Introduction to .NET Route Symbols

Consulting Services, SmartPlant 3D

Agenda

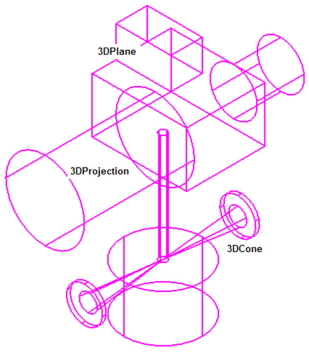
- Overview
- Symbol Definition/ Symbol Occurrences/ Flavor
- Creating VB .Net Symbol Definitions
- Defining Symbol Definition Class, Inputs, Aspects and Outputs
  - Symbol Geometry Helper Component
- Modification of Symbol Definition
- Synchronization between Model & Catalog
- Symbol Deployment and Guidelines
- Debugging your code
- Open Discussion / Follow up Questions

Symbols: What is it ?

……...Reusability of Graphical Entities
Probably the most used entity by Smart 3D

Example:

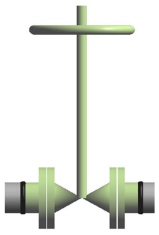
- Piping
- Electrical
- Hvac
- Equipment
- Structure



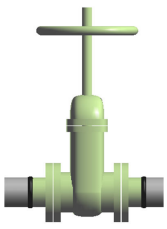
Symbol Overview

Functionality

- Support Multiple Aspects: allows to have several different representations of the same component
- It is up to the symbol definition designer to define the aspects that the symbol will support
- The aspects to be displayed is controlled by the client tier (Format View dialog)



Simple Physical Aspect

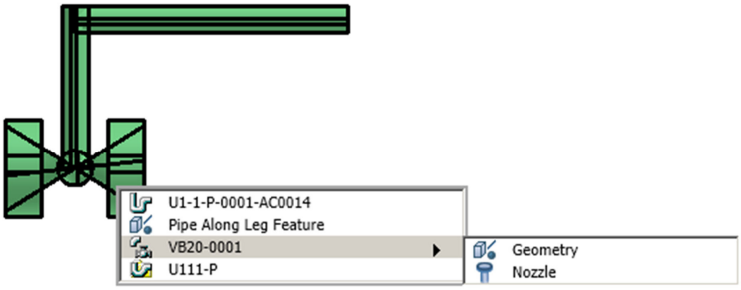


Detailed Physical Aspect

Symbol Overview

Functionality

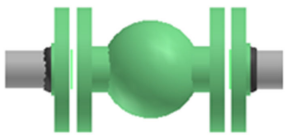
- Support the functionality of penetrating a symbol occurrence at locate time to identify some of its members or constituent



Symbol Overview

Functionality

- Support Nesting: Nesting means that a symbol can be an output of another symbol



SP3DBallValve.CBallValve

SP3DOP9.COP9

Symbol Overview

Terminology

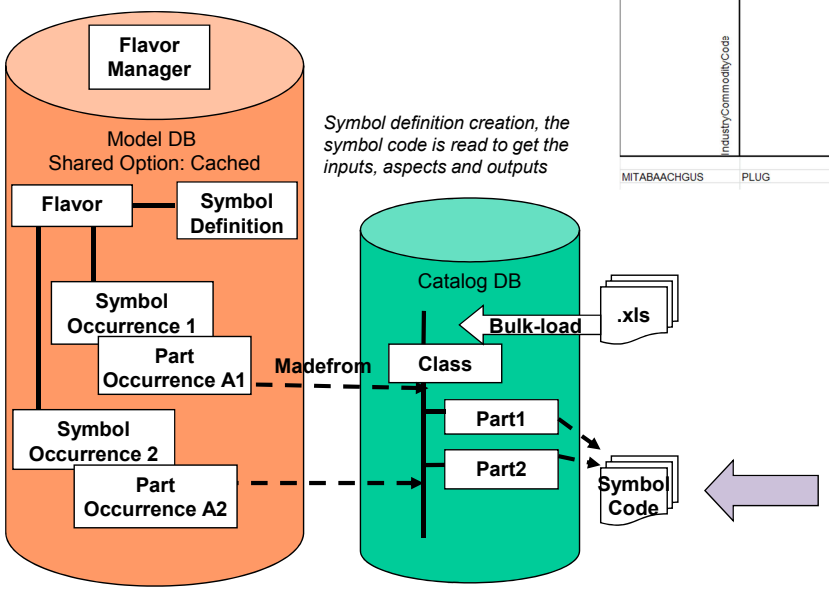
- **Symbol Definition** is a persisted object that describes and implements the behavior shared by all the symbol occurrences referencing it
- **Symbol Occurrence** is an object that has a transformation matrix (offset/rotation)
- **Flavor** holds and caches the actual symbol geometry
  - Flavor is like a cell in Microstation cell library
  - Cache is based on catalog part (identity), progid & dll version, and symbol Inputs (parameters)
  - Cache is used when same part is placed in the model next time with same inputs [symbol not re-evaluated]
  - Most symbols are cached

Symbol Overview

Pre-defined parts are defined in the Catalog

CatalogData.xls

PartClassType	SymbolDefinition	UserClassName	OccClassName	SymbolIcon								
PipeComponentClass	Asm,Ingr.S3D.Content.RS.CPlugType1	Plug Type1	Plug Type1									
	IndustryCommodityCode	CommodityType	GeometryType	GraphicalRepresentationOrNot	SymbolDefinition	Npd[1]	NpdUnitType[1]	FacetoEnd				
MITABAACHGUS	PLUG		220			1	in	1in				



Benefits of using Flavors

- Sharing the Geometry
  - Geometry is created once and shared by many occurrences; less data to store
  - Each occurrence has a transformation matrix
- Symbols are Parametric
  - Same program can be used for generating different flavors for different set of inputs
- Synchronization
  - A symbol can be modified and all the existing occurrences can be updated to show the new symbol

.NET Symbol Definition

- Create a new class which is derived from CustomSymbolDefinition
  - A base class is available to reduce coding for creating a symbol definition
- Declare symbol inputs as member variables
  - Identify properties which control symbol geometry
- Declare symbol aspects as member variables
  - Identify which aspects the symbol is going to support
  - Users can add more aspects. For example,
    - Simple Physical – Geometry of the symbol
    - Detailed Physical - Detailed geometry of the symbol
    - Operation – Geometry of the operation on the object
    - Insulation – Geometry of the insulation on the object
    - Maintenance - Space for maintenance access on the object
- Declare symbol outputs as attributes on each Aspect
- Construct outputs to all aspects
  - Each symbol definition must override the **ConstructOutputs** method from the base class. For each aspect, constructs persistent graphics objects and add them as outputs of the symbol.

Inputs

Catalog Part

Dimensional Parameters
(from Catalog Part or from Occurrence)

Outputs

Named output objects representing **Graphics & Ports**.
[Physical aspect]
Cylinder, Cone
Operator
Port1, Port2

[Insulation aspect]
Cylinder

[Operation aspect]
Cylinder

.NET Symbol Definition

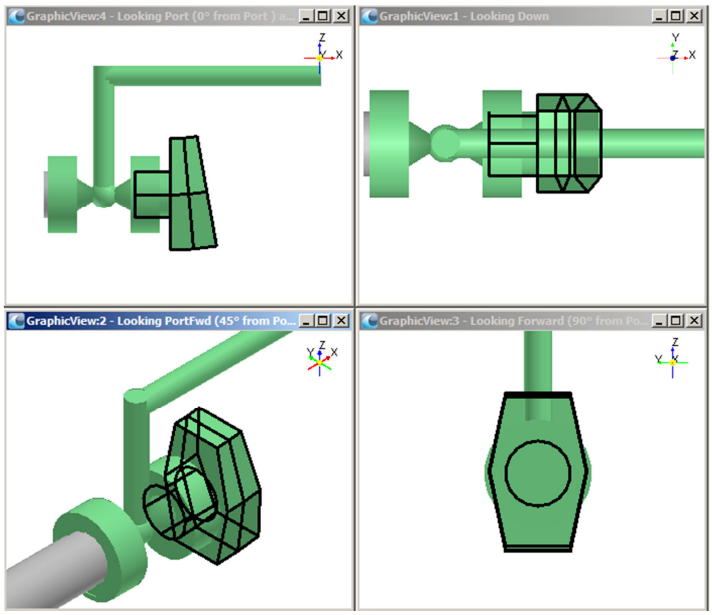
```
Public Class BallValve : Inherits CustomSymbolDefinition

    '----------------------------------------------------------------------------
    'DefinitionName/ProgID of this symbol is "SP3DBallValve,Ingr.SP3D.Piping.BallValve"
    '----------------------------------------------------------------------------

#Region "Definition of Inputs"

    <InputCatalogPart(1)> Public m_oPartInput As InputCatalogPart
    <InputDouble(2, "FacetoFace", "Face to Face", 0.31)> Public m_dFaceToFaceInput As InputDouble
    <InputDouble(3, "OperatorHeight", "Stem Height", 0.15)> Public m_dOperatorHeightInput As InputDouble
    <InputDouble(4, "OperatorLength", "Handle Length", 0.75)> Public m_dOperatorLengthInput As InputDouble
    <InputDouble(5, "InsulationThickness", "Insulation Thickness", 0.025)> Public m_dInsulationThicknessInput As InputDouble

#End Region

#Region "Definitions of Aspects and their outputs"

    <Aspect("Physical", "Physical Aspect", AspectID.SimplePhysical)> _
    <SymbolOutput("Ball", "Body")> _
    <SymbolOutput("Stem", "Stem")> _
    <SymbolOutput("Handle", "Handle")> _
    <SymbolOutput("PNoz1", "PNoz1")> _
    <SymbolOutput("PNoz2", "PNoz2")> _
    <SymbolOutput("Operator", "Operator occurrence")> _
    Public m_oPhysicalAspect As AspectDefinition

#End Region
```


Create a VB.net Class Library Project

- Use Microsoft Visual Studio 2010
  - Choose a .net language → VB.net / C#
    - VB.net is preferred. Most samples are in VB.net. Our examples will be VB.net.
- Project Type
  - Use a Windows “Class Library” project template
  - Create the new symbol in a new **Class Library** Project.
    - Choose File → New Project

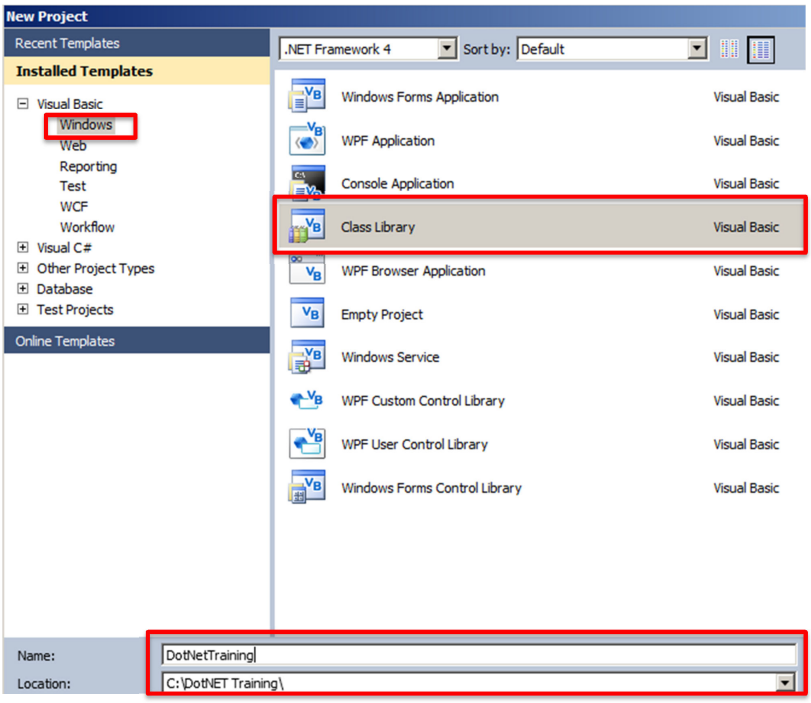
Piping Component Symbol

Lab 1 - Prismatic Torus Plug



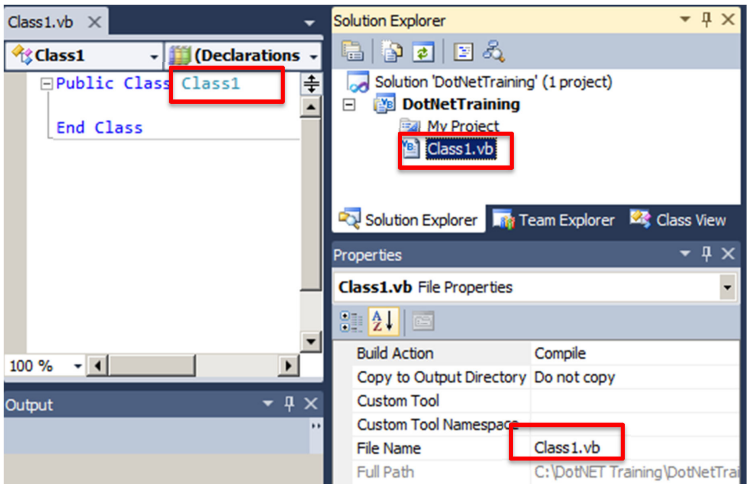
Create a VB.net Class Library Project

- Choose **File** > **New Project**
- Use **Visual Basic** > **Windows** from Project Types
- Pick **Class Library** template
- Specify **Name** of Project
- “DotNetTraining”
- Hit **OK**
- Project gets created



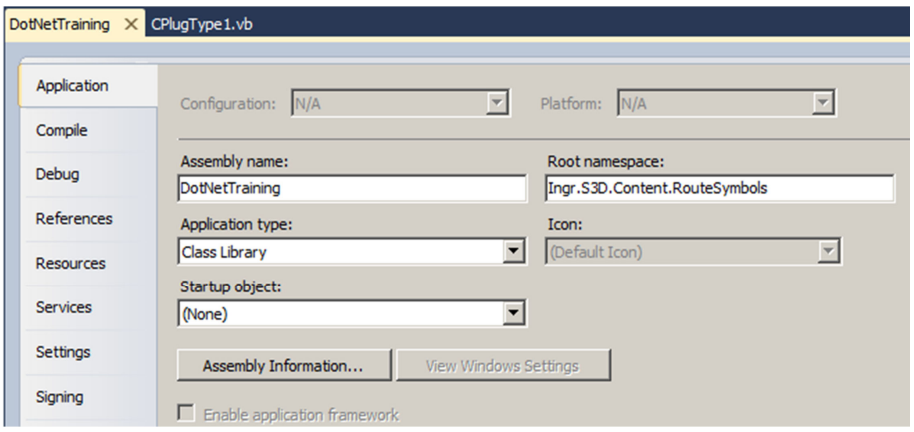
Create a VB.net Class Library Project

- It creates a Class1.vb which has an empty class named Class1.
- Rename the Name of the class from Class1 to a name of your choice and also rename the filename accordingly.
- Lets say we change it to “CPlugType1”, for the Route symbol we will be writing here.

