Introduction to .NET Naming Rules

Consulting Services, SmartPlant 3D

Agenda

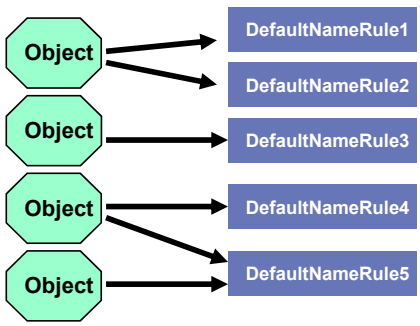
- Overview
- Naming Rule Data Model
- Creation of Naming Rules
- Generic Property and Relationship Access
- Modification of Naming Rules
- Deployment and Guidelines
- Debugging your code
- Open Discussion / Follow up Questions

Naming Rules: Overview

- Customized program that creates custom names of business objects
- Name can be comprised of the following:
  - Object type
  - Object properties
  - Primary organizational relationships
    - System
    - Space
    - Assembly
    - Work Breakdown
  - Other relationships
    - Connection

Naming Rules: Overview

- Naming rules are associated to an object class
- Multiple rules can be defined for each class of object
  - If multiple rules are provided in the catalog, the user must select the rule to apply when objects are created
- A rule can also be shared by multiple object classes



Naming Rules: Reference Data

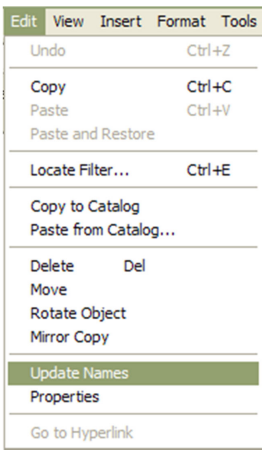
- Mapping of rules to object classes
  Example: GenericNamingRules.xls
  [Installation]\CatalogData\BulkLoad\Datafiles

Object Type Name in GUI VB or NET ProgID

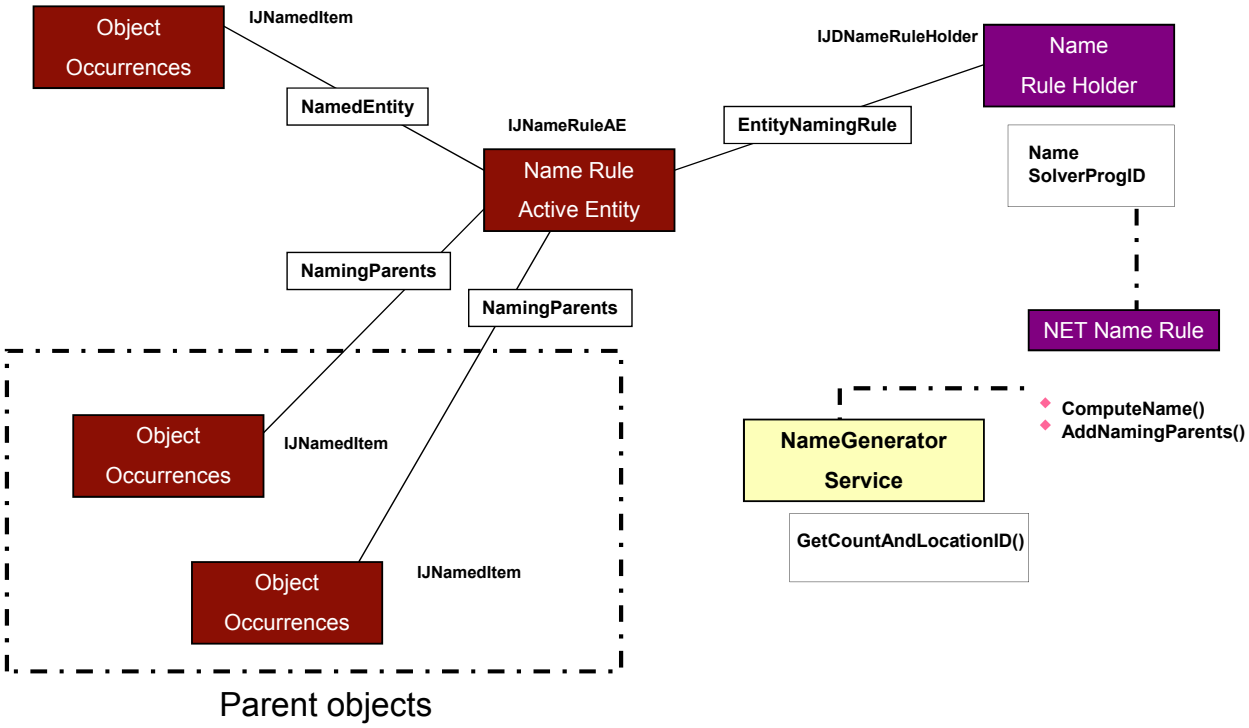
Head	TypeName	Name	SolverProgID
Start			
	CPMPipeRun	DefaultNameRule	RouteRunNameRules.PipeRunNameRule
	CArea	DescriptionRule	SpaceRules.AreaDescRule
	CArea	SpaceRule	SpaceRules.AreaSpaceRule
	CZone	DescriptionRule	SpaceRules.ZoneDescRule
	CZone	SpaceRule	SpaceRules.ZoneSpaceRule
	CPInterferenceVolume	DescriptionRule	SpaceRules.IFVolumeDescRule
	CPInterferenceVolume	SpaceRule	SpaceRules.IFVolumeSpaceRule
	CHgrSupportComponent	DefaultNameRule	HgrSupNameRule.CompNameRule
	CHgrSupport	Tag NameRule	HgrSupNameRule.TagNameRule
	CSPGElevationPlane	Index	GSNamingRules.IndexNameRule
	CSPGElevationPlane	Position	GSNamingRules.PositionNameRule
	CSPGGridPlane	Index	GSNamingRules.IndexNameRule
	CSPGGridPlane	Position	GSNamingRules.PositionNameRule
	CSPGGridPlane	Alphanumeric and Percent	GSNamingRules.AlphaNPercent

Naming Rules: Re-Compute

- Naming rule semantic is responsible for calling the ComputeName Method when the following actions occur:
  - A property on the named object is modified.
  - A property on the naming parent object is modified.
  - System parent or assembly parent is changed.
- Use the Update Name to re-fire the naming rule semantic to all objects in the select set



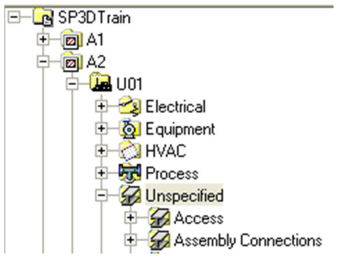
Naming Rules: Data Model

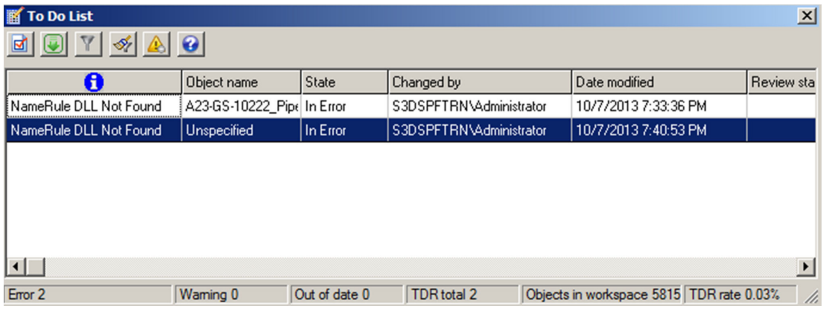


Naming Rules: Error Handling

If a name rule returns an error, the generic semantic will trap the error

- Set the name to “unspecified”
- Add the object to the To Do list

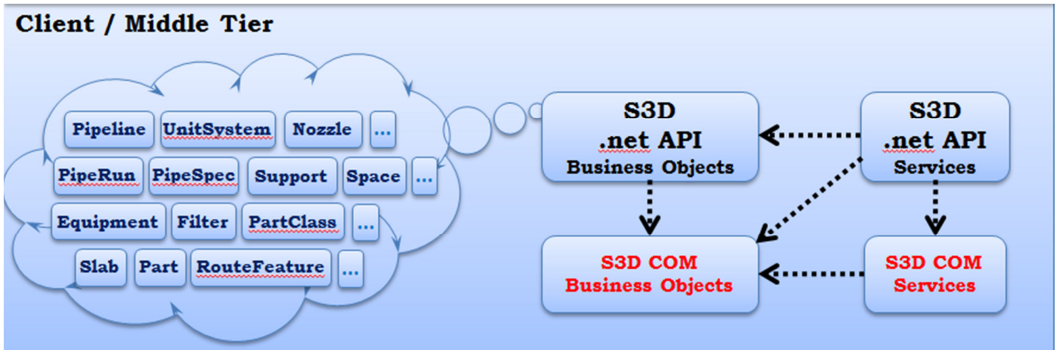




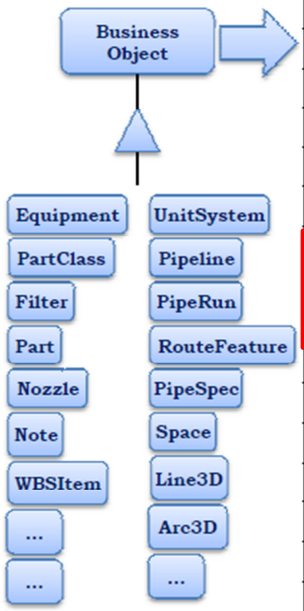
*** Error ***
Time : 10/05/13 08:58:17
ID : 0x80004005(-2147467259)(16389)
Source : NameRuleCompute.cpp
Desc : SymbolConfig map files do not contain an entry for name rule with SolverProgId : NetNaminRuleTraining,Ingr.SP3D.Content.NamingRule.CPipeline
Help : NameRuleCompute.cpp;482

