S3D .net API

Quick Overview of Smart 3D .net API
From the perspective of writing
Code based Labels / Reports

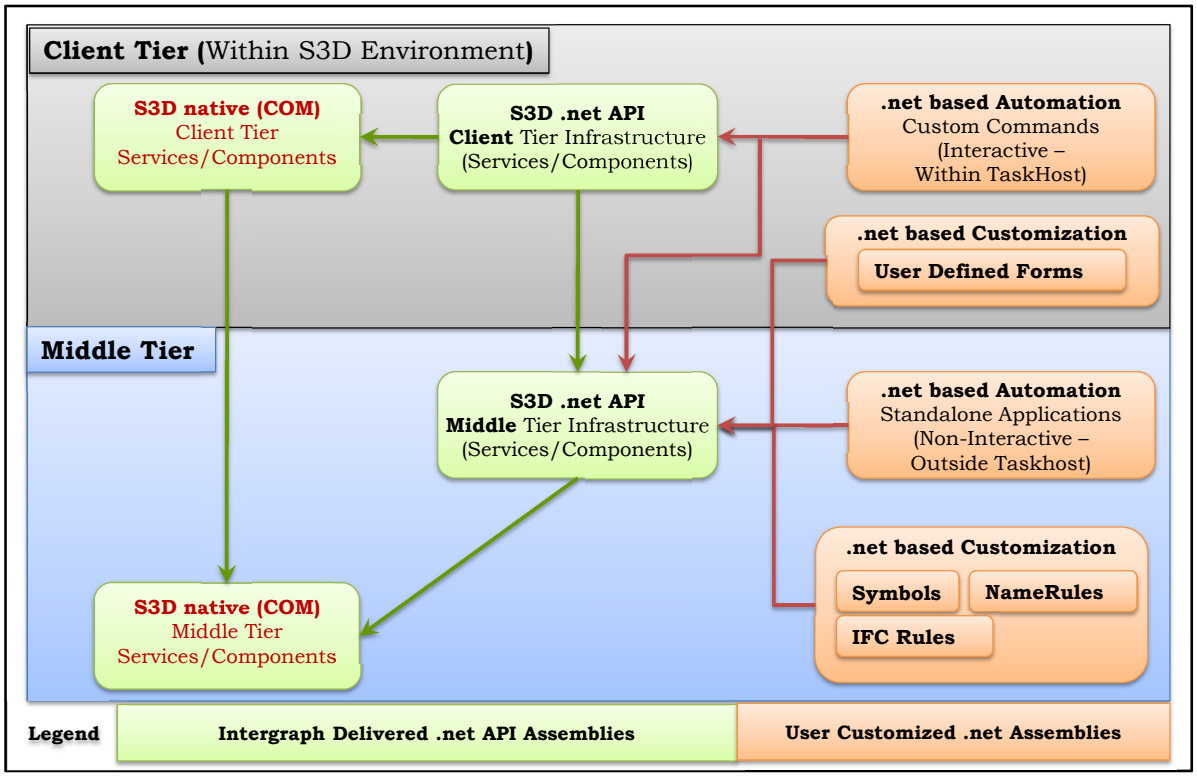
Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Topics of Interest from Reports/Labels perspective

- S3D .net API Architecture.
  - Leverages **.net** methodologies, and provides **.net** based Automation capability.
  - Useful components delivered to you as **.net** Assemblies.
  - Use Visual Studio v2010 - **VB.net** or **C#** or any **.net** language.
- API available for your use in your Code based Labels / Reports
- Middle Tier Services/Components
- Metadata – Classes, BOCs, Interfaces, Properties, Codelists …
- Get Properties, Access Relationships
- Access Discipline specific objects and properties.
- Fully documented with API Help integrated into Visual Studio Environment. F1 help works and shows context sensitive help.

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

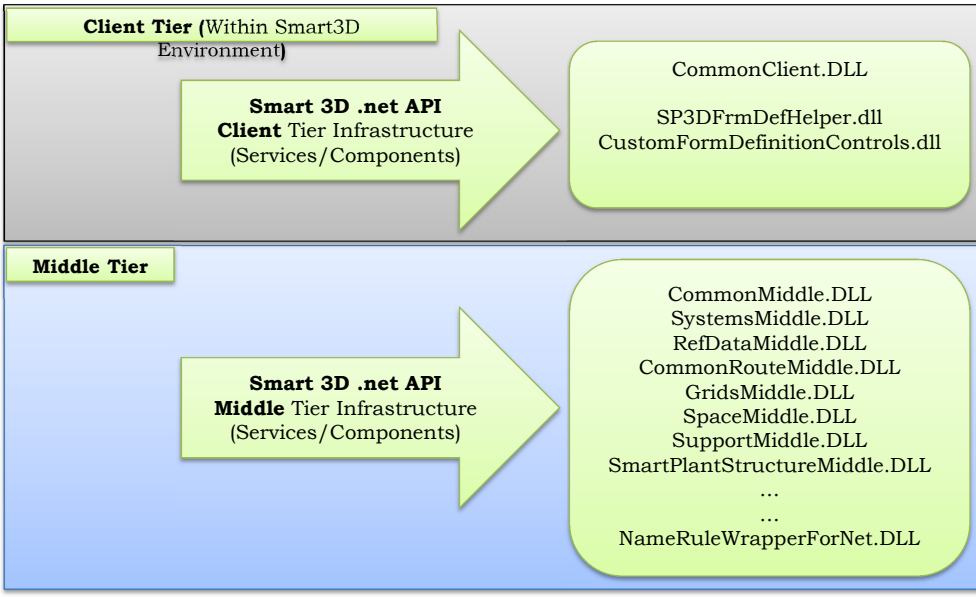
S3D .net API Architecture



Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Smart3D .net API delivered Assemblies

All of the Smart 3D .net API is delivered in assemblies in
$Smart3D\Core\Container\Bin\Assemblies\Release\



Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Labels & Reports using .NET

Overview

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Overview – Smart3D Labels & Reports

COM

- COM Labels and Reports are created using Smart3D User Interface
- Simple for direct properties and relationship/edge based properties
- Limited when complex logic is involved

VB6

- Prior to S3D V2014, Labels and Reports are written using Visual Basic 6.0 utilizing COM API for cases where direct properties based output may not help and complex logic and computation is involved.