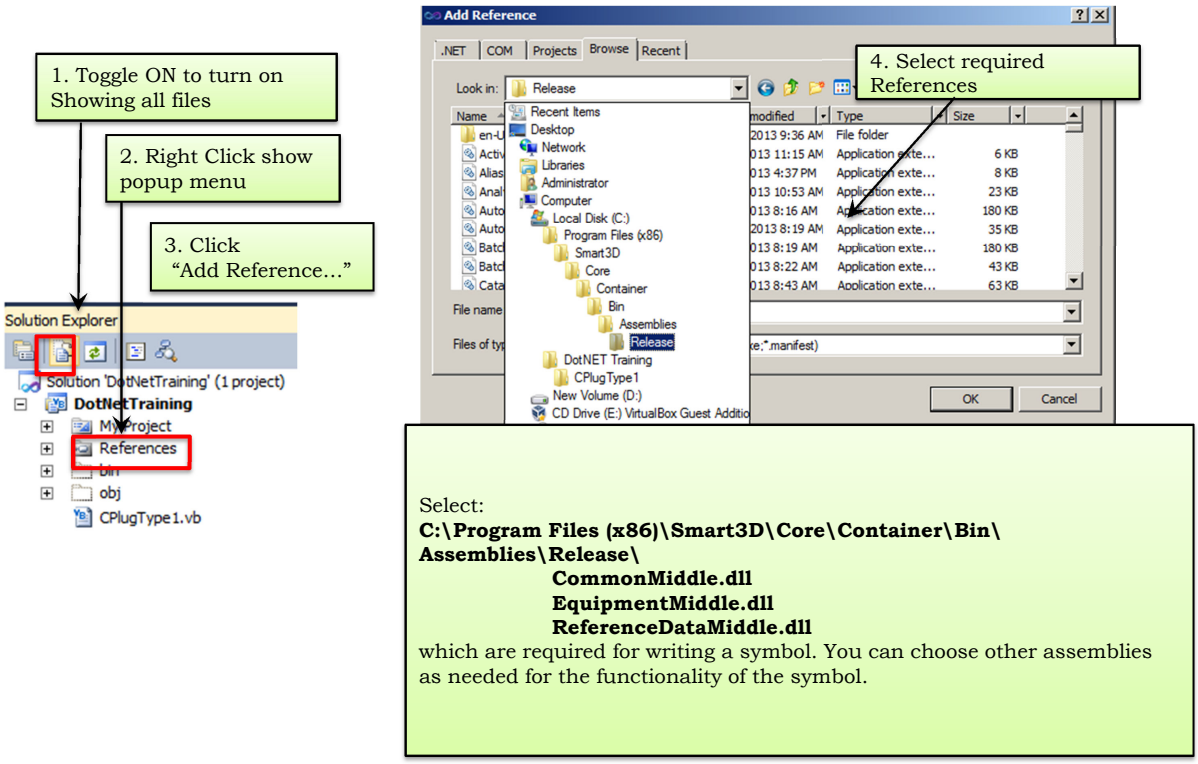


Create a VB.net Class Library Project

- Right Click the Project > Properties
- On the **Properties > Application** set the **Root Namespace** for this project
- Rename the **Root namespace:** from *ClassLibrary1* to *Ingr.S3D.Content.RouteSymbols* for this class.
- Intergraph recommends the following format for your company to follow <CompanyName>.S3D.Content.<Specialization>



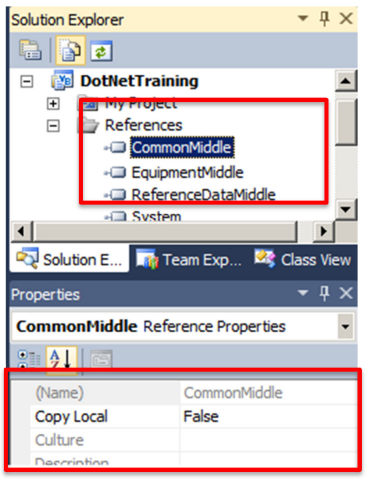
Attach Required References



Select:
C:\Program Files (x86)\Smart3D\Core\Container\Bin\Assemblies\Release
CommonMiddle.dll
EquipmentMiddle.dll
ReferenceDataMiddle.dll
which are required for writing a symbol. You can choose other assemblies as needed for the functionality of the symbol.

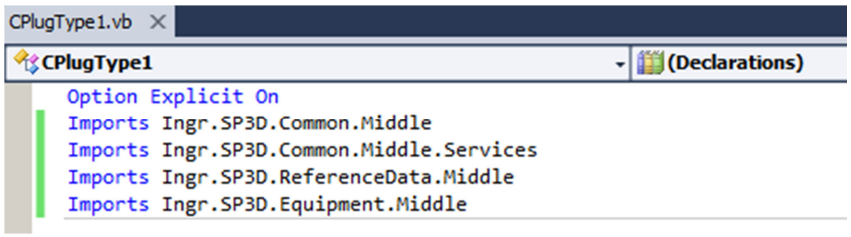
References: Copy Local Setting

- Typically, Visual Studio adds references with "**Copy Local : True**" setting, i.e. it is copied locally for the project's use.
- Change it as "**Copy Local : False**" in the Properties tab of the assembly reference.
- Otherwise, when you install updates of the S3D product(s), your projects may continue using old copies of the references copied at the time of adding the reference.



Import Required Namespaces

- Add the required “Imports” statements at the top of the file to enable you use the types in that namespace without using the full name of the class when needed.



- This lets you just use

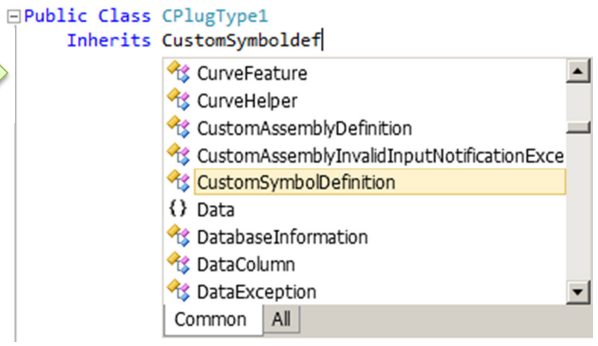
  Dim oPart As Part

  instead of :

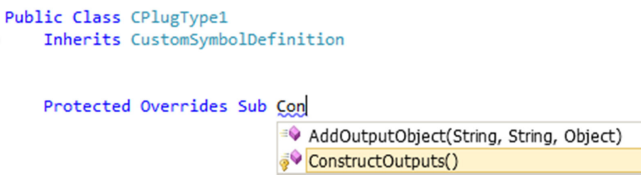
  Dim oPart As Ingr.SP3D.ReferenceData.Middle.Part

Base CustomSymbolDefinition Class

- To be a symbol you must ***inherit*** from the class **CustomSymbolDefinition**



- We then provide the ***override implementation*** for the properties/methods



  i.e. the **ConstructOutputs()**

Definition of Inputs

- The inputs for the symbol definition is added at the beginning of the class definition.

#Region “Name”
Tag for organizing code

```
#Region "Definition of Inputs"


#End Region
```

#End Region
Is a Tag for closing the #Region Tag

Definition of Inputs

- Declare Input parameters as member variables in the Custom Symbol Definition class
- Available Attributes:
  - InputCatalogPart(Index)
  - InputDouble(Index, Name, Description, Default Value, Optional)
  - InputString(Index, Name, Description, Default Value, Optional)

```
<InputCatalogPart(1)> Public m_oPartDef As InputCatalogPart

<InputDouble(2, "FacetoEnd", "Face to End", 0.15)> Public m_dFacetoEnd As InputDouble
```

Definition of Inputs

```
Public Class CPlugType1
    Inherits CustomSymbolDefinition

#Region "Definition of Inputs"

    <InputCatalogPart(1)> Public m_oPartDef As InputCatalogPart
    <InputDouble(2, "FacetoEnd", "FacetoEnd", 0.15)> Public m_dFacetoEnd As InputDouble

#End Region
```

InputCatalogPart Attribute
Define the global member catalog part.
- This is always the 1st attribute defined

InputDouble Attribute
Define a global member attribute that is a Double type

- 2 is the index in the inputs list
- “FacetoEnd” is the name of the attribute
- “Face to End” is the description of the attribute
- 0.15 is the default value

Definition of Aspects and their Outputs

- Declare Aspects as member variables in the Custom Symbol Definition class
- Declare Outputs as attributes on each Representation
  - Aspect(Name, Description, Aspect ID)
  - SymbolOutput(Name, Description)

```
<Aspect("Physical", "Physical Aspects", AspectID.SimplePhysical)> _
  <SymbolOutput("PipePort1", "PipePort1")> _
  <SymbolOutput("Body1", "Body1")> _
  <SymbolOutput("Body2", "Body2")> _
  Public m_oPhysicalAspect  As AspectDefinition
```

Definition of Aspects and their Outputs

```
Public Class CPlugType1
    Inherits CustomSymbolDefinition

#Region "Definition of Inputs"

    <InputCatalogPart(1)> Public m_oPartDef As InputCatalogPart
    <InputDouble(2, "FacetoEnd", "FacetoEnd", 0.15)> Public m_dFacetoEnd As InputDouble

#End Region

#Region "Definition of Aspects and their Outputs"

    'Physical Aspect
    <Aspect("Physical", "Physical Aspects", AspectID.SimplePhysical)> _
    <SymbolOutput("PipePort1", "PipePort1")> _
    <SymbolOutput("Body1", "Body1")> _
    <SymbolOutput("Body2", "Body2")> _
    Public m_oPhysicalAspect As AspectDefinition

#End Region
```

Define each Aspect the symbol will support
- “Physical” is the Aspect name
- “Physical Aspect” is the description
- AspectID.SimplePhysical is the Aspect Id

Definition of Aspects and their Outputs

```
Public Class CPlugType1
    Inherits CustomSymbolDefinition

#Region "Definition of Inputs"

    <InputCatalogPart(1)> Public m_oPartDef As InputCatalogPart
    <InputDouble(2, "FacetoEnd", "FacetoEnd", 0.15)> Public m_dFacetoEnd As InputDouble

#End Region

#Region "Definition of Aspects and their Outputs"

    'Physical Aspect
    <Aspect("Physical", "Physical Aspects", AspectID.SimplePhysical)> _
    <SymbolOutput("PipePort1", "PipePort1")> _
    <SymbolOutput("Body1", "Body1")> _
    <SymbolOutput("Body2", "Body2")> _
    Public m_oPhysicalAspect As AspectDefinition

#End Region
```

Define each output for the Aspect
- "Body1“ is the name of the output
- "Body1“ is the description of the output

Definition of ConstructOutputs() Sub

```
#Region "Construction of outputs of all aspects"

    Protected Overrides Sub ConstructOutputs()

        Dim oPart As Part = Nothing
        Dim oConnection As SP3DConnection = Nothing
        Dim dFacetoEnd As Double = 0.0


    End Sub
#End Region
```

Define local variables for the subroutine

Definition of ConstructOutputs() Sub

```
#Region "Construction of outputs of all aspects"

    Protected Overrides Sub ConstructOutputs()

        Dim oPart As Part = Nothing
        Dim oConnection As SP3DConnection = Nothing
        Dim dFacetoEnd As Double = 0.0

        'Get Input values

        oPart = m_oPartDef.Value
        dFacetoEnd = m_dFacetoEnd.Value

        oConnection = OccurrenceConnection

        Dim oPipePortDef1 As PipePortDef
        oPipePortDef1 = oPart.PortDefinitions.Item(0)
```

Get the part from the catalog

Get the port definition from the part

Definition of ConstructOutputs() Sub

Define local variables for the subroutine

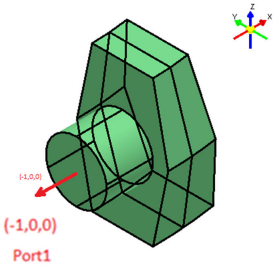
```
Dim oGeomHlpr As New SymbolGeometryHelper()
Dim oXAxis As New Vector(1, 0, 0), oYAxis As New Vector(0, 1, 0), oZAxis As New Vector(0, 0, 1)
Dim oNegXAxis As New Vector(-1, 0, 0)
Dim oLocation1 As New Position(-dFacetoEnd, 0, 0)

m_oPhysicalAspect.Outputs("PipePort1") = _
        New PipeNozzle(oPart, oConnection, False, 1, oLocation1, oNegXAxis, 0.0, False)
```

Add port to the output collection

New(**oPart As Ingr.SP3D.ReferenceData.Middle.Part**,
oConnection As Ingr.SP3D.Common.Middle.Services.SP3DConnection,
bLightWeighGraphics As Boolean,
m_Index As Integer,
oPos As Ingr.SP3D.Common.Middle.Position,
oNormal As Ingr.SP3D.Common.Middle.Vector,
dLength As Double,
bIsFacePosition As Boolean)

Creates a pipe nozzle for a given part at the specified location and orientation and with the predefined length. lightweight graphic of the nozzle.

oPart: *Part of the nozzle to be created.*



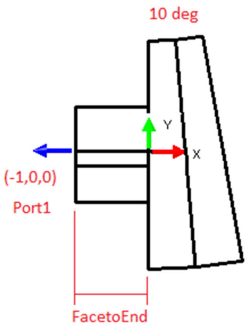
Definition of ConstructOutputs() Sub

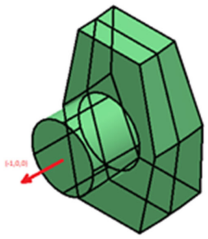
```
oGeomHlpr.SetOrientation(oXAxis, oYAxis)
oGeomHlpr.MoveToPoint(oLocation1)

m_oPhysicalAspect.Outputs("Body1") = _
        oGeomHlpr.CreateCylinder(oConnection, oPipePortDef1.PipingOutsideDiameter / 2.0, dFacetoEnd)
```

CreateCylinder(**oConnection As Ingr.SP3D.Common.Middle.Services.SP3DConnection**, dRadius As Double, dLength As Double) As Ingr.SP3D.Common.Middle.Projection3d

Creates a cylinder with radius and length. Length is the direction of the primary vector.

oConnection: *SP3DConnection object.*



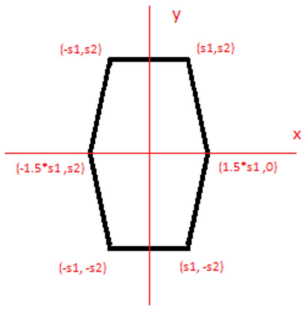


Definition of ConstructOutputs() Sub

```
Dim oList As New List(Of Position)
Dim dsize1 As Double = oPipePortDef1.PipingOutsideDiameter / 2.0
Dim dsize2 As Double = 0.04
Dim dradius As Double = 0.15
```