Naming Rules: Business Object

- Abbreviated as **BO**. BusinessObjects are the engineering things like Equipment, PipeRun, PipeFeature, ControlPoint and so on
- A .net BusinessObject is also referred to as ***Wrapper***, as it essentially wraps the Business Object and provides property/method access
- Provides many useful methods and properties, including generic access to properties and relationships

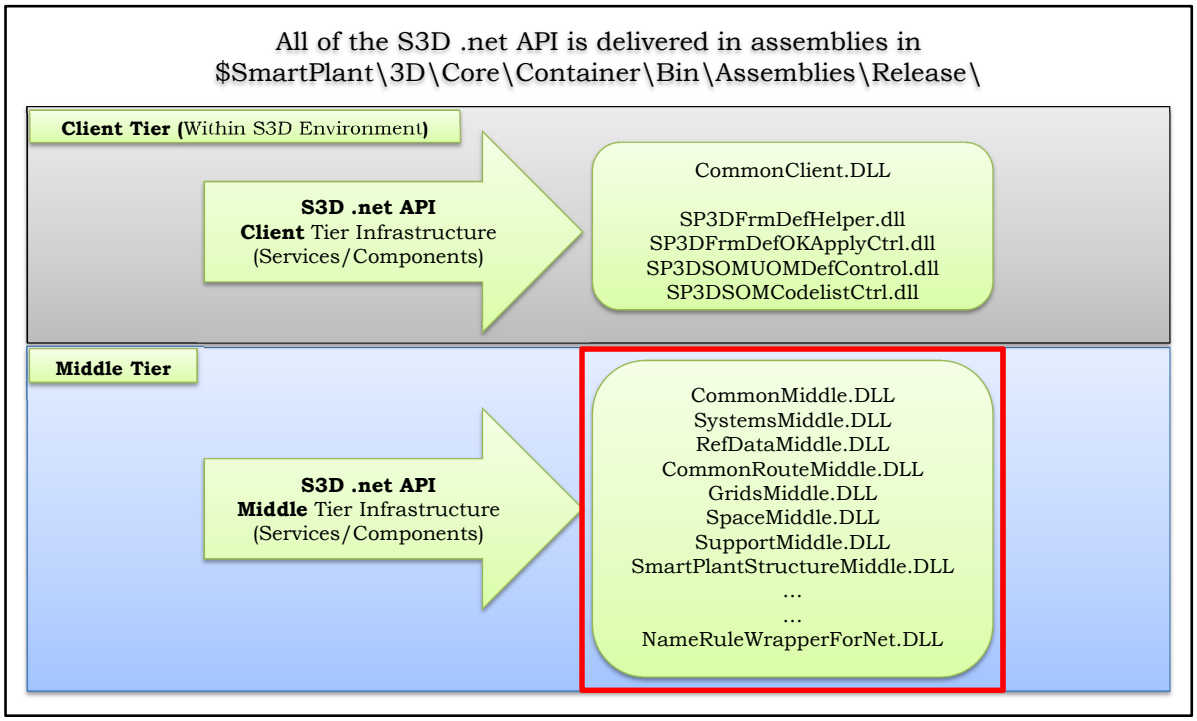


Naming Rules: Business Object



AccessControl	Gets AccessControl to current user
ApprovalStatus	Gets ApprovalStatus of the object
ClassInfo	Gets ClassInfo (Metadata) of this object
DBConnection	Gets Connection to which this BO belongs.
Delete()	Deletes the Object (if permissions allow)
GetAllProperties	Gets PropertyValue ReadOnlyCollection of all properties
GetPropertyValue()	Gets PropertyValue given Property Details
SetPropertyValue()	Sets PropertyValue given Property Details and value
GetRelationship ()	Gets RelationshipCollection given RelationshipInfo.
IsTypeofBOC ()	Checks if this BO is of given BO Classification type.
Notes ()	Gets Notes associated with this BO (as ReadOnlyCollection).
RemoveAllNotes ()	Removes all Notes associated with this BO.
RemoveNote ()	Removes a given Note object associated with this BO.
ObjectID	Gets OID of the Object
PermissionGroup	Get/Set the PermissionGroup to which this object belongs
Relationships ()	Gets read only collection of RelationCollection.
SupportsInterface()	Check if the BO supports a given interface name
ToString()	Gets name if BO is IJNamedItem, OID if not Named Item.
UserClassInfo()	Gets ClassInfo (Metadata) for user bulkloaded partclass occs

Naming Rules: S3D.net API delivered Assemblies



Naming Rules: Creation

- Can be defined using
  - **.NET** (the new way using S3D .NET API – VB or C#)
  - **VB6.0** (The old way – COM API)

- A base class from which any customized name rule should be derived
- This class "***NameRuleBase"*** exists in CommonMiddle.dll
- It has 2 abstract methods which must be implemented in any customized name rule
  - **GetNamingParents()** - Specify Parents which drive the name of the object in question.
  - **ComputeName() –** Compose the name using
    - This objects attributes.
    - Attributes of Object's Parents.

Helper methods on NameRuleBase used in ComputeName

- ***NameRuleBase*** has helper methods which can be used in their customized name rules code when needed.
  - ***GetPart*** method returns a part if the object passed in is a part occurrence.
  - ***GetPartNumber*** is to get the part number of a part.
  - ***GetTypeString*** to get its class name of the object if the object is not a PartOccurence.
  - ***GetCountAndLocationID*** returns a unique count from name generator and locationID for the name string passed in.
  - ***Get/SetNamingParentsString*** takes name rule ActiveEntity as input and allows you to get/set the namebasis for the active entity passed in.
  - ***GetTypeString*** returns the type string for catalog and model items.
  - ***Get/SetName*** allows to get or set the name of the business object passed in.
  - ***GetParent*** returns the parent of the given object based on the enumerated hierarchy types (System/WBS/Space/Assembly/PG/VolumeNamedSpace/Analysis)

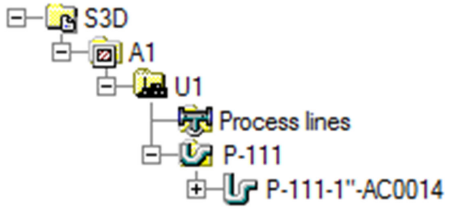
Create a VB.net Class Library Project

- 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 naming rule in a new **Class Library** Project.
    - Choose File → New Project

Naming Rules

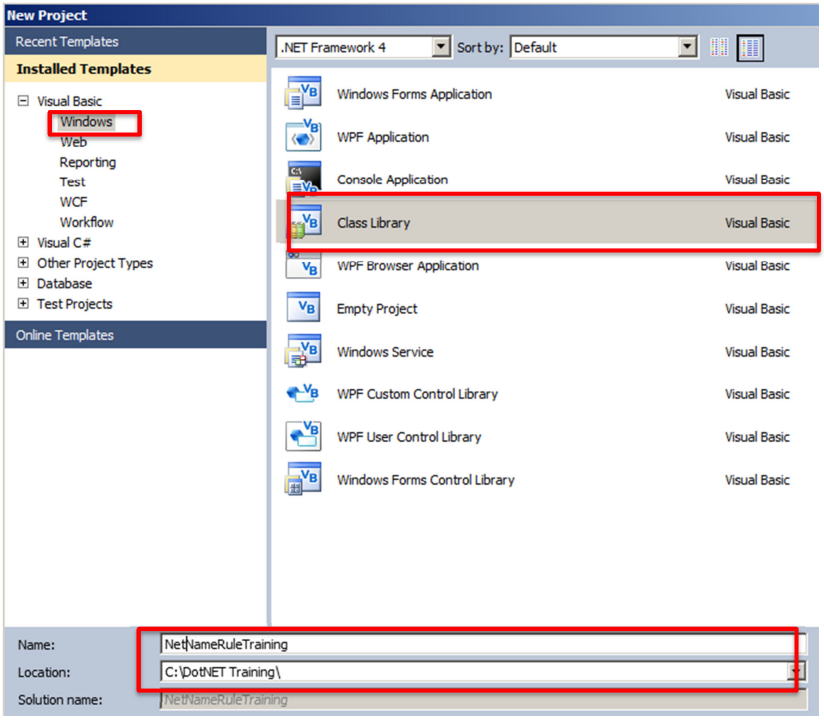
Lab 1 – Pipeline Naming Rule

Pipeline Object:
Fluid Code + Sequence Number



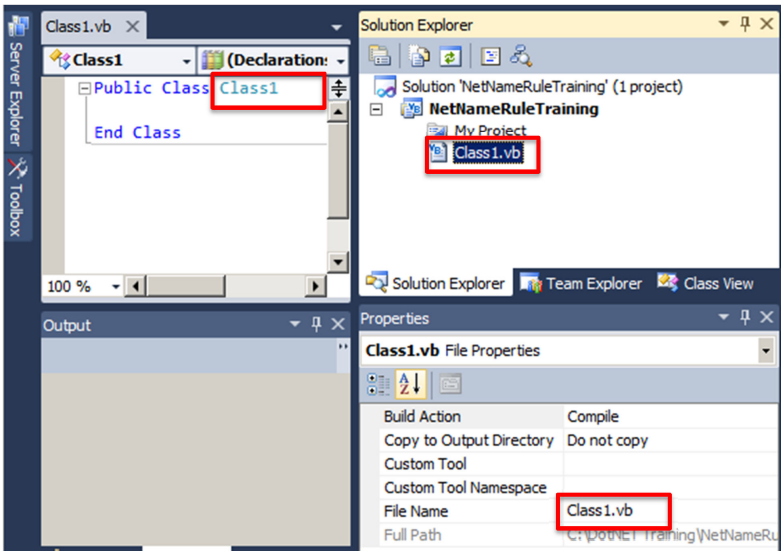
Create a VB.net Class Library Project

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



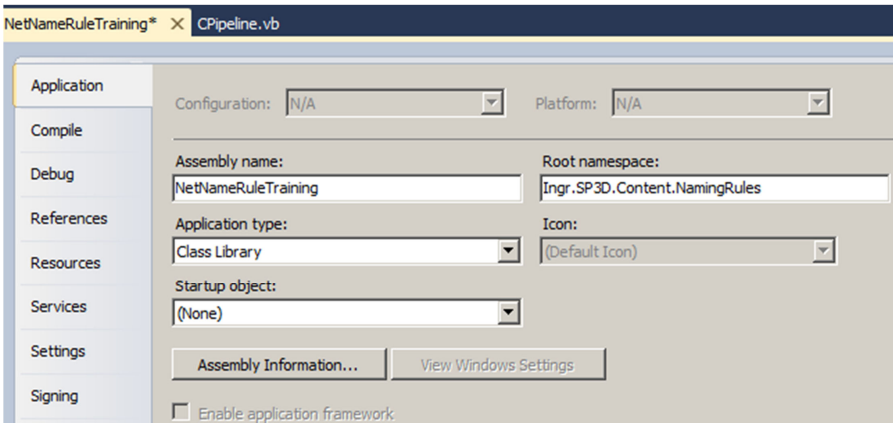
Create a VB.net Class Library Project

- 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 “CPipeline”, for the Naming Rule we will be writing here.

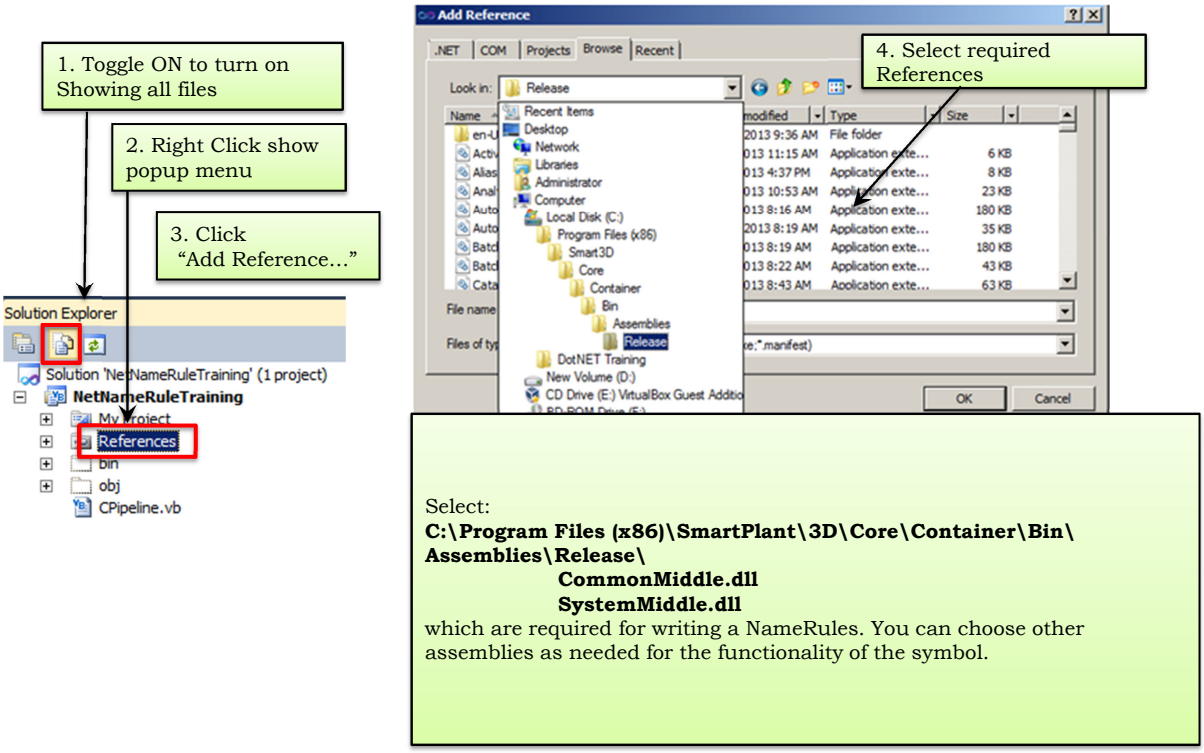


Create a VB.net Class Library Project

- Right Click the Project > Properties
- On the **Properties > Application** set the **Root Namespace** for this project
- Rename the **Root namespace:** from *ClassLibrary1* to *Ingr.SP3D.Content.NamingRules* for this class.
- Intergraph recommends the following format for your company to follow <CompanyName>.SP3D.Content.<Specialization>



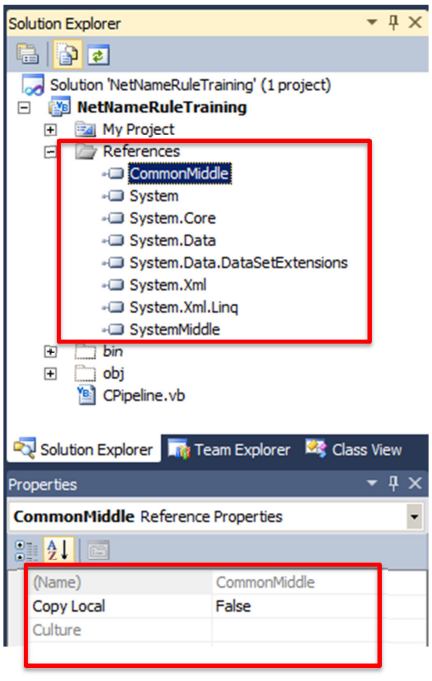
Attach Required References



Select:
C:\Program Files (x86)\SmartPlant\3D\Core\Container\Bin\Assemblies\Release
CommonMiddle.dll
SystemMiddle.dll
which are required for writing a NameRules. You can choose other assemblies as needed for the functionality of the symbol.

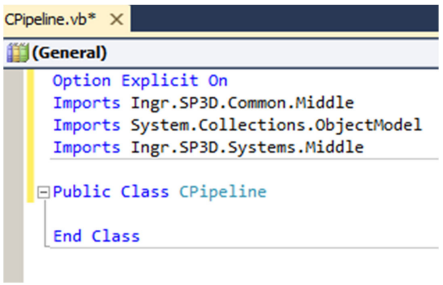
References: Copy Local Setting

- 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

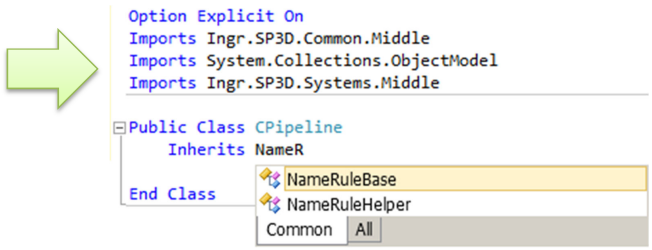
  Dim oBO As BusinessObject

  instead of :

  Dim oBO As Ingr.SP3D.Common.Middle. BusinessObject

Base NameRuleBase Class

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



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

  ComputeName() and **GetNamingParents()** methods.

```
Public Class CPipeline
   Inherits NameRuleBase

   Public Overrides Sub ComputeName(ByVal oEntity As BusinessObject, _
                ByVal oParents As ReadOnlyCollection(Of BusinessObject), _
                ByVal oActiveEntity As BusinessObject)

   End Sub

   Public Overrides Function GetNamingParents(ByVal oEntity As
                BusinessObject) As Collection(Of BusinessObject)

   End Function
End Class
```

Definition of GetNamingParents() Function

- This method is used to return the related parent object or objects that are used in forming a name

```
Public Overrides Function GetNamingParents(ByVal oEntity As
            BusinessObject) As Collection(Of BusinessObject)



End Function
```

- All the Naming Parents that need to participate in an objects naming are added here to the Collection

Definition of GetNamingParents() Function

```
Public Overrides Function GetNamingParents(ByVal oEntity As Ingr.SP3D.Common.Middle.BusinessObject) _
        As System.Collections.ObjectModel.Collection(Of Ingr.SP3D.Common.Middle.BusinessObject)

    GetNamingParents = Nothing

End Function
```

Return an Empty Collection

Definition of ComputeName() Sub

This method is used to format the name of an object

```
Public Overrides Sub ComputeName
(
ByVal oEntity As BusinessObject,  _                          'Object being named
ByVal oParents As ReadOnlyCollection(Of BusinessObject), _   'Naming Parents collection
ByVal oActiveEntity As BusinessObject                        'Naming rules active entity
)
```

```
        Dim oBO As BusinessObject
        strObjName As String
        oBO = oEntity
'……
        'Add code here to format the name….
'……..
        oEntity.SetPropertyValue(strObjectName, "IJNamedItem, "Name")
```


Generic Property Access – Get/Set Properties

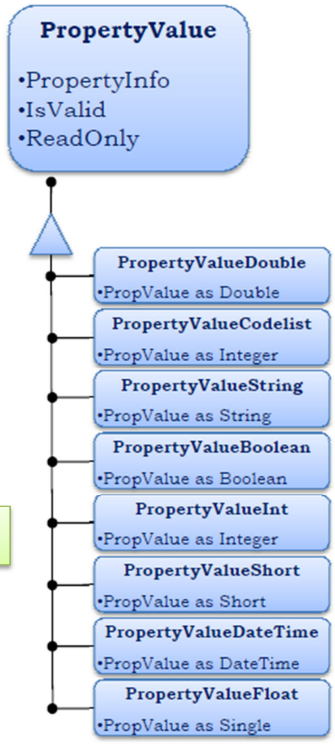
- The base class of all the Smart 3D Objects provides ***Generic Property Access*** mechanism to access any property on it
- To access a property, you need to know its InterfaceName and AttributeName
- GetPropertyValue() is the method to get object property value

BusinessObject.GetPropertyValue("InterfaceName", "Name")

- SetPropertyValue() is the method to modify object property value

BusinessObject.SetPropertyValue(Value,"InterfaceName","Name")

- PropertyValue Class is the BaseClass of different propertyValueXXX classes supported by S3D, i.e. String, Double, Codelist, etc…

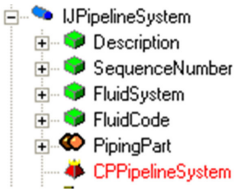


Definition of ComputeName() Sub

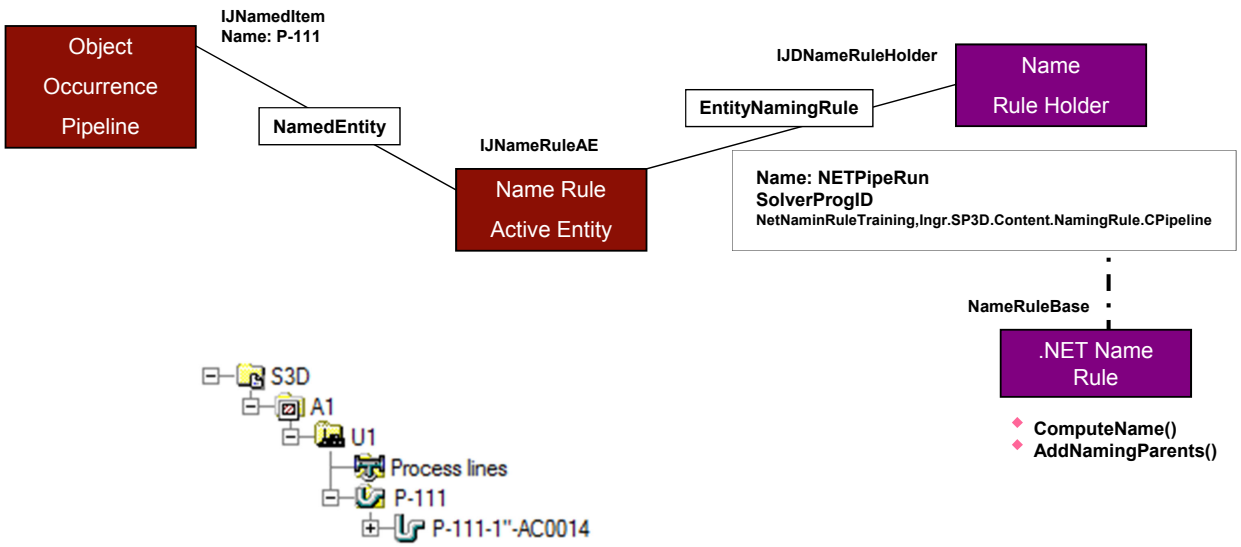
```
Public Overrides Sub ComputeName(ByVal oEntity As BusinessObject, _
            ByVal oParents As ReadOnlyCollection(Of BusinessObject), _
            ByVal oActiveEntity As BusinessObject)

    Dim strSeqNumber As String
    Dim strFluidCode As String
    Dim oStrPropValue As PropertyValueString
    Dim oCLPropValue As PropertyValueCodelist

    oCLPropValue = oEntity.GetPropertyValue("IJPipelineSystem", "FluidCode")
    strFluidCode = oCLPropValue.PropertyInfo.CodeListInfo.GetCodelistItem(oCLPropValue.PropValue).Name

    oStrPropValue = oEntity.GetPropertyValue("IJPipelineSystem", "SequenceNumber")
    strSeqNumber = oStrPropValue.PropValue

    Dim strPipelineName As String = strFluidCode & "-" & strSeqNumber
    oEntity.SetPropertyValue(strPipelineName, "IJNamedItem", "Name")

End Sub
```

Define local variables

Use the Generic Property Access

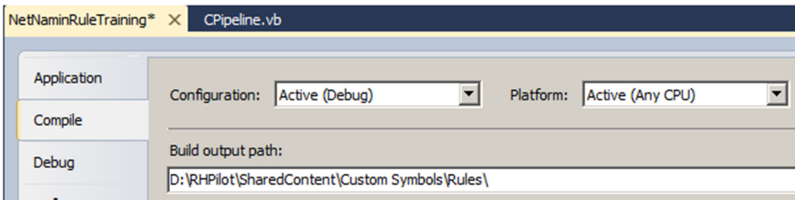


Naming Rule Data Model

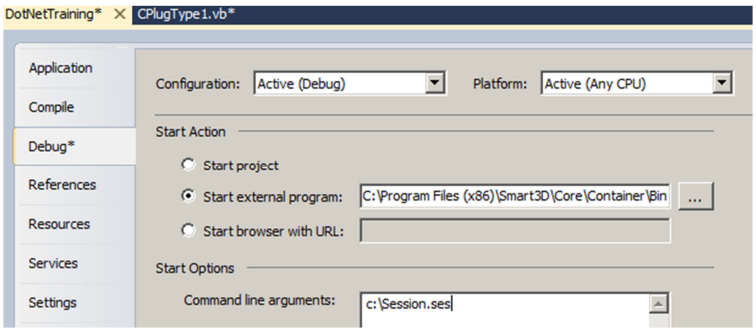


Build and Save Project

- 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***



Deploying & Testing the NamingRule

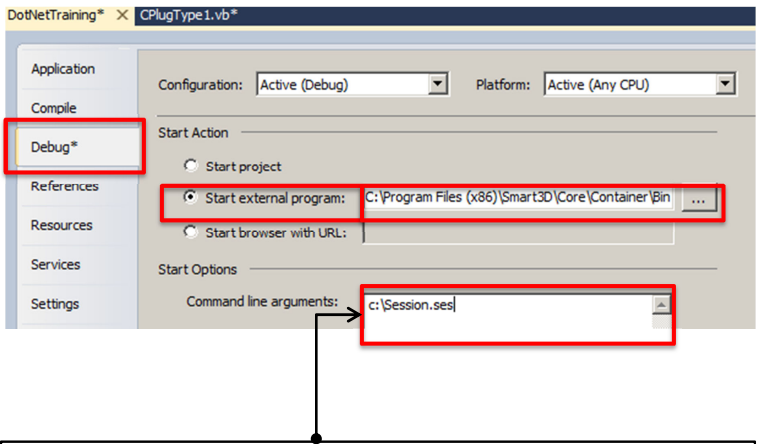
- Save the Custom Symbol DLL to the ***[SharedContent Directory]\Custom Symbols*** folder
- In Project Management Task, select the catalog you want to update then invoke menu item
  - **Tools > Update Custom Symbol Configuration**
    - The command creates or updates the file called ***CustomSymbolConfig.xml*** 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:**

- 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>
```


NamingRules sheet

- Open the [Install Product]\ CatalogData\BulkLoad\Datafiles\GenericNamingRules.xls
- Make sure to remove the Read-Only setting on the file
- Add a new record in the NamingRules sheet as shown below:

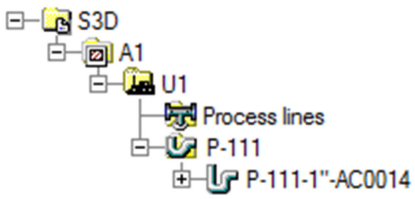
Head	TypeName	Name	SolverProgID
Start			
a	CPPipelineSystem	NET Pipeline	NetNameRuleTraining,Ingr.SP3D.Content.NamingRule.CPipeline

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

Naming Rules

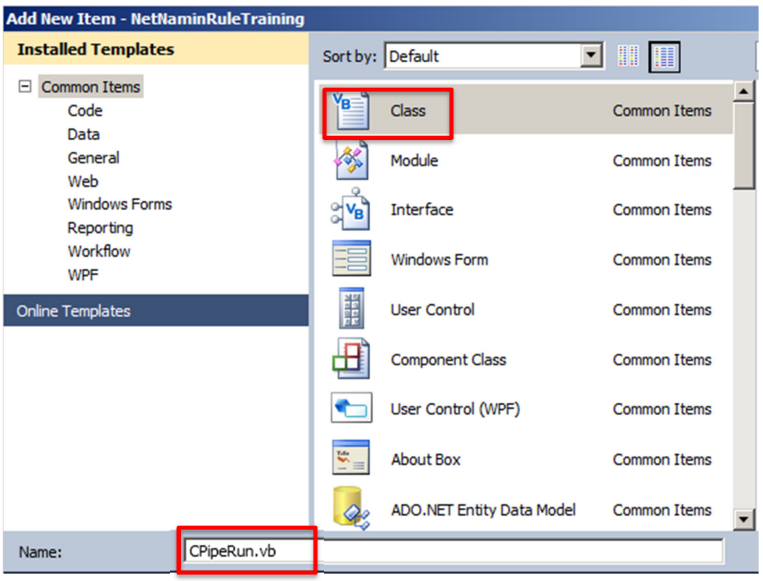
Lab 2 – PipeRun Naming Rule

PipeRun Object:
Parent Name + NPD + NPD Units + Spec Name

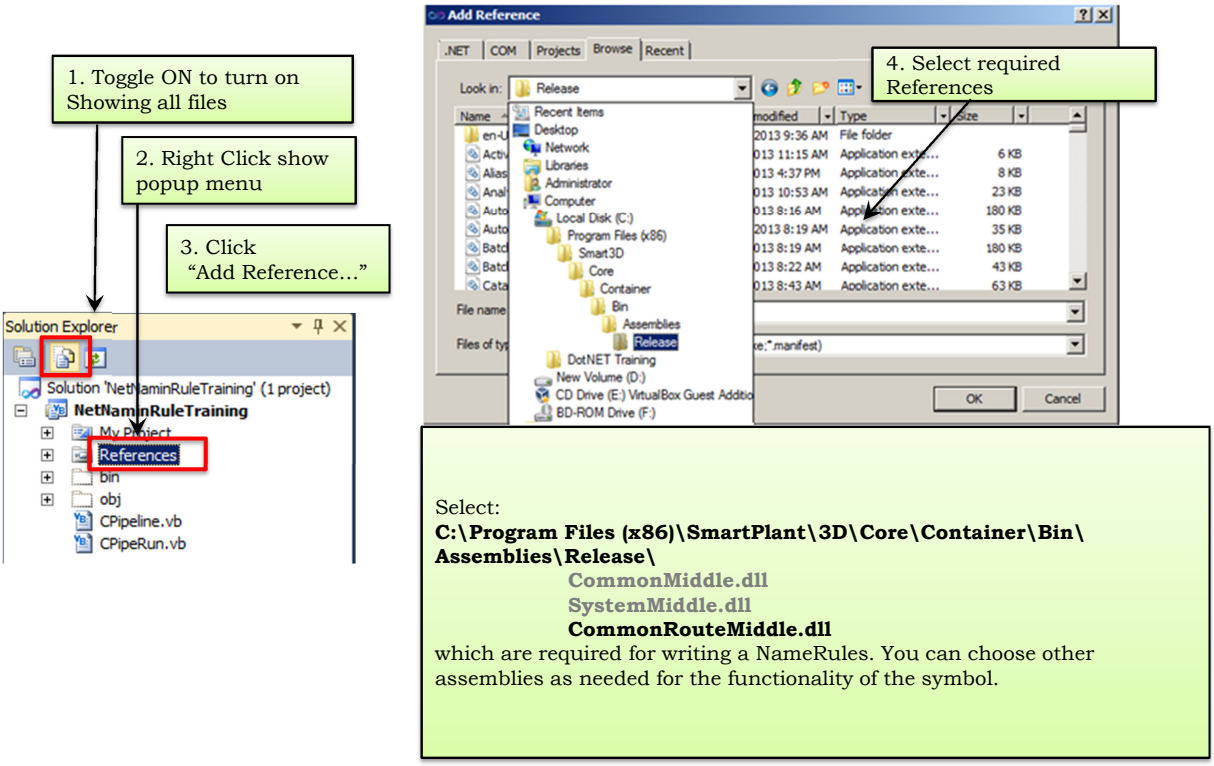


Create a VB.net Class Library Project

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



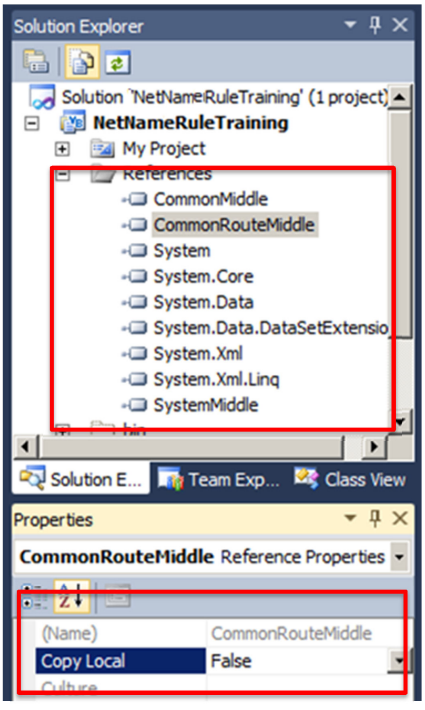
Attach Required References



Select:
C:\Program Files (x86)\SmartPlant\3D\Core\Container\Bin\Assemblies\Release
CommonMiddle.dll
SystemMiddle.dll
CommonRouteMiddle.dll
which are required for writing a NameRules. You can choose other assemblies as needed for the functionality of the symbol.

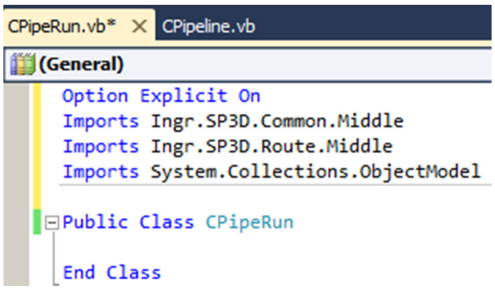
References: Copy Local Setting

- 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

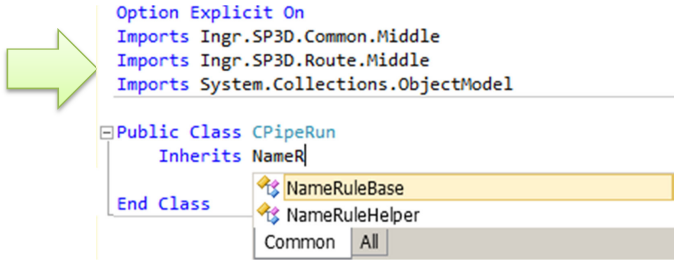
  Dim oBO As BusinessObject

  instead of :

  Dim oBO As Ingr.SP3D.Common.Middle.BusinessObject

Base NameRuleBase Class

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



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

  ComputeName() and **GetNamingParents()** methods.

```
Public Class CPipeRun
   Inherits NameRuleBase


   Public Overrides Sub ComputeName(ByVal oEntity As BusinessObject, _
                ByVal oParents As ReadOnlyCollection(Of BusinessObject), _
                ByVal oActiveEntity As BusinessObject)

   End Sub

   Public Overrides Function GetNamingParents(ByVal oEntity As
                BusinessObject) As Collection(Of BusinessObject)

   End Function
End Class
```

Definition of GetNamingParents() Function

- This method is used to return the related parent object or objects that are used in forming a name

```
Public Overrides Function GetNamingParents(ByVal oEntity As
            BusinessObject) As Collection(Of BusinessObject)



End Function
```

- All the Naming Parents that need to participate in an objects naming are added here to the Collection

Definition of GetNamingParents() Function

```
Public Overrides Function GetNamingParents(ByVal oEntity As BusinessObject) As Collection(Of BusinessObject)

    Dim oParent As BusinessObject
    Dim oParentColl As New Collection(Of BusinessObject)

    GetNamingParents = Nothing

    oParent = GetParent(HierarchyTypes.System, oEntity)
    oParentColl.Add(oParent)
    GetNamingParents = oParentColl

End Function
```

Define local variables

Use GetParent method

Protected Function **GetParent**(ByVal *oHierarchyType* As Ingr.SP3D.Common.Middle.HierarchyTypes, ByVal *oChild* As Ingr.SP3D.Common.Middle.BusinessObject) As Ingr.SP3D.Common.Middle.BusinessObject
Member of Ingr.SP3D.Common.Middle.NameRuleBase

Summary:
Gets a parent BusinessObject in the hierarchy specified for a child BusinessObject. "Volume_NamedSpace" is not implemented.

Parameters:
oHierarchyType: Hierarchy to use for finding the parent. PermissionGroup hierarchy is not supported.
oChild: BusinessObject that is a child in the specified hierarchy.

Return Values:
BusinessObject that is the parent.

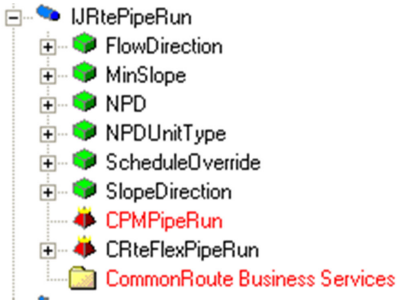
Definition of ComputeName() Sub

```
Public Overrides Sub ComputeName(ByVal oEntity As BusinessObject, _
                                 ByVal oParents As ReadOnlyCollection(Of BusinessObject), _
                                 ByVal oActiveEntity As BusinessObject)

    Dim strNPD As String
    Dim strNPDUnit As String
    Dim strSpecName As String
    Dim StrParentName As String

    Dim oPipeRun As PipeRun

    oPipeRun = oEntity
    strNPD = CStr(oPipeRun.NPD.Size)
    strNPDUnit = oEntity.GetPropertyValue("IJRtePipeRun", "NPDUnitType").ToString

    If LCase(strNPDUnit) = "in" Then
        strNPDUnit = Chr(34)
    End If
```

Define local variables

Use Route API

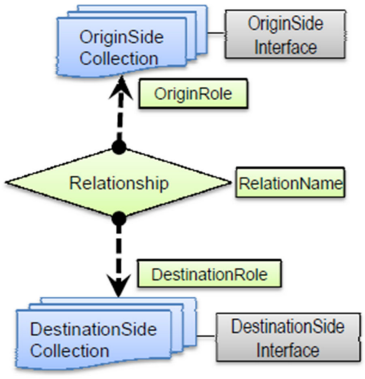
Use the Generic Property Access



Generic Relationship Access – Traverse Relationships

- **BusinessObject**, the base class of all the SmartPlant Objects, provides ***Generic Relationship Access*** mechanism to access any related Object
- To access related objects, all you need to know is the **RelationshipName** and the **RoleName** collection you are interested to access

BusinessObject.GetRelationship(RelationName, RoleName)



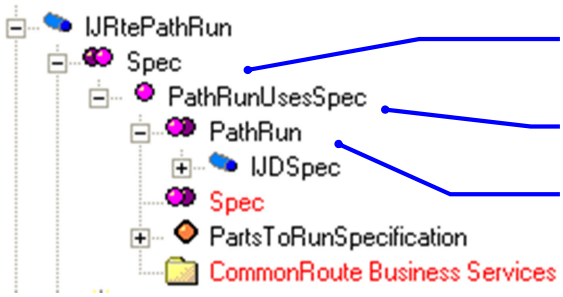
Definition of ComputeName() Sub

```
Dim oRelationCollection As RelationCollection
Dim oSpec As BusinessObject

oRelationCollection = oEntity.GetRelationship("PathRunUsesSpec", "Spec")
oSpec = oRelationCollection.TargetObjects.Item(0)


strSpecName = oSpec.GetPropertyValue("IJDPipeSpec", "SpecName").ToString
```

Define local variables

Use Generic Relationship Access

Use the Generic Property Access



Origin Role Name

Relation

Destination Role Name

Definition of ComputeName() Sub

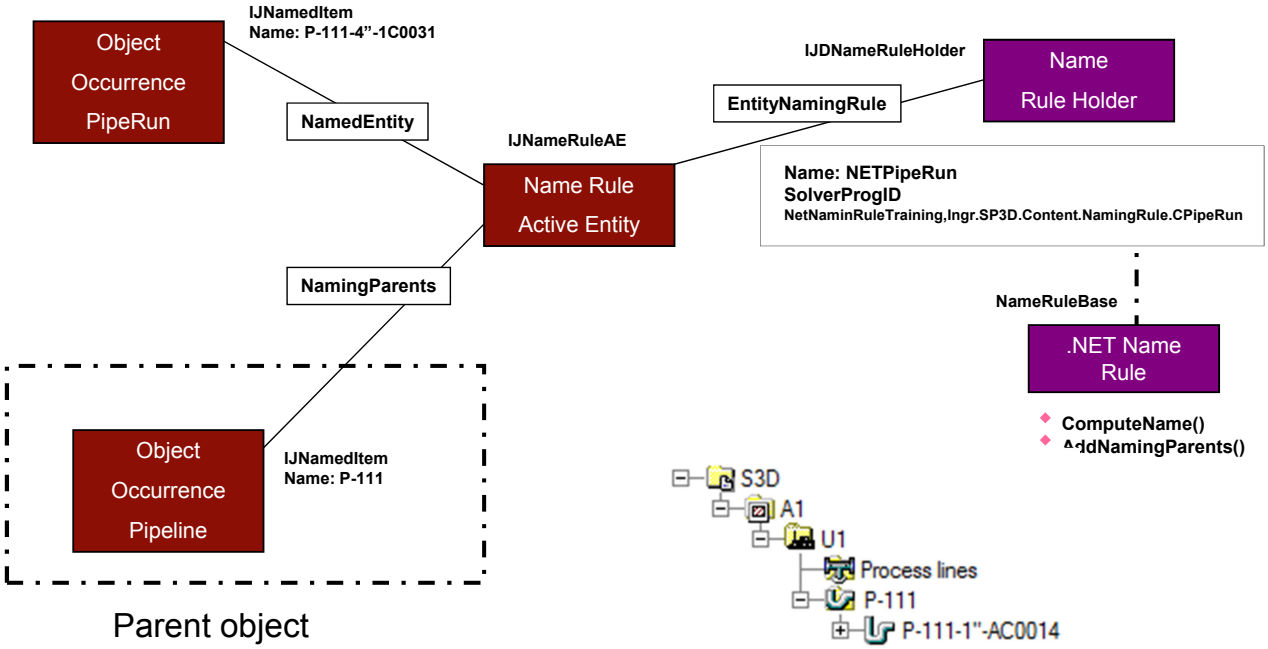
```
StrParentName = oParents.Item(0).GetPropertyValue("IJNamedItem", "Name").ToString


oEntity.SetPropertyValue(StrParentName & "-" & _
                         strNPD & strNPDUnit & "-" & strSpecName, "IJNamedItem", "Name")

End Sub
```

Use the Generic Property Access

Use the Generic Property Access

Public Overrides Function ToString() As String
Returns PropertyValue as string. For PropertyValueString, returns the string value as it is. For PropertyValueCodelist, returns the ShortString value. For PropertyValueBoolean, returns "True" or "False" string. For PropertyValueDateTime, returns string as per system's current locale. For PropertyValueDouble, returns the formatted value if the UOMType is not undefined. For PropertyValueInt, PropertyValueShort, and PropertyValueFloat, returns the value formatted as string as it is.

Naming Rule Data Model



NamingRules sheet

- Open the GenericNamingRules.xls
- Add a new record in the NamingRules sheet as shown below:

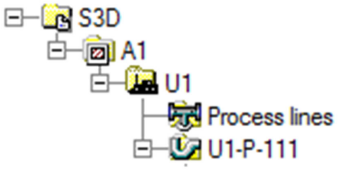
	A	B	C	D
	! Back to Index			
Head	TypeName	Name	SolverProgID	
Start				
a	CPMPipeRun	NET PipeRun	NetNameRuleTraining,Ingr.SP3D.Content.NamingRule.CPipeRun	
	CPPipelineSystem	NET Pipeline	NetNameRuleTraining,Ingr.SP3D.Content.NamingRule.CPipeline	

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

Naming Rules

Lab 3 - Modification - Pipeline Naming Rule

Pipeline Object:
Unit System Name + Fluid Code + Sequence Number



Modification – GetNamingParents() Function

```
Public Overrides Function GetNamingParents(ByVal oEntity As BusinessObject) As Collection(Of BusinessObject)

    GetNamingParents = Nothing

    Dim oChild As BusinessObject
    Dim oParent As BusinessObject
    Dim oParentColl As New Collection(Of BusinessObject)

    oChild = oEntity
    Do
        oParent = GetParent(HierarchyTypes.System, oChild)
        oChild = oParent
    Loop Until TypeOf oChild Is UnitSystem Or Not TypeOf oParent Is ISystemChild

    If oChild Is Nothing Then
        GetNamingParents = Nothing
    Else
        oParentColl.Add(oChild)
        GetNamingParents = oParentColl
    End If

End Function
End Class
```

Define local variables

Get Parent System

Loop until Parent is a Unit System or Plant

Modification of ComputeName() Sub

```
Dim strUnitName As String

If TypeOf oParents.Item(0) Is UnitSystem Then
    oStrPropValue = oParents.Item(0).GetPropertyValue("IJNamedItem", "Name")
    strUnitName = oStrPropValue.PropValue
Else
    strUnitName = "----"
End If
```

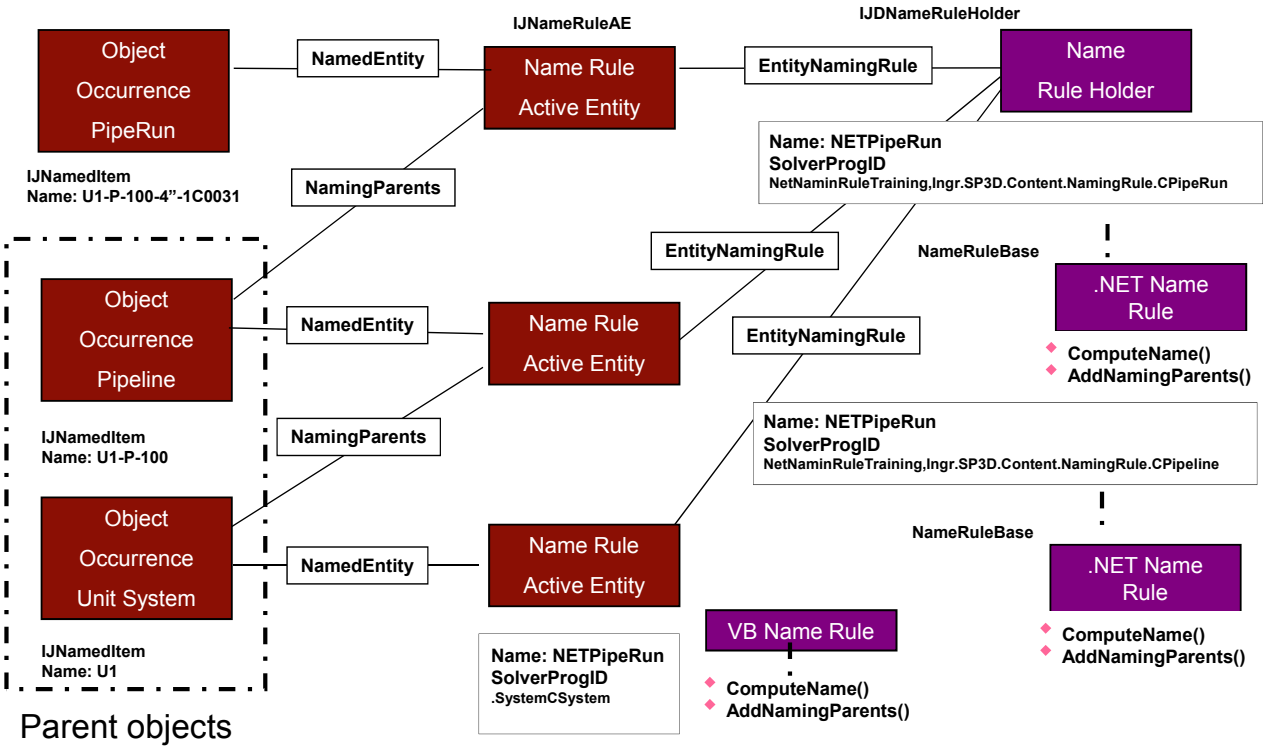
Define local variable

Use the Generic Property Access

```
Dim strPipelineName As String = strUnitName & "-" & strFluidCode & "-" & strSeqNumber
oEntity.SetPropertyValue(strPipelineName, "IJNamedItem", "Name")
```

Add Unit Name to the string

Naming Rule Data Model

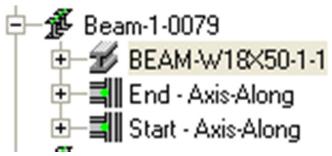


Naming Rules

Lab 4 – Member Part Naming Rule

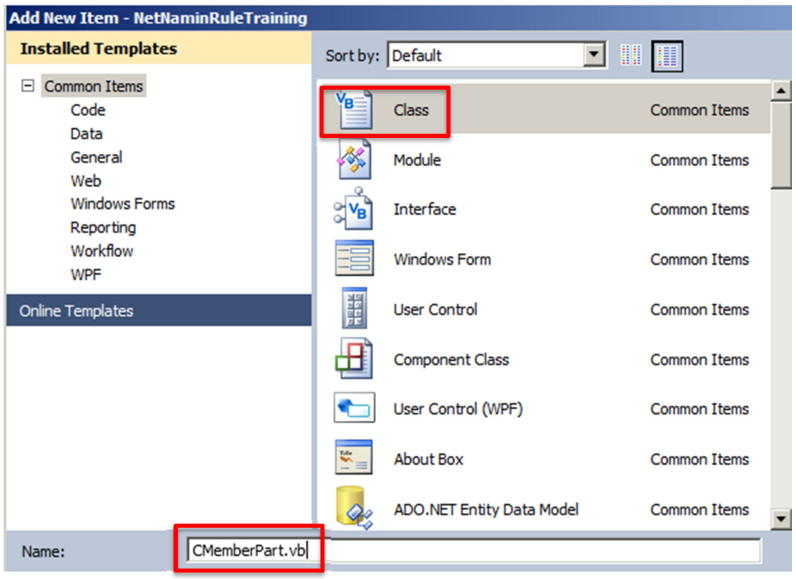
Member Part Object:

Short Description of the Member Category Select List + Section Name + Location + IndexCounter

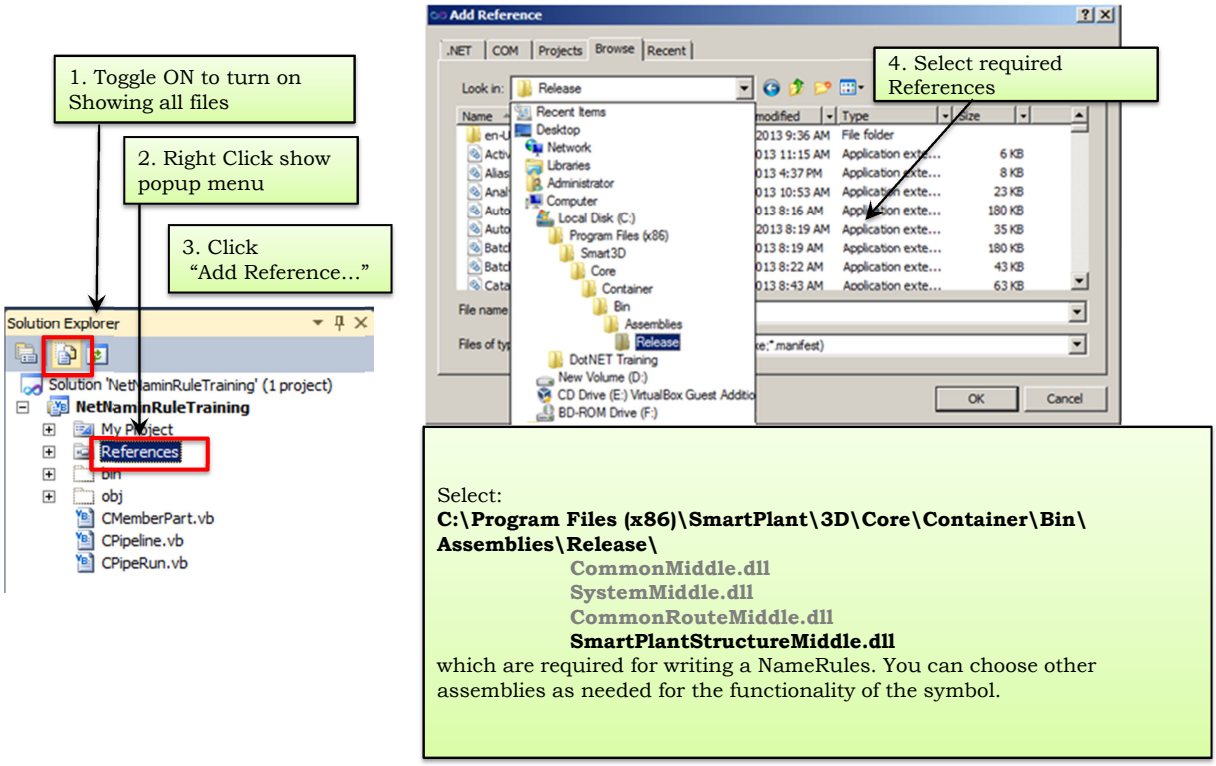


Create a VB.net Class Library Project

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



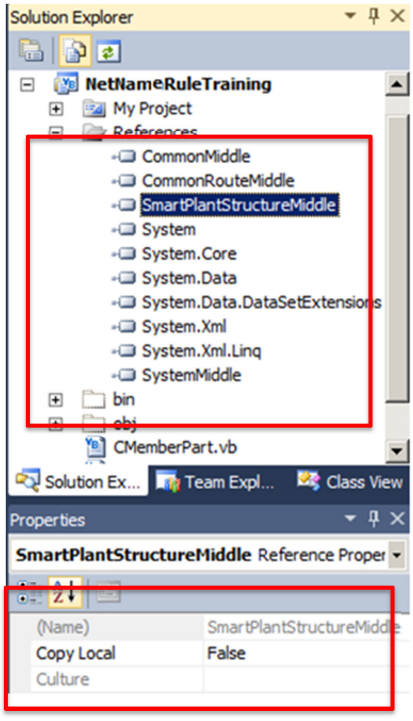
Attach Required References



Select:
C:\Program Files (x86)\SmartPlant\3D\Core\Container\Bin\Assemblies\Release
CommonMiddle.dll
SystemMiddle.dll
CommonRouteMiddle.dll
SmartPlantStructureMiddle.dll
which are required for writing a NameRules. You can choose other assemblies as needed for the functionality of the symbol.

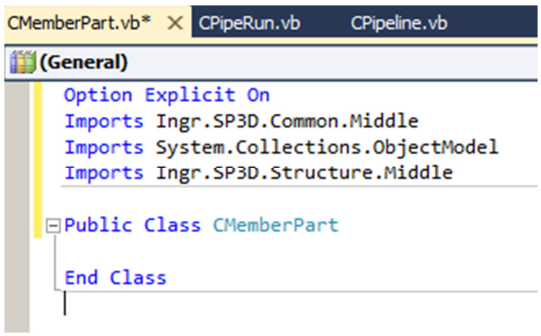
References: Copy Local Setting

- 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

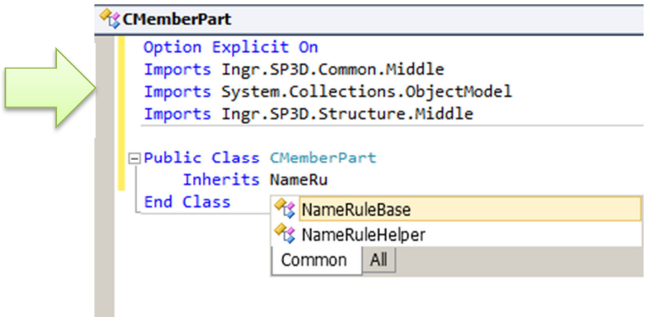
  Dim oBO As BusinessObject

  instead of :

  Dim oBO As Ingr.SP3D.Common.Middle. BusinessObject

Base NameRuleBase Class

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



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

  ComputeName() and **GetNamingParents()** methods.

```
Public Class CMemberPart
   Inherits NameRuleBase


   Public Overrides Sub ComputeName(ByVal oEntity As BusinessObject, _
                ByVal oParents As ReadOnlyCollection(Of BusinessObject), _
                ByVal oActiveEntity As BusinessObject)

   End Sub

   Public Overrides Function GetNamingParents(ByVal oEntity As
                BusinessObject) As Collection(Of BusinessObject)

   End Function
End Class
```

Definition of GetNamingParents() Function

- This method is used to return the related parent object or objects that are used in forming a name

```
Public Overrides Function GetNamingParents(ByVal oEntity As
            BusinessObject) As Collection(Of BusinessObject)



End Function
```

- All the Naming Parents that need to participate in an objects naming are added here to the Collection

Definition of GetNamingParents() Function

```
Public Overrides Function GetNamingParents(ByVal oEntity As Ingr.SP3D.Common.Middle.BusinessObject) _
        As System.Collections.ObjectModel.Collection(Of Ingr.SP3D.Common.Middle.BusinessObject)

    GetNamingParents = Nothing

End Function
```

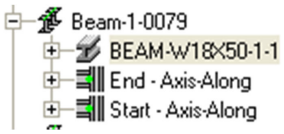
Return an Empty Collection

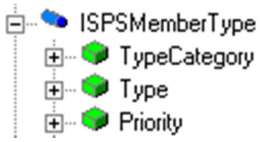
Definition of ComputeName() Sub

```
Public Overrides Sub ComputeName(ByVal oEntity As BusinessObject, _
                                 ByVal oParents As ReadOnlyCollection(Of BusinessObject), _
                                 ByVal oActiveEntity As BusinessObject)

    Dim strMemType As String
    Dim oCLPropValue As PropertyValueCodelist

    oCLPropValue = oEntity.GetPropertyValue("ISPSMemberType", "TypeCategory")
    strMemType = UCase(oCLPropValue.PropertyInfo.CodeListInfo.GetCodelistItem(oCLPropValue.PropValue).Name)
```

Define local variables

Use the Generic Property Access



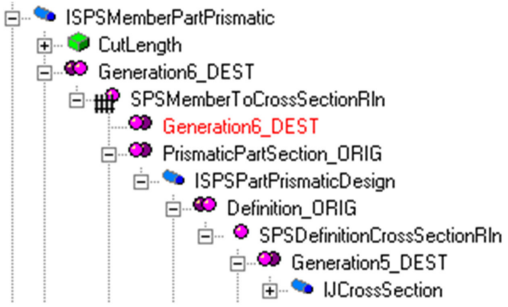


Definition of ComputeName() Sub

```
Dim strSectionName As String
Dim oBO As BusinessObject
Dim oRelationCollection As RelationCollection
Dim oStrPropValue As PropertyValueString

oRelationCollection = oEntity.GetRelationship("SPSMemberToCrossSectionRln", "Generation6_DEST")
oBO = oRelationCollection.TargetObjects.Item(0)

oRelationCollection = oBO.GetRelationship("SPSDefinitionCrossSectionRln", "Definition_ORIG")
oBO = oRelationCollection.TargetObjects.Item(0)
```

Define local variables

Use Generic Relationship Access



Use the Generic Property Access

```
oStrPropValue = oBO.GetPropertyValue("IStructCrossSection", "SectionName")
strSectionName = UCase(oStrPropValue.PropValue)
```

Definition of ComputeName() Sub

```
Dim strBaseName As String
strBaseName = strMemType + "-" + strSectionName

Dim lCount As Long
Dim strLocation As String = vbNullString

GetCountAndLocationID(strBaseName, lCount, strLocation)

oEntity.SetPropertyValue(strBaseName + "-" + strLocation + "-" + Format(lCount, "0000"), "IJNamedItem", "Name")
```

Set BaseName

Get Location and a Counter

Use the Generic Property Access

▲ 1 of 2 ▼ GetCountAndLocationID(**sBaseName As String**, ByRef lCount As Long, ByRef sLocationID As String)
Gets count and location ID from the NameGeneratorService.
sBaseName: *BaseNameString for which next counter is requested.*

Naming Counter Service

Naming counter service - generates a unique counter given a basis string

GetCountAndLocationID(**sBaseName As String**, ByRef lCount As Long, ByRef sLocationID As String)
Gets count and location ID from the NameGeneratorService.
sBaseName: *BaseNameString for which next counter is requested.*

```
Dim strBaseName As String
strBaseName = strMemType + "-" + strSectionName

Dim lCount As Long
Dim strLocation As String = vbNullString
GetCountAndLocationID(strBaseName, lCount, strLocation)
```

GetCountAndLocationID(**sBaseName As String**, iRange As Integer, ByRef lCount As Long, ByRef sLocationID As String)
Gets count and location ID using the range specified from NameGeneratorService.
sBaseName: *BaseNameString for which next counter is requested.*

```
Dim strBaseName As String
strBaseName = strMemType + "-" + strSectionName

Dim lCount As Long
Dim strLocation As String = vbNullString
GetCountAndLocationID(strBaseName, 5, lCount, strLocation)
```

Naming Counter Service Architecture

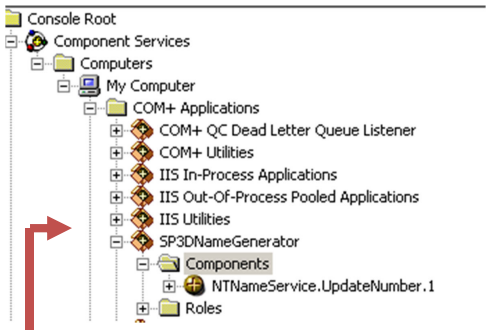
S3D (S3DHost.exe)

Property	Value
Name	U1-S-111
Name Rule	NETPipeline
Type	Pipeline System
Parent System	U1
Description	
Sequence Number	111
Fluid Requirement	Steam
Fluid Type	S, Steam
Correlation Status	Not correlated

GetCountAndLocation()

NameGeneratorService

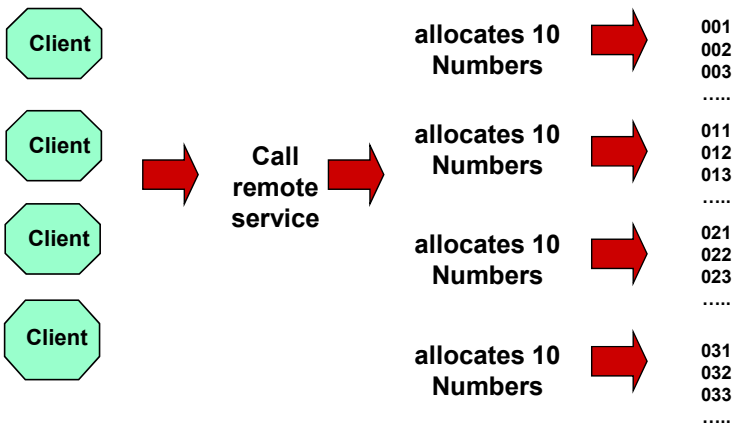
SP3DNameGenerator COM+ Application on the Administrator machine

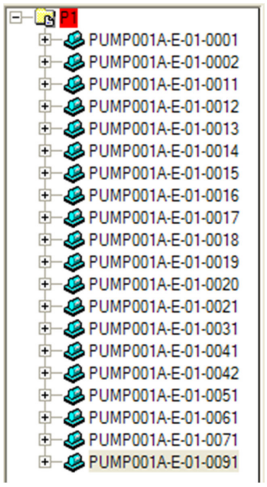


UpdateNumber

Naming Counter Service

Default Range is 10





NamingRules sheet

- Open the GenericNamingRules.xls
- Add a new record in the NamingRules sheet as shown below:

! Back to Index

Head	TypeName	Name	SolverProgID
Start			
a	CSPSMemberPartPrismatic	NET MemberPart	NetNameRuleTraining,Ingr.SP3D.Content.NamingRule.CMemberPart
	CPMPipeRun	NET PipeRun	NetNameRuleTraining,Ingr.SP3D.Content.NamingRule.CPipeRun
	CPPipelineSystem	NET Pipeline	NetNameRuleTraining,Ingr.SP3D.Content.NamingRule.CPipeline

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