Smart P&ID 2019

Automation Programming with C# Labs

Version 2019 **July 2020** **DPID2-TP-100030A**

Table of Contents

Preface

This document is a user's guide for Smart P&ID 2019 automation programming with C# labs.

Send documentation comments or suggestions to PPMdoc@hexagon.com

General Instructions For Labs:

1. You will need to reference following dlls for your lab programs, which are located at "…\Program Files (x86)\SmartPlant\P&ID Workstation\bin".

(1) Intergraph Smart P&ID Logical Model Automation – LLAMA.DLL
(2) Intergraph Smart P&ID Placement Automation – Plaice.DLL
(3) Intergraph Smart P&ID Automation – PIDAuto.DLL
(4) Intergraph SmartP PID Foreign Calculation Adapter - LMForeignCalc.DLL

2. Some constants need to be defined to hold SP_ID of some items, of course you need to place these items first. I provide an assembly for you to place into a drawing at the beginning of this course. Examples are:

```
String CONST_SPID_ModelItem As String = "C76EF274525A4345A6ACE1D179362899"

String CONST_SPID_ItemNote As String = "9A3B02C271754A8BB46DC4D02F9F0954"

String CONST_SPID_OPC As String = "A8EC523227A4F3AB480E9AB39205BCC"

String CONST_SPID_Vessel As String = "C76EF274525A4345A6ACE1D179362899"

String CONST_SPID_PipeRun As String = "8B283FA8472F4E3BABB6AF573DF161F4"

String CONST_SPID_PipingComp As String = "59D6251324574734B9883C8E89E57B4E"

String CONST_SPID_OfflineInstrument As String = "7EAB72658BA04FD8BD67CFEB4D96DD37"

String CONST_SPID_InlineInstrument As String = "BC21A415E803496EBDA87129F5F5F540"

String CONST_SPID_LabelPersist As String = "B9E88D821E8145269E5B398B858555A8"

String CONST_SPID_Drawing = "A8A04604CF314E33A58CBF8B7ED585E6";
```

1. INITIALIZE LMADATASOURCE

a) Purpose

To initialize LMADataSource with different methods and access some properties of it.

b) Problem Statement

Write a standalone application to initialize the LMADataSource with New LMADatasource and PIDDatasource, then access some properties of it, such as ProjectNumber, SiteNote, etc.

c) Solution

1. Using `new LMADataource` to initialize `LMADataSource`.
2. Use `Console.WriteLine` method to print out the required properties.

◊ **Example Code**

```
LMADataSource datasource = new LMADataSource();

Console.WriteLine("DataSource.SiteNode = " + datasource.SiteNode);
Console.WriteLine("DataSource.ProjectNumber = " + datasource.ProjectNumber);
Console.WriteLine("DataSource.IsSatellite = " + datasource.IsSatellite());

datasource = null;
```

2. Change Site and Plant

a) Purpose

To change active site and active plant within LLAMA program.

b) Problem Statement

Place a Vessel into active drawing, then close the drawing. Then switch the smartplant to another site and another plant, then create a new drawing and place another Vessel in it.

c) Solution

Change the site and plant within your program to get access to that Vessel.

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

Console.WriteLine("DataSource.SiteNode = " + datasource.SiteNode);
Console.WriteLine("DataSource.ProjectNumber = " + datasource.ProjectNumber);

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);
Console.WriteLine("Vessel ItemTag  ==>  " + objVessel.Attributes["ItemTag"].Value);

datasource.SiteNode = "\\\\SPID-TRN\\Hexagon_Site\\Ini File\\SmartPlantV4.ini";
datasource.ProjectNumber = "TSPL1002!TSPL1002";

Console.WriteLine("DataSource.SiteNode = " + datasource.SiteNode);
Console.WriteLine("DataSource.ProjectNumber = " + datasource.ProjectNumber);

//Get the CONST_SPID_Vessel of a vessel from TSPL1002: 321C066E734F42CEA60844E22C0907DA
objVessel = datasource.GetVessel( “321C066E734F42CEA60844E22C0907DA” );
Console.WriteLine("Vessel ItemTag  ==>  " + objVessel.Attributes["ItemTag"].Value);