Define some variables for constructing the prismatic torus

```
oList.Add(New Position(-dsize1, -dsize2, 0))
oList.Add(New Position(dsize1, -dsize2, 0))
oList.Add(New Position(1.5 * dsize1, 0, 0))
oList.Add(New Position(dsize1, dsize2, 0))
oList.Add(New Position(-dsize1, dsize2, 0))
oList.Add(New Position(-1.5 * dsize1, 0, 0))
```



10 degree sweep angle

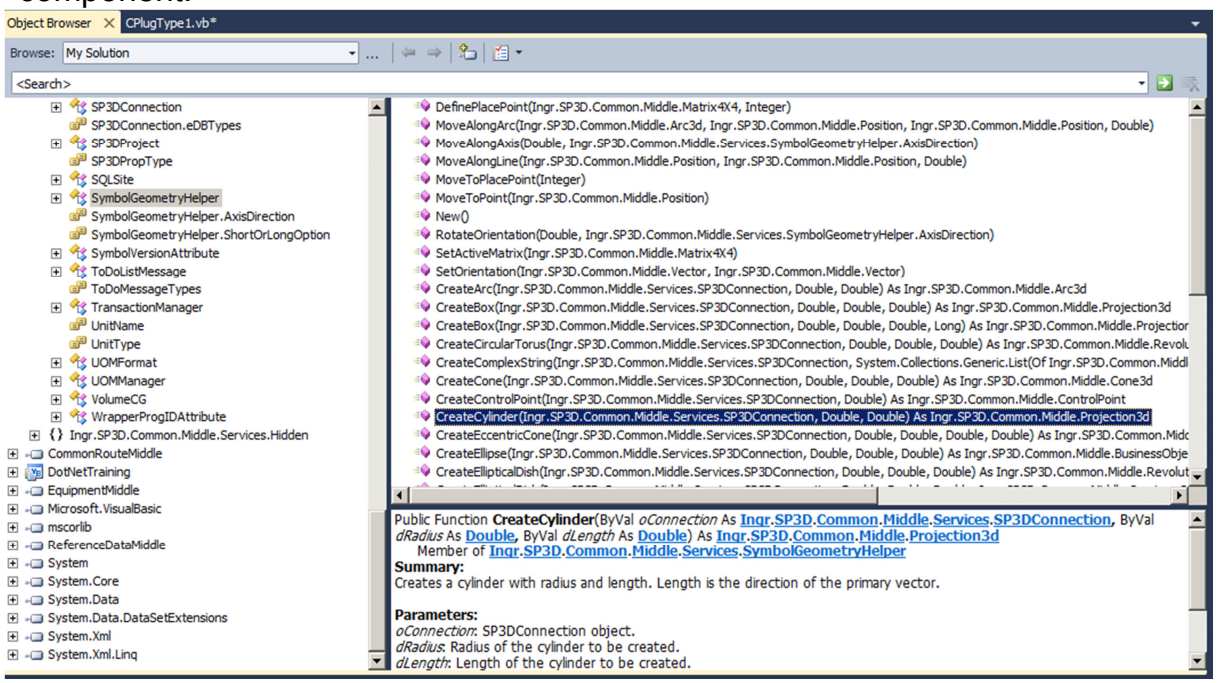
```
m_oPhysicalAspect.Outputs("Body2") = _
    oGeomHlpr.CreateNEdgePrismaticTorus(oConnection, oList, dradius, 10 * Math.PI / 180.0)
```

CreateNEdgePrismaticTorus(**oConnection As Ingr.SP3D.Common.Middle.Services.SP3DConnection**,
oVertices As System.Collections.Generic.List(Of Ingr.SP3D.Common.Middle.Position),
dRadius As Double,
dSweepAngle As Double) As Ingr.SP3D.Common.Middle.Revolution3d

Creates a torus of a polygon cross section. The cross section definition should be completed with a list of positions.
Note: Vertices must be defined in the global XY plane.
oConnection: *SP3DConnection object.*

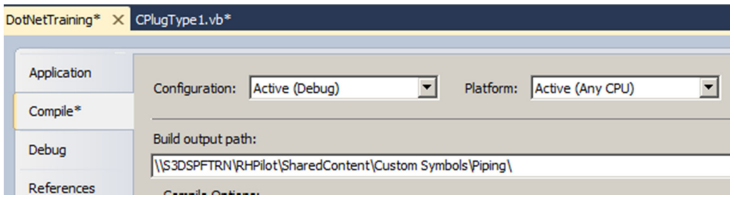
Symbol Geometry Helper Component

The **Object Browser** shows all available features of SymbolGeometryHelper component.

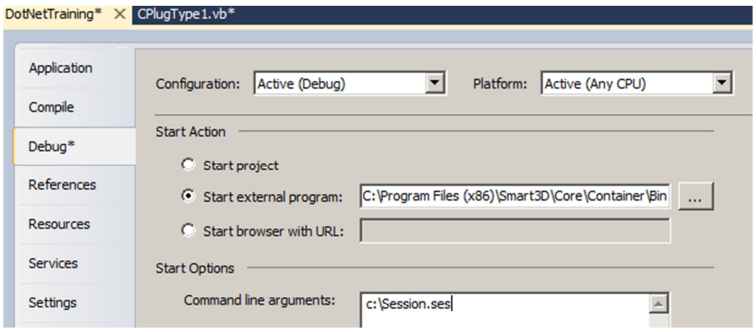


Build and Save Project

- On the **Properties > Compile** page set ***Build output path:*** to the ***[SharedContent Directory]\Custom Symbols*** folder



- On the **Properties > Debug** page under ***Start Action*** select ***Start external program:*** and assign ***[Product Directory]\3D\Container\Bin\TaskHost.exe*** or ***S3DHOST.exe***. Optionally you can assign a Session file to start up with. Under the ***Start Options*** select ***Command line arguments:*** assign ***[Path to Session file]\Session.ses***



Part Definition Creation

- Open the [Install Product]\ CatalogData\BulkLoad\Datafiles\Ten_Specs_CatalogData.xls
- Make sure to remove the Read-Only setting on the file
- Create a new part class using the new symbol definition DotNetTraining,Ingr.S3D.Content.RouteSymbols.CPlugType1
- Add 1” size male threaded plug as shown below

	A	B	C	D	E	F	G
1	! Back to Index						
2	Definition	PartClassType	SymbolDefinition	UserClassName	OccClassName	SymbolIcon	
3	a	PipeComponentClass		Plug Type1	Plug Type1		
4	CommodityPart						
5	Head	IndustryCommodityCode	CommodityType	GeometryType	GraphicalRepresentationOrNot	SymbolDefinition	MaterialGrade
6	Start						
7	a	PlugType-01	PLUG	220		DotNetTraining,Ingr.S3D.Content.RouteSymbols.CPlugType1	150

PlugType1

Part Definition Creation

- Add 1” size male threaded plug as shown below

GeometricIndustryStandard	PartDataBasis	ValveManufacturer	ValveModelNumber	ValveTrim	FlangeFaceSurfaceFinish	SurfacePreparation	ManufacturingMethod	MiscRequisitionClassification	PipingPointBasis[1]	Rd[1]	PressureRating[1]	EndPreparation[1]	EndStandard[1]	ScheduleThickness[1]	FlowDirection[1]	PipingNote1	DryWeight	DryCogX	DryCogY	DryCogZ	WaterWeight	WaterCogX	WaterCogY	WaterCogZ	SurfaceArea	VolumetricCapacity	Npd[1]	NpdUnitType[1]	FacetoEnd
41									5			331	5		4		0.43lbm	0	0	0							1	in	1in

PlugType1

Piping Commodity Filter

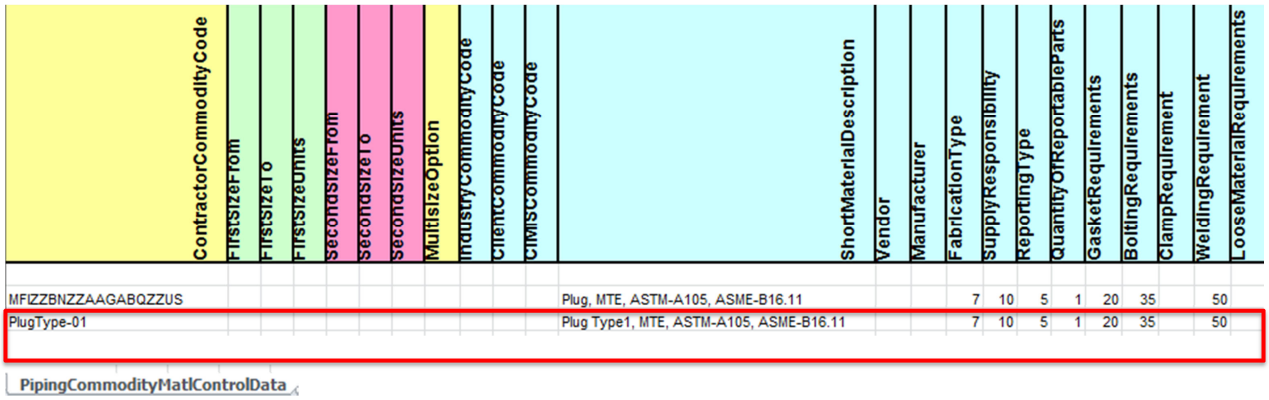
- Open the [Install Product]\CatalogData\BulkLoad\Datafiles\
- Ten_Specs_SpecificationData.xls. Make sure to remove the Read-Only setting on the file.
- Add a new option for this plug in the PipingCommodityFilter sheet for spec AC0014 as shown below:

SpecName	ShortCode	OptionCode	FirstSizeFrom	FirstSizeTo	FirstSizeUnits	SecondSizeFrom	SecondSizeTo	SecondSizeUnits	MultisizeOption	Comments	SelectionBasis	FluidCode	JacketedPipingBasis	MaximumTemperature	MinimumTemperature	EngineeringTag	CommodityCode
AC0014																	
	Plug	1	0.75	2	in						1						MEIZZBNZZAAGABQZZUS
	Plug	11	1	1	in						1						PlugType-01

PipingCommodityFilter

Piping Commodity Material Control Data

- Add a new record in the piping commodity material control data sheet as shown below:



ContractorCommodityCode	FirstSizeFrom	FirstSizeTo	FirstSizeUnits	SecondSizeFrom	SecondSizeTo	SecondSizeUnits	MultiSizeOption	IndustryCommodityCode	ClientCommodityCode	CMMSCommodityCode	ShortMaterialDescription	Vendor	Manufacturer	FabricationType	SupplyResponsibility	ReportingType	QuantityOfReportableParts	GasketRequirements	BoltingRequirements	ClampRequirement	WeldingRequirement	LooseMaterialRequirements
MFIZZBNZZAAGABQZZUS											Plug, MTE, ASTM-A105, ASME-B16.11			7	10	5	1	20	35		50	
PlugType-01											Plug Type1, MTE, ASTM-A105, ASME-B16.11			7	10	5	1	20	35		50	

PipingCommodityMatlControlData

- Save the file
- Use the Bulkload utility, load the two workbook files into the catalog using the Add/Modify/Delete mode

Deploying & Testing the Symbol

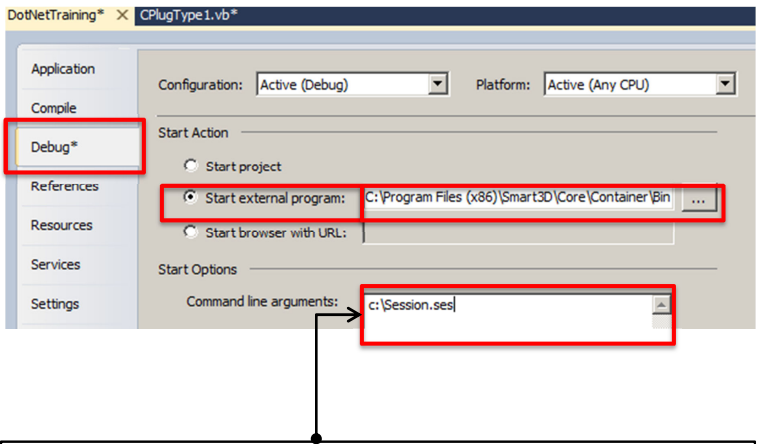
- Save the Custom Symbol DLL to the ***[SharedContent Directory]\Custom Symbols*** folder
- In Project Management Task, select the catalog you want to update then invoke menu item
  - **Tools > Update Custom Symbol Configuration**
    - The command creates or updates the file called ***CustomSymbolConfig.xml*** in the ***[SharedContent Directory]\Xml*** folder
    - ***CustomSymbolConfig.xml*** contains entries of ProgID, CLSID, and DLL name for each class in the custom DLLs. After ***CustomSymbolConfig.xml*** is created, the software uses the custom DLLs from the ***[SharedContent Directory]\Custom Symbols*** folder.

Moving/Copying/Opening Projects across machines

- **.NET** is not like **COM** in many respects.
- When you open a Project in a new machine, if the 3D installation folder in the new machine is not the same as that of the machine where the project was originally saved, you will encounter broken references.
- This is due to the fact that the references added to the project from the original machine are not found at those folder locations in the new machine.
- Whereas, in COM, the system goes by the registry to auto correct the DLL file references.
- To solve this situation, You will have to open and fix the project files in the new machine to use the corrected paths.
- Using Standard installation paths for the developers in your automation team is a good idea to avoid this problem.

Debugging

- Add the full **Core\Runtime** and **GeometryTopology\Runtime** paths to your Path Environment Variable first.
- Inside Visual Studio 2010, **Project > Project Properties**
- Choose the following settings and **save**.
- Now select **Debug>Run menu item in Visual Studio.**



TIP : You can also specify the session file name in the **Command line arguments** field, with which the application will start in that session file directly instead of the normal open session file dialog.

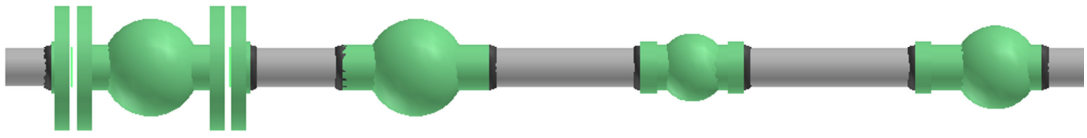
Debugging

For Developer Studio **Express Edition,** you can be setup by specifying the details of the Executable in the <project>.vbproj.user file. Optionally you can also specify the Session File name in StartArguments section.