.NET

- From S3D V2014 and onwards, .NET based infrastructure is available for writing Labels and Reports using .NET API. User can write code in VB.Net/C#.

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Major Steps involved in .NET based Label

1. Create a COM Label in Smart 3D Catalog Task
2. Code the Label in .NET environment using VS 2010, .NET Framework 4 (VB.NET or C#)
3. Specify ProgID of the .NET based Label in Label RQE file. (Assemblyname, NameSpace.ClassName)

```
<RUN_TIME
 Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetLabel"
 Action=""
 Arg="" />
```

4. Copy the Label DLL in [SharedContent]\CustomSymbols\ folder
5. From Project Management, Run “Update Custom Symbol Configuration”
6. Verify Label

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Major Steps involved in .NET based Report

1. Create a Blank SpreadSheet Report in Smart 3D Drawings & Reports Task
2. Copy the Report to Catalog and give it a name . (This will create Reports files in [Shared Content])
3. Code the Report in .NET environment using VS 2010, .NET Framework 4 (VB.NET or C#)
4. Specify ProgID of the .NET based Report in Report RQE file. (Assemblyname, NameSpace.ClassName)

```
<RUN_TIME
 Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetLabel"
 Action=""
 Arg="" />
```

5. Copy the Report Label DLL in [SharedContent]\CustomSymbols\
6. From Project Management, Run “Update Custom Symbol Configuration”
7. Create the Report in D&R Task, Selecting from Catalog the Report created in Step 2
8. Setup, Configure and Design the Report in D&R Task
9. Again Copy to Catalog for updating final changes made.
10. Verify the Report

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Create a VB.net Class Library Project

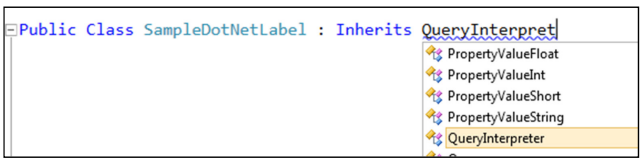
- Use Microsoft Visual Studio 2010.
  - Choose a .net language → VB.net / C#
    - VB.net is preferred, since users were familiar with VB6 language. Most samples are in VB.net. Our course examples will be VB.net.

- Project Type
  - Use a Windows "Class Library" project template
  - Can create the new Label / Report in
    - an existing **Class Library** project you already have with other Labels/Reports you have developed.
      - Choose File → Open Project
    - a new **Class Library** Project.
      - Choose File → New Project

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

QueryInterpreter – The Base class

- In order to be executed by Smart3D Runtime, a Code based Label/Report Class must ***inherit*** from QueryInterpreter Class



- We then ***access the required properties*** and provide the ***override implementation*** for the properties/methods, i.e. the Label characteristics and functionality.
  - **EvaluateOnly, InputObjects** [you can access these properties]
  - **Execute(string,string)** [you have to override this]
  - **ExecuteDelegatedQuery()** [this is available for you to call, in case of Delegated Query reports]

```
Public Class SampleDotNetLabel : Inherits QueryInterpreter

    Public Overrides Function Execute(ByVal action As String, ByVal argument As String) As System.Data.DataTable
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Methods & Properties of QueryInterpreter Class

– You must override the **Execute** Function

```
Public MustOverride Function Execute(ByVal action As String, ByVal argument As String) As System.Data.DataTable
    Member of Ingr.SP3D.Reports.Middle.QueryInterpreter
```

– Execute Function has 2 arguments : **Action** As string, **Argument** As String

– Can be configured from the RQE file.

– Your code can use these values to customize its logic

– Basics of the Implementation

- Define the columns of data (i.e. the header row) which your Label or Report will output (typically, for Label, it will be one column; for report it can be many column)
- Process the Object(s) and produce the data for that columns (i.e. the actual data itself) (typically, for Label, it will be one object; for report it can be many objects)
- Return the data as a datatable

Refer example (VB6 though) Piping MTO

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

S3D .Net Labels & Reports

LAB

–

S3D .NET Label

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

LAB - Overview

Goal : Create a Label which gets largest NPD of all the PipeRuns of a Pipeline

- In this LAB we will learn
  - Writing a Label in VB.net
    - Creating Project, manage its settings
    - Adding required references,
    - Importing required namespaces,
    - Inheriting from QueryInterpreter class,
    - Providing override implementation – Label functionality
  - Deploying the Label
  - Running the Label
  - Debugging the Label

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Major Steps involved in .NET based Label

1. Create a COM Label in Smart 3D Catalog Task
2. Code the Label in .NET environment using VS 2010, .NET Framework 4 (VB.NET or C#)
3. Specify ProgID of the .NET based Label in Label RQE file. (Assemblyname, NameSpace.ClassName)