objVessel = null;
datasource = null;
```

3. ACCESS ALL ITEMTYPES

a) Purpose

To access all ItemTypes within a Plant

b) Problem Statement

Access to an LMADatasource, then print out all Item Types within that LMADatasource.
There are total 48 Item Types in Smart P&ID v2019, which includes:

AreaBreak
Drawing
DrawingProject
DrawingVersion
DuctRun
DuctingComp
DuctingPoint
EquipComponent
Equipment
EquipmentOther
Exchanger
GlobalDrawing
History
HydraulicCircuit
InstrLoop
Instrument
ItemNote
Label
LabelPersist
Mechanical
ModelItem
ModelItemClaim
ModelItemClaimOffline
ModelItemClaimRep
ModelItemLookup
Note
Nozzle
OPC
Package
PipeRun
Pipeline
PipingComp
PipingPoint
PlantItem
PlantItemGroup
PlantItemGroupOther
Representation
RepresentationLookup
Revision
Room
RoomComponent
SafetyClass
SignalPoint
SignalRun
System
Task
TaskItemProperty
Vessel

c) Solution

1. Get object LMADatasource
2. Loop through LMADatasource.ItemTypes

◊ **Example code**

Dim datasource As LMADataSource
Dim i As Integer

```
LMADataSource datasource = new LMADataSource();

Console.WriteLine("Total ItemTypes = " + datasource.TypeNames.Count());

for (int i = 1; i <= datasource.TypeNames.Count(); i++)
    Console.WriteLine(datasource.TypeNames.Item(i));

datasource = null;
```

4. IDENTIFY AN ITEM IN THE DATABASE USING SP_ID AND READ ITS PROPERTIES

Purpose

To access a vessel using SP_ID values and read its properties

b) Problem Statement

Place a vessel. Write a standalone application to retrieve the following properties of the vessel:
SP_ID, EquipmentSubClass, EquipmentType, aabbcc_code, Class, Item TypeName, volumeRating, and volumeRating in SI units.

c) Solution

1. Define a LMVessel object and get the object using `LMADataSource.GetVessle` method.
2. Use `Console.WriteLine` method to out put the required properties.

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);

Console.WriteLine("Vessel.Id = " + objVessel.Id);
Console.WriteLine("Vessel EquipmentSubclass  ==>
        "+ objVessel.Attributes["EquipmentSubclass"].Value);
Console.WriteLine("Vessel EquipmentType  ==>
        "+ objVessel.Attributes["EquipmentType"].Value);
Console.WriteLine("Vessel aabbcc_code  ==>
        "+ objVessel.Attributes["aabbcc_code"].Value);
Console.WriteLine("Vessel Class  ==>
        "+ objVessel.Attributes["Class"].Value);
Console.WriteLine("Vessel ItemTypeName  ==>
        "+ objVessel.Attributes["ItemTypeName"].Value);
Console.WriteLine("Vessel VolumeRating  ==>
        "+ objVessel.Attributes["VolumeRating"].Value);
Console.WriteLine("Vessel VolumeRating in SI units  ==>
        " + objVessel.Attributes["VolumeRating"].SIValue);

objVessel = null;
datasource = null;
```

5. IDENTIFY AN ITEM IN THE DATABASE AND MODIFY ITS PROPERTIES

Purpose

To modify its properties of items in the database

b) Problem Statement

Place a vessel. Write a standalone application to modify the following property of the vessel:
Name

c) Solution

1. Get the vessel object using `LMADataSource.GetVessel` method.
2. Change the value of required properties
3. Use `LMVessel.Commit` to commit the change to database

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);

datasource.BeginTransaction();

objVessel.Attributes["Name"].Value = "Vessel 7";
objVessel.Attributes["DesignBy"].Value = "By B";
objVessel.Commit();

datasource.CommitTransaction();

Console.WriteLine("Done.");

objVessel = null;
datasource = null;
```

6. INIT OBJECTS READ ONLY

a) Purpose

To use property of `LMADatasource: InitObjectsReadonly`

b) Problem Statement

Place a Piperun. Write a standalone application to get `LMPiperun`, then set the `InitObjectsReadonly` to `true`, and check if the property "Name" can be changed with drawing close and `New LMADatasource()` is used.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

datasource.InitObjectsReadonly = true;

LMPipeRun objPiperun = datasource.GetPipeRun(CONST_SPID_PipeRun);

datasource.BeginTransaction();

objPiperun.Attributes["Name"].Value = "TEST1";
objPiperun.Commit();

datasource.CommitTransaction();

Console.WriteLine("PipeRun Name  ==>  " + objPiperun.Attributes["Name"].Value);

objPiperun = null;
datasource = null;
```

7. ROLLBACK

a) Purpose

To rollback a transaction by automation program

Problem Statement

Place a Piperun. Write a standalone application to get LMPiperun, then change the property “Name” of the piperun and CommitTransaction, then change the property “Name” again, but this time RollbackTransaction, check which value is commited.

c) Solution

1. Define a LMPipeRun object and get the object using LMADataSource.GetPipeRun method.
2. Using LMADataSource.RollbackTransaction method.

