef"
m_oEquipCADHelper.OccurrenceRootClass = orcEquipment
. . .
```

**OccurrenceRootClassGUID Property**
**Description**

The OccurrenceRootClassGUID property returns the CLSID of the Custom Assembly Occurrence root class and is read-only.

**Syntax**
object.OccurrenceRootClassGUID

**Remarks**
This property must be used in IJDUserSymbolServices_InitialSymbolDefinition method of the CAD for setting the AggregatorClsid. The helper assigns the OccurrenceRootClassGUID internally when the property OccurrenceRootClass is let, which must occur previously (usually done in the Class_Initialize method).

```
' ////////////////////////////////////
' Use case for get

Private Sub IJDUserSymbolServices_InitializeSymbolDefinition(oSymbolDefinition As
IMSSymbolEntities.IJDSymbolDefinition)
    . . .
    Dim oPropertyDescriptions As IJDPropertyDescriptions
    Dim oMemberDescriptions As IJDMemberDescriptions
    Dim oMemberDescription As IJDMemberDescription
    Dim oAggregatorDescription As IJDAggregatorDescription

    ' Set up the aggregator
    Set oAggregatorDescription = oSymbolDefinition
    oAggregatorDescription.AggregatorClsid = m_oEquipCADHelper.OccurrenceRootClassGUID()
    . . .
End Sub
```

### ProjectName Property

**Description**
The ProjectName property sets or returns the Custom Assembly Definition Visual Basic project name. This project name, as part of a ProgID, must be referenced in the column 'Definition' for each part in the bulkloaded spreadsheets.

**Syntax**
object.ProjectName
Remarks
The length of ClassName concatenated to ProjectName must not exceed 39 characters.

```
' //////////////////////////////////////////
' Use case for let:

Private Sub Class_Initialize()
    . . .
    Set m_oEquipCADHelper = New CADServices
    m_oEquipCADHelper.ProjectName = "MyEqpProjName"
    m_oEquipCADHelper.ClassName = "CMyPartClassDef"
    m_oEquipCADHelper.OccurrenceRootClass = orcEquipment
    . . .
End Sub

' Use case for get:

Private Function IJDUserSymbolServices_GetDefinitionName(ByVal definitionParameters As Variant) As String
    . . .
    IJDUserSymbolServices_GetDefinitionName = m_oEquipCADHelper.ProjectName & "." &
m_oEquipCADHelper.ClassName
    . . .
End Function
```

**Methods**

### CheckMemberConditional Method

**Description**

The CheckMemberConditional method checks whether the member is conditional based on the CanBeDeleted flag in the MakeMemberDeletable method.

**Syntax**

object.CheckMemberConditional*(MemberDescription)*

| Parameter | Data Type | Description |
|---|---|---|
| MemberDescription | IJDMemberDescription | Required. This argument specifies the equipment or equipment component member. |

**Remarks**

The implementation of the smart-occurrence in Equipment provides the ability to delete members that are considered optional.

In order to accomplish this:

1. The type of member must implement the interface IJDeletableMember. This is done by default for all type of ports, shapes, and equipment components.

2. When the property is changed, call MakeMemberDeletable to forward the setting to the equipment.

3. When the condition of the member is evaluated, call CheckMemberConditional

Notes: From the user standpoint, the workflow is as follows:

1. Edit the member to be deleted

2. Set the CanBeDeleted flag in MakeMemberDeletable to True and OK the change.

3. Use the Delete command to delete the member.

At this point, the member cannot "come back", unlike a regular conditional member.

```
' ////////////////////////////////////
' Use case

Public Sub CMConditionalDP1(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As Boolean)
   . . .
   IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)
   . . .
End Sub
```

**CreateEquipmentComponent Method**

**Description**
The CreateEquipmentComponent method creates an equipment component and adds it to the Equipment.

**Syntax**
object.CreateEquipmentComponent*(MemberDescription, ResourceManager, EquipCompPartNumber, EquipComponentOccName)*

| Parameter | Data Type | Description |
|---|---|---|
| MemberDescription | IJDMemberDescription | Required. This argument specifies the equipment or equipment component member. |
| ResourceManager | Object | Required. Resource Manager of the Model or Catalog connection depending on whether created under plant/ship or catalog. |
| EquipCompPartNumber | String | Required. |
| EquipComponentOccName | String | Required. |

**Remarks**
This method must be called from the CMConstruct method of a member.

```
' ///////////////////////////////////////
' Use case:

Public Sub CMConstructTyB(ByVal pMemberDescription As IJDMemberDescription, _
                        ByVal pResourceManager As IUnknown, _
                        ByRef pObject As Object)
   . . .
   'Create Equipment Component
   Set pObject = m_oEquipCADHelper.CreateEquipmentComponent(pMemberDescription, pResourceManager,
"FrontEndTypeB 01-EC", "TyB")
   . . .
End Sub
```

**CreateNozzleFromPH Method**

**Description**
The CreateNozzleFromPH method creates a pipe nozzle from a nozzle placeholder defined in the equipment symbol.

**Syntax**
object.CreateNozzleFromPH*(MemberDescription, ResourceManager, Nozzle, NozzleIndex)*

| Parameter | Data Type | Description |
|---|---|---|
| MemberDescription | IJDMemberDescription | Required. This argument specifies the equipment or equipment component member. |
| ResourceManager | Object | Required. Resource Manager of the Model or Catalog connection depending on whether created under plant/ship or catalog. |
| Nozzle | Object | Required. This argument specifies the nozzle object. |
| NozzleIndex | long | Required. This argument specifies the nozzle object index. |

**Remarks**

It is assumed that a place holder with the same name of the member or same index (if specified) is created as an output by the symbol. The nozzle data are retrieved from the part at the given index (internally retrieved by the name if not specified).
This type of creation is used when the position and the orientation of the nozzle are driven totally or partially by the symbol along with the rest of the geometry. The nozzle is considered to be placed by point.
Accordingly, all or some of the properties, such as nozzle placement type nozzle position and orientation, must be made read-only in attribute management functions calls.
Note: Creating a nozzle directly using the factory is discouraged. Apparent successful behavior must not be considered. The methods of the Equipment CAD Helper perform complementary operations that provide internal behavior common with other nozzles.

```
' ////////////////////////////////////
' Use case

Public Sub CMConstructNozzleVesselFoundationPort8(ByVal pMemberDescription As IJDMemberDescription, _
                        ByVal pResourceManager As IUnknown, _
                        ByRef pObject As Object)
   . . .
   'Create Nozzle
   m_oEquipCADHelper.CreateNozzleFromPH pMemberDescription, pResourceManager, pObject, 8
   . . .
End Sub
```

**CreateNozzleGivenIndex Method**

**Description**

The CreateNozzleGivenIndex method creates the nozzle given the index of the nozzle on the Equipment Part.

**Syntax**
object.CreateNozzleGivenIndex*(MemberDescription, NozzleIndex, ResourceManager, PortType, Nozzle, LightWeightGraphics)*

| Parameter | Data Type | Description |
|---|---|---|
| MemberDescription | IJDMemberDescription | Required. This argument specifies the equipment or equipment component member. |
| NozzleIndex | long | Required. This argument specifies the nozzle object index. |
| ResourceManager | Object | Required. Resource Manager of the Model or Catalog connection depending on whether created under plant/ship or catalog. |
| PortType | DistribPortType | Required. |
| Nozzle | Object | Required. This argument specifies the nozzle object. |
| LightWeightGraphics | Boolean | Required. |

**Remarks**
No placeholder from the symbol is required. The nozzle data are retrieved from the part at the given index.
This type of creation is used in conjunction with the creation of another member of type shape relative to which an orientation is given. This allows giving more control to the symbol designer and at the same time more freedom to the positioning and orientation of the nozzle by the user.
Any placement type can be used to place the nozzle.
Accordingly, none or some of the properties such as nozzle placement type or nozzle position and orientation can be made read-only in attribute management functions calls.

```
' ////////////////////////////////
' Use case

Public Sub CMConstructNozzleVVANoz11(ByVal pMemberDescription As IJDMemberDescription, _
                                ByVal pResourceManager As IUnknown, _
                                ByRef pObject As Object)
   . . .
   Dim oOrientation As IJNozzleOrientation
   Dim oNozzle As IJDNozzle

   GetDimensionsFromSymbolArray pMemberDescription.CAO

   'Create Nozzle
   m_oEquipCADHelper.CreateNozzleGivenIndex pMemberDescription, 1, pResourceManager,
DistribPortType_DUCT, pObject, False

   Set oNozzle = pObject
   oNozzle.Length = 0.1905

   'Create the nozzle orientation and set it on the nozzle
   Set oOrientation = . . .

End Sub
```

## CreateOrientationAndSetRelations Method

### Description

The CreateOrientationAndSetRelations method creates a nozzle orientation and sets the relations with the reference geometry and nozzle.

### Syntax

object.CreateOrientationAndSetRelations*(ReferenceGeometry, Nozzle)*

| Parameter | Data Type | Description |
|---|---|---|
| ReferenceGeometry | Object | Required. |
| Nozzle | Object | Required. This argument specifies the nozzle object. |

### Remarks

This method is called in the CMContruct method of the nozzle being oriented. It is assumed that a reference geometry (a shape) has already been created as member.