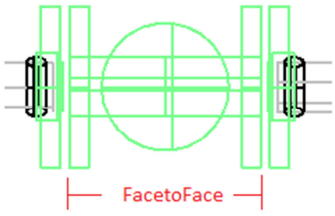
```
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
<PropertyGroup>
   <StartAction>Program</StartAction>
   <StartProgram>C:\Program Files (x86)\Smart3D\Core\Container\Bin\S3DHost.exe</StartProgram>
   <StartArguments>C:\session.ses</StartArguments>
</PropertyGroup>
</Project>
```


Stock Instrument Symbol

Lab 2 - Instrument Temperature Control Valves



Flanged Ends Welded Ends Socked Ends SockedxWelded Ends



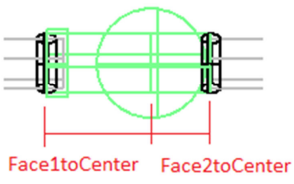
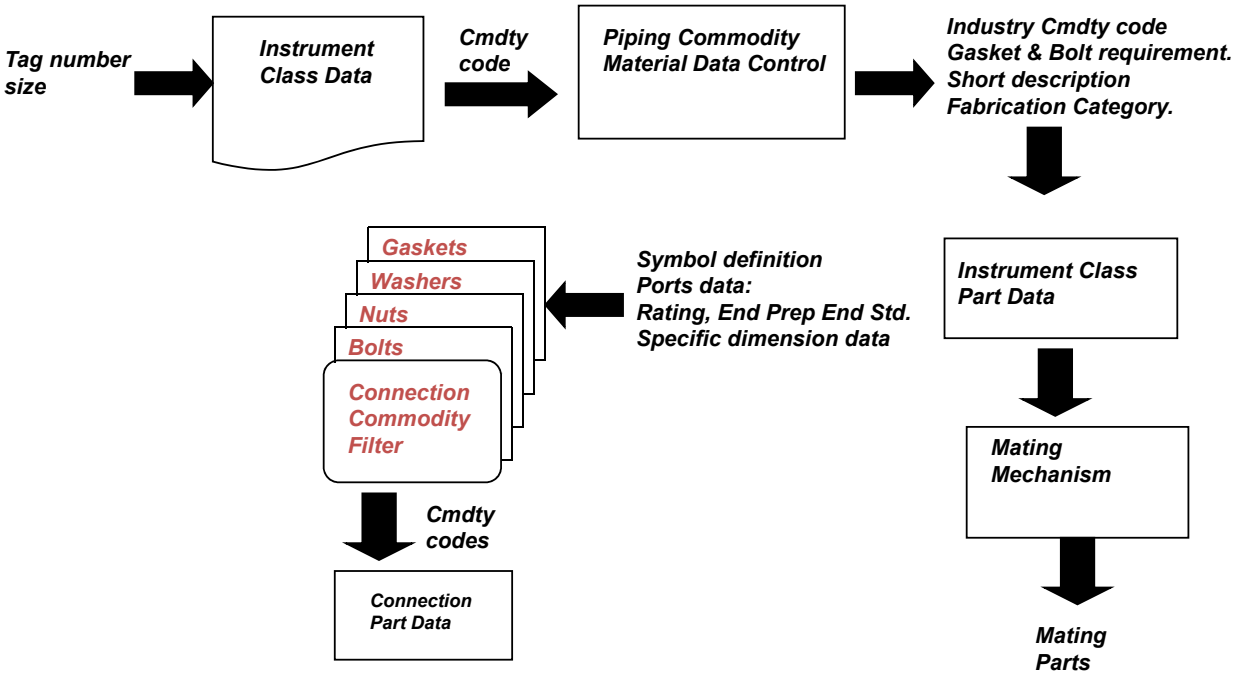


Table Look Up for a Stock Instrument



43

Table Look Up for a Stock Instrument

Instrument class data

TagNumber	GenericTagNumber	FirstSizeFrom	FirstSizeTo	FirstSizeUnits	RequisitionType	ContractorCommodityCode
BV-1001		15	50	mm	5	INVDABAVRAWA

Piping Commodity Material Control Data

ContractorCommodityCode	FirstSizeFrom	FirstSizeTo	FirstSizeUnits	IndustryCommodityCode	ShortMaterialDescription
INVDABAVRAWA				INVDABAVRAWA	3-Way Ball valve, T-Full Port, Carbon Steel,

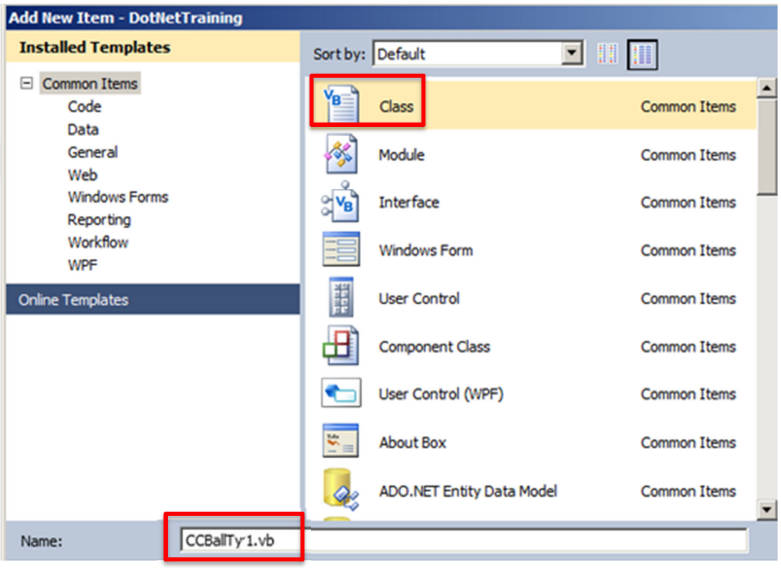
3 way valve class

IndustryCommodityCode	FirstSizeSchedule	SecondSizeSchedule	CommodityType	GeometryType	GraphicalRepresentationOrNot	SymbolDefinition	MaterialGrade	LiningMaterial	GeometricIndustryStandard	RequisitionType	PipingPointBasis[1]	Id[1]	PressureRating[1]	EndPreparation[1]	EndStandard[1]	ScheduleThickness[1]	FlowDirection[1]	PipingPointBasis[2]	Id[2]	PressureRating[2]	EndPreparation[2]	EndStandard[2]	ScheduleThickness[2]	FlowDirection[2]	PipingPointBasis[3]	Id[3]	PressureRating[3]	EndPreparation[3]	EndStandard[3]	ScheduleThickness[3]	FlowDirection[3]	Npd[1];Primary	NpdUnitType[1]	Npd[2];Secondary	NpdUnitType[2]	Npd[3];Secondary	NpdUnitType[3]	FacetoCenter	MajorBodyDiameter	OperatorHeight	OperatorLength
INVDABAVRAWA			BAL3W	75			150			5			150	21	5		3			150	21	5		3			150	21	5		3	15	mm	15	mm	15	mm	65	60	110	140
																																20	mm	20	mm	20	mm	75	65	110	140

44

Create a VB.net Class Library Project

- Choose **Project** > **Add Class**
- Pick **Class Library** template
- Specify **Name** of Class
- CCBallTy1
- Hit **OK**.



Import Required Namespaces

- Add the required “Imports” statements at the top of the file to enable you use the types in that namespace without using the full name of the class when needed.

```
CBallTy1
Option Explicit On
Imports System.Math
Imports Ingr.SP3D.Common.Middle
Imports Ingr.SP3D.Common.Middle.Services
Imports Ingr.SP3D.ReferenceData.Middle
Imports Ingr.SP3D.Equipment.Middle
```

- This lets you just use

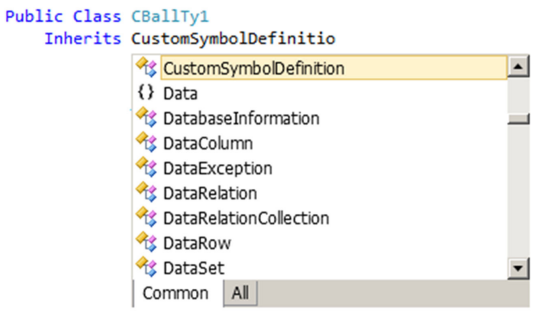
  Dim oPart As Part

  instead of :

  Dim oPart As Ingr.SP3D.ReferenceData.Middle.Part

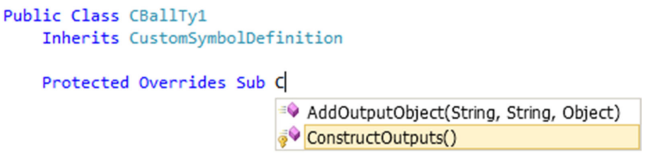
Base CustomSymbolDefinition Class

- To be a symbol you must ***inherit*** from the class **CustomSymbolDefinition**



- We then provide the ***override implementation*** for the properties/methods

  i.e. the **ConstructOutputs()**



Base CustomSymbolDefinition Class

```
<SymbolVersion("1.0.0.0")> _
Public Class CBallTy1
    Inherits CustomSymbolDefinition
```

SymbolVersion Attribute
Specify the symbol version.

Public Class **SymbolVersionAttribute**
Inherits System.Attribute
Member of Ingr.SP3D.Common.Middle.Services
Summary:
SymbolVersionAttribute class defined on a CustomSymbolDefinition class is used to specify the version number of a symbol. The version should be specified as a string in the format of #.#.#.#.

Definition of Inputs

- Declare Input parameters as member variables in the Custom Symbol Definition class
  - InputCatalogPart(Index)
  - InputDouble(Index, Name, Description, Default Value, Optional)
  - InputString(Index, Name, Description, Default Value, Optional)

```
<InputCatalogPart(1)> Public m_oPartDef As InputCatalogPart
  <InputDouble(2, "FacetoFace", "FacetoFace", 0.3, True)> Public m_dFacetoFace As InputDouble
  <InputDouble(3, "Face1toCenter", "Face1toCenter", 0.3, True)> Public m_dFace1toCenter As InputDouble
  <InputDouble(4, "Face2toCenter", "Face2toCenter", 0.3, True)> Public m_dFace2toCenter As InputDouble
  <InputDouble(5, "CylHeight", "CylinderHeight", 0.2)> Public m_dCylHeight As InputDouble
```


Definition of Inputs

```
<SymbolVersion("1.0.0.0")> _
Public Class CBallTy1
    Inherits CustomSymbolDefinition


#Region "Definition of Inputs"

    <InputCatalogPart(1)> Public m_oPartDef As InputCatalogPart
    <InputDouble(2, "FacetoFace", "FacetoFace", 0.3, True)> Public m_dFacetoFace As InputDouble
    <InputDouble(3, "Face1toCenter", "Face1toCenter", 0.3, True)> Public m_dFace1toCenter As InputDouble
    <InputDouble(4, "Face2toCenter", "Face2toCenter", 0.3, True)> Public m_dFace2toCenter As InputDouble
    <InputDouble(5, "CylHeight", "CylinderHeight", 0.2)> Public m_dCylHeight As InputDouble

#End Region
```

InputCatalogPart Attribute
Define the global member catalog part
- This is always the 1st attribute defined

InputDouble Attribute
Define a global member attribute that is a Double type

- 2 is the index in the inputs list
- “FacetoFace” is the name of the attribute
- “FacetoFace” is the description of the attribute
- 0.3 is the default value
- True is an Optional Input

Definition of Aspects and their Outputs

- Declare Representations as member variables in the Custom Symbol Definition class
- Declare Outputs as attributes on each Representation
  - Aspect(Name, Description, Aspect ID)
  - SymbolOutput(Name, Description)

```
<Aspect("Physical", "Physical Aspects", AspectID.SimplePhysical)> _
  <SymbolOutput("Body0", "Body0")> _
  <SymbolOutput("Body1", "Body1")> _
  <SymbolOutput("Body2", "Body2")> _
  <SymbolOutput("PipePort1", "PipePort1")> _
  <SymbolOutput("PipePort2", "PipePort2")> _
  Public m_oPhysicalAspect As AspectDefinition
```


Definition of Aspects and their Outputs

```
#Region "Definition of Aspects and their Outputs"

    'Physical Aspect
    <Aspect("Physical", "Physical Aspects", AspectID.SimplePhysical)> _
    <SymbolOutput("Body0", "Body0")> _
    <SymbolOutput("Body1", "Body1")> _
    <SymbolOutput("Body2", "Body2")> _
    <SymbolOutput("PipePort1", "PipePort1")> _
    <SymbolOutput("PipePort2", "PipePort2")> _
    Public m_oPhysicalAspect As AspectDefinition

#End Region
```

Define each Aspect the symbol will support
- “Physical” is the Aspect name
- “Physical Aspect” is the description
- AspectID.SimplePhysical is the Aspect Id

Definition of Aspects and their Outputs

```
#Region "Definition of Aspects and their Outputs"

    'Physical Aspect
    <Aspect("Physical", "Physical Aspects", AspectID.SimplePhysical)> _
    <SymbolOutput("Body0", "Body0")> _
    <SymbolOutput("Body1", "Body1")> _
    <SymbolOutput("Body2", "Body2")> _
    <SymbolOutput("PipePort1", "PipePort1")> _
    <SymbolOutput("PipePort2", "PipePort2")> _
    Public m_oPhysicalAspect As AspectDefinition

#End Region
```

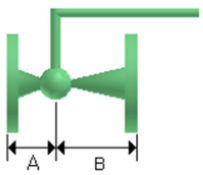
Define each output for the Aspect
- "Body1“ is the name of the output
- "Body1“ is the description of the output

Part Data Basis

Part Data Basis value is intended to be used by the applicable symbol to recognize those components that require special treatment in terms of interpreting the dimensional data.