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMPipeRun objPiperun = datasource.GetPipeRun(CONST_SPID_PipeRun);

datasource.BeginTransaction();

objPiperun.Attributes["Name"].Value = "TEST1";
objPiperun.Commit();

datasource.CommitTransaction();

Console.WriteLine("PipeRun Name  ==>  " + objPiperun.Attributes["Name"].Value);

datasource.BeginTransaction();

objPiperun.Attributes["Name"].Value = "TEST2";
objPiperun.Commit();

Console.WriteLine("PipeRun Name  ==>  " + objPiperun.Attributes["Name"].Value);

datasource.RollbackTransaction();

objPiperun = null;
datasource = null;
```

8. PROPAGATION

a) Purpose

To set propagation to True or False from automation program

b) Problem Statement

Place a PipeRun, then place couple branch PipeRuns to this piperun. Write a standalone application to modify the property “SupplyBy” of the first PipeRun with Propagation set to True and modify the property “CleaningReqmts” with Propagation set to False.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMPipeRun objPiperun = datasource.GetPipeRun(CONST_SPID_PipeRun);

datasource.BeginTransaction();

datasource.PropagateChanges = true;

objPiperun.Attributes["SupplyBy"].Value = "By D";
objPiperun.Commit();

datasource.PropagateChanges = false;

objPiperun.Attributes["CleaningReqmts"].Value = "CC1";
objPiperun.Commit();

datasource.CommitTransaction();

objPiperun = null;
datasource = null;
```

9. Access LMAAttributes Collection

Purpose

To access LMAAttributes collection of LLAMA object.

Problem Statement

Place a Vessel and get the LMVessel object, then loop through its Attributes collection.

Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);

Console.WriteLine("\tTotal attributes for Vessel = " + objVessel.Attributes.Count);

Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for Vessel");

foreach (LMAAttribute objAttr in objVessel.Attributes)
        Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
        Console.WriteLine("ProcessAlternateDesign.Max.Pressure  ==>
        " + objVessel.Attributes["ProcessAlternateDesign.Max.Pressure"].Value + "\n");
        Console.WriteLine("\tTotal attributes for Vessel = " +
objVessel.Attributes.Count);
        Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for Vessel");

foreach (LMAAttribute objAttr in objVessel.Attributes)
        Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);