```
<RUN_TIME Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetLabel"
   Action=""    Arg="" />
```

4. Copy the Label DLL in [SharedContent]\CustomSymbols\ folder
5. From Project Management, Run "Update Custom Symbol Configuration"
6. Verify Label

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Create a VB.net Class Library Project

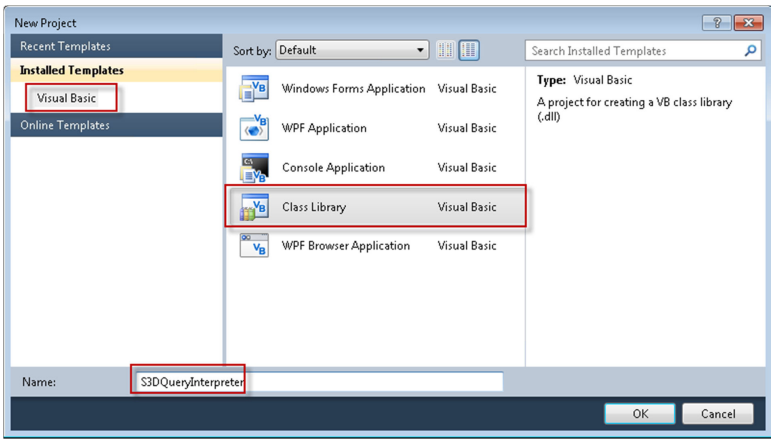
- Use Microsoft Visual Studio 2010.
  - Choose a .net language → VB.net / C#
    - VB.net is preferred. Most samples are in VB.net. Our examples will be VB.net.
- Project Type
  - Use a Windows "Class Library" project template
  - Can create the new Label in
    - an existing **Class Library** project you already have with other commands you have developed.
      - Choose File → Open Project
    - a new **Class Library** Project.
      - Choose File → New Project

  As this is our first Lab for Label we choose New Project.

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Create a VB.net Class Library Project

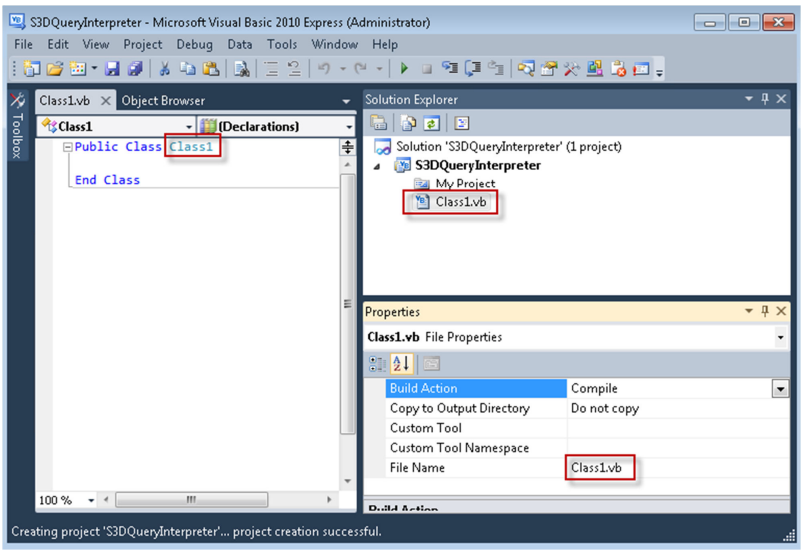
- Open Microsoft Visual Basic 2010 Express from Windows Programs
- Choose **File** > **New Project**
- Use **Visual Basic**
- Pick **Class Library** template
- Specify **Name** of Project (S3DQueryInterpreter)
- Hit **OK**.
- Project gets created.
- …



Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

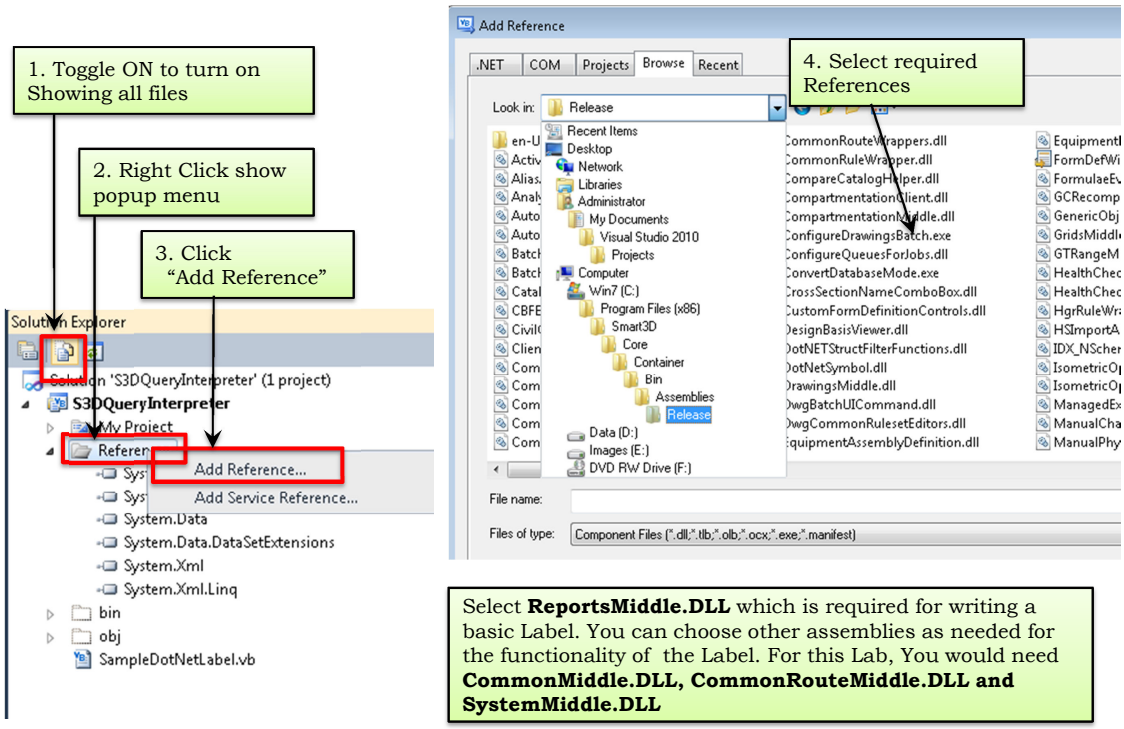
Create a VB.net Class Library Project

- 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.
- Let us change it to "SampleDotNetLabel", for the sample **Label** we will be writing here.



Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Attach Required References

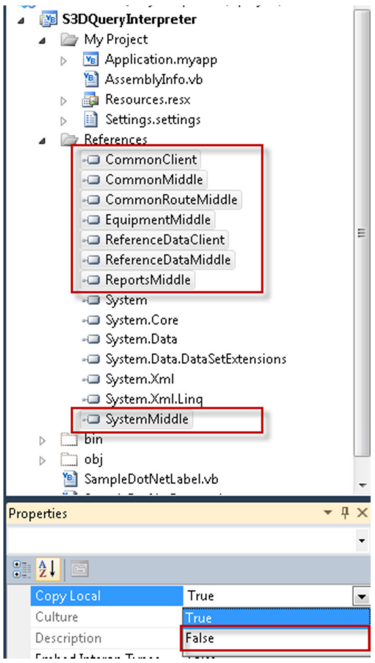


Select **ReportsMiddle.DLL** which is required for writing a basic Label. You can choose other assemblies as needed for the functionality of the Label. For this Lab, You would need **CommonMiddle.DLL, CommonRouteMiddle.DLL and SystemMiddle.DLL**

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

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.



Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

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.

```
Imports System
Imports System.Collections
Imports System.Collections.ObjectModel
Imports Ingr.SP3D.Common.Middle
Imports Ingr.SP3D.Common.Middle.Services
Imports Ingr.SP3D.Reports.Middle
Imports Ingr.SP3D.Route.Middle
Imports Ingr.SP3D.Systems.Middle
```

- This lets you just use

  Dim oPipeline as **Pipeline**

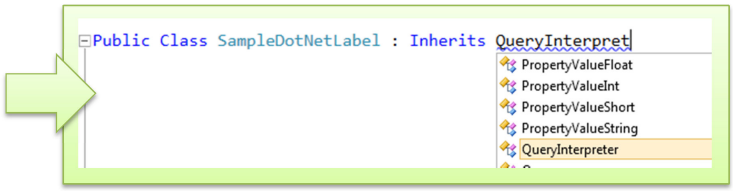
  instead of

  Dim oPipeline as **Ingr.SP3D.Systems.Middle.Pipeline**

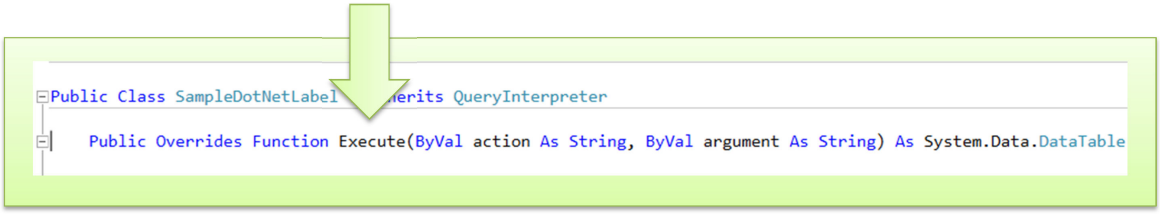
Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Inherit from QueryInterpreter class

- In order to be executed, Label Class must ***inherit*** from ***QueryInterpreter*** Class



- We then provide the ***override implementation*** for the method **Execute**



- At Runtime, Function Execute parameters values are referred from Arg and Action Attributes in Label RQE file.

```
<RUN_TIME
 Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetLabel"
 Action=""
 Arg="" />
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Methods/Properties of Query Interpreter

Complete logic of the label will be implemented under method ***Execute***

```
Public MustOverride Function Execute(ByVal action As String, ByVal argument As String) As System.Data.DataTable
    Member of Ingr.SP3D.Reports.Middle.QueryInterpreter
```

If we observe the function ***Execute***, function is expected to return a DataTable (System.Data.DataTable). Hence, the responsibility here is to return a DataTable with Rows/Columns.

The Objects for which the Label is being invoked are accessible using Property ***InputObjects***

In case of Label, the InputObjects count would be 1.

```
Public Property InputObjects As System.Collections.ObjectModel.Collection(Of Ingr.SP3D.Common.Middle.BusinessObject)
    Member of Ingr.SP3D.Reports.Middle.QueryInterpreter
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Label Requirement

We are implementing Label as below:

For a Pipeline object provided as Input to Label, determine largest NPD from all Runs of the Pipeline and return the NPD

If Runs with different NPD units are found (Inches or Millimeter), the label should still be able to determine the largest.

Label should be able to return the Largest NPD in the units that user has specified in label RQE file (Arg = “in” or Arg= “mm”).

```
<RUN_TIME Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetLabel"
   Action=""    Arg="in" />
```

```
<RUN_TIME Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetLabel"
   Action=""    Arg=“mm" />
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Label Implementation – Overview

We get all Runs of Pipeline, get MaxNPD, and the Run of that MaxNPD

We take care of checking if NPD found in Millimeters is greater than the Max NPD, If so then make this NPD as Max NPD

Once all Runs have been iterated & Max NPD is found, we check if user has specified any value for “Arg” in Label RQE file for an desirable unit to be returned for NPD

If nothing is specified, we return NPD in Inches

Once Max NPD in desirable unit is determined , we create a new row in DataTable and Set Max NPD value for Column name “NPD”. Then we add the row to the DataTable

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Label Implementation

- Our SampleDotNetLabel Class will inherit from QueryInterpreter

```
Public Class SampleDotNetLabel : Inherits QueryInterpreter
```

- Declare a global variable m_DataTable in which we will be adding Rows & Columns

```
Dim m_DataTable As DataTable
```

- Override Execute Function (type **Overrides** and pick **Execute** from list shown)

```
Public Overrides Function Execute(ByVal action As String, ByVal argument As
String) As System.Data.DataTable
```

- Declare a List of Type Column (QueryInterpreter.Column)
- Create New Column for NPD and Add it to the List
- Then InitializeDateTable with the List of Column (In our case it is just one Column – later we extend it to handle Delegated Query – but we don’t need any changes here for that extension)

```
Dim FieldColumnList As New List(Of Column)

Dim recFieldNPD As New Column("NPD", GetType(System.String))
FieldColumnList.Add(recFieldNPD)

m_DataTable = InitializeDataTable(FieldColumnList)
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Label Implementation

- Query Interpreter has a Boolean Property named EvaluateOnly

```
Public Property EvaluateOnly As Boolean
    Member of Ingr.SP3D.Reports.Middle.QueryInterpreter
```

- In Reports for example, the design of report requires only column names to be displayed at the design time

So during Design stage EvaluateOnly property is set to True by Report Framework before calling your code.

Hence, if we find **EvaluateOnly = True**, we just return DataTable with Columns definitions only (i.e. no data)

```
If (EvaluateOnly) Then Return m_DataTable
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Label Implementation

```
For Each oBO In InputObjects
  Dim dMaxNPD As Double = -1, dNPDInMM As Double, strMaxNPD As String = ""
  Dim oUOMMgr As UOMManager = MiddleServiceProvider.UOMMgr

  'Given a Pipeline as Input, Return Largest NPD,
  If (TypeOf oBO Is Pipeline) Then
    Dim oPipeline As Pipeline = oBO

    'Pipeline might have 0 runs, if there are no runs found, return nothing
    If (oPipeline.SystemChildren.Count < 1) Then Return Nothing

    Dim oNPD As NominalDiameter, oPipeRun As PipeRun, oMaxNPDPipeRun As PipeRun
    For Each oSysChild As ISystemChild In oPipeline.SystemChildren 'iterating all SystemChildren of Pipeline
      If (TypeOf oSysChild Is PipeRun) Then 'Process PipeRuns
        oPipeRun = oSysChild
        oNPD = oPipeRun.NPD

        'Pipeline can have Runs of different NPD units (in or mm),
        'we first parse NPD value string (Size + Unit) as UnitType.NPD,
        'which returns NPDUnit value in DBU (Database Units) – meters, which we convert to Distance (mm)
        dNPDInMM = oUOMMgr.ConvertDBUtoUnit(UnitType.Distance, _
            oUOMMgr.ParseUnit(UnitType.Npd, oNPD.Size.ToString & oNPD.Units),
            UnitName.DISTANCE_MILLIMETER)
```

We process InputObjects (will be just 1 for Label situation, can be 1 or more for Reports situation)

So, theoretically, we don't need For … Next for **Label** Situation. We still keep it here, to make this code usable in a **Report**.