```
' ///////////////////////////////////
' Use case

Public Sub CMConstructNozzleVVANoz11(ByVal pMemberDescription As IJDMemberDescription, _
                                ByVal pResourceManager As IUnknown, _
                                ByRef pObject As Object)
   . . .
   'Create the nozzle orientation and set it on the nozzle
   Set oOrientation = m_oEquipCADHelper.CreateOrientationAndSetRelations(Nothing, oNozzle)

   'Set the default values
   oOrientation.PlacementType = Axial

   oOrientation.N1 = 0.9875
   oOrientation.N2 = 0.123
   oOrientation.OR1 = PI / 2
   oOrientation.OR2 = 0#

   Set oNozzle = Nothing
   Set oOrientation = Nothing
   . . .
End Sub
```

## CreateShape Method

### Description

The CreateShape method creates a shape. The shape name is equivalent to a part number.

### Syntax

object.CreateShape*(MemberDescription, ResourceManager, ShapeName, ShapeOccName)*

| Parameter | Data Type | Description |
|---|---|---|
| MemberDescription | IJDMemberDescription | Required. This argument specifies the equipment or equipment component member. |
| ResourceManager | Object | Required. Resource Manager of the Model or Catalog connection depending on whether created under plant/ship or catalog. |

| | | |
|---|---|---|
| ShapeName | String | Required. |
| ShapeOccName | String | Required. |

**Remarks**

The positioning and orientation of the shape can be freed like a manually added shape; or can be made a function of the symbol inputs; or a mix of both.
Accordingly, either none or some of the properties, such as IsAssociative, position, and orientation, can be made read-only in attribute management functions calls.

```
' //////////////////////////////////////////
' Use case

Public Sub CMConstructDP1(ByVal pMemberDescription As IJDMemberDescription, _
                          ByVal pResourceManager As IUnknown, _
                          ByRef pObject As Object)
    . . .
    Dim oDatumShape As IJShape
    Dim oDesignEquipment As IJDesignEquipment

    'Create Datum Shape DP1
    Set oDatumShape = m_oEquipCADHelper.CreateShape(pMemberDescription, pResourceManager, "DatumShape
001", "DP1")
    If Not oDatumShape Is Nothing Then
        Set pObject = oDatumShape
        oDatumShape.RepresentationId = ReferenceGeometry

        Set oDesignEquipment = pMemberDescription.CAO
        oDesignEquipment.AddShape oDatumShape

        GetDimensionsFromSymbolArray oDesignEquipment
        'Use a private method to position and orient
        PositionAndOrientDP1 oDesignEquipment, oDatumShape
    End If

    Set oDesignEquipment = Nothing
    Set oDatumShape = Nothing
    . . .
End Sub
```

### GetMemberDescriptionFromChild Method

**Description**

The GetMemberDescriptionFromChild method returns the member description of a Custom Assembly Occurence aggregator (Equipment or EquipmentComponent) child.

**Syntax**

object.GetMemberDescriptionFromChild*(Child)*

| Parameter | Data Type | Description |
|---|---|---|
| Child | IJDAttributes | Required. This argument specifies the child attribute of the member. |

**Remarks**

This method is useful in the attribute management implementation methods.

### GetNozzleDataFromPart Method

**Description**
The GetNozzleDataFromPart method returns the port definition from the part at a given nozzle index.

**Syntax**
object.GetNozzleDataFromPart*(Part, Portindex)*

| Parameter | Data Type | Description |
|---|---|---|
| Part | IJDPart | Required. |
| Portindex | long | Required. |

**Remarks**
This method can be used when the placement and orientation of the nozzle depend on its own dimensions.

### GetNozzlePlaceHolderByIndex Method

**Description**
The GetNozzlePlaceHolderByIndex method returns the nozzle placeholder from the symbol at a given nozzle index.

**Syntax**
object.GetNozzlePlaceHolderByIndex*(SmartOcc, NozzleIndex)*

| Parameter | Data Type | Description |
|---|---|---|
| SmartOcc | IJSmartOccurrence | Required. |
| NozzleIndex | long | Required. This argument specifies the nozzle object index. |

**Remarks**
This method can be used in CMConditional of a nozzle that is dependant upon a nozzle placeholder that is itself variable output of the symbol (may or may not exist).

```
' ///////////////////////////////////////
' Use case

Public Sub CMConditionalNozzleN11(ByVal pMemberDesc As IJDMemberDescription, ByRef IsNeeded As Boolean)
   . . .

   'The existence of this nozzle depends on the existence of a nozzle placeholder in the symbol with the same index.
   If m_oEquipCADHelper.GetNozzlePlaceHolderByIndex(pMemberDesc.CAO, 1) Is Nothing Then
      IsNeeded = False
   Else
      IsNeeded = True
      IsNeeded = m_oEquipCADHelper.CheckMemberConditional(pMemberDesc)
   End If
   . . .
End Sub
```

### GetNozzlePlaceHolderByName Method

**Description**
The GetNozzlePlaceHolderByName method returns the nozzle placeholder from the symbol for a given symbol output name.

**Syntax**
object.GetNozzlePlaceHolderByName*(SmartOcc, outputName)*

| Parameter | Data Type | Description |
|---|---|---|
| SmartOcc | IJSmartOccurrence | Required. |
| outputName | String | Required. This argument specifies the name of the symbol. |

**Remarks**
This method can be used in CMConditional of a nozzle that is dependant upon a nozzle placeholder that is itself variable output of the symbol (may or may not exist).

### GetSymbolArrayOfInputs Method

**Description**
The GetSymbolArrayOfInputs method returns the symbol array of inputs.
This method allows for creating and updating non-nozzle members in the Custom Assembly Definition that take the same inputs as the symbol.

**Syntax**
object.GetSymbolArrayOfInputs*(SmartOcc)*

| Parameter | Data Type | Description |
|---|---|---|
| SmartOcc | IJSmartOccurrence | Required. |

**Remarks**
This method allows for mimic of the inputs management made in the symbol.

### InstanciateDefinition Method

**Description**
The InstanciateDefinition method instantiates the Custom Assembly Definition as a Symbol Definition.

**Syntax**
object.InstanciateDefinition*(CodeBase, DefParameters, ResourceManager)*

| Parameter | Data Type | Description |
|---|---|---|
| CodeBase | String | Required. This argument is the location where an archive of the software distribution exists. |
| DefParameters | VARIANT | Required. This argument specifies the definition parameters for the symbol. |
| ResourceManager | Object | Required. Resource Manager of the Model or Catalog connection depending on whether created under plant/ship or catalog. |

**Remarks**
The InstanciateDefinition method returns a persistent symbol definition object based on indicated parameter input.

If more than one Codebase is specified, then any one of the multiple URLs is valid, and any one of them may be used at random (for load-balancing reasons).

```
' ////////////////////////////////////////
' Use case:

Private Function IJDUserSymbolServices_InstanciateDefinition(ByVal CodeBase As String, _
                                                             ByVal defParams As Variant, _
                                                             ByVal pResourceMgr As Object) As Object
    . . .
    Dim oSymbolDefinition As IJDSymbolDefinition

    Set oSymbolDefinition = m_oEquipCADHelper.InstanciateDefinition(CodeBase, defParams, pResourceMgr)
    IJDUserSymbolServices_InitializeSymbolDefinition oSymbolDefinition
    Set IJDUserSymbolServices_InstanciateDefinition = oSymbolDefinition
    . . .
End Function
```

### IsShapeFreeToTransform Method

**Description**

The IsShapeFreeToTransform method returns a flag indicating whether the Shape follows the symbol inputs or not.

**Syntax**

object.IsShapeFreeToTransform*(Shape)*

| Parameter | Data Type | Description |
|---|---|---|
| Shape | Object | Required. |

**Remarks**

This method is called in the CMEvaluateGeometry of the shape member and checks for the state of the IsAssociative property.

This functionality is provided in order to mimic some PDS behavior for the datum point: When a datum point is moved from the position driven by the EDEN symbol, it looses its ability to follow the symbol inputs. In the software change the IsAssociative property to False, assuming that the following method is used and that the internal logic is coded.

To avoid misleading the user, shapes that are created as members and do not implement such a mechanism should define the IsAssociative property as read-only.

```
' ////////////////////////////////////
' Use case

Public Sub CMEvaluateGeometryDP1(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
    . . .
    Dim oEquipment As IJEquipment
    Dim oDatumShape As IJShape

    Set oDatumShape = oPropertyDescription.Object

    'Test if this DatumShape follows the symbol inputs
    If m_oEquipCADHelper.IsShapeFreeToTransform(oDatumShape) = False Then
        Set oEquipment = oPropertyDescription.CAO
        GetDimensionsFromSymbolArray oEquipment
        PositionAndOrientDP1 oEquipment, oDatumShape
    Else
        'Do nothing, the equipment will transform the shape for us
    End If
    Set oDatumShape = Nothing
    Set oEquipment = Nothing
    . . .
End Sub
```

### LogError Method

**Description**
The LogError method logs to the middle tier error log when it is set to exhaustive setting.

**Syntax**
object.LogError*(Description)*

| Parameter | Data Type | Description |
|---|---|---|
| Description | String | Required. This argument is the error string description of the log. |

**Remarks**
This method allows tracing or logging errors and warnings only when the middle tier error log is set to Exhaustive.

### MakeMemberDeletable Method

**Description**
The MakeMemberDeletable method sets the Custom Assembly member deletable or non-deletable depending on the setting of the CanBeDeleted flag.

**Syntax**
object.MakeMemberDeletable*(MemberDescription, Child, CanBeDeleted)*

| Parameter | Data Type | Description |
|---|---|---|
| MemberDescription | IJDMemberDescription | Required. This argument specifies the equipment or equipment component member. |
| Child | IJDAttributes | Required. This argument specifies the child attribute of the member. |
| CanBeDeleted | Boolean | Required. This argument is a Boolean specifying whether the member can be deleted. |

**Remarks**
The implementation of the smart-occurrence in Equipment symbols provides the ability to delete members that are considered optional.
In order to accomplish this:
  1. The type of member must implement the interface IJDeletableMember. This is done by default for all type of ports, shapes, and equipment components.
  2. When the property is changed, call MakeMemberDeletable to forward the setting to the equipment.
  3. When the condition of the member is evaluated, call CheckMemberConditional.
Note: For a user, the workflow is as follows::
  1. Edit the member to be deleted.
  2. Set the CanBeDeleted argument to True and OK the change.
  3. Use the Delete command to delete the member.
At this point, the member cannot "come back", unlike a regular conditional member.

```
' ////////////////////////////////////////
' Use case
```

```
Private Function IJEquipUserAttrMgmt_OnAttributeChange(ByVal pIJDAttrs As IJDAttributes, ByVal
CollAllDisplayedValues As Object, ByVal pAttrToChange As IJEquipAttrDescriptor, ByVal varNewAttrValue As
Variant) As String
    . . .
    Dim oMemberDescription As IJDMemberDescription

    Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)

    Select Case oMemberDescription.name
        Case "DP1"
            Select Case UCase(pAttrToChange.InterfaceName)
                Case "IJDELETABLEMEMBER"
                    If UCase(pAttrToChange.AttrName) = "CANBEDELETED" Then
                        m_oEquipCADHelper.MakeMemberDeletable oMemberDescription, pIJDAttrs,
CBool(varNewAttrValue)
                    End If
                Case Else
            End Select
        Case Else
    End Select

    Set oMemberDescription = Nothing

    IJEquipUserAttrMgmt_OnAttributeChange = ""
    . . .
End Function
```

**PositionAndOrientShape Method**

**Description**
The PositionAndOrientShape method is used to position and orient the shape of Equipment symbols.

**Syntax**
object.PositionAndOrientShape*(Equipment, Shape, Position, XAxis, YAxis)*

| Parameter | Data Type | Description |
|---|---|---|
| Equipment | IJEquipment | Required. |
| Shape | Object | Required. |
| Position | IJDPosition | Required. |
| XAxis | IJDVector | Required. |
| YAxis | IJDVector | Required. |

**Remarks**
This method positions and orients the shape of equipment in the coordinate system.

```
' ////////////////////////////////////////
' Use case

Private Sub PositionAndOrientDP1(Equipment As IJEquipment, Shape As IJShape)

    Dim oPosition As IJDPosition
    Set oPosition = New DPosition
    oPosition.Set 0, 0, 0

    'We want:
    'the X (primary) of the shape on the Z or Elevation of the equipment (ECS)
    'the Y (secondary) of the shape on the Y or North of the equipment (ECS)
    m_oEquipCADHelper.PositionAndOrientShape Equipment, Shape, oPosition, m_oEast, m_oNorth

    Set oPosition = Nothing

End Sub
```

**SetMemberPropertiesAsReadOnly Method**

**Description**
The SetMemberPropertiesAsReadOnly method sets all the properties of a member, except IJEFWCorrelation interface and IJNamedItem properties names, as read-only.

**Syntax**
object.SetMemberPropertiesAsReadOnly*(MemberDescription, CollAllDisplayedValues, IncludesName)*

| Parameter | Data Type | Description |
|---|---|---|
| MemberDescription | IJDMemberDescription | Required. This argument specifies the equipment or equipment component member. |
| CollAllDisplayedValues | Object | Required. This argument is the collection of displayed property values. |
| IncludesName | Boolean | Required. This argument specifies a Boolean indicating whether the name will be included in the member properties. |

**Remarks**
This is a shortcut method to setup properties (attributes) for members of the dimension, position, and orientation that are driven by the symbol or the CAD.

**SetSmartItemAsInputToSymbol Method**

**Description**
The SetSmartItemAsInputToSymbol method sets the SmartItem (Part) as input to the Custom Assembly Occurrence symbol in order for it to be updated after a bulkload.
This method must be called by the CMSetInputs of the aggregator.

**Syntax**
object.SetSmartItemAsInputToSymbol*(AggregatorDescription)*

| Parameter | Data Type | Description |
|---|---|---|
| AggregatorDescription | IJDAggregatorDescription | Required. |

**Remarks**
This method is a mandatory requirement for Equipment and Equipment Component symbols.

```
' ////////////////////////////////////////////
' Use case

Public Sub CMSetInputsEquipment(ByVal pAggregatorDescription As IJDAggregatorDescription)
```

```
    . . .
    m_oEquipCADHelper.SetSmartItemAsInputToSymbol pAggregatorDescription
    . . .
End Sub
```

### TransformNozzleWrtPH Method

**Description**

The TransformNozzleWrtPH method keeps the Nozzle, as a member, always positioned as the matching nozzle placeholder in the symbol.

This method prevents free transformation of the Nozzle when it is called in its CMEvaluate method.

**Syntax**

object.TransformNozzleWrtPH*(PropertyDescription, Nozzle, NozzleIndex)*

| Parameter | Data Type | Description |
|---|---|---|
| PropertyDescription | IJDPropertyDescription | Required. |
| Nozzle | Object | Required. This argument specifies the nozzle object. |
| NozzleIndex | long | Required. This argument specifies the nozzle object index. |

```
' //////////////////////////////////
' Use case

Public Sub CMEvaluateGeometryNozzleN22(ByVal oPropertyDescription As IJDPropertyDescription, pObject As Object)
. . .

'Transform the nozzle so that it behaves like a rigid body inside the equipment
m_oEquipCADHelper.TransformNozzleWrtPH oPropertyDescription, pObject, 2
. . .
End Sub
```

# Equipment User Attribute Management Service

## IJEquipUserAttrMgmt

**Description**

The IJEquipUserAttrMgmt interface is the notification interface triggered by the property page. This interface allows the Custom Assembly Definition (CAD) that implements it to override (at the level of the occurrence) the behavior defined in the metadata.
Note: This interface must be implemented by the CAD in order to interact with the property pages. The property pages call these methods from the ProgID of the CAD.
Following defines its library and filename:
SP3DEquipCADHelperInterfaces library
SP3DEquipCADHelperInterfaces.dll
Ingr SmartPlant 3D Equipment CAD Helper Interfaces v 1.0 Library

**Methods**

**OnAttributeChange Method**

**Description**
The OnAttributeChange method is used each time an attribute is changed.

**Syntax**
object.OnAttributeChange*(pIJDAttrs, CollAllDisplayedValues, pAttrToChange, varNewAttrValue)*

| Parameter | Data Type | Description |
|---|---|---|
| pIJDAttrs | IJDAttributes | Required. This argument specifies the collection of attributes to be changed. |
| CollAllDisplayedValues | Object | Required. This argument is the collection of displayed property values. |
| pAttrToChange | IJEquipAttrDescriptor | Required. This argument specifies the actual property to be changed. |
| varNewAttrValue | VARIANT | Required. This argument specifies the new attribute value. |

**Remarks**
This method is used to validate property values. It should not contain middle tier logic.

**OnPreLoad Method**

**Description**
The OnPreLoad method is used immediately before the property-page attributes are set.

**Syntax**
object.OnPreLoad*(pIJDAttrs, CollAllDisplayedValues)*

| Parameter | Data Type | Description |
|---|---|---|
| pIJDAttrs | IJDAttributes | Required. This argument specifies the collection of attributes to be set. |
| CollAllDisplayedValues | Object | Required. This argument is the collection of displayed property values. |

**Remarks**
This method must contain the logic for read-only properties.
Properties, which are defined in the metadata as read-only, cannot be reset as read/write.

```
' ////////////////////////////////////////
' Use case

Private Function IJEquipUserAttrMgmt_OnPreLoad(ByVal pIJDAttrs As IJDAttributes, ByVal
CollAllDisplayedValues As Object) As String
    . . .
    Dim oMemberDescription As IJDMemberDescription

    IJEquipUserAttrMgmt_OnPreLoad = "ERROR"
    Set oMemberDescription = m_oEquipCADHelper.GetMemberDescriptionFromChild(pIJDAttrs)
    Dim oAttrCollection As Collection
    Dim oAttributeDescriptor As IJEquipAttrDescriptor
    Dim m As Long
    Set oAttrCollection = CollAllDisplayedValues
    Select Case oMemberDescription.name
        Case "DP1"
            For m = 1 To oAttrCollection.Count
                Set oAttributeDescriptor = oAttrCollection.Item(m)
                Select Case UCase(oAttributeDescriptor.InterfaceName)
                    Case "IJUADATUMSHAPE"
                        oAttributeDescriptor.AttrState = oAttributeDescriptor.AttrState Or adsReadOnly
                    Case Else
                        '
                End Select
            Next
            Set oAttrCollection = Nothing
        Case Else

    End Select

Set oMemberDescription = Nothing
    IJEquipUserAttrMgmt_OnPreLoad = ""
    Exit Function
ErrorHandler:
    IJEquipUserAttrMgmt_OnPreLoad = "ERROR"
    HandleError MODULE, METHOD
End Function
```

**OnPreCommit Method**

**Description**
The OnPreCommit method is used prior to committing the attribute changes.

**Syntax**
object.OnPreCommit*(pIJDAttrs, CollAllDisplayedValues)*

| Parameter | Data Type | Description |
|---|---|---|
| pIJDAttrs | IJDAttributes | Required. This argument specifies the collection of attributes to be changed. |
| CollAllDisplayedValues | Object | Required. This argument is the collection of displayed property values. |

# Custom Assembly Definition

The features and the behaviors of the custom assembly occurrence are defined by the custom assembly definition. The members of custom assembly are specified by several set of data.

- Construction and initialization of a member
  - Constructor (CMConstruct)
  - Set of input arguments (CMSetInputs)
- Relationship that exists between the custom assembly and a member.
- Controlled properties (or portions) of the member
- Controlled Interface
- Evaluate function.

This information is stored by the custom assembly definition. This information is accessible through following interfaces:

- **IJDMemberDescriptions** interface
  It manages a collection of MemberDescription for the custom assembly
- **IJDMemberDescription** interface
  It describes a list of custom methods: the CMConditionnal custom method checks if the member should exist. The CMConstruct custom method creates the member
- **IJDPropertyDescriptions** interface
  For a member, a set of properties can be controlled. It is the same behavior than the one exposed on the aggregator.

The **RelationshipClsid** property specifies the clsid of a particular relationship type it exists between the CAO and the member. We assume that the CAO is always the origin of the relation.
When a predefined relationship has been selected, the relationclsid is automatically set by the CAD.

**CMConditional** is the custom method that checks if an optional member is needed or not. This method is used during the creation and the compute of the CAO.

**CMConstruct** is the custom method in charge of the creation of the CAO member object. This method is used only when the member object has to be created..

**CMFinalConstruct** is an optional custom method in charge of connecting additional relationship between the created member and other objects of the model.

**CMSetInputs** is the custom method that set the input arguments to the member object.

## IJDPropertyDescriptions

This is the interface to access properties on a member or the aggregator of a custom assembly definition

### Methods

AddProperty (name as String, dispid as Long, InterfaceID as String, CMEvaluate as Variant, libCookie as Variant, varProcessTime as PDProcessTime) ppPD as IJDPropertyDescription

Description: check if the given property is set in the description of a property/member description.

Parameters:

[in] name — Name of the property.

[in] dispid — Dispatch id of the property.

[in] InterfaceID — Id of the interface controlled by the property.

[in, optional] CMEvaluate — Name of the custom method to be called to evaluate the property.

[in, optional] libCookie — Identification of the library, hosting the custom method.

[in, optional, defaultvalue (igPROCESS_PD_BEFORE_SYMBOL_UPDATE)] varProcessTime — Time at which the custom method needs to be excuted with respect to the update of the aggregated symbol. This is valid only in the context, where the property is relative to the Aggregator section of a custom assembly definition.

[out,retval, optional] ppPD — Created property.

Add (pPD as IJDPropertyDescription) ppInternalPD as IJDPropertyDescription

Description: Add a property from a property description.

[in] pPD — Interface on the property allocated by the caller.

[out,retval,optional] ppInternalPD — Interface on the internal property stored in the collection.

Remove (key as Variant)

Description: Delete an item from the collection specified by key (index or name).

[in] key — key (index or name) of the item in the collection.

RemoveByDispid (dispid as Long)

Description: Delete an item from the collection specified by dispatch id.

[in] dispid — Dispatch id of the item in the collection.

RemoveAll ()

Description: Delete all the items from the collection.

### Properties

Item (key as Variant) as IJDPropertyDescription

Description: Gets the item specified by key (index or name).

Modifiability: Read

[in] key — Key (index or name) of the item in the collection.

ItemByDispid (dispid as Long) as IJDPropertyDescription

Description: Gets the item specified by dispatch id).

Modifiability: Read

[in] dispid — Dispatch id of the item.

Count () as Long

Description: Gets the count of items in the collection.

Modifiability: Read

## IJDPropertyDescription

The IJDPropertyDescription interface is the interface to manipulate a property on a member or the aggregator of a custom assembly definition

**Methods**

IsPropertySet (prop as Long) pProp as Boolean
Description: check if the given property is set in the description of a property/member description.
Parameters:
[in] prop A mask on the property to check.
[out, retval] pProp True/False if the property is set or not.

Reset ()
Description: Reset the description of a property/member description.

SetCMEvaluate (libraryCookie as Variant, method as Variant)
Description: Declare the CMEvaluate custom method used to evaluate the property .
Parameters:
[in] libraryCookie Local id of the library, hosting the custom method.
[in] method Dispatch id of the custom method within the library.

**Properties**

*Name () as String*

| | |
|---|---|
| Description: | Sets or gets the name of a property/member of a custom assembly definition. |
| Modifiability: | Read/Write |

Description () as String
Description: Sets or gets the description of a property/member of a custom assembly definition.
Modifiability: Read/Write

Properties () as Long
Description: Sets or gets the properties mask of a property/member of a custom assembly definition.
Modifiability: Read/Write

Dispid () as Long
Description: Sets or gets the Dispatch Id of a property of a custom assembly definition.
Modifiability: Read/Write

InterfaceId () as String
Description: Set or gets the Interface Id of a property of a custom assembly definition.
Modifiability: Read/Write

ProcessTime () as PDProcessTime
Description: Set or gets the time to process the property.
Modifiability: Read/Write

Definition () as Object
Description: Gets the custom assembly definition.
Modifiability: Read

Parent ( ) as Object
Description: Gets the parent PropertyDescriptions object.
Modifiability: Read

Object ( ) as Object
Description: Gets at run-time the member object, when the property is used in the context of a member, or the custom assembly occurrence, when the property is used in the context of the aggregator.
Modifiability: Read

CAO ( ) as Object
Description: Gets the custom assembly occurrence at run-time.
Modifiability: Read

# SPSMemberSystemLinear

The Member System is a sequential set of member parts that share a common axis. It is the design parent of other entities, initially the Member Part and two Frame Connections. As a graphic entity, it consists of an axis that is locatable but not displayable. This curve is the geometrical axis for all Member Parts in this system. The MemberSystem is a SystemChild. Member System supports IJDesignParent but not IJSystem. The purpose is to restrict the class of children that the Member System can own. By using a custom relation, we are also able to add children to an approved system, as in the case of a planning split.

If a user deletes a Member Part, the entire Member System is deleted even if it was split into two parts.

This object keeps it's own copy of the associated logical joint. When the user moves the axis it transforms these positions and the AxisSolver can then compute a constrained position. The notification semantic then updates the physical Axis, after the FrameConnections have computed the offsets, which are delta-vectors between the logical model and the physical model. These offset vectors are computed by the FrameConnection macros.

The Member System class defines generic "system" behavior and is also a specific axis curve type. Implementation of this class should re-use common "System" functions, while also implementing the specific class for the curve type. A linear Member System supports IJLine. The physical axis (offset) is this line, which is kept up to date by the AxisNotifySemantic.

**SPSMemberFactory**
The member factory object provides the means for constructing a Member System and/or a stand-alone Member Part Prismatic

**Methods**
CreateMemberSystemPrismaticLinear( IUnknown ResourceManager, ISPSMemberSystem **MemberSystem )

| | |
|---|---|
| Description: | Creates a MemberSystem entity, a MemberPart entity and all required second-class objects. |
| Parameters: | [in] ResourceManager object related to the data store.<br>[out,retval] MemberSystem the created Member System |

GetDesignPartsAtPosition( double x, double y, double z, ISPSMemberPartPrismatic **PreviousPart, ISPSMemberPartPrismatic **NextPart )

| | |
|---|---|
| Description: | Returns one or two child MemberParts at the given location. If the position is at the start of the MemberSystem, only NextPart is set. If the position is along the MemberSystem, then PreviousPart and NextPart are set. If the position is at the end of the MemberSystem, only PreviousPart is set. |
| Modifiability | Read |
| Parameters: | [in] x x coordinate of desired position<br>[in] y y coordinate of desired position<br>[in] z z coordinate of desired position |

**ISPSLogicalAxis**
Use the methods in this interface to store, retrieve, and manipulate the logical end points of the member system. The logical end positions are the same as the ending Joints.

**Methods**
SetLogicalStartPoint( double x, double y, double z )

| | |
|---|---|
| Description: | Sets the coordinates of the member system's logical axis start point. |
| Parameters: | [in] x x coordinate value of the start position.<br>[in] y y coordinate value of the start position<br>[in] z z coordinate value of the start position |

GetLogicalStartPoint( double *x, double *y, double *z )

Description: Gets the coordinates of the member system's logical axis start point.

Parameters: [out] x x coordinate value of the start position.
[out] y y coordinate value of the start position
[out] z z coordinate value of the start position

**ISPSMemberType**

Member Parts and the Member System implement this interface. Setting these properties on a MemberSystem will also modify the corresponding properties on its MemberParts. Setting these properties on a MemberPart that is a child of a MemberSystem will also modify the properties on the MemberSystem.

**Properties**

TypeCategory( long TypeCategory )

Description: Sets or returns the Member's TypeCategory. Values should be those defined in the StructuralMemberTypeCategory code list.

Modifiability Read/Write

Parameters:

Type( long Type )

Description: Sets or returns the Member's Type. Values should be those defined in the StructuralMemberType code list.

Modifiability Read/Write

Parameters:

Priority( long Type )

Description: Sets or returns the Member's Priority. Values should be those defined in the StructuralMemberPriority code list.

Modifiability Read/Write

Parameters:

**ISPSMemberPartPrismatic**

A Member Part is a contiguous structural element whose volume is described by sweeping a pre-defined cross-section along a curve, which is usually a line. This kind of geometry is a prismatic solid. Additional operations may be applied to trim or extend the solid.

**Properties**

MemberSystem( ISPSMemberSystem **MemberSystem )

Description: Returns the MemberSystem that is the design parent of the MemberPart, if it is a child of a MemberSystem.

Modifiability Read

Parameters:

MemberType( ISPSMemberType **MemberType )

Description: Returns the ISPSMemberType interface for the Member Part

Modifiability Read

Parameters:

MaterialDefinition( ISPSMemberType **MemberType )

Description: Sets or returns the MaterialDefinition related to the Member Part

Modifiability Read/Write

Parameters:

CrossSection( ISPSCrossSection **CrossSection )

Description: Returns the ISPSCrossSection interface for the Member Part.

Modifiability Read

Parameters:

PointAtEnd( SPSMemberAxisPortIndex whichEnd, IJPoint **Point )

Description: Returns the IJPoint interface of the logical connection object at the part's end, if the part is a member of a MemberSystem.

Modifiability Read

Parameters: [in] whichEnd tells which end at which the corresponding point is to be returned

**ISPSAxisRotation**

ISPSAxisRotation is the interface used to set and retrieve information related to rotation of SPS Members

**Properties**

Mirror( VARIANT_BOOL bMirror )

Description: Sets or returns whether the cross-section orientation is mirrored. If so, the cross-section's horizontal axis is reversed.

Modifiability Read/Write

Parameters:

**Methods**

SetOrientationVector( double x, double y, double z )

Description: Uses the input orientation vector to set the BetaAngle.

Parameters: [in] x x value of orientation vector
[in] y y value of orientation vector
[in] z z value of orientation vector

# Spatial Entity

The Space Generic Entity is an aggregatable persistent object. This can be aggregated by different kinds of space objects like Area, Zone, Interference Volume and Drawing Volume. It implements IJSpaceCreation interface to able to set and get the inputs and to get the Active Entity used to create the space object. It will have a relation with Space Generic Geometry object, which has graphics. The spatial entity is created by the SpaceFactory object.

**IJDSpacePrimitiveFactory**
Creates a Space Primitive Entity

**Methods**
CreateSpacePrimitive ( byval Part as IJDPart , byval ResourceMgr as Unknown ) as Object
Description: Creates the the SpacePrimitive object and add it to Resource manager.Establishes the relation between SpacePrimitive and Part.
Parameters:
[in] Part Part object to which new SpacePrimitive object associates
[in] ResourceMgr IUnknown Pointer to Resource Manager

**IJDSpaceFolderFactory**
Creates spatial node based on the input object type and input nodetype

**Methods**
CreateEntity ( byval ObjectType as SpaceFactoryObjectTypes , byval nodeType as SpaceNodeType , byval SpaceName as String , byval Parent as Unknown , byval Connection as Unknown ) as Object
Description: Creates spatial node based on the input object type and input nodetype
Parameters:
[in] ObjectType Type of the SpaceFactory Object (see specific types definition)
[in] nodeType Type of the root node
[in] SpaceName Name of the SpaceFolder Object
[in] Parent The parent of the object to be created
[in] Connection

**IJDSpaceFactory**
Creates a Spatial Entity

**Methods**
CreateEntity ( byval strSpaceTypeProgID as String , byval Parent as Unknown , byval SpacePart as LPPART ) as Object
Description: Creates Spatial Entity based on the input type
Parameters:
[in] strSpaceTypeProgID Type of the space to be created
[in] Parent parent of the created space
[in] SpacePart Part to be connected to the space

**IJSpaceCreation**
Set/remove the inputs on the space

**Methods**
SetInputs ( byval bstrAEProgID as String , byval CreationType as short , byval ParentCollection as Object , byval vMiscArg as Variant )
Description: Set the inputs before creating space
Parameters:
[in] bstrAEProgID ProgID of the active entity to be created
[in] CreationType CreationType (2pts/4pts/etc)
[in] ParentCollection Points in the case of Space By Points Cmd etc...

[in] vMiscArg Miscellanious

**SpaceAssociationAEFactory Object**

Get (create, if not exists) the AssociationAE associated with the Object(Space or Graphic Entity) and to disconnect either the space or the graphic entity from AE

**Properties**

ActiveEntity ( ) as Object
Description: It gets the AssociationAE associated with the Object(Space or Graphic Entity).It creates the AE if it does not exist, already
Modifiability: Read Only

**IJSpaceAssociationAE**

Set the spaces or objects on the AssociationAE.It helps to associate a collection of spaces with the desired graphic object

**Properties**

ObjectMatrix ( ) as IJDT4x4
Description: Gets/sets the object matrix
Modifiability: Read/Write

AssociatedSpaces ( ) as IJElements
Description: Gets/sets the requested spaces on the AssociationAE
Modifiability: Read/Write

# NamingRulesHelper Object

This is the helper object that implements the IJDNamingRulesHelper interface to query the naming rules for an object type, to create naming relations, and to query for the active naming rule. This is implemented in the middle tier so that both application commands and business objects can use this implementation.

**References**
Object Library: Ingr Sp3d Generic NamingRules Helper 1.0

**Interfaces**

| Interface Name | lang | Description |
|---|---|---|
| IJDNamingRulesHelper | vb/c | This is the helper interface with the methods that can be used by application commands and business objects for defining naming rules for their objects. |

**IJDNamingRulesHelper**
This is a helper interface that can be used to query the naming rules for an object type, to create naming relations, and to query for the active naming rule. The functionality of this interface is accessed by adding a project reference to the "Ingr Sp3d Generic NameRuleSemantics 1.0 Type Library".
This interface inherits from IDispatch.

**When To Use**
The Visual Basic® NamingRulesHelper Object implements all of the helper functions. This implementation can be used as long as the applications are using the generic naming rules semantic.

**Methods**

GetEntityNamingRulesGivenName ( byval strEntityName as String ) as IJElements
Description: It returns a reference (as NamingRules) to the IJElements interface of the first object in a collection of the naming rules available in the catalog database for the given object name input.
Parameters:
[in] strEntityName Class(object) name(internal name).

GetEntityNamingRulesGivenProgID ( byval strEntityProgID as String ) as IJElements
Description: It returns a reference (as NamingRules) to the IJElements interface of the first object in a collection of the naming rules available in the catalog database for the given object class ProgID input.
Parameters:
[in] strEntityProgID Object class ProgID.

AddNamingRelations ( byval pDispEntity as Object , byval pNameRuleHolder as IJDNameRuleHolder ) as IJNameRuleAE
Description: Adds naming relations "NamedEntity" and "EntityNamingRule" after creating the Active Entity and returns a reference (as pActiveEntity) to the interface of the active entity object created. The method deletes the Active Entity if it is there before creating the new one so it can also be used to delete the relations. If nothing is sent as the pNameRuleHolder argument, the method deletes the existing relations.
Parameters:
[in] pDispEntity The IDispatch interface of the object to be named.
[in] pNameRuleHolder The interface of the NamingRule.

GetActiveNamingRule ( byval pDispEntity as Object ) as IJDNameRuleHolder
Description: This method returns a reference (as pNameRuleHolder) to the interface of the active naming rule that is being used for naming the input object from the relations. pNameRuleHldr will be nothing if there are no active naming rules on the object.

Parameters:
[in] pDispEntity The IDispatch interface of the named object.

IsGeneratedNameUnique ( byval oEntity as LPDISPATCH , byval oFilter as IJSimpleFilter , byval strGenName as String , optional byval strIID as String , optional byval strAttributeName as String ) as Boolean

Description: This method returns a boolean value (as pVal) indicating whether the generated name is unique in the domain specified by the user through the oFilter. True indicates the name is unique.
The optional arguments strIID and strAttribute Name are to be provided by the users of this function. They are provided so as to give an option to the user to specify the Interface and also the Attribute of the object on which the name uniqueness has to be ensured.

Parameters:
[in] oEntity The IDispatch interface of the named object.
[in] oFilter The interface of the Filter to use in determining the uniqueness.
[in] strGenName The generated name string.
[in] strIID An optional IID as a string to help in making the determination. If the IID is provided then strAttributeName has to be provided. Default value is null string.
[in] strAttributeName An optional AttributeName as a string to help in making the determination. Default value is null string.

Return error codes:
E_FILTER_NOT_SPECIFIED The Filter was not specified.

# Attribute Helper service

**CollectionHlp**
The role of this object is to operate on one instantiated collection of attributes. A CollectionHlp object is returned by most of the methods of the IJDAttributes and IJAttributes interfaces. A collection of attributes maps to an interface definition, i.e., it gathers all the properties that belong to an interface.

**References**
Object Library: Ingr SmartPlant 3D Attributes 1.0 Type Library

**Interfaces**

| Interface Name | lang | Description |
|---|---|---|
| IJDAttributesCol | vb/c | Visual Basic® Interface used to manipulate a collection of attributes. |

**IJDAttributesCol**
This interface is used to get information from an item or items in a collection of attributes.
This interface inherits from IDispatch.

**When To Use**
Call this interface when you want to:
Access an item of a collection of attributes.
Access all the items of a collection of attributes.
Count the items of a collection.
Get the metadata about a collection of attributes.

**Properties**

Item ( byval VItem as Variant ) as IJDAttribute

| | |
|---|---|
| Description: | Returns the IJDAttribute interface of the attribute as ppAttribute. Note that: The For Each loop is the preferred implementation to iterate through a collection instead of using a simple index because the DispatchID is NOT a sequential list (1, 2, 3, ...). |
| Modifiability: | Read Only |
| Parameters: | |
| [in] VItem | The VItem can be the DispatchID of the attribute or its name. |
| Return error codes: | |
| S_OK | Operation succeeded. |
| E_FAIL | Operation failed (no detail). |

_EnumItem ( ) as LPUNKNOWN

| | |
|---|---|
| Description: | Enumerates all the attributes of this collection by returning ppEnumUnk. |
| Modifiability: | Read Only |
| Return error codes: | |
| S_OK | Operation succeeded. |
| E_FAIL | Operation failed (no detail). |

InterfaceInfo ( ) as IJDInterfaceInfo

| | |
|---|---|
| Description: | Returns ppInfo, the IJDInterfaceInfo interface of an InterfaceInfo Object for this collection. |
| Modifiability: | Read Only |
| Return error codes: | |
| S_OK | Operation succeeded. |
| E_FAIL | Operation failed (no detail). |

Count ( ) as Long
Description: Returns the number of attributes of this Collection.
Modifiability: Read Only
Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).

**IJDAttributes**
This interface is used to get a CollectionOfAttributes property. This interface is implemented by any component that is attributes-enabled and aggregates the AttributeHelper object.

**<u>When To Use</u>**
Call this interface when you want to access the CollectionOfAttributesproperty of an object.

**Properties**

CollectionOfAttributes( byval InterfaceType as Variant ) as IJDAttributesCol
Description: Returns a pointer (ppIAttributesCol) to the IJDAttributesCol interface of the CollectionHlp Object (collection of attributes).
If the UserTypeCLSID property was set to an acceptable value, the method checks to see that this collection is allowed for this UserType according to the metadata.
If UserTypeCLSID is set to CLSID_NULL, the method only checks to see that this collection/Interface is described in the metadata.
Modifiability: Read Only
Parameters:
[in] InterfaceType The InterfaceType is a variant that contains a string with the formatted hexa value of the IID : "{24E1A26B-1275-11d2-A684-00A0C96F81B9}", or with the interface name IID : "IJGeometry", or a GUID structure.
Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).
E_NOINTERFACE The interface is not implemented by the UserType class. The AttributesCol is set to NULL in this case.

Count ( ) as Long
Description: Returns the number of collections of this object.
Modifiability: Read Only
Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).

**Attribute**
The role of this object is to operate on one instantiated attribute. The Attribute object is returned by most of the methods of the IJDAttributesCol interface.

**References**
Object Library: Ingr SmartPlant 3D Attributes 1.0 Type Library

**Interfaces**

| Interface Name | lang | Description |
|---|---|---|
| IJDAttribute | vb/c | Visual Basic® Interface used to manipulate an attribute |

IJDAttribute
This interface is used to manipulate the value of an attribute.
This interface inherits from IDispatch.

**When To Use**

Call this interface when you want to:
Access the value of an attribute.
Get the metadata about an attribute.

**Properties**

| | |
|---|---|
| Value ( ) as Variant | |
| Description: | Allows you to get or set the value of an attribute. The method using this property is the generic way to access the value of an attribute. It is not responsible to check and see if the caller is allowed to write in this field. If one uses put_Value with Val.vt = VT_NULL or VT_EMPTY, the attribute is removed from the database. For Hierarchical Code Lists, if one uses put_Value with val.vt = VT_BSTR (implying that the ShortString value has been passed), it is automatically converted to the ValueID (val.vt = VT_I4). If one uses get_Value on a removed attribute, the returned variant will have its vt flag set to VT_EMPTY. This confusion of the VT_EMPTY and VT_NULL flag allows us to save database space. See the Specific Types Definition below for the definitions. |
| Modifiability: | Read/Write |
| Return error codes: | |
| S_OK | Operation succeeded. |
| E_FAIL | Operation failed (no detail). |

| | |
|---|---|
| AttributeInfo ( ) as IJDAttributeInfo | |
| Description: | Returns the IJDAttributeInfo interface of an AttributeInfo object for this attribute. |
| Modifiability: | Read Only |
| Return error codes: | |
| S_OK | Operation succeeded. |
| E_FAIL | Operation failed (no detail). |

**Specific Types Definition**

| | |
|---|---|
| Enum tagSQLTypes | |
| SQL_VB_CHAR = 1 | // CHAR, VARCHAR, DECIMAL, NUMERIC = VT_BSTR = SQL_C_CHAR = SQL_CHAR |
| SQL_VB_LONG = 4 | // long int = VT_I4 = SQL_C_LONG = SQL_INTEGER |
| SQL_VB_SHORT = 5 | // shrt int = VT_I2 = SQL_C_SHORT = SQL_SMALLINT |
| SQL_VB_FLOAT = 7 | // float = VT_R4 = SQL_C_FLOAT = SQL_REAL |
| SQL_VB_DOUBLE = 8 | // double = VT_R8 = SQL_C_DOUBLE = SQL_DOUBLE |
| SQL_VB_BIT = -7 | // boolean = VT_BOOL = SQL_C_BIT |
| SQL_VB_DATE = 9 | // date = VT_DATE = SQL_C_DATE |
| End Enum | |

Note about tagSQLTypes : The type of the attribute is defined in the METADATALib in terms of SQL_C_Types. The value of an attribute is a VARIANT. We use the correspondence table above. If the type of the VARIANT does not match the VT type, we try to coerce it using MS API VariantChangeType. If the attribute is hard coded, the coercion is done by the MS API invoke.

**IJDCodeListMetaData**

This interface is used to access the codelist metadata and is exported in the COM map of the business object that aggregates the attribute helper. The method calls are delegated to the POM.
This interface inherits from IDispatch.

**When To Use**

Call this interface when you want to access the metadata about a codelist.

**Properties**

ShortStringValue ( byval TableName as String , byval ValueID as Long ) as String
Description: Gets the short string of a codelist.
Modifiability: Read Only
Parameters:
[in] TableName Name of the table. Can be either Namespace:TableName (i.e., PackageName:TableName) or just TableName. When there are two tables with same name in different packages and no namespace is specified, an error will be returned.
[in] ValueID Index of the codelist in the table.
Return error codes:
S_OK Operation succeeded, ShortString returned.
S_FALSE Operation succeeded, no ShortString returned.
E_FAIL (1) No TableName is provided; (2) Duplicated TableNames are found in Metadata database (need Namespce); (3) Operation failed for other reasons.
Note: This API returns S_FALSE if the CodelistTable does not exist or the CodelistTable does not have ValueID as its value.

LongStringValue ( byval TableName as String , byval ValueID as Long ) as String
Description: Gets the long text string of a codelist.
Modifiability: Read Only
Parameters:
[in] TableName Name of the table. Can be either Namespace:TableName (i.e., PackageName:TableName) or just TableName. When there are two tables with same name in different packages and no namespace is specified, an error will be returned.
[in] ValueID Index of the codelist in the table.
Return error codes:
S_OK Operation succeeded, longString returned.
S_FALSE Operation succeeded, no longString returned.
E_FAIL (1) No TableName is provided; (2) Duplicated TableNames are found in Metadata database (need Namespce); (3) Operation failed for other reasons.
Note: This API returns S_FALSE if the CodelistTable does not exist or the CodelistTable does not have ValueID as its value.

ParentValueID ( byval TableName as String , byval ValueID as Long ) as Long
Description: Gets the ParentValueID of a codelist. Returns -1 in case a valid ValueID does not have a ParentValueID.
Modifiability: Read Only
Parameters:
[in] TableName Name of the table. Can be either Namespace:TableName (i.e., PackageName:TableName) or just TableName. When there are two tables with same name in different packages and no namespace is specified, an error will be returned.
[in] ValueID Index of the codelist in the table.
Return error codes:
S_OK Operation succeeded, ParentValueID returned.
S_FALSE Operation succeeded, no ParentValueID returned.
E_FAIL (1) No TableName is provided; (2) Duplicated TableNames are found in Metadata database (need Namespce); (3) Operation failed for other reasons.
Note: This API returns S_FALSE if the CodelistTable does not exist or the CodelistTable does not have ValueID as its value.

CodelistValueCollection ( byval TableName as String ) as IJDInfosCol
Description: Returns (pEnumCodeList as RetVal) the IJDInfosCol interface of the first item of the collection of tables. The IJDInfosCol is a collection of IJDCodelistValue.
Modifiability: Read Only
Parameters:

[in] TableName Name of the table. Can be either Namespace:TableName (i.e., PackageName:TableName) or just TableName. When there are two tables with same name in different packages and no namespace is specified, an error will be returned.
Return error codes:
S_OK Operation succeeded.
E_INVALIDARG No TableName provided.
E_FAIL (1) Duplicated TableNames are found in Metadata database (need Namespce); (2) Operation failed for other reasons.
Note: This API returns a codelist value collection cotaining "Unidentified" if a non-existing Codelist table name is passed in.

ChildValueCollection ( byval TableName as String , byval ValueID as Long ) as IJDInfosCol
Description: Returns (pEnumCodeList as RetVal) the IJDInfosCol interface of the first item of the collection of tables associated with a specific ValueID. The IJDInfosCol
Modifiability: Read Only
Parameters:
[in] TableName Name of the table. Can be either Namespace:TableName (i.e., PackageName:TableName) or just TableName. When there are two tables with same name in different packages and no namespace is specified, an error will be returned.
[in] ValueID Index of the codelist in the table.
Return error codes:
S_OK Operation succeeded.
S_FALSE TableName does not have a ChildTable.
E_FAIL (1) TableName has duplicates in Metadata; (2) Operation failed for other reasons (no detail).

ParentTable ( byval TableName as String ) as String
Description: Gets ParentTable name of a given a codelist table.
Modifiability: Read Only
Parameters:
[in] TableName Name of the table. Can be either Namespace:TableName (i.e., PackageName:TableName) or just TableName. When there are two tables with same name in different packages and no namespace is specified, an error will be returned.
Return error codes:
S_OK Operation succeeded, ParentTable returned.
S_FALSE Operation succeeded, no ParentTable returned.
E_CL_TABLENAMEDUPLICATED TableName has duplicates in Metadata database.
E_FAIL More than one ParentTable name is found (require namespace); Operation failed (no detail).

ChildTable ( byval TableName as String ) as String
Description: Gets ChildTable name of a given a codelist table.
Modifiability: Read Only
Parameters:
[in] TableName Name of the table. Can be either Namespace:TableName (i.e., PackageName:TableName) or just TableName. When there are two tables with same name in different packages and no namespace is specified, an error will be returned.
Return error codes:
S_OK Operation succeeded, ChildTable returned.
S_FALSE Operation succeeded, no ChildTable returned.
E_CL_TABLENAMEDUPLICATED TableName has duplicates in Metadata database.
E_FAIL More than one ChildTable name is found (require namespace); Operation failed (no detail).

TableDescription ( byval TableName as String ) as String
Description: Gets the description of the codelist table.
Modifiability: Read Only
Parameters:
[in] TableName Name of the table. Can be either Namespace:TableName (i.e., PackageName:TableName) or just TableName. When there are two tables with same name in different packages and no namespace is specified, an error will be returned.
Return error codes:
S_OK Operation succeeded, TableDescription returned.
S_FALSE Operation succeeded, no TableDescription returned.
E_CL_TABLENAMEDUPLICATED TableName has duplicates in Metadata database.
E_FAIL More than one ChildTable name is found (require namespace); Operation failed (no detail).

TableCollection ( ) as Unknown
Description: Returns (pEnumCodeList as RetVal) the IUnknown interface of the first item of the collection of tables. Gets an enumerated collection of CodeList tables.
Modifiability: Read Only
Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).
Note: This API returns S_OK no matter if a TableCollection is reurned or not.

ValueIDByShortString ( byval TableName as String , byval ShortStringValue as String ) as Long
Description: Returns the ValueID of a codelist entry given the codelist TableName and the ShortStringValue of the entry.
Modifiability: Read Only
Parameters:
[in] TableName Name of the table. Can be either Namespace:TableName (i.e., PackageName:TableName) or just TableName. When there are two tables with same name in different packages and no namespace is specified, an error will be returned.
[in] ShortStringValue The short string value of a codelist.
Return error codes:
S_OK Operation succeeded, ValueId returned.
S_FALSE Operation succeeded, no ValueId returned.
E_INVALIDARG No TableName or ShortString is provided.
E_FAIL More than one TableName is found in Metadata database (require namespace); Operation failed (no detail).

# Relation Helper service

### DRelationHelper

In the MS repository model of relationships, the Automation object CollectionHelper can be retrieved from any component that is relationships-enabled by getting the CollectionRelations property of the interface that the relationship is established to.

**References**
Object Library: Ingr SmartPlant 3D Relation 1.0 Type Library

**Interfaces**

| Interface Name | lang | Description |
|---|---|---|
| IJDAssocRelation | vb/c | Visual Basic® Interface used to access a CollectionOfRelations property. |
| IJDTargetObjectCol | vb/c | Dual interface to manipulate the collection of target objects. |
| IJDRelationshipCol | vb/c | Dual interface to manipulate the collection of relationships. |

**IJDAssocRelation**
This interface accesses the Collection of Relations in which a business object participates. It should be implemented by any business object that is relationship-enabled.
The relationship types are defined between interfaces of the two participant objects, and that relationships are gathered per homogenous collections. The Core uses this alternative accessor as an interface on the business object where both the interface and the property are input arguments when asking for the collection. This interface inherits from IDispatch.

**When To Use**
Call this interface when you want to access a collection of relationships on a business object.

**Properties**

CollectionRelations ( byval InterfaceID as Variant , byval CollectionName as String ) as Object

| | |
|---|---|
| Description: | Returns the IDispatch interface of the Collection of relationships. This collection should implement the interfaces IJDRelationshipCol and IJDTargetObjectCol. If using the provided RelationHelper Object, the returned object is of the type CollectionHelper Object. |
| Modifiability: | Read Only |
| Parameters: | |
| [in] InterfaceID | IID that the collection is associated to. This variant contains a string with the formatted hexa value of the IID : "{24E1A26B-1275-11d2-A684-00A0C96F81B9}" or with the interface name IID : "IJGeometry", or a GUID structure. |
| [in] CollectionName | Name of the collection. |
| Return error codes: | |
| S_OK | Operation succeeded. |
| E_FAIL | Operation failed (no detail). |

**IJDRelationshipCol**
This is one of the two basic interfaces that collections of relationships should implement.
This interface inherits from IDispatch.

**When To Use**
Use this interface to manage the relationships that belong to a particular relationship collection. This includes the set of relationships that:
Is of the same type.
Is attached to a particular source object.

Have objects playing the same role, have the same origin, or the same destination in the relationship.
With this interface, you can:
Get a count of the number of relationships in the collection.
Add and remove relationships to and from the collection.
If the collection is sequenced (which requires it to be an origin collection), place a relationship in a specific spot in the collection sequence or modify the sequencing of the collection.
Retrieve a specific relationship from the collection.
Obtain information about the collection and the relation to which it is associated.

**Methods**

Add ( byval TargetObject as Unknown , byval Name as String ) as IJDRelationship

| | |
|---|---|
| Description: | Adds a relationship between the source object containing this collection of relationships and the given target object. Returns the IJDRelationship interface (CreatedRelationship) of the created relationship. If the business object is aggregating a RelationHelper Object, this object is a RelationshipHelper Object. Following the Repository API, if the relationship is of the ordered type, the added relationship is always added at the end of the existing ones. |
| Parameters: | |
| [in] TargetObject | Target Object to be connected. |
| [in] Name | Name of the relationship. This requires the relation to support naming. |
| Return error codes: | |
| S_OK | Operation succeeded. |
| S_FAIL | Operation failed (no detail). |
| E_OBJECTS_NOT_WITHIN_SAME_DB | The error is returned when DBContainment flag on relation metadata is WITHIN_DB and a relation is being created between objects belonging to different databases. |

Insert ( byval TargetObject as Unknown , byval Index as Long , byval Name as String ) as IJDRelationship

| | |
|---|---|
| Description: | Adds a relationship between the source object containing this collection of relationships and the given target object. Returns the IJDRelationship interface (CreatedRelationship) of the inserted relationship. If the business object is aggregating a RelationHelper Object, this object is a RelationshipHelper Object. This method can only be used when the origin side of the relation supports ordering. |
| Parameters: | |
| [in] TargetObject | Target object to be connected. |
| [in] Index | Index of the new relationship. |
| [in] Name | Name of the relationship. |
| Return error codes: | |
| S_OK | Operation succeeded. |
| S_FAIL | Operation failed (no detail). |

IsSourceOrigin ( )

| | |
|---|---|
| Description: | Returns if the source (i.e., the object that the collection has been retrieved from) is the origin of the relationships contained by the collection. |
| Return error codes: | |
| S_OK | Source is origin in the relationships. |
| S_FALSE | Source is destination in the relationships. |

Remove ( byval TargetItem as Variant )

| | |
|---|---|
| Description: | Remove a relationship. |
| Parameters: | |
| [in] TargetItem | Identifies the Relationship to be removed by an index of type long or by a string (BSTR) when the relation supports unique naming and requires the collection to be the origin of the relation. |

Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).

Move ( byval oldIndex as Long , byval newIndex as Long )
Description: Move a relationship in a sequenced origin colelction.
Parameters:
[in] oldIndex Identifies the relationship to be moved by it's index.
[in] newIndex Identifies the index to which the relation should be moved.
Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).

Refresh ( )
Description: Refresh the collection with the current data from the database.
Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).
Note: That method refreshs only a non associative collection. The method does nothing for an associative relation.

**Properties**

Count ( ) as Long
Description: Returns the count of relationships.
Modifiability: Read Only
Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).

Infos ( InterfaceID as Variant , pCollectionName as String )
Description: Returns the name of the collection and the interface that the collection is associated to.
Modifiability: Read Only
Parameters:
[out] InterfaceID The IID of the interface with which the collection is associated.
[out] pCollectionName The name of the collection.
Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).

Item ( byval TargetItem as Variant ) as IJDRelationship
Description: Returns the IJDRelationship interface of an object describing the requested relationship. If using the provided helpers, this object is a RelationshipHelper.
Modifiability: Read Only
Parameters:
[in] TargetItem Either the name or the index.
Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).
Note: The TargetItem value identifies the relationship to be returned by a string (BSTR) when the relation supports unique naming and requires the collection to be origin of the relation or by an index of type long.

ItemByKey ( byval Key as String ) as IJDRelationship
Description: Returns the IJDRelationship interface of an object describing the requested relationship. If using the provided helpers, this object is a RelationshipHelper.

Modifiability: Read Only
Parameters:
[in] Key The relation key relative to the origin collection.
Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).
Note: This property requires the collection to be the origin of the relation.

Source ( ) as Unknown
Description: Returns the IUnknown interface of the source object. This is the object that the collection of relationships is associated to.
Modifiability: Read Only
Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).

Type ( ) as Variant
Description: Returns the GUID identifying the relation to which the current collection is associated. Then the interface IJRelationMetaData on the source of the collection permits access to the complete meta-data information of this relation type.
Modifiability: Read Only
Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).

### IJDTargetObjectCol

This is one of the two basic interfaces that collections of relationships should implement.
With this interface, you can:

Get a count of the number of destinations in the collection.
Add and remove relationships to and from the collection.
If the collection is sequenced (which requires it to be an origin collection), place a relationship in a specific spot in the collection sequence, or modify the sequencing of the collection.
Retrieve a specific relationship from the collection.
Obtain information about the collection and the relation with which it is associated.

This interface inherits from IDispatch.

### When To Use

Use this interface to manage the objects that are the destination of a particular relationship collection. This is the set of objects that are related to the source object (from which the current collection has been retrieved) by relationships:
of the same type.
attached to this particular source object.
where the objects in the relationship play the same role, origin, or destination.

### Methods

Add ( byval TargetObject as Unknown , byval Name as String , byval CreatedRelationship as IJDRelationship )
Description: Adds a relationship between the source object containing this collection of relationships and the given target object. Following the Repository API, if the relationship is of the ordered type, the added relationship is always added at the end of the existing ones.
Parameters:

[in] TargetObject Target Object to be connected.
[in] Name Name of the relationship.
[in] CreatedRelationship Pointer to the created relationship. If the business object is aggregating a RelationHelper , this object is a RelationshipHelper.
Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).
E_OBJECTS_NOT_WITHIN_SAME_DB The error is returned when DBContainment flag on relation metadata is WITHIN_DB and a relation is being created between objects belonging to different databases.

Insert ( byval TargetObject as Unknown , byval Index as Long , byval Name as String , byval CreatedRelationship as IJDRelationship )
Description: Adds a relationship between the source object containing this collection of relationships and the given target object. This method could only be used when the origin side of the relationship supports ordering.
Parameters:
[in] TargetObject Target object to be connected.
[in] Index Index of the new relationship.
[in] Name Name of the relationship.
[in] CreatedRelationship Pointer to the created relationship. If the business object is aggregating a RelationHelper, this object is a RelationshipHelper.
Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).

IsSourceOrigin ( )
Description: Returns if the source (i.e., the object that the collection has been retrieved from) is the origin of the relationships contained by the collection.
Return error codes:
S_OK Source is origin in the relationships.
S_FALSE Source is destination in the relationships.

Move ( byval ActualIndex as Long , byval NewIndex as Long )
Description: Moves the relationship to another location (for sequenced relations).
Parameters:
[in] ActualIndex The index before the move where it actually is.
[in] NewIndex The index to move it to.
Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).

Remove ( byval TargetItem as Variant )
Description: Removes a relationship.
Parameters:
[in] TargetItem Identifies the Relationship to be removed by: - a string (BSTR) when the relation supports unique naming (requiring the collection to be the origin of the relation). - an index (long).
Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).

EnumTargetMoniker ( byval ppEnumMoniker as LPENUMMONIKER * )
Description: Enumerates monikers of target objects.
Parameters:
[in] ppEnumMoniker Enumerates monikers of target objects. This enumeration will be sometimes useful in

avoiding binding all target objects. This enumeration can be used in VB also (see code example below).

Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).

**Properties**

Count ( ) as Long
Description: Returns the count of target entities.
Modifiability: Read Only
Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).

Infos ( byval InterfaceID as Variant ) as String
Description: Returns the name of the collection and the interface that the collection is associated to.
Modifiability: Read Only
Parameters:
[in] InterfaceID The InterfaceID value passed out is the IID of the interface with which the collection is associated.
Return error codes:
S_OK Operation succeeded.
S_FAIL Operation failed (no detail).

Item ( byval TargetItem as Variant ) as Unknown
Description: Returns the IUnknown interface of a target object.
Modifiability: Read Only
Parameters:
[in] TargetItem TargetItem value passed in identifies the Relationship to be removed by: - a string (BSTR) when the relation supports unique naming (requiring the collection to be the origin of the relation). - an index (long).
Return error codes:
S_OK Operation succeeded.
E_ACCESSDENIED Access to the target is denied.
E_FAIL Operation failed (no detail).

Source ( ) as Unknown
Description: Returns the IUnknown interface of the source object.
Modifiability: Read Only
Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).

Type ( ) as Variant
Description: Returns the GUID identifying the relationship with which the current collection is associated. Then use the interface IJRelationMetaData on the source of the collection to have access to the complete metadata information of this relation type.
Modifiability: Read Only
Return error codes:
S_OK Operation succeeded.
E_FAIL Operation failed (no detail).

## SP3D References Tool

The software consists of hundreds of type libraries that provide the programmatic interfaces to the data model and its underlying data. These libraries consist of the data model's interfaces and their methods and properties.

The ability to integrate user-definable components into the environment is a key capability of the software. The mechanism of creating custom commands provides this extensibility.
To reference the available type libraries in Visual Basic:

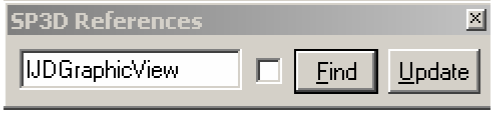
- Click **Project > References**.

To perform the task of referencing your type libraries more quickly and efficiently:

- Click **Project > SP3D References**.

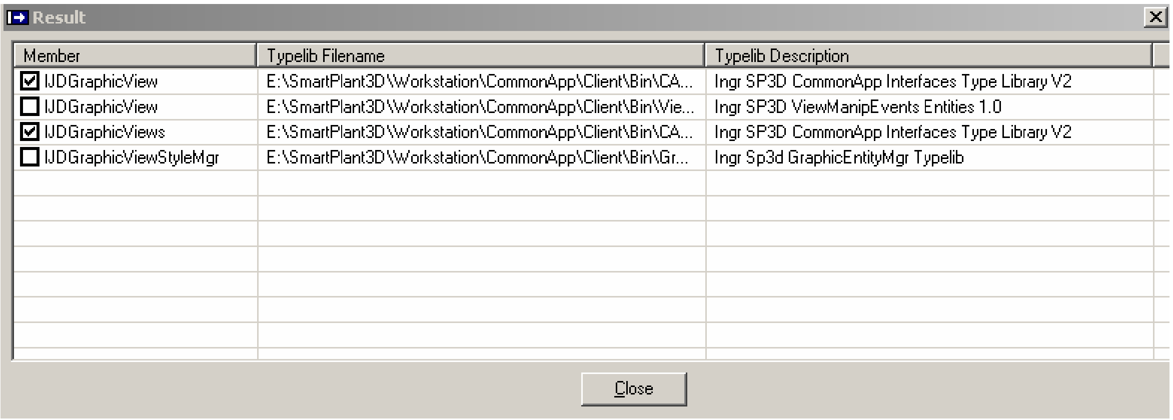
**Using the SP3D References Tool**
The SP3D References tool is a very useful utility that you can use to locate and reference type libraries quickly and easily. You only need to know the name of your class object or variable in which to perform a search.

1. Open Visual Basic.
2. Click **Add-Ins > Add-In Manager**....
3. Select **SP3D References** and make sure that the **Loaded/Unloaded** and **Load on Startup** boxes under **Load Behavior** are both checked.
4. Click **OK**.
5. Click **Project > SP3D References** to invoke the dialog.



IJDGraphicView

6. Enter a class or variable name to search..
7. Click **Find**.



| Member | Typelib Filename | Typelib Description |
|---|---|---|
| ☑ IJDGraphicView | E:\SmartPlant3D\Workstation\CommonApp\Client\Bin\CA... | Ingr SP3D CommonApp Interfaces Type Library V2 |
| □ IJDGraphicView | E:\SmartPlant3D\Workstation\CommonApp\Client\Bin\Vie... | Ingr SP3D ViewManipEvents Entities 1.0 |
| ☑ IJDGraphicViews | E:\SmartPlant3D\Workstation\CommonApp\Client\Bin\CA... | Ingr SP3D CommonApp Interfaces Type Library V2 |
| □ IJDGraphicViewStyleMgr | E:\SmartPlant3D\Workstation\CommonApp\Client\Bin\Gr... | Ingr Sp3d GraphicEntityMgr Typelib |

8. Check the appropriate type libraries.

Note: If this is the first time that you have invoked the tool, it begins reading your system to generate a data file that contains information about all existing registered type libraries.

## Debugging Your Code

No matter how carefully you create your code, errors can occur. To handle these errors, you need to add error-handling code to your procedures.
You perform the process of locating and fixing bugs in applications by *debugging* the code. Visual Basic provides several tools to help analyze how your application operates. These debugging tools are useful in locating the source of bugs, but you can also use the tools to experiment with changes to your application or to learn how other applications work.

Note: You must add the TaskHost project to the integrated development environment (IDE) before you can debug your Visual Basic project.

Before you can use the TaskHost project, you must set new paths in your computer's environment variables. Click Start -> Settings -> Control Panel -> System. Select the Advanced tab and then click Environment Variables. Finally add the following path statements according to the location in which you installed the software:

PATH=[*Product Directory*]\Core\Runtime; [*Product Directory*]\GeometryTopology\Runtime

**Adding the TaskHost Project to your Project**

1. Open your Visual Basic .vbp project to debug.
2. Click File > Add Project.
3. Select the Existing tab.
4. Open SP3DTaskHost.vbp in the following path: ..\Debug\Container\Src\Host
5. In the Project window, right-click over SP3DTaskHost and then select Set as Start Up.
6. Right-click again on SP3DTaskHost and then select SP3DtaskHost Properties...
7. On the Project Properties dialog, change the Project Type to Standard EXE.
8. Set the breakpoint in your project to debug.
9. Click Run and wait for processing to begin. Your Visual Basic project becomes active when the breakpoint is reached.
10. Click to view <your project>, which returns you back to the code view. Then step through your code.

**Important**

Do not stop the debug process by clicking the End command. If you end processing this way, you will throw an exception, crash all the software that is running, and lose your changes.
To safely end processing, click File > Exit from the SmartPlant 3D TaskHost software.