objAttr = null;
objVessel = null;
datasource = null;
```

10. ACCESS ITEMATTRIBUTIONS IN DETAILS

a) Purpose

To access ItemAttributions of different items in details

b) Problem Statement

Place all kinds of Smart P&ID items, such as Vessel, Mechanical, Heat Exchanger, then print their ItemAttributions information in details in format of Excel, which includes attribution format, index if codelist, calculation ProgID and validation ProgID

c) Solution

1. Get Items
2. Needs access LMAAttribute.ISPAttribute
3. Print result in Excel

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);

Excel.Application objExcel =
Activator.CreateInstance(Type.GetTypeFromProgID("Excel.Application")) as
Excel.Application;

objExcel.Visible = true;
Excel.Workbook xlWorkbook = objExcel.Workbooks.Add();
Excel.Worksheet xlWorksheet = objExcel.Worksheets["SHEET1"] as Excel.Worksheet;
long row = 1;
xlWorksheet.Cells[row, 1] = "ItemType";
xlWorksheet.Cells[row, 2] = "Attribute Name";
xlWorksheet.Cells[row, 3] = "Format";
xlWorksheet.Cells[row, 4] = "IsCodeList";
xlWorksheet.Cells[row, 5] = "CodeList Index";
xlWorksheet.Cells[row, 6] = "Calculation ProgID";
xlWorksheet.Cells[row, 7] = "Validation ProgID";
row++;

xlWorksheet.Cells[row, 1] = "Total attributions for Vessel: " +
objVessel.Attributes.Count;
row++;

long codeListCount = 0;
foreach (LMAAttribute objAttr in objVessel.Attributes)
{
     xlWorksheet.Cells[row, 1] = objVessel.AsLMAItem().ItemType;
     xlWorksheet.Cells[row, 2] = objAttr.Name;
     xlWorksheet.Cells[row, 3] = objAttr.ISPAttribute.Attribution.Format;
     try
     {
            codeListCount = 0;
            codeListCount = objAttr.ISPAttribute.Attribution.ISPEnumAtts.Count;
     }
```

```
        catch (Exception ex)
        {
               ex.ToString();
        }

        if (codeListCount > 0)
               xlWorksheet.Cells[row, 4] = "True";
        else

        xlWorksheet.Cells[row, 4] = "False";
        xlWorksheet.Cells[row, 5] = objAttr.Index;
        xlWorksheet.Cells[row, 6] = objAttr.ISPAttribute.Attribution.CalculationProgID;
        xlWorksheet.Cells[row, 7] = objAttr.ISPAttribute.Attribution.ValidationProgID;
        row++;
        }
        row++;

        try
        {
               Object objValue = objVessel.Attributes["ProcessDesign.Max.Pressure"];
        }
        catch (Exception ex)
        {
               MessageBox.Show(ex.GetBaseException().ToString());
         }

        xlWorksheet.Cells[row, 1] = "Total attributions for Vessel: " +
        objVessel.Attributes.Count;

        row++;
        foreach (LMAAttribute objAttr in objVessel.Attributes)
        {
               xlWorksheet.Cells[row, 1] = objVessel.AsLMAItem().ItemType;
               xlWorksheet.Cells[row, 2] = objAttr.Name;
               xlWorksheet.Cells[row, 3] = objAttr.ISPAttribute.Attribution.Format;
               try
               {
                      codeListCount = 0;
                      codeListCount = objAttr.ISPAttribute.Attribution.ISPEnumAtts.Count;
               }
               catch (Exception ex)
               {
                      ex.ToString();
               }
               if (codeListCount > 0)
                      xlWorksheet.Cells[row, 4] = "True";
               else
                      xlWorksheet.Cells[row, 4] = "False";
                      xlWorksheet.Cells[row, 5] = objAttr.Index;
                      xlWorksheet.Cells[row, 6] =
                      objAttr.ISPAttribute.Attribution.CalculationProgID;
                      xlWorksheet.Cells[row, 7] =
                      objAttr.ISPAttribute.Attribution.ValidationProgID;
                     row++;
                 }
```

```
objExcel.Workbooks[1].SaveAs(Environment.GetEnvironmentVariable("TEMP") +
"\\ItemAttributions.xls");

xlWorkbook.Close(true);
objExcel.Quit();

MessageBox.Show("Lab 10 - Access Item Attributions In Details complete.");

objVessel = null;
xlWorksheet = null;
xlWorkbook = null;
objExcel = null;
datasource = null;
```

11. Collect Items from the Database Using Filters

a) Purpose

To access objects created through SPID using filters

b) Problem Statement

Place a piperun and give it a TagSuffix value. Retrieve the piperun by filtering on the TagSuffix value = “P” and populate the Name property with value “P-Run”

c) Solution

1. Define LMAFilter and LMACriterion
2. Add LMACriterion to LMAFilter
3. Call LMPipeRuns.Collect method by using the LMAFilter

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "TagSuffix";
criterion.Operator = "=";
criterion.ValueAttribute = "P";

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PipeRun";
objFilter.Criteria.Add(criterion);

LMPipeRuns objPiperuns = new LMPipeRuns();
objPiperuns.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal PipeRuns filtered = " + objPiperuns.Count);

datasource.BeginTransaction();

foreach (LMPipeRun objPiperun in objPiperuns)
{
       Console.WriteLine("PipeRun TagSuffix  ==>  " +
       objPiperun.Attributes["TagSuffix"].Value);
       Console.WriteLine("PipeRun Id = " + objPiperun.Id);
       objPiperun.Attributes["Name"].Value = "P-Run";
       objPiperun.Commit();
}

datasource.CommitTransaction();