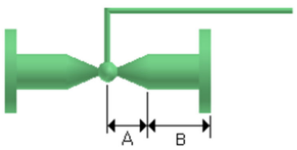
	IndustryCommodityCode	CommodityType	GeometryType	SymbolDefinition	MirrorBehaviorOption	GeometricIndustryStandard	PartDataBasis
	VCKAHBVZZAAGAODZZZZZZ	BALL	15	SP3DBallValve.CBallValve		41	25
	VCKAHABZZAAGZZZZZZZZZ	BALL	15	SP3DBallValve.CBallValve		41	13
	VCKAMABZZAAGADAZZZZZZ	BALL	15	SP3DBallValve.CBallValve		41	353

Part Design Basis 13



A=Face 1 to Center
B=Face 2 to Center

Part Design Basis 353



A=Valve Face to Center
B=Nipple Length

Part Design Basis 25

A=Seat to Seat

Definition of ConstructOutputs() Sub

```
Protected Overrides Sub ConstructOutputs()

    Dim oPart As Part = Nothing
    Dim oConnection As SP3DConnection = Nothing
    Dim dFacetoFace As Double = 0.0, dFace1toCenter As Double = 0.0, dFace2toCenter As Double = 0.0
    Dim dCylHeight As Double = 0.0

    oPart = m_oPartDef.Value
    dCylHeight = m_dCylHeight.Value

    oConnection = OccurrenceConnection

    Dim dPartDatabasis As Long
    Dim oPropertyValueCodelist As PropertyValueCodelist = Nothing
    oPropertyValueCodelist = oPart.GetPropertyValue("IJDPipeComponent", "PartDataBasis")
    dPartDatabasis = oPropertyValueCodelist.PropValue
```

Define local variables for the subroutine

Get the part from the catalog

Get the PartDataBasis value using Business Object Generic Property Access method

Definition of ConstructOutputs() Sub

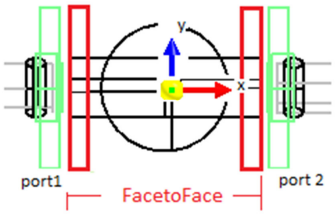
```
'================================================
' Construction of Physical Aspect
'================================================

Dim oGeomHlpr As New SymbolGeometryHelper()
Dim oXAxis As New Vector(1, 0, 0), oYAxis As New Vector(0, 1, 0), oZAxis As New Vector(0, 0, 1)
Dim oNegXAxis As New Vector(-1, 0, 0), oNegYAxis As New Vector(0, -1, 0), oNegZAxis As New Vector(0, 0, -1)

Dim oOrigin As New Position(0, 0, 0)
oGeomHlpr.ActivePosition = oOrigin
```

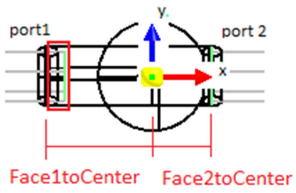
Define local variables for the subroutine

Set current position as (0,0,0)

Definition of ConstructOutputs() Sub

```
' ================================== Main Body Output ==================================

Select Case dPartDatabasis
    Case 1, 5
        dFacetoFace = m_dFacetoFace.Value

        m_oPhysicalAspect.Outputs("Body0") = oGeomHlpr.CreateSphere(oConnection, dFacetoFace / 3.0)
        oGeomHlpr.MoveToPoint(New Position(-0.5 * dFacetoFace, 0, 0))
        m_oPhysicalAspect.Outputs("Body1") = oGeomHlpr.CreateCylinder(oConnection, dCylHeight / 4, dFacetoFace / 2.0)
        m_oPhysicalAspect.Outputs("Body2") = oGeomHlpr.CreateCylinder(oConnection, dCylHeight / 4, dFacetoFace / 2.0)

    Case 10
        dFace1toCenter = m_dFace1toCenter.Value
        dFace2toCenter = m_dFace2toCenter.Value
        dFacetoFace = dFace1toCenter + dFace2toCenter

        m_oPhysicalAspect.Outputs("Body0") = oGeomHlpr.CreateSphere(oConnection, dFacetoFace / 3.0)
        oGeomHlpr.MoveToPoint(New Position(-dFace1toCenter, 0, 0))
        m_oPhysicalAspect.Outputs("Body1") = oGeomHlpr.CreateCylinder(oConnection, dCylHeight / 4, dFace1toCenter)
        m_oPhysicalAspect.Outputs("Body2") = oGeomHlpr.CreateCylinder(oConnection, dCylHeight / 4, dFace2toCenter)
End Select
```



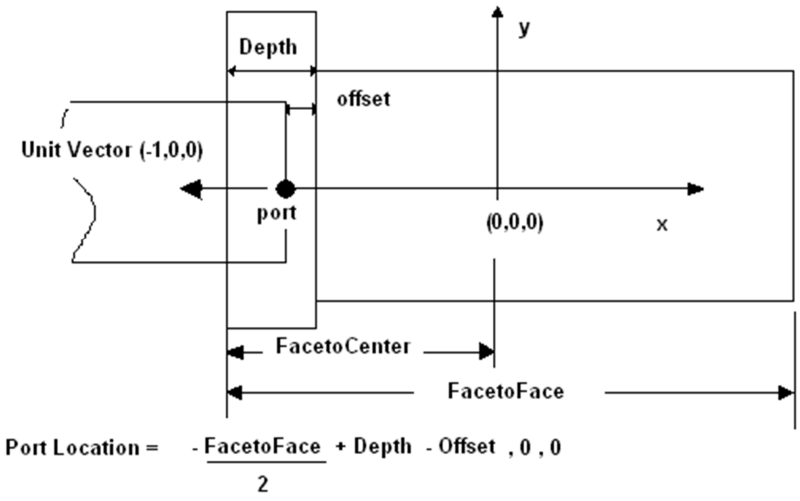


Definition of ConstructOutputs() Sub

```
' ================================== Place the Ports ==================================

Dim oPipePortDef1 As PipePortDef, oPipePortDef2 As PipePortDef

If (CType(oPart.PortDefinitions.Item(0), PipePortDef).PortIndex = 1) Then
    oPipePortDef1 = oPart.PortDefinitions.Item(0)
    oPipePortDef2 = oPart.PortDefinitions.Item(1)
Else
    oPipePortDef1 = oPart.PortDefinitions.Item(1)
    oPipePortDef2 = oPart.PortDefinitions.Item(0)
End If
```

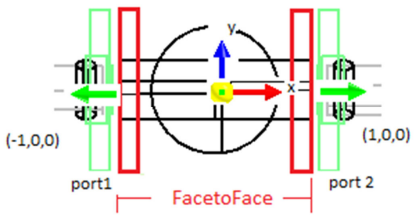
Get the ports from the part definition

Piping Port



Definition of ConstructOutputs() Sub

```
Select Case dPartDatabasis
    Case 1, 5
        Dim oPlacementPos As New Position(-0.5 * dFacetoFace _
                                + oPipePortDef1.SeatingOrGrooveOrSocketDepth - oPipePortDef1.FlangeProjectionOrSocketOffset, 0, 0)
        m_oPhysicalAspect.Outputs("PipePort1") = _
                New PipeNozzle(oPart, oConnection, False, 1, oPlacementPos, oNegXAxis, 0.0, False)

        oPlacementPos.Set(0.5 * dFacetoFace _
                                - oPipePortDef2.SeatingOrGrooveOrSocketDepth + oPipePortDef2.FlangeProjectionOrSocketOffset, 0, 0)
        m_oPhysicalAspect.Outputs("PipePort2") = _
                New PipeNozzle(oPart, oConnection, False, 2, oPlacementPos, oXAxis, 0.0, False)
    Case 10
        Dim oPlacementPos As New Position(-dFace1toCenter _
                                + oPipePortDef1.SeatingOrGrooveOrSocketDepth - oPipePortDef1.FlangeProjectionOrSocketOffset, 0, 0)
        m_oPhysicalAspect.Outputs("PipePort1") = _
                New PipeNozzle(oPart, oConnection, False, 1, oPlacementPos, oNegXAxis, 0.0, False)

        oPlacementPos.Set(dFace2toCenter _
                            - oPipePortDef2.SeatingOrGrooveOrSocketDepth + oPipePortDef2.FlangeProjectionOrSocketOffset, 0, 0)
        m_oPhysicalAspect.Outputs("PipePort2") = _
                New PipeNozzle(oPart, oConnection, False, 2, oPlacementPos, oXAxis, 0.0, False)
End Select
```



Part Class Definition

- Open the [Install Product]\ CatalogData\BulkLoad\Datafiles\Instrument Data.xls
- Make sure to remove the Read-Only setting on the file
- Create a new part class using the new symbol definition DotNetTraining,Ingr.S3D.Content.RouteSymbols.CCBallTy1
- Add four 1” size control valve parts

Definition	PartClassType	SymbolDefinition	UserClassName	OccClassName
a	InstrumentsClass	DotNetTraining.Ingr.S3D.Content.RouteSymbols.CBallTy1	Control Ball Valve Type1	Control Ball Valve Type1
CommodityPart				
Head	IndustryCommodityCode	CommodityType	GeometryType	
Start				
a	CVBallType1-01	Temperature control valve	15	
a	CVBallType1-02	Temperature control valve	15	
a	CVBallType1-03	Temperature control valve	15	
a	CVBallType1-04	Temperature control valve	15	

CBallValveType1

Part Class Definition

- Add four 1” size control valve parts with different port data

MaterialGrade	LiningMaterial	BendAngle	BendRadius	GeometricIndustryStandard	BendRadiusMultiplier	PartDataBasis	DryCogX	DryCogY	DryCogZ	WaterWeight	WaterCogX	WaterCogY	WaterCogZ	SurfaceArea	VolumetricCapacity	PipingPointBasis[1]	Id[1]	PressureRating[1]	EndPreparation[1]	EndStandard[1]	ScheduleThickness[1]	FlowDirection[1]	PipingPointBasis[2]	Id[2]	PressureRating[2]	EndPreparation[2]	EndStandard[2]	ScheduleThickness[2]	FlowDirection[2]
150						5	0	0	0		0	0	0					150	21	5		3			150	21	5		3
150						5	0	0	0		0	0	0						301	5	S-STD	3				301	5	S-STD	3
150						5	0	0	0		0	0	0					800	421	5		3			800	421	5		3
150						10	0	0	0		0	0	0					800	421	5		2				301	5	S-STD	1

CBallValveType1

Part Class Definition

- Add four 1” size control valve parts with different port data

DryWeight	Npd[1]	NpdUnitType[1]	Npd[2]	NpdUnitType[2]	FacetoFace	Face1toCenter	Face2toCenter	CylHeight
2.2Kg	1	in	1	in	128mm			80mm
1.5Kg	1	in	1	in	128mm			80mm
2Kg	1	in	1	in	90mm			80mm
2.1Kg	1	in	1	in		70mm	40mm	80mm

CBallValveType1

Instrument Class Data

- Add four stock instruments in the InstrumentClassData sheet as shown below:

Head Start	TagNumber	GenericTagNumber	SpecName	FIrstSizeFrom	FIrstSizeTo	FIrstSizeUnIts	SecondSizeFrom	SecondSizeTo	SecondSizeUnIts	MultISizeOption	RequIsitionType	ContractorCommodityCode	InstrumentType	GeometryType
a	CV-01			1	1	in					5	CVBallType1-01		15
a	CV-02			1	1	in					5	CVBallType1-02		15
a	CV-03			1	1	in					5	CVBallType1-03		15
a	CV-04			1	1	in					5	CVBallType1-04		15

InstrumentClassData

Piping Commodity Material Control Data

- Add new records in the piping commodity material control data sheet as shown below:

Head	ContractorCommodityCode	ShortMaterialDescription	FabricationType	GasketRequirements	BoltingRequirements	ClampRequirement	WeldingRequirement
Start							
a	CVBallType1-01	Control Ball Valve Type1 with Lever operator, Flanged Ends	7	5	5		50
a	CVBallType1-02	Control Ball Valve Type1 with Lever operator, Welded Ends	7	20	35		5
a	CVBallType1-03	Control Ball valve Type1 with Lever operator, Socked Ends	7	20	35		5
a	CVBallType1-04	Control Ball Valve Type1 with Lever operator, SockedxWelded Ends	7	20	35		5

PipingCommodityMatlControlData

- Save the file
- Use Bulkload utility, load the workbook file into the catalog using the Add/Modify/Delete mode

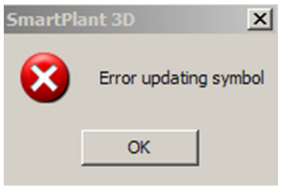
Troubleshooting Symbols

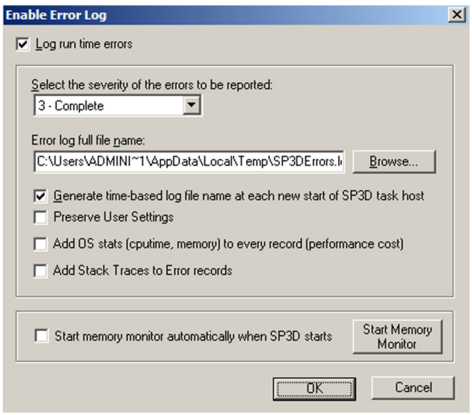
Sources of Errors

- Setting incorrect parameter values in the catalog. For example, Missing the outside pipe diameter in the generic table and therefore the pipe outside diameter is set to zero
- Check for incomplete or wrong definition
- Check for wrong versions of a symbol definition
- Check for input mismatches
- Check for output mismatches

Troubleshooting Symbols

Generic message dialog box





***** Error *****

Time : 06/27/05 13:00:39

ID : 0x80004005(-2147467259)(16389)

Source : SP3DPIA.Geom3d

Desc : Unexpected failure in constructing the .NET object.

Info : System.ArgumentException: Value does not fall within the expected range. at IngrGeom3D.GeometryFactoryClass.CreateByCenterNormalRadius(Object pConnection, Double CenterX, Double CenterY, Double CenterZ, Double NormalX, Double NormalY, Double NormalZ, Double Radius)

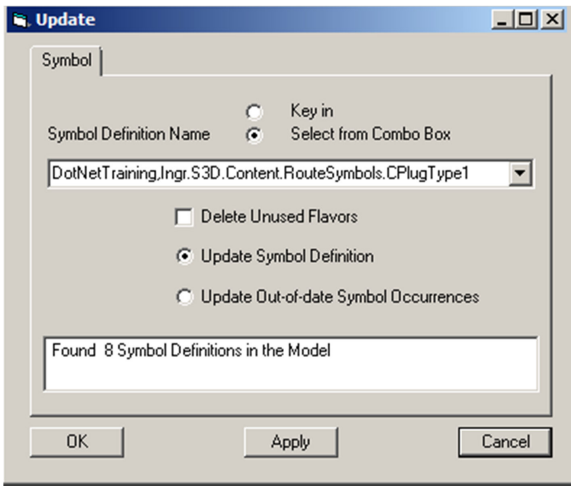
at Ingr.SP3D.Common.Middle.Circle3d..ctor(Position posCenter, Vector vecNorm, Double Radius)

at Ingr.SP3D.Common.Middle.Services.SymbolGeometryHelper.CreateCylinder(SP3DConnection oConnection, Double dRadius, Double dLength)

Troubleshooting Symbols

Custom Command

Update Symbol Occurrences after making changes on the Symbol Definition (SymbolTestCmds.CUpdateSymbolDefinition)



Symbol Modification Rules

- Symbol Inputs
  - Name and index of an input cannot be modified
  - An input type cannot be changed from parameter to object or vice versa
  - When an input is a parameter, its type cannot be changed from string to value or vice versa
  - An input cannot be deleted
  - A new input can be added which means that next available index will be used for the new input
  - Default value of an input can be changed
  - An optional input cannot be made mandatory input

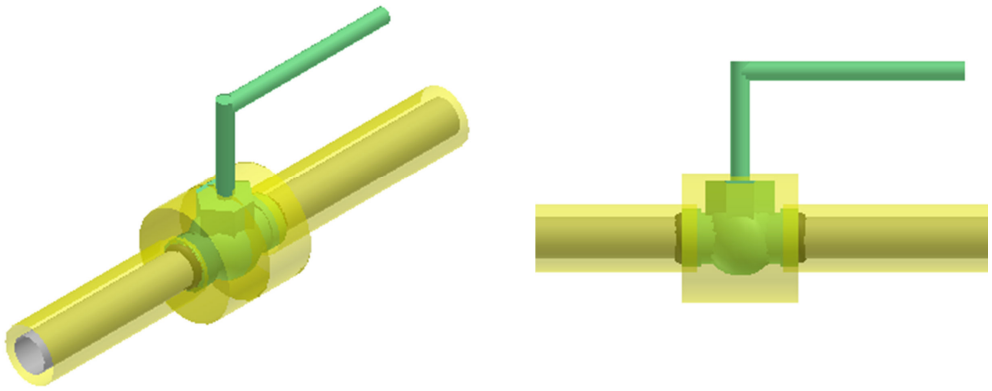
Symbol Modification Rules

- Symbol Aspects and Outputs
  - Name of an aspect cannot be modified
  - Aspect id cannot be changed
  - An Aspect cannot be deleted
  - A new Aspect can be added
  - One Output must be defined in the physical aspect
  - Name of an output of a static symbol cannot be modified
  - An output of a static symbol cannot be deleted
  - A new output can be added to a representation

Stock Instrument Symbol

Lab 3 - Symbol Modification of Instrument Control Valve

- Add a Lever Operator
- Add Insulation Geometry



Symbol Modification - Definition of Inputs

```
#Region "Definition of Inputs"

    <InputCatalogPart(1)> Public m_oPartDef As InputCatalogPart
    <InputDouble(2, "FacetoFace", "FacetoFace", 0.3, True)> Public m_dFacetoFace As InputDouble
    <InputDouble(3, "Face1toCenter", "Face1toCenter", 0.3, True)> Public m_dFace1toCenter As InputDouble
    <InputDouble(4, "Face2toCenter", "Face2toCenter", 0.3, True)> Public m_dFace2toCenter As InputDouble
    <InputDouble(5, "CylHeight", "CylinderHeight", 0.2)> Public m_dCylHeight As InputDouble
    <InputDouble(6, "HandwheelAngle", "HandwheelAngle", 0.0)> Public m_dHandwheelAngle As InputDouble
    <InputDouble(7, "OffsetFrmValCen", "OffsetFrmValCen", 0.2)> Public m_dOffsetFrmValCen As InputDouble
    <InputDouble(8, "InsulationThickness", "Insulation Thickness", 0.025)> Public m_dInsulationThickness As InputDouble
#End Region
```

Add three additional inputs

Symbol Modification - Definition of Outputs

```
#Region "Definition of Aspects and their Outputs"

    'Physical Aspect
    <Aspect("Physical", "Physical Aspects", AspectID.SimplePhysical)> _
    <SymbolOutput("Body0", "Body0")> _
    <SymbolOutput("Body1", "Body1")> _
    <SymbolOutput("Body2", "Body2")> _
    <SymbolOutput("PipePort1", "PipePort1")> _
    <SymbolOutput("PipePort2", "PipePort2")> _
    <SymbolOutput("Body3", "Body3")> _
    <SymbolOutput("Operator", "Operator occurrence")> _
    Public m_oPhysicalAspect As AspectDefinition
```

Add two additional outputs In the Physical Aspect

Symbol Modification - Definition of Aspects

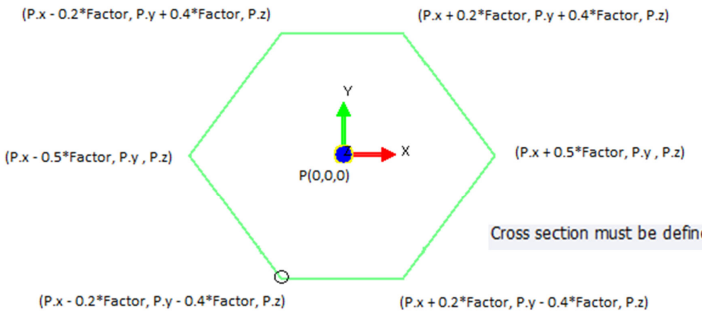
Add Insulation Aspect

```
'Insulation Aspect
<Aspect("Insulation", "Insulation Aspect", AspectID.Insulation)> _
<SymbolOutput("Body1Insulation", "Body1Insulation")> _
<SymbolOutput("Body2Insulation", "Body2Insulation")> _
Public m_oInsulationAspect As AspectDefinition
```

Add two additional outputs in the Insulation Aspect

Symbol Modification - Definition of ConstructOutputs()

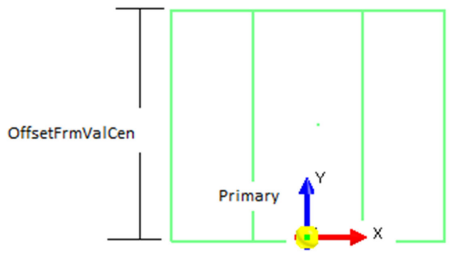
```
oGeomHlpr.MoveToPoint(oOrigin)
oGeomHlpr.SetOrientation(oYAxis, oZAxis)

'Define points for constructing the LineString3d for Body3
Dim oPointColl As New List(Of Position)
Dim dFactor As Double = dCylHeight / 1.5
oPointColl.Add(New Position(oOrigin.X - 0.2 * dFactor, oOrigin.Y - 0.4 * dFactor, oOrigin.Z))
oPointColl.Add(New Position(oOrigin.X + 0.2 * dFactor, oOrigin.Y - 0.4 * dFactor, oOrigin.Z))
oPointColl.Add(New Position(oOrigin.X + 0.5 * dFactor, oOrigin.Y, oOrigin.Z))
oPointColl.Add(New Position(oOrigin.X + 0.2 * dFactor, oOrigin.Y + 0.4 * dFactor, oOrigin.Z))
oPointColl.Add(New Position(oOrigin.X - 0.2 * dFactor, oOrigin.Y + 0.4 * dFactor, oOrigin.Z))
oPointColl.Add(New Position(oOrigin.X - 0.5 * dFactor, oOrigin.Y, oOrigin.Z))

Dim oCrossSection As LineString3d
oCrossSection = oGeomHlpr.CreateLineString(Nothing, oPointColl, True)
```



Cross section must be defined in the global XY plane.

Symbol Modification - Definition of ConstructOutputs()

```
Dim dOffsetFrmValCen As Double = 0.0
dOffsetFrmValCen = m_dOffsetFrmValCen.Value
m_oPhysicalAspect.Outputs("Body3") = _
    oGeomHlpr.CreateProjectedPolygonFromCrossSection(oConnection, oCrossSection, dOffsetFrmValCen, True)
```



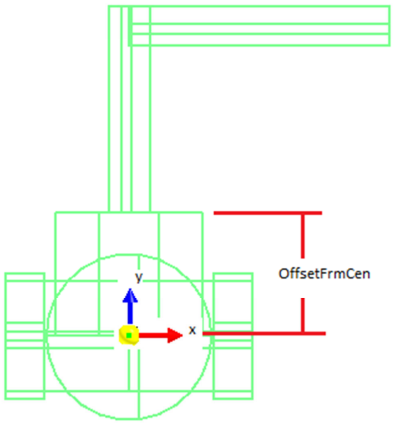
CreateProjectedPolygonFromCrossSection(**oConnection As Ingr.SP3D.Common.Middle.Services.SP3DConnection,**
oCrossSection As Ingr.SP3D.Common.Middle.BusinessObject,
dLength As Double,
bCappedEnds As Boolean) As Ingr.SP3D.Common.Middle.Projection3d

Creates a projection given either a linestring or a complex string. Projection is created in the direction of the primary axis.
Note: Cross section must be defined in the global XY plane.

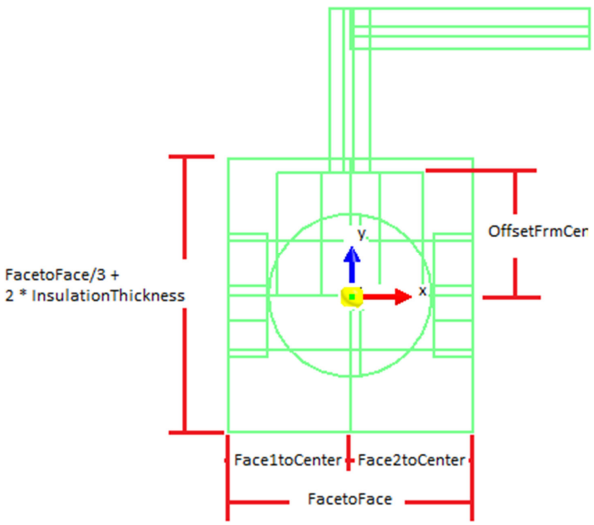
oConnection: *SP3DConnection.*

Symbol Modification - Definition of ConstructOutputs()

```
Dim dHandwheelAngle As Double = 0.0
dHandwheelAngle = m_dHandwheelAngle.Value

Dim oPipeComp As IPipeComponent
Dim oOperator As ComponentOcc, oOperatorPart As IPart

oPipeComp = oPart
oOperatorPart = oPipeComp.ValveOperator
oOperator = New ComponentOcc(oOperatorPart, oConnection)
oOperator.Origin = New Position(0, dOffsetFrmValCen, 0)
oOperator.SetOrientation(New Vector(Cos(dHandwheelAngle), 0, Sin(dHandwheelAngle)), New Vector(0, 1, 0))
m_oPhysicalAspect.Outputs("Operator") = oOperator
```



Symbol Modification - Definition of ConstructOutputs()



Symbol Modification - Definition of ConstructOutputs()

```
        '================================================
        ' Construction of Insulation Aspect
        '================================================

        Dim dParInsulationThickness As Double
        dParInsulationThickness = m_dInsulationThickness.Value

        oGeomHlpr.ActivePosition = oOrigin
        oGeomHlpr.SetOrientation(oXAxis, oYAxis)

        Select Case dPartDatabasis
            Case 1, 5
                dFacetoFace = m_dFacetoFace.Value
                oGeomHlpr.MoveToPoint(New Position(-0.5 * dFacetoFace, 0, 0))
                m_oInsulationAspect.Outputs("Body1Insulation") = _
                    oGeomHlpr.CreateCylinder(oConnection, dFacetoFace / 3.0 + 2 * dParInsulationThickness, dFacetoFace / 2.0)
                m_oInsulationAspect.Outputs("Body2Insulation") = _
                    oGeomHlpr.CreateCylinder(oConnection, dFacetoFace / 3.0 + 2 * dParInsulationThickness, dFacetoFace / 2.0)
            Case 10
                dFace1toCenter = m_dFace1toCenter.Value
                dFace2toCenter = m_dFace2toCenter.Value
                dFacetoFace = dFace1toCenter + dFace2toCenter

                oGeomHlpr.MoveToPoint(New Position(-dFace1toCenter, 0, 0))
                m_oInsulationAspect.Outputs("Body1Insulation") = _
                    oGeomHlpr.CreateCylinder(oConnection, dFacetoFace / 3.0 + 2 * dParInsulationThickness, dFace1toCenter)
                m_oInsulationAspect.Outputs("Body2Insulation") = _
                    oGeomHlpr.CreateCylinder(oConnection, dFacetoFace / 3.0 + 2 * dParInsulationThickness, dFace2toCenter)
        End Select

    End Sub

#End Region
```


Symbol Modification - Definition of ConstructOutputs()

```
<SymbolVersion("2.0.0.0")> _
Public Class CBallTy1
    Inherits CustomSymbolDefinition
```

SymbolVersion Attribute

Public Class **SymbolVersionAttribute**
Inherits System.Attribute
Member of Ingr.SP3D.Common.Middle.Services

Summary:
SymbolVersionAttribute class defined on a CustomSymbolDefinition class is used to specify the version number of a symbol. The version should be specified as a string in the format of #.#.#.#.

Symbol Modification - Synchronization

Synchronize Model with the Catalog Command

- S3D supports update of the definition persisted in the database and optional update of the existing symbol occurrences placed in the model automatically
  - Project management->Tools->Synch Model With Catalog command
  - Synch log is useful to see what definitions were found out of date and how many occurrences were updated

- Governed by the version number of the Assembly and the Symbol Definition Class.
  - .NET assembly version format: *<major version>.<minor version>.<build number>.<revision>*
  - If only **minor version** updated, the definition persisted in the database is updated.
  - If **major version** updated, apart from the definition being updated, the existing occurrences are also updated.

Custom Interface sheet

- Check and add Interface IJUAOffsetFrmValCen in the CustomInterfaces sheet

InterfaceName	AttributeName	AttributeUserName	Type	UnitsType	PrimaryUnits	OnPropertyPage	ReadOnly	SymbolParameter
IJUAOffsetFrmValCen	OffsetFrmValCen	Offset from Valve Centerline	Double	Distance	in	TRUE	FALSE	OffsetFrmValCen

CustomInterfaces

Part Class Definition

- Add one lever operator for a 1” size valve

Definition	PartClassType	SymbolDefinition	SymbolIcon
m	ValveOperatorClass	SP3DOP9.COP9	SymbolIcons\SP3DOP9.gif

CommodityPart	ValveOperatorNumber	SymbolDefinition	MirrorBehaviorOption	DimensionalBasis	ValveOperatorIsRotatable	DryWeight	DryCogX	DryCogY	DryCogZ	ValveSize	ValveSizeUnits	OperatorHeight	LeverLength	LeverLength1
Head														
Start														
a	LeverOP-9-1				5		0	0	0	1	in	3.625in	6.875in	0in
END														

Operator9

Piping Commodity Material Control Data

- Modify records in the piping commodity material control data sheet as shown below:

MultiportValveOpReq	ValveOperatorType	ValveOperatorGeoIndStd	ValveOperatorCatalogPartNumber
	9	5	LeverOP-9-1
	9	5	LeverOP-9-1
	9	5	LeverOP-9-1
	9	5	LeverOP-9-1

PipingCommodityMatlControlData

- Save the file
- Use Bulkload utility, load the workbook file into the catalog using the Add/Modify/Delete mode

Part Class Definition

- Modify the Occurrence Class Definition by adding occurrence attributes

SymbolIcon	OA:Rotation	OA:InsulationThickness
SymbolDefinition	MaterialGrade	LiningMaterial
	150	
	150	
	150	
	150	

CBallValveType1

Part Class Definition

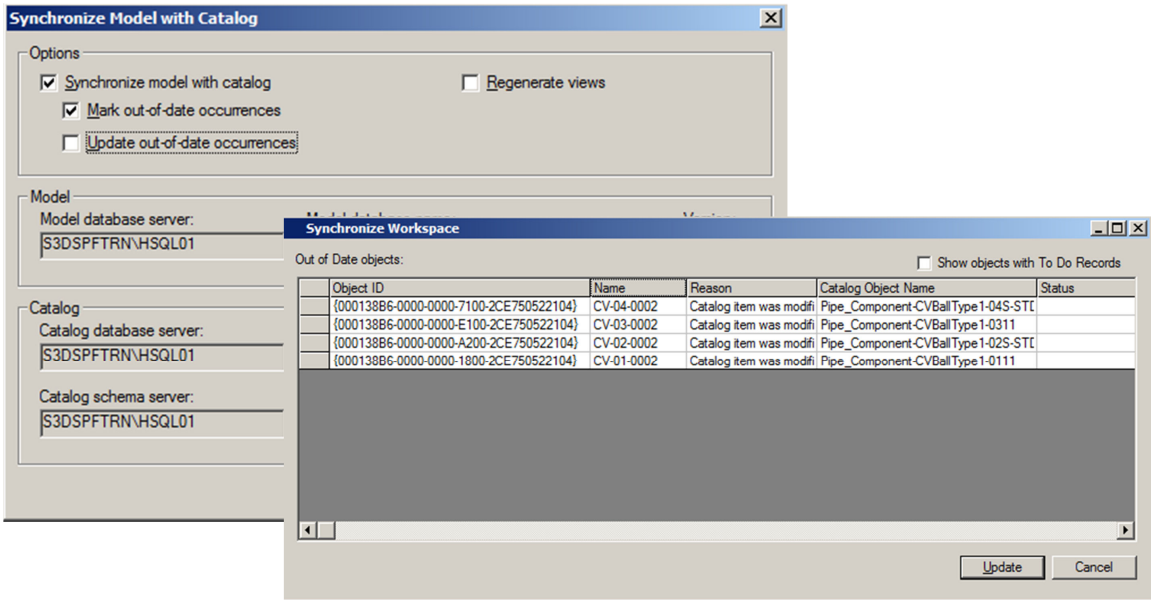
- Modify records by adding OffsetFrmValCen values as shown below:

FacetoFace	Face1toCenter	Face2toCenter	CylHeight	OffsetFrmValCen
128mm			80mm	64mm
128mm			80mm	64mm
90mm			80mm	45mm
	70mm	40mm	80mm	45mm

CBallValveType1

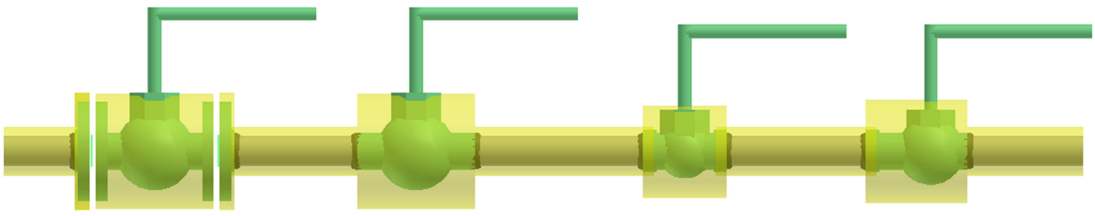
Synchronize Model with Catalog Command

- Mark out of date occurrences
- Synchronize Workspace Command



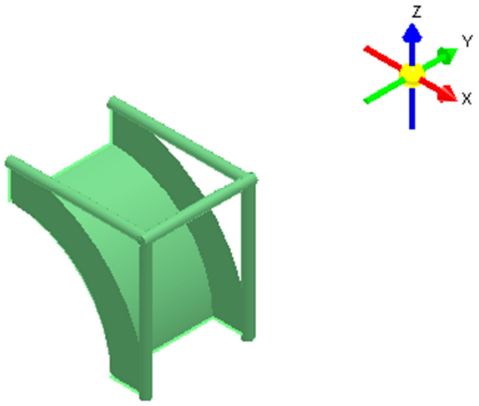
Synchronize Model with Catalog Command

- Four Instrument Temperature Control Valves with lever operator



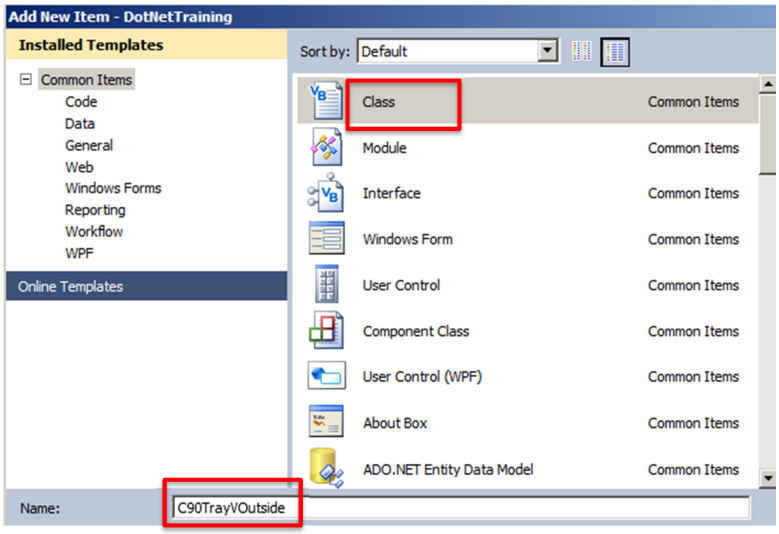
CableTray Symbol

Lab 4 - Fixed 90 deg Vertical Outside CableTray Symbol



Create a VB.net Class Library Project

- Choose **Project** > **Add Class**
- Pick **Class Library** template
- Specify **Name** of Class
- C90TrayVOutside
- Hit **OK**.



Import Required Namespaces

- Add the required “Imports” statements at the top of the file to enable you use the types in that namespace without using the full name of the class when needed.

C90TrayVOutside

```
Option Explicit On
Imports System.Math
Imports System.Collections.ObjectModel
Imports Ingr.SP3D.Common.Middle
Imports Ingr.SP3D.Common.Middle.Services
Imports Ingr.SP3D.Equipment.Middle
Imports Ingr.SP3D.ReferenceData.Middle
```

- This lets you just use

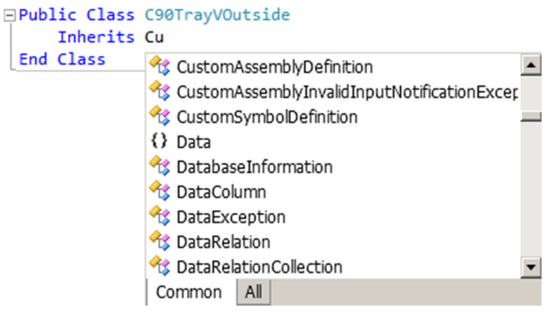
  Dim oPart As Part

  instead of :

  Dim oPart As Ingr.SP3D.ReferenceData.Middle.Part

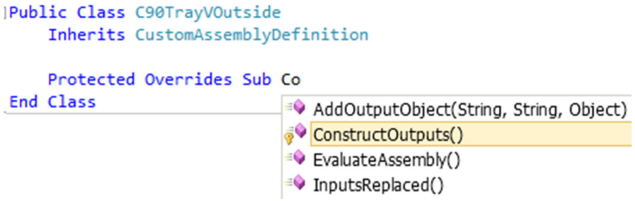
Base CustomSymbolDefinition Class

- To be a symbol you must ***inherit*** from the class **CustomSymbolDefinition**



- We then provide the ***override implementation*** for the properties/methods

  i.e. the **ConstructOutputs()**



Base CustomSymbolDefinition Class

```
<SymbolVersion("1.0.0.0")> _
<VariableOutputs()> _
Public Class C90TrayVOutside
    Inherits CustomSymbolDefinition
```

SymbolVersion Attribute
VariableOutputs Attribute

Public Class **SymbolVersionAttribute**
Inherits System.Attribute
Member of Ingr.SP3D.Common.Middle.Services
Summary:
SymbolVersionAttribute class defined on a CustomSymbolDefinition class is used to specify the version number of a symbol. The version should be specified as a string in the format of #.#.#.#.

Public Class **VariableOutputsAttribute**
Inherits System.Attribute
Member of Ingr.SP3D.Common.Middle
Summary:
Attribute defined on a CustomSymbolDefinition class defining that the symbol has variable outputs.

Definition of Inputs

- Declare Input parameters as member variables in the Custom Symbol Definition class
  - InputCatalogPart(Index)
  - InputDouble(Index, Name, Description, Default Value, Optional)
  - InputString(Index, Name, Description, Default Value, Optional)

```
<InputCatalogPart(1)> Public m_oPart As InputCatalogPart
  <InputDouble(2, "FrameRadius", "Frame Radius", 0.015)> Public m_dFrameRadius As InputDouble
```

Definition of Inputs

```
<SymbolVersion("1.0.0.0")> _
<VariableOutputs()> _
Public Class C90TrayVOutside
    Inherits CustomSymbolDefinition

#Region "Definition of Inputs"

    <InputCatalogPart(1)> Public m_oPart As InputCatalogPart
    <InputDouble(2, "FrameRadius", "Frame Radius", 0.015)> Public m_dFrameRadius As InputDouble

#End Region
```

InputCatalogPart Attribute
Define the global member catalog part
- This is always the 1st attribute defined

InputDouble Attribute
Define a global member attribute that is a Double type.
- 2 is the index in the inputs list
- "Frameradius" is the name of the attribute
- "Frame radius" is the description of the attribute
- 0.015 is the default value

Definition of Aspects and their Outputs

- Declare Representations as member variables in the Custom Symbol Definition class
- Declare Outputs as attributes on each Representation
  - Aspect(Name, Description, Aspect ID)
  - SymbolOutput(Name, Description)

```
<Aspect("Physical", "Physical Aspect", AspectID.SimplePhysical)> _
  <SymbolOutput("TrayPort1", "Tray Port 1")> _
  <SymbolOutput("TrayPort2", "Tray Port 2")> _
  Public m_oPhysicalAspect As AspectDefinition
```

Definition of Aspects and their Outputs

```
#Region "Definitions of Aspects and their outputs"

    ' Physical Aspect
    <Aspect("Physical", "Physical Aspect", AspectID.SimplePhysical)> _
    <SymbolOutput("TrayPort1", "Tray Port 1")> _
    <SymbolOutput("TrayPort2", "Tray Port 2")> _
    Public m_oPhysicalAspect As AspectDefinition

#End Region
```

Define each Aspect the symbol will support
- "Physical" is the Aspect name
- "Physical Aspect" is the description
- AspectID.SimplePhysical is the Aspect Id

Definition of ConstructOutputs() Sub

```
#Region "Construction of outputs of all aspects"

    Protected Overrides Sub ConstructOutputs()

        Dim oTrayPart As Part, oConnection As SP3DConnection
        Dim oTrayPort1 As CableTrayPortDef, oTrayPort2 As CableTrayPortDef

        Dim dFrameRadius As Double
        Dim dWidth As Double, dDepth As Double, dBendRad As Double

        ' ================================== Get Input values ==============================
        oConnection = OccurrenceConnection
        dFrameRadius = m_dFrameRadius.Value

        oTrayPart = m_oPart.Value
        If (CType(oTrayPart.PortDefinitions.Item(0), CableTrayPortDef).PortIndex = 1) Then
            oTrayPort1 = oTrayPart.PortDefinitions.Item(0)
            oTrayPort2 = oTrayPart.PortDefinitions.Item(1)
        Else
            oTrayPort1 = oTrayPart.PortDefinitions.Item(1)
            oTrayPort2 = oTrayPart.PortDefinitions.Item(0)
        End If
```

Define local variables for the subroutine

Get the part and ports from the catalog

Definition of ConstructOutputs() Sub

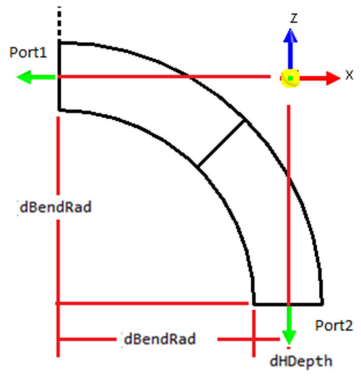
```
Dim oPropertyValueDouble As PropertyValueDouble
' Get Property Values from the part and port1
oPropertyValueDouble = oTrayPart.GetPropertyValue("IJCableTrayPart", "BendRadius")
dBendRad = oPropertyValueDouble.PropValue
oPropertyValueDouble = oTrayPort1.GetPropertyValue("IJCableTrayPort", "ActualWidth")
dWidth = oPropertyValueDouble.PropValue
oPropertyValueDouble = oTrayPort1.GetPropertyValue("IJCableTrayPort", "ActualDepth")
dDepth = oPropertyValueDouble.PropValue
```

Get the Part Definition Data using Business Object Generic Property Access method

Definition of ConstructOutputs() Sub

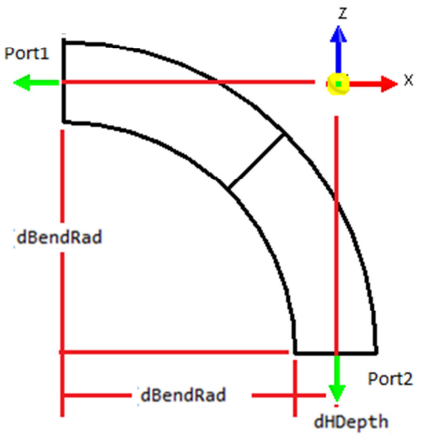
```
'================================================
' Construction of Physical Aspect
'================================================

Dim dHDepth As Double, dHWidth As Double, dAngle As Double
Dim dFacetoCenter As Double, dTubeLength As Double

dHDepth = dDepth / 2
dHWidth = dWidth / 2
dFacetoCenter = (dBendRad + dHDepth)
dTubeLength = (dBendRad + dDepth)
dAngle = Math.PI / 2.0

Dim oXAxis As New Vector(1, 0, 0), oYAxis As New Vector(0, 1, 0), oZAxis As New Vector(0, 0, 1)
Dim oNegXAxis As New Vector(-1, 0, 0), oNegYAxis As New Vector(0, -1, 0), oNegZAxis As New Vector(0, 0, -1)
```

Define local variables for the subroutine



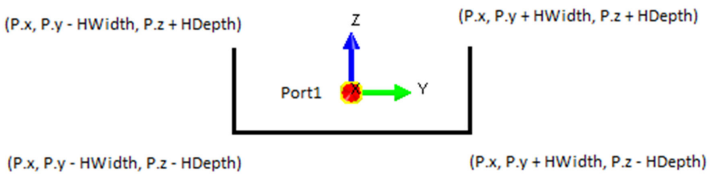
Definition of ConstructOutputs() Sub

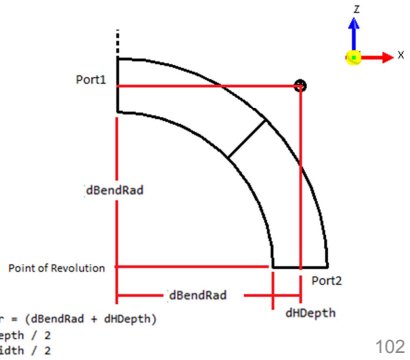
```
' ================================== Place the Ports ==============================
' Define the port positions
Dim oPort1location As New Position(-dFacetoCenter, 0, 0)
Dim oPort2location As New Position(0, 0, -dFacetoCenter)

' Place Nozzle 1
Dim oCableTrayPort = New CableTrayPort(oTrayPart, oConnection, 1, oPort1location, oNegXAxis)
oCableTrayPort.RadialVector = oZAxis
m_oPhysicalAspect.Outputs("TrayPort1") = oCableTrayPort

' Place Nozzle 2
oCableTrayPort = New CableTrayPort(oTrayPart, oConnection, 2, oPort2location, oNegZAxis)
oCableTrayPort.RadialVector = oXAxis
m_oPhysicalAspect.Outputs("TrayPort2") = oCableTrayPort
```



Definition of ConstructOutputs() Sub

```
' ================================== Main Body Output ==================================

Dim oGeomHlpr As New SymbolGeometryHelper()
'Define point and axis of revolution
oGeomHlpr.ActivePosition = New Position(-dFacetoCenter, 0, -dBendRad - dHDepth)
oGeomHlpr.SetOrientation(oYAxis, oZAxis)

'Define U shape cross section
Dim oPointsColl As New Collection(Of Position)
oPointsColl.Add(New Position(oPort1location.X, oPort1location.Y + dHWidth, oPort1location.Z + dHDepth))
oPointsColl.Add(New Position(oPort1location.X, oPort1location.Y + dHWidth, oPort1location.Z - dHDepth))
oPointsColl.Add(New Position(oPort1location.X, oPort1location.Y - dHWidth, oPort1location.Z - dHDepth))
oPointsColl.Add(New Position(oPort1location.X, oPort1location.Y - dHWidth, oPort1location.Z + dHDepth))
Dim oLineStr = New LineString3d(oPointsColl)
Dim oElbow = oGeomHlpr.CreateSurfaceofRevolution(oConnection, oLineStr, dAngle)
m_oPhysicalAspect.Outputs("Body1") = oElbow
oPointsColl.Clear()
```

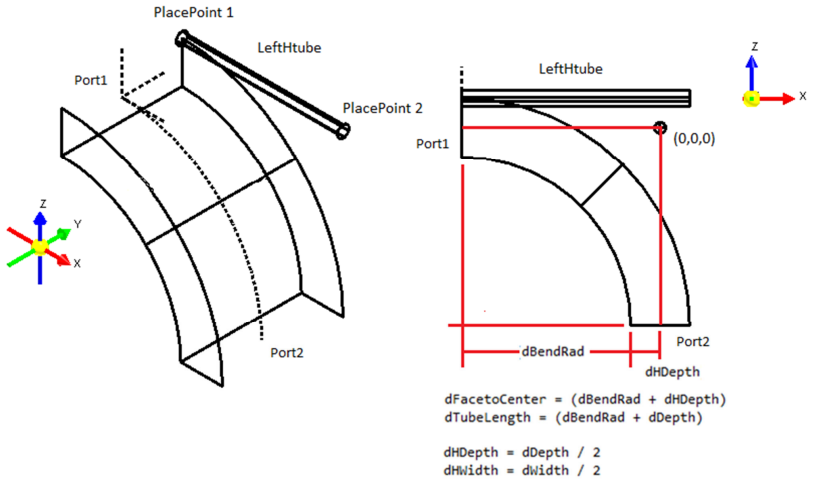





```
dFacetoCenter = (dBendRad + dHDepth)
dHDepth = dDepth / 2
dHWidth = dWidth / 2
```

Definition of ConstructOutputs() Sub

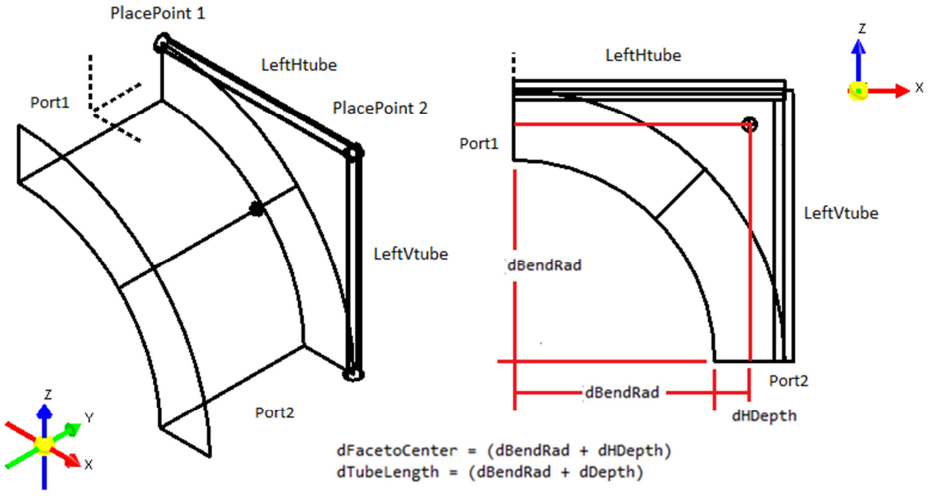
```
' ================================== Rails Outputs ====================================

' Reset current position to the left top side of port1
oGeomHlpr.ActivePosition = New Position(-dFacetoCenter, dHWidth, dHDepth)
'Save current position as Point1
oGeomHlpr.DefinePlacePoint(New Matrix4X4(oGeomHlpr.GetActiveMatrix()), 1)
oGeomHlpr.SetOrientation(oXAxis, oZAxis)
m_oPhysicalAspect.Outputs("LeftHtube") = oGeomHlpr.CreateCylinder(oConnection, dFrameRadius, dTubeLength)
'Save current position as Point2
oGeomHlpr.DefinePlacePoint(New Matrix4X4(oGeomHlpr.GetActiveMatrix()), 2)
```




```
dFacetoCenter = (dBendRad + dHDepth)
dTubeLength = (dBendRad + dDepth)

dHDepth = dDepth / 2
dHWidth = dWidth / 2
```


Definition of ConstructOutputs() Sub

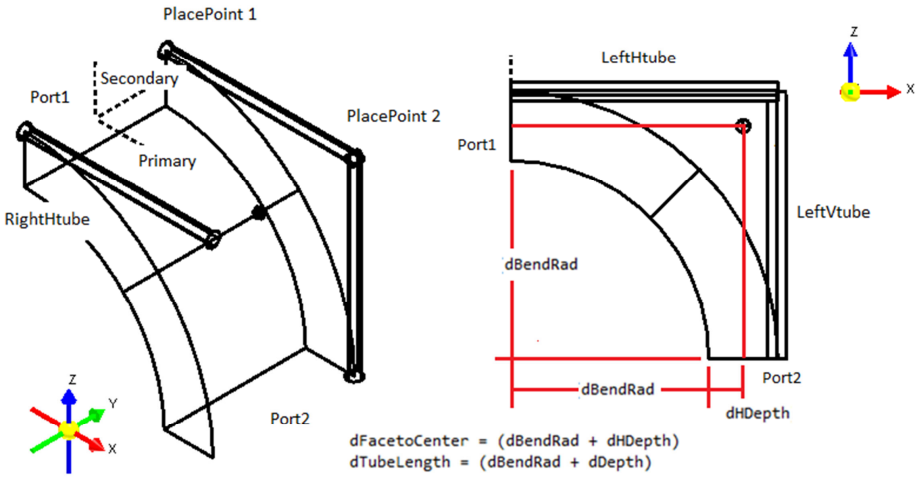
```
oGeomHlpr.SetOrientation(oNegZAxis, oXAxis)
m_oPhysicalAspect.Outputs("LeftVtube") = oGeomHlpr.CreateCylinder(oConnection, dFrameRadius, dTubeLength)
```




```
dFacetoCenter = (dBendRad + dHDepth)
dTubeLength = (dBendRad + dDepth)

dHDepth = dDepth / 2
dHWidth = dWidth / 2
```

Definition of ConstructOutputs() Sub

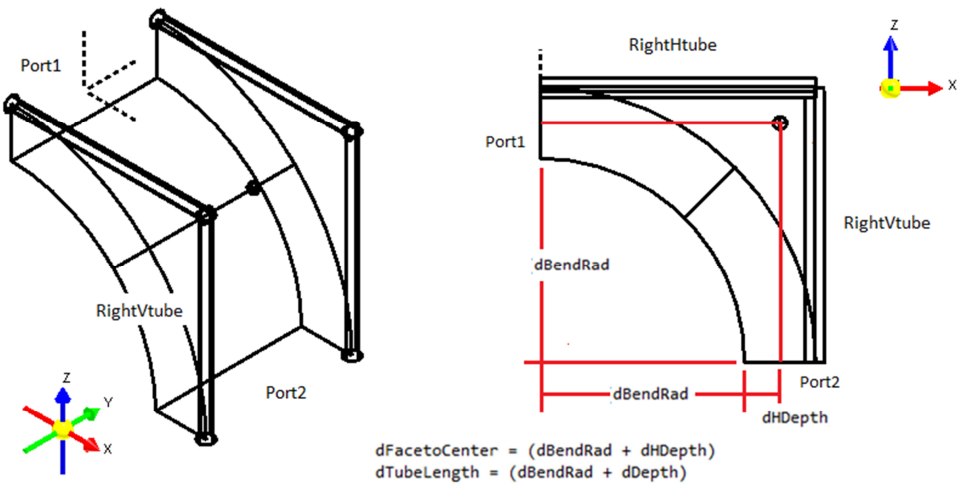
```
'Move back current position to point1
oGeomHlpr.MoveToPlacePoint(1)
oGeomHlpr.MoveAlongAxis(-dWidth, SymbolGeometryHelper.AxisDirection.Secondary)
oGeomHlpr.SetOrientation(oXAxis, oZAxis)
m_oPhysicalAspect.Outputs("RightHtube") = oGeomHlpr.CreateCylinder(oConnection, dFrameRadius, dTubeLength)
```




```
dFacetoCenter = (dBendRad + dHDepth)
dTubeLength = (dBendRad + dDepth)

dHDepth = dDepth / 2
dHWidth = dWidth / 2
```


Definition of ConstructOutputs() Sub

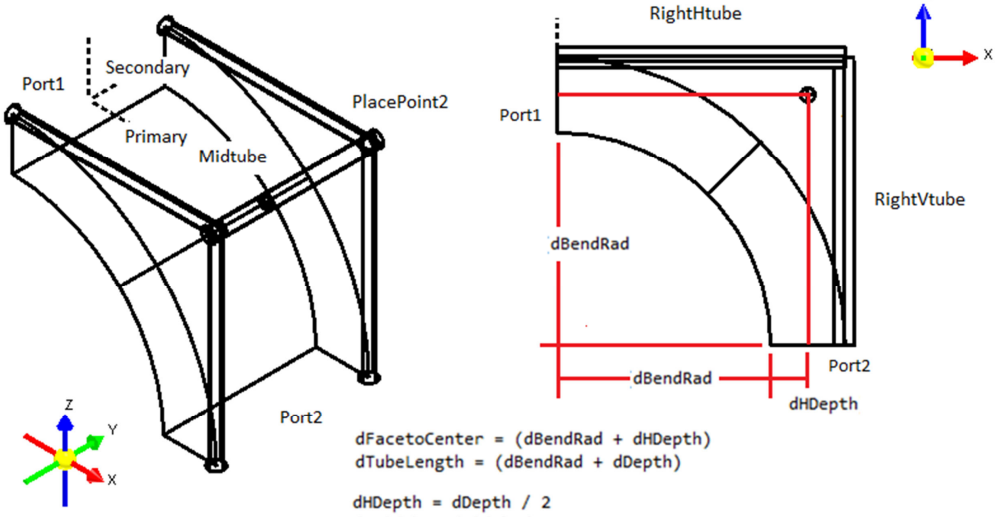
```
' Set the orientation vector
oGeomHlpr.SetOrientation(oNegZAxis, oXAxis)
m_oPhysicalAspect.Outputs("RightVtube") = oGeomHlpr.CreateCylinder(oConnection, dFrameRadius, dTubeLength)
```




```
dFacetoCenter = (dBendRad + dHDepth)
dTubeLength = (dBendRad + dDepth)

dHDepth = dDepth / 2
dHWidth = dWidth / 2
```

Definition of ConstructOutputs() Sub

```
        'Move back current position to point2
        oGeomHlpr.MoveToPlacePoint(2)
        oGeomHlpr.MoveAlongAxis(dFrameRadius, SymbolGeometryHelper.AxisDirection.Secondary)
        oGeomHlpr.SetOrientation(oNegYAxis, oZAxis)
        m_oPhysicalAspect.Outputs("Midtube") = _
            oGeomHlpr.CreateCylinder(oConnection, dFrameRadius, dWidth + 2 * dFrameRadius)

    End Sub
#End Region

End Class
```




```
dFacetoCenter = (dBendRad + dHDepth)
dTubeLength = (dBendRad + dDepth)

dHDepth = dDepth / 2
dHWidth = dWidth / 2
```


Part Class Definition

- Open the [Install Product]\ CatalogData\BulkLoad\Datafiles\CableTray.xls
- Make sure to remove the Read-Only setting on the file
- Create a new part class using the new symbol definition DotNetTraining,Ingr.S3D.Content.RouteSymbols.C90TrayVOutside
- Add a fixed 90 degree 12”x4” Vertical Outside CableTray Part as shown below:

Definition	PartClassType	SymbolDefinition
a	CableTrayClass	DotNetTraining,Ingr.S3D.Content.RouteSymbols.C90TrayVOutside

			Common Key Inputs				Component Specific Inputs										
Head	PartNumber	PartDescription	Manufacturer	Material	TrayType	ComponentType	Length	LoadSpanClassification	RungSpacing	TangentLength	BendAngle	BendRadius	MirrorBehaviorOption	PartDataBasis	InsertionDepth	ReplacementPartNumber
Start																	
a	4P-12-90VOF12	4P-12-90VOF12	174	10	5	20		25			90Deg	12in	5			
End																	

CT90VOBendFrame

Part Class Definition

- Add a fixed 90 degree 12”x4” Vertical Outside CableTray Part as shown below:

								Port Data												
NominalWidth	NominalDepth	ReducingSize	SymbolDefinition	DryWeight	DryCogX	DryCogY	DryCogZ	NominalWidth[1]	NominalDepth[1]	ActualWidth[1]	ActualDepth[1]	LoadWidth[1]	LoadDepth[1]	NominalWidth[2]	NominalDepth[2]	ActualWidth[2]	ActualDepth[2]	LoadWidth[2]	LoadDepth[2]	FrameRadius
12in	4in			5Kg	0	0	0	12in	4in	12.125in	4.188in	12in	4in	12in	4in	12.125in	4.188in	12in	4in	15mm

CT90VOBendFrame

Custom Interface sheet

- Add new Interface in the CustomInterfaces sheet

Head	InterfaceName	AttributeName	AttributeUserName	Type	UnitsType	PrimaryUnits	OnPropertyPage	ReadOnly	SymbolParameter
Start									
	IJUAFrameRadius	FrameRadius	Frame Radius	Double	Distance	in	TRUE	FALSE	FrameRadius

CustomInterfaces

R-ClassNodeDescribes sheet

- Add new record in R-ClassNodeDescribes sheet

Head	RelationSource	RelationDestination
Start		
	CableTrayStraights	Straight
!	Reducers	
	CTStraightReducers	CTStraightReducer
	CTRightReducers	CTRightReducer
	CTLeftReducers	CTLeftReducer
!	Bends	
	CTHorizontalBends	CT90HBend
	CTHorizontalBends	CT60HBend
	CTHorizontalBends	CT45HBend
	CTHorizontalBends	CT30HBend
!		
a	CTVerticalBends	CT90VOBendFrame
	CTVerticalBends	CT90VOBend
	CTVerticalBends	CT60VIBend

R-ClassNodeDescribes