Ensure what we got is a Pipeline

We Exit if Pipeline has no SystemChildren

Process Children, of PipeRun type (supports can be under pipeline and hence this check). We Get NPD of Each PipeRun

We get NPD value in **mm** for each piperun.

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Label Implementation

We use MiddleServiceProvider.UOMMgr for Converting DBU (DatabaseUnit) to a Unit, Parsing values

Public Function **ParseUnit**(ByVal *iUnitType* As Ingr.SP3D.Common.Middle.Services.UnitType, ByVal *sParseString* As String) As Double
Member of Ingr.SP3D.Common.Middle.Services.UOMManager

Summary:
Parses a text expression in the context of the specified unit type (iUnitType) and returns the value in database units.

Parameters:
iUnitType: Unit type used to interpret the string.
sParseString: String to parse.

for Parsing values

Public Function **ConvertDBUtoUnit**(ByVal *iUnitType* As Ingr.SP3D.Common.Middle.Services.UnitType, ByVal *dDBU* As Double, ByVal *iUnitName* As Ingr.SP3D.Common.Middle.Services.UnitName) As Double
Member of Ingr.SP3D.Common.Middle.Services.UOMManager

Summary:
Converts a number expressed in internal database units into the specified unit of measure, overriding the default unit for this type.

Parameters:
iUnitType: Target unit type (i.e., DISTANCE, etc.).
dDBU: The DBU value to be converted.
iUnitName: Target unit name (i.e., FEET) which must be of type iUnitType.

for Converting DBU (DatabaseUnit) to a Unit

```
            If (dNPDInMM > dMaxNPD) Then
               dMaxNPD = dNPDInMM
               oMaxNPDPipeRun = oPipeRun
            End If
         End If
      Next 'For Each oSysChild of Pipeline
```

We converted all NPD values to mm, so that we can compare correctly and find maxNPD.
We also remember which Run has that MaxNPD.

Label Implementation

```
      'we provided facility (Argument in RQE) to return Max NPD in a specific unit that a user may want.
      ' If Argument = "in" or "" – we return in inches,  If Argument = "mm"  - we return in mm
      If ((argument.ToLower = "") Or (argument.ToLower = "in")) Then _
         strMaxNPD = oUOMMgr.ConvertDBUtoUnit(UnitType.Npd, (dMaxNPD * 0.001),
               UnitName.NPD_INCH).ToString & " in“

      If (argument.ToLower = "mm") Then strMaxNPD = oUOMMgr.ConvertDBUtoUnit(UnitType.Npd, (dMaxNPD
            * 0.001), UnitName.NPD_MILLIMETER) & " mm"

      'data row to hold the data for current object
      Dim currentRow As DataRow = m_DataTable.NewRow()
      currentRow.SetField("NPD", strMaxNPD)
      'Add the new rows to the master data table
      m_DataTable.Rows.Add(currentRow)
   End If
End If
Next

'we will insert code here for Extending this lab functionality to handle Delegated Query.

Return m_DataTable
```

We must return data (NPD String) as per the need – in / mm setting specified by Argument

We create a new Row in our DataTable, and set column values

Later on…

Finally, we return the DataTable we created.

Deploying & Running Label

- At this time, our Label is ready to run or debug.
- Build and Save the Label Assembly DLL to [SharedContent]\CustomS ymbols\, and Run Update Custom Symbols Configuration command from Project Management
- Create a COM Label from S3D Catalog Task.

1. Edit Label RQE file in XML Editor and Add ProgID for the .NET Label we built in this Lab

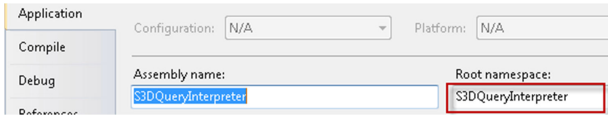
```
<RUN_TIME
 Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetLabel"
 Action=""
 Arg="" />
```

ProgID syntax should be *AssemblyName,NameSpace.Class*

the **Assembly (DLL) Name** but without extension,

the **NameSpace** of Label Class

if you have not explicitly created your class in a different namespace, it goes to project's Root NameSpace (see it in Application Tab of Project Properties).



the **ClassName**

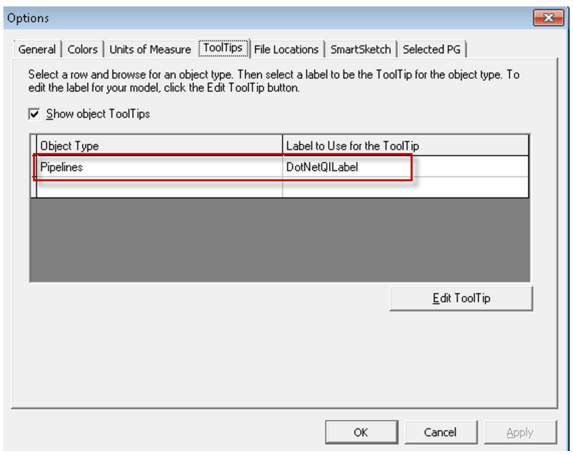
2. Edit Label RFM file in XML Editor and Add DATA Column "NPD". This "NPD" name should match the Column that we returned with DataTable

```
<DATA
 column="NPD"
 ToParse="no"
 visible="yes" />
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Deploying & Running Labels

- Once Label is configured and Assembly is deployed, Test the Label
- You can test the label in various ways. Easiest option is to map the label as ToolTip
  - In S3D Session, Go to Tools>Options and then under ToolTips Tab, Add Pipelines as ObjectTypes and Map the Label against it.



  - From Locate Filter, Select a Pipeline, hover mouse to so the pipeline is selected and check if label resolves to the expected Largest NPD

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Debugging the Label

Approach #1: Attach to an already running S3DHost process.

Approach #2: Specify Executable to start which invokes the command class. (details next)

Note : Developer Studio **Express Edition** limits its UI from doing both of these approaches. However, Approach #2 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>[InstallDir]\3D\Core\Container\Bin\S3DHost.exe </StartProgram>
  <StartArguments>FullPathToSessionFile</StartArguments>
</PropertyGroup>
</Project>
```

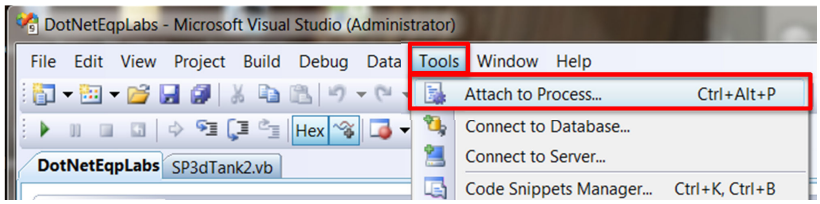

Debugging the Symbol Code

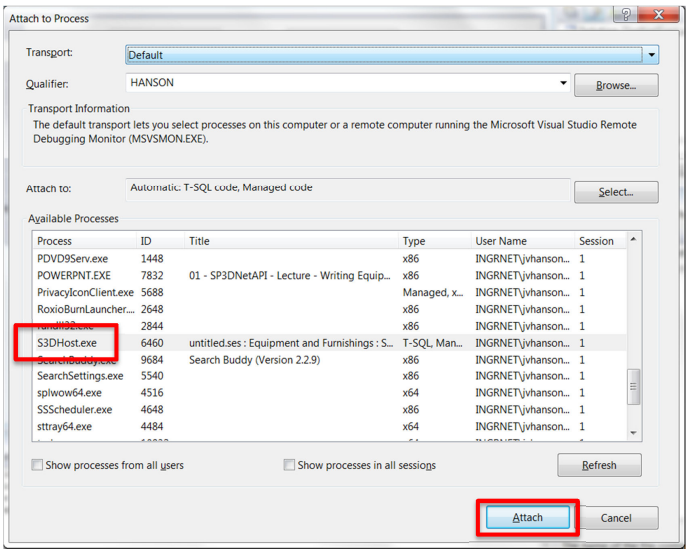
Approach #1: Attach to an already running S3DHost process.

Inside Visual Studio, choose **Tools > Attach To Process**,

Select **S3DHost.exe** process and press **Attach**.

Inside **Smart 3D**, invoke the commands as described earlier to test.





Debugging the Symbol Code…

Approach #2: Start a new Smart3D process from within the debugger.

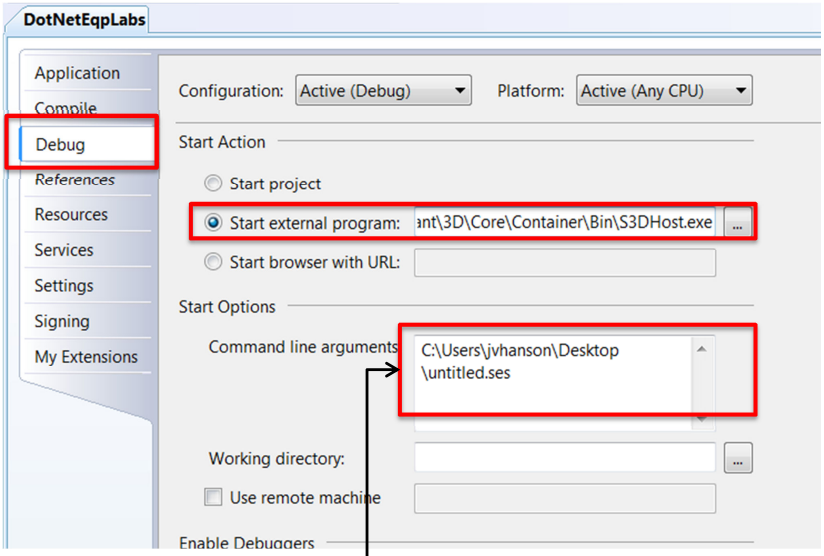
Note: you need to add to your system Path env variable, the value of Path setting from Registry node HKLM\Software\[Wow6432Node\]Microsoft\Windows\CurrentVersion\App Paths\S3DHost.exe)

Inside Visual Studio, **Project > Project Properties**

Choose the following settings and **Save**.

Now select **Debug>Run menu item in Visual Studio.**

Inside **Smart3D**, invoke the commands as described earlier to test.



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.

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

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.
- Installation : Your Setup Installer must only include your custom DLLs. Do not include (i.e. re-deliver) ANY of the Smart 3D delivered API DLLs because the Smart 3D Product delivery (ServicePack/HotFixes) is responsible for those.

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Further improvements to the Label

Here, we enhance the Label to get Properties specified through delegated query in RQE file. **QueryInterpreter's ExecuteDelegatedQuery** method executes COM queries specified in RQE and return those properties (as a DataTable). We then merge them into our DataTable, which we return.

Public Function **ExecuteDelegatedQuery**(ByVal *objectToQuery* As Ingr.SP3D.Common.Middle.BusinessObject) As System.Data.DataTable
Member of Ingr.SP3D.Reports.Middle.QueryInterpreter

```
'Execute the Delegated Query.
If (oMaxNPDPipeRun IsNot Nothing) Then
   Dim DelegatedDataTable As System.Data.DataTable = ExecuteDelegatedQuery(oMaxNPDPipeRun)
   If (DelegatedDataTable IsNot Nothing) Then
      If (DelegatedDataTable.Rows.Count >= 1) Then
         Dim row As DataRow = DelegatedDataTable.Rows.Item(0)
         'merge the column value pair to data table, excluding the oid column
         For Each col As DataColumn In row.Table.Columns
            If (col.ColumnName.ToLower() <> "oid") Then
               currentRow.SetField(col.ColumnName, row(col.ColumnName))
            End If
         Next
      End If
   End If
   'Add the new rows to the master data table
   m_DataTable.Rows.Add(currentRow)
End If

Return m_DataTable
```

Note that, we have control over which object to send as input for the Delegated Query execution

Implement this code in your Label

This code is
- Executing the delegated query with the Run of Max NPD
- Going through the rows returned by Delegated Query
- Merging data from Delegated Query into DataTable we return

This line of code exists already from our current implementation

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Properties from Delegated Query in RQE file

Add a query in Label RQE

```
....
....
<RUN_TIME Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetLabel" Action="" Arg="mm"/>

<RETURNED_PROPERTIES>
  <RETURNED_PROPERTY Name="Name" SQLType="BStr">
    <PATHS>
      <PATH SourceType="*" SourceBOC="" DestinationInterface="IJNamedItem"
            DestinationProperty="Name" DestinationBOC="" Concatenate="No" PathSeparator="\"/>
    </PATHS>
  </RETURNED_PROPERTY>
</RETURNED_PROPERTIES>

</REPORT_QUERY>
```

Make relevant changes to Label definition interactively to add the new data (Name) to display
[this interactive action modifies the RFM file contents].

You can now test the enhanced Label output using Tool Tip.

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

S3D .Net Labels & Reports

LAB

–

S3D .NET Report

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

LAB - Overview

- The goal of this lab is to make a Report for each Equipment, its Name, a list of its nozzles, their NPD and their position. We also extend the report using a Delegated Query to get Connected Run name for each Nozzle.

- In this LAB we will learn
  - Writing a Report in VB.net
    - Adding a new class to existing Project
    - Adding required references,
    - Importing required namespaces,
    - Inheriting from QueryInterpreter class,
    - Providing override implementation – Report functionality
  - Deploying the Report
  - Running the Report
  - Debugging the Report

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Major Steps involved in .NET based Report

1. Create a Blank SpreadSheet Report in Smart 3D Drawings & Reports Task
2. Copy the Report to Catalog and give it a name . (This will create Reports files in [Shared Content])
3. Code the Report in .NET environment using Visual Studio, .NET Framework 4 (VB.NET or C#)
4. Specify ProgID of the .NET based Report in Report RQE file. (Assemblyname, NameSpace.ClassName)

```
<RUN_TIME Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetReport"
   Action=""    Arg="" />
```

5. Copy the Report Label DLL in [SharedContent]\CustomSymbols\
6. From Project Management, Run “Update Custom Symbol Configuration”
7. Create the Report in D&R Task, Selecting from Catalog the Report created in Step 2
8. Setup, Configure and Design the Report in D&R Task
9. Again Copy to Catalog for updating final changes made.
10. Verify the Report

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Open Project, add new Class

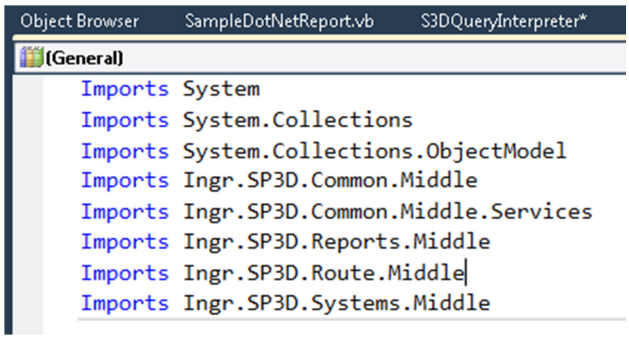
As this is our 2nd Lab for Reports we choose existing Project which we created for the 1st Lab.

- Start Visual Basic
- Choose **File** > **Open Project**
- Choose the Project we previously created
- Choose Project Menu, Add New Class named “SampleDotNetReport”, for the sample **Report** we will be writing here.
- Add the new required References (add EquipmentMiddle.DLL)
- Ensure Copy Local for all Smart3D references is set to False
- Add required name spaces to the new Class at the top

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

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 oPipeline as **Pipeline**

  instead of

  Dim oPipeline as **Ingr.SP3D.Systems.Middle.Pipeline**

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Inherit from QueryInterpreter class

- In order to be executed, our Class SampleDotNetReport must ***inherit*** from ***QueryInterpreter*** Class
- Remember to use report instead of Label in this pictures

```
Public Class SampleDotNetReport : Inherits QueryInterpreter
```

- We then provide the ***override implementation*** for the method ***Execute***

```
Public Class SampleDotNetReport : Inherits QueryInterpreter

    Public Overrides Function Execute(ByVal action As String, ByVal argument As String) As
System.Data.DataTable
```

- At Runtime, Function Execute parameters values are referred from Arg and Action Attributes in Report RQE file.

```
<RUN_TIME Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetReport"
   Action=""    Arg="" />
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Methods/Properties of Query Interpreter

Complete logic of the label will be implemented under method ***Execute***

```
Public MustOverride Function Execute(ByVal action As String, ByVal argument As String) As System.Data.DataTable
    Member of Ingr.SP3D.Reports.Middle.QueryInterpreter
```

If we observe the function ***Execute***, function is expecting a DataTable (System.Data.DataTable) to be returned. Hence, the responsibility here is to return a DataTable with Rows/Columns.

The input objects considered by the system for generating the Report can be accessed using Property ***InputObjects***

In case of Report, the InputObjects count could be one or more.

```
Public Property InputObjects As System.Collections.ObjectModel.Collection(Of Ingr.SP3D.Common.Middle.BusinessObject)
    Member of Ingr.SP3D.Reports.Middle.QueryInterpreter
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Report Implementation

Our SampleDotNetReport Class will inherit from QueryInterpreter

```
Public Class SampleDotNetReport : Inherits QueryInterpreter
```

Declare a class variable m_DataTable in which we will be adding Rows & Columns

```
Dim m_DataTable As DataTable
```

Override Execute Function

```
Public Overrides Function Execute(ByVal action As String, ByVal argument As String) As
System.Data.DataTable
```

Declare and initialize a new List of Type Column (QueryInterpreter.Column)

Create New Columns for EqpName, NozzleName, NPD, CoordX,Y,Z and add them to the List

Then InitializeDateTable with the List of Columns

```
'Create a record DataColumn list that will contain all the columns(DataColumns)
Dim DataColumnList As New List(Of Column)
DataColumnList.Add(New Column("EqpName", GetType(System.String)))
DataColumnList.Add(New Column("NozzleName", GetType(System.String)))
DataColumnList.Add(New Column("NozzleNPD", GetType(System.String)))
DataColumnList.Add(New Column("CoordX", GetType(System.Double)))
DataColumnList.Add(New Column("CoordY", GetType(System.Double)))
DataColumnList.Add(New Column("CoordZ", GetType(System.Double)))

'create and initialize the Data Table with the Columns
m_DataTable = InitializeDataTable(DataColumnList)
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Report Implementation

- Query Interpreter base class has a Boolean Property named EvaluateOnly

```
Public Property EvaluateOnly As Boolean
    Member of Ingr.SP3D.Reports.Middle.QueryInterpreter
```

- In Reports for example, the design of report requires only column names to be displayed at the design time

So during Design stage EvaluateOnly property is set to True by Report Framework before it calls our Execute function.

Hence, we add a condition that if EvaluateOnly = True, just return DataTable with Columns but no data

```
If (EvaluateOnly) Then Return m_DataTable
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Report Implementation

Now Iterate each object from InputObjects

```
For Each oBO In InputObjects
```

(Since this is a Report so there could multiple objects in InputObjects)

Determine if the object is an Equipment

```
'Check if oBO Is Equipment, if yes, get its Piping Ports
If (TypeOf oBO Is Equipment) Then

  Dim oEqp As Equipment = oBO
  Dim oPorts As ReadOnlyCollection(Of IPort) = oEqp.GetPorts(PortType.Piping)
```

Get the Data we are interested to report for each Piping Nozzle

```
'Get Type Required Data for Each Port
For Each oPipePort As IPipePort In oPorts

Dim sNozName As String = oPipePort.ToString() 'Get Name of Nozzle
Dim oNozNPD As NominalDiameter = oPipePort.NPD 'Get Nozzle NPD
'Location of Nozzle by Typecasting to IDistributionPort
Dim oDistribPort As IDistributionPort = CType(oPipePort, IDistributionPort)
Dim dX = oDistribPort.Location.X
Dim dY = oDistribPort.Location.Y
Dim dZ = oDistribPort.Location.Z
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Report Implementation

Add data to the datatable

```
    'data row to hold the data for current object
    Dim currentRow As DataRow = m_DataTable.NewRow()
    currentRow.SetField("EqpName", oEqp.ToString)
    currentRow.SetField("NozzleName", sNozName)
    currentRow.SetField("NozzleNPD", oNozNPD.Size.ToString & " " & oNozNPD.Units)
    currentRow.SetField("CoordX", dX)
    currentRow.SetField("CoordY", dY)
    currentRow.SetField("CoordZ", dZ)

    'Add the new rows to the master data table
    m_DataTable.Rows.Add(currentRow)
  Next ' For each oPipePort
 End If
Next ' next object in InputObjects
```

Finally, we return the datatable

```
Return m_DataTable
```

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Further improvements to the Report

Here, we enhance the Label to get Properties specified through delegated query in RQE file. **QueryInterpreter's ExecuteDelegatedQuery** method executes COM queries specified in RQE and return those properties (as a DataTable). We then merge them into our DataTable, which we return.

Public Function **ExecuteDelegatedQuery**(ByVal *objectToQuery* As Ingr.SP3D.Common.Middle.BusinessObject) As System.Data.DataTable
Member of Ingr.SP3D.Reports.Middle.QueryInterpreter

```
currentRow.SetField("CoordZ", dZ)

'Execute the Delegated Query.
Dim DelegatedDataTable As System.Data.DataTable = ExecuteDelegatedQuery(CType(oPipePort, BusinessObject))
'(user added columns and returned properties from the Delegated Query)
If (DelegatedDataTable IsNot Nothing) Then
   If (DelegatedDataTable.Rows.Count >= 1) Then
      Dim row As DataRow = DelegatedDataTable.Rows.Item(0)
      'merge the column value pair to data table, excluding the oid column
      For Each col As DataColumn In row.Table.Columns
         If (col.ColumnName.ToLower() <> "oid") Then
            currentRow.SetField(col.ColumnName, row(col.ColumnName))
         End If
      Next
   End If
End If

'Add the new rows to the master data table
m_DataTable.Rows.Add(currentRow)
```

Implement this code in your Report.

In our example, we add a delegated query to get Connected run Name for a given Nozzle.

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.

Properties from Delegated Query in RQE file

....
....
<RUN_TIME Progid="S3DQueryInterpreter,S3DQueryInterpreter.SampleDotNetReport" Action="" Arg=""/>

Add a query in Report RQE

```
<RETURNED_PROPERTIES>
 <RETURNED_PROPERTY Name="Connected Run" SQLType="BStr">
   <PATHS>
    <PATH SourceType="IJDistribPort" SourceBOC="" DestinationInterface="IJNamedItem" DestinationProperty="Name" DestinationBOC=""
          Concatenate="No" PathSeparator="\">
     <STROKES>
      <STROKE Interface="IJLogicalDistPort" RelationCollection="LogDistConn" Recursive="No" Filter="First" IsVirtualRelationship="No"
          ExitValue="1"/>
      <STROKE Interface="IJDesignParent" RelationCollection="DistribConnection" Recursive="No" Filter="First" IsVirtualRelationship="No"
          ExitValue="1"/>
      <STROKE Interface="IJDistribConnection" RelationCollection="Parts" Recursive="No" Filter="First" IsVirtualRelationship="No"
          ExitValue="1"/>
      <STROKE Interface="IJRtePathGenPart" RelationCollection="Owner" Recursive="No" Filter="First" IsVirtualRelationship="No" ExitValue="1"/>
     </STROKES>
    </PATH>
   </PATHS>
  </RETURNED_PROPERTY>
</RETURNED_PROPERTIES>
```

This delegated query gets Connected Run Name for a given Piping Nozzle.

</REPORT_QUERY>

Make relevant changes to Report definition interactively to add the new data (Connected Run Name) to report [this interactive action modifies the respective files involved in the report template].
You can now test the enhanced report output.

Smart 3D .net API Training – © 2008. Intergraph Corporation. All Rights Reserved.