criterion = null;
objFilter = null;
objPiperuns = null;
datasource = null;
```

12. Collect Items from the Database Using Filters with Multiple Criteria

Purpose

To access objects created through SPID using filters with multiple criteria

b) Problem Statement

Place three piperuns and set OperFluidCode=”KD” for one piperun, TagSuffix = “PT” for another pipe run and Name=”V” for another pipe run. Retrieve the three piperuns by filtering using Multiple Criteria.

c) Solution

1. Dim LMAFilter and LMACriterion
2. Add multiple LMACriterion to LMAFilter
3. Call LMPipeRuns.Collect method by using the LMAFilter

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterions criteria = new LMACriterions();
criteria.AddNew("1stOne");
criteria["1stOne"].SourceAttributeName = "ItemTag";
criteria["1stOne"].Operator = "like";
criteria["1stOne"].ValueAttribute = "%K%";
criteria.AddNew("2ndOne");
criteria["2ndOne"].SourceAttributeName = "TagSuffix";
criteria["2ndOne"].Operator = "like";
criteria["2ndOne"].ValueAttribute = "P_";
criteria["2ndOne"].Conjunctive = false;
criteria.AddNew("3rdOne");
criteria["3rdOne"].SourceAttributeName = "Name";
criteria["3rdOne"].Operator = "!=";
criteria["3rdOne"].ValueAttribute = null;
criteria["3rdOne"].Conjunctive = false;

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PipeRun";
objFilter.Criteria.Add(criteria["1stOne"]);
objFilter.Criteria.Add(criteria["2ndOne"]);
objFilter.Criteria.Add(criteria["3rdOne"]);

Console.WriteLine("\tCriteria count = " + objFilter.Criteria.Count);

LMPipeRuns objPiperuns = new LMPipeRuns();
objPiperuns.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal PipeRuns filtered = " + objPiperuns.Count);

foreach (LMPipeRun objPiperun in objPiperuns)
{
     Console.WriteLine("PipeRun.Id = " + objPiperun.Id);
     Console.WriteLine("PipeRun ItemTag  ==>  " +
     objPiperun.Attributes["ItemTag"].Value);
Console.WriteLine("PipeRun TagSuffix  ==>  " +
objPiperun.Attributes["TagSuffix"].Value);
Console.WriteLine("PipeRun Name  ==>  " + objPiperun.Attributes["Name"].Value);
}

criteria = null;
objFilter = null;
objPiperuns = null;
datasource = null;
```

13. USING FILTERS WITH CRITERIA ON SELECT LIST DATA

a) Purpose

To access objects created through SPPID using filters with multiple criteria

b) Problem Statement

Place two piperuns and set NominalDiameter=2” for the piperuns. Then delete one piperun from model. Retrieve the active piperun by filtering using Criteria on ItemStatus and NominalDiameter.

c) Solution

Need to find the index for ItemStatus=”Active” and NominalDiameter=2”.

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterions criteria = new LMACriterions();
criteria.AddNew("1stOne");
criteria["1stOne"].SourceAttributeName = "ItemStatus";
criteria["1stOne"].Operator = "=";
criteria["1stOne"].ValueAttribute ="1";
criteria.AddNew("2ndOne");
criteria["2ndOne"].SourceAttributeName = "NominalDiameter";
criteria["2ndOne"].Operator = "=";
criteria["2ndOne"].ValueAttribute = "5064"; // 2"
criteria["2ndOne"].Conjunctive = true;

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PipeRun";
objFilter.Criteria.Add(criteria["1stOne"]);
objFilter.Criteria.Add(criteria["2ndOne"]);

Console.WriteLine("criteria count = " + objFilter.Criteria.Count);

LMPipeRuns objPiperuns = new LMPipeRuns();
objPiperuns.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal PipeRuns filtered = " + objPiperuns.Count);

foreach (LMPipeRun objPiperun in objPiperuns)
{
     Console.WriteLine("PipeRun.Id = " + objPiperun.Id);
     Console.WriteLine("PipeRun ItemStatus  ==>  " +
     objPiperun.Attributes["ItemStatus"].Value);
      Console.WriteLine("PipeRun NominalDiameter  ==>  " +
     objPiperun.Attributes["NominalDiameter"].Value);
}

criteria = null;
objFilter = null;
objPiperuns = null;
datasource = null;
```

14. Using Compound Filter

a) Purpose

To access objects created through SPID using compound filter

b) Problem Statement

Place six piperuns and set NominalDiameter=1”, 2”, and 3” for the piperuns. Then, delete three piperuns from model. Retrieve the piperuns with ItemStatus=”Active” and NominalDiameter equals 1” or 2” by using compound filter.

c) Solution

Comound allows conjunctive as both “And” and “Or”.

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "ItemStatus";
criterion.Operator = "=";
criterion.ValueAttribute = "1";
LMAFilter objChildFilter1 = new LMAFilter();
objChildFilter1.ItemType = "PipeRun"
objChildFilter1.Name = "Filter 1";
objChildFilter1.Criteria.Add(criterion);

LMACriterions criteria = new LMACriterions();
criteria.AddNew("1stOne");
criteria["1stOne"].SourceAttributeName = "NominalDiameter";
criteria["1stOne"].Operator = "=";
criteria["1stOne"].ValueAttribute = "5032"; // = 1"
criteria.AddNew("2ndOne");
criteria["2ndOne"].SourceAttributeName = "NominalDiameter";
criteria["2ndOne"].Operator = "=";
criteria["2ndOne"].ValueAttribute = 5064; // 2"
criteria["2ndOne"].Conjunctive = false;
LMAFilter objChildFilter2 = new LMAFilter();
objChildFilter2.Conjunctive = false;
objChildFilter2.ItemType = "PipeRun";
objChildFilter2.Name = "Filter 2";
objChildFilter2.Criteria.Add(criteria["1stOne"]);
objChildFilter2.Criteria.Add(criteria["2ndOne"]);

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PipeRun";
objFilter.FilterType = 1; // 1 for compound filter, 0 for simple filter
objFilter.ChildLMAFilters.Add(objChildFilter1);
objFilter.ChildLMAFilters.Add(objChildFilter2);
objFilter.Conjunctive = true; // AND

LMPipeRuns objPiperuns = new LMPipeRuns();
objPiperuns.Collect(datasource, null, "", objFilter);
```

```
Console.WriteLine("\tTotal PipeRuns filtered = " + objPiperuns.Count);

foreach (LMPipeRun objPiperun in objPiperuns)
{
       Console.WriteLine("PipeRun ItemStatus = " +
       objPiperun.Attributes["ItemStatus"].Value);

       Console.WriteLine("PipeRun NominalDiameter = " +
       objPiperun.Attributes["NominalDiameter"].Value);
 }

 criterion = null;
 criteria = null;
 objFilter = null;
 objChildFilter1 = null;
 objChildFilter2 = null;
 objPiperuns = null;
 datasource = null;
```

15. Collect Filters from Datasource

a) Purpose

To collect all filters in SPID from datasource

b) Problem Statement

Write a standalone application to retrieve all filters in SPPID from datasource. Display the Item Type and the first Criterion (if one exists) in the filter for those of ItemType = "Instrument".

c) Solution

1. Define LMAFilter
2. Call LMADataSource.Filters method to get all LMAFilters in database
3. Use For … Next to loop through the LMAFilters and print out required properties

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

Collection objFiltersCollection = datasource.Filters; //using VBA
Console.WriteLine("\tTotal filters found = " + objFiltersCollection.Count());

foreach (LMAFilter objFilter in objFiltersCollection)
if (objFilter.ItemType.Equals("Instrument") && objFilter.Criteria != null)
     if (objFilter.Criteria.Count >= 1)
     {
             Console.WriteLine("objFilter.ItemType = " + objFilter.ItemType +
             "\t\tobjFilter.Name = " + objFilter.Name);
             Console.WriteLine("\tSourceAttributeName = " +
             objFilter.Criteria[1].SourceAttributeName);
             Console.WriteLine("\tOperator = " + objFilter.Criteria[1].Operator);
             Console.WriteLine("\tValueAttribute = " +
             objFilter.Criteria[1].ValueAttribute);
     }

 LMAFilter objFilterEq = datasource.Filters.Item("Active Equipment") as LMAFilter;
 LMEquipments objEquipments = new LMEquipments();
 objEquipments.Collect(datasource, null, "", objFilterEq);

 Console.WriteLine("\tTotal found using pre-defined filter for Active Equipment = " +
 objEquipments.Count);

 objFiltersCollection = null;
 objFilterEq = null;
 objEquipments = null;
 datasource = null;
```

16. ACCESS SELECTLIST DATA

a) Purpose

To get familiar with LMAEnumAttList and LMAEnumratedAttributes objects in LLAMA.

b) Problem Statement

Write a standalone application to retrieve all Select List Data in SPID from datasource. Display properties, such as ListName, DependName, DependID. Then loop through all Select List Value of each Select List Data, display properties, such as Name and Index.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMAEnumAttLists objEnumLists = datasource.CodeLists;

Console.WriteLine("\tTotal SelectList Data found = " + objEnumLists.Count);

// loop through the code lists and their enumerated attributes
foreach (LMAEnumAttList objEnumList in objEnumLists)
{
       Console.WriteLine(String.Empty);
       Console.WriteLine("SelectList name = " + objEnumList.ListName + "\tDependName = "
       + objEnumList.DependName + "\tDependID = " + objEnumList.DependID);
       Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for
       EnumAttList.EnumeratedAttribute");

       foreach (LMAEnumeratedAttribute objEnumAttr in
              objEnumList.get_EnumeratedAttributes())
              Console.WriteLine(objEnumAttr.Name + "  ==>  " + objEnumAttr.Index);
 }


 objEnumLists = null;
 datasource = null;
```

17. CREATE FILTER WITH SELECT LIST DATA IN CRITERIA

a) Purpose

To create a filter with select list data in criteria, learn how to resolve the select data to its index dynamically.

b) Problem Statement

Place couple piping valves in drawing. Write a standalone application to collect all Ball Valves.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "PipingCompType";
criterion.Operator = "=";
criterion.ValueAttribute = datasource.CodeLists["Piping Component
     Type"].get_EnumeratedAttributes().GetItemIndex("Ball valve");

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PipingComp";
objFilter.Criteria.Add(criterion);

LMPipingComps objPipingComps = new LMPipingComps();
objPipingComps.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal Ball Valves filtered = " + objPipingComps.Count);

criterion = null;
objFilter = null;
objPipingComps = null;
datasource = null;
```

18. READ HISTORY PROPERTY OF MODELITEM

a) Purpose

To read the history data belongs to a modelitem.

b) Problem Statement

Place a Vessel. Write a standalone application to read the history data belongs to this Vessel.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMModelItem objModelItem = datasource.GetModelItem(CONST_SPID_ModelItem);

LMHistories objHistories = objModelItem.Histories;

Console.WriteLine("\tTotal Histories = " + objHistories.Count);

foreach (LMHistory objHistory in objHistories)
{
     Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for History");

     foreach (LMAAttribute objAttr in objHistory.Attributes)
            Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
}

objModelItem = null;
objHistories = null;
datasource = null;
```

19. Read Status Property of ModelItem

a) Purpose

To read the Status datas belongs to a modelitem

b) Problem Statement

Place a Vessel with some Status data populated. Write a standalone application to read the status data belongs to this Vessel.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMModelItem objModelItem = datasource.GetModelItem(CONST_SPID_ModelItem);

LMStatuses objStatuses = objModelItem.Statuses;

Console.WriteLine("\tTotal Statuses = " + objStatuses.Count);

foreach (LMStatus objStatus in objStatuses)
{
       Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for Status");

       foreach (LMAAttribute objAttr in objStatus.Attributes)
             Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
 }

 objModelItem = null;
 objStatuses = null;
 datasource = null;
```

20. READ CASE PROPERTY OF MODELITEM

a) Purpose

To read Case data of a modelitem

b) Problem Statement

Place a Vessel with some Case data populated. Write a standalone application to read the case data belongs to this Vessel.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMModelItem objModelItem = datasource.GetModelItem(CONST_SPID_ModelItem);

LMCases objCases = objModelItem.Cases;

Console.WriteLine("\tTotal Cases = " + objCases.Count);

foreach (LMCase objCase in objCases)
{
       Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for Case");

       foreach (LMAAttribute objAttr in objCase.Attributes)
              Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
              Console.WriteLine("\tTotal Case.CaseProcesses = " +
              objCase.CaseProcesses.Count);

       foreach (LMCaseProcess objCaseProcess in objCase.CaseProcesses)
       {
              Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for
              CaseProcess");

              foreach (LMAAttribute objAttr in objCaseProcess.Attributes)
                     Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
        }

       Console.WriteLine("\tTotal Case.CaseControls = " + objCase.CaseControls.Count);

       foreach (LMCaseControl objCaseControl in objCase.CaseControls)
       {
              Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for
              CaseControl");

              foreach (LMAAttribute objAttr in objCaseControl.Attributes)
                     Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
        }
  }

  objModelItem = null;
  objCases = null;
  datasource = null;
```

21. ACCESS ITEMNOTE

a) Purpose

To access an ItemNote.

b) Problem Statement

Place an ItemNote. Write a standalone application to read the properties of this ItemNote.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMItemNote objItemNote = datasource.GetItemNote(CONST_SPID_ItemNote);

Console.WriteLine("\tTotal ItemNote attributes = " + objItemNote.Attributes.Count);

// loop through the item note attributes
Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for ItemNote");

foreach (LMAAttribute objAttr in objItemNote.Attributes)
       Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
       Console.WriteLine("\tTotal ItemNote.Notes = " + objItemNote.Notes.Count);

// loop through the note attributes
foreach (LMNote objNote in objItemNote.Notes)
{
       Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for ItemNote.Note");

       foreach (LMAAttribute objAttr in objNote.Attributes)
              Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
 }

 objItemNote = null;
 datasource = null;
```

22. Access OPC

a) Purpose

To access an OPC

b) Problem Statement

Place an OPC and its PairOPC in another drawing. Write a standalone application to read the properties of this OPC and its PairOPC.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMOPC objOPC = datasource.GetOPC(CONST_SPID_OPC);

Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for OPC");

foreach (LMAAttribute objAttr in objOPC.Attributes)
     Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);

LMOPC objPairOPC = objOPC.pairedWithOPCObject;

Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for Paired OPC");

foreach (LMAAttribute objAttr in objPairOPC.Attributes)
     Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);

objOPC = null;
objPairOPC = null;
datasource = null;
```

23. Filter for Histories

a) Purpose

To filter for histories by TimeStamp and ItemType

b) Problem Statement

Set the active plant with some items placed. Write a standalone application to filter Histories.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterions criteria = new LMACriterions();
criteria.AddNew("1stOne");
criteria["1stOne"].SourceAttributeName = "TimeStamp";
criteria["1stOne"].Operator = ">";
criteria["1stOne"].ValueAttribute = "7/19/12 8:00:00 AM";
criteria.AddNew("2ndOne");
criteria["2ndOne"].SourceAttributeName = "ModelItem.ModelItemType";
criteria["2ndOne"].Operator = "=";
criteria["2ndOne"].ValueAttribute = 29; // 29 is the index for 'PlantItem'
criteria["2ndOne"].Conjunctive = true;

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "History";
objFilter.Criteria.Add(criteria["1stOne"]);
objFilter.Criteria.Add(criteria["2ndOne"]);

LMHistories objHistories = new LMHistories();
objHistories.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal Histories filtered = " + objHistories.Count);

foreach (LMHistory objHistory in objHistories)
{
     Console.WriteLine("objHistory TimeStamp  ==>  " +
     objHistory.Attributes["TimeStamp"].Value);
     Console.WriteLine("objHistory.ModelItemObject ModelItemType  ==>  " +
     objHistory.ModelItemObject.Attributes["ModelItemtype"].Value);
 }

 objFilter = null;
 objHistories = null;
 datasource = null;
```

24. CHANGE PROPERTIS AT DIFFERNET OBJECT LEVELS

a) Purpose

To access “Name” property at different object level.

b) Problem Statement

Place a vessel. Write a standalone application to change “Name” property at Equipment object level, and see how it changes the output for Vessel object

c) Solution

1. Use LMADataSource.GetVessel and LMADataSource.GetEquipment methods to obtain object Vessel and Equipment with same SP_ID
2. Change property “Name” value of Equipment object, then obtain Vessel object again to see how it changes the property “Name” value of Vessel

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMEquipment objEquipment = datasource.GetEquipment(CONST_SPID_Equipment);

Console.WriteLine("\tTotal Equipment attributes = " + objEquipment.Attributes.Count);

Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for Equipment");

foreach (LMAAttribute objAttr in objEquipment.Attributes)
       Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);

Console.WriteLine("\tTotal Vessel attributes = " + objVessel.Attributes.Count);


Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for Vessel");

foreach (LMAAttribute objAttr in objVessel.Attributes)
       Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
       Console.WriteLine("Equipment.Name = " + objEquipment.Name);
       Console.WriteLine("Vessel.Name = " + objVessel.Name);

datasource.BeginTransaction();

objEquipment.Attributes["Name"].Value = "Lab-24";
objEquipment.Commit();

objVessel = datasource.GetVessel(CONST_SPID_Vessel);
Console.WriteLine("Equipment.Name = " + objEquipment.Name);
Console.WriteLine("Vessel.Name = " + objVessel.Name);

datasource.CommitTransaction();

objEquipment = null;
objVessel = null;
datasource = null;
```

25. READ CASEPROPERTY OF VESSEL

a) Purpose

To access case properties of Vessel.

b) Problem Statement

Place a vessel. Populate the some case property value of the vessel. Write a standalone application to access the case and caseprocess of the vessel and read properties of the case.

c) Solution

1. Dim LMVessel
2. Vessel is associated several LMCases, if Case Class is Case Process, then this Case can have two CaseProcesses associated with it, depends on Quality, which can be Maximun or Minimum, then a one to one filtered relationship is found for the Vessel and Case property

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);

Console.WriteLine("Vessel ProcessDesign.Min.Pressure  ==>  " +
     objVessel.Attributes["ProcessDesign.Min.Pressure"].Value);

objVessel = null;
datasource = null;
```

26. READ FLOW DIRECTION OF PIPERUN

Purpose

To obtain Flow Direction of Piperun

b) Problem Statement

Place a Piperun. Write a standalone application to obtain Flow Direction information about the Piperun.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMPipeRun objPiperun = datasource.GetPipeRun(CONST_SPID_PipeRun);

Console.WriteLine("PipeRun FlowDirection  ==>  " +
     objPiperun.Attributes["FlowDirection"].Value);

objPiperun = null;
datasource = null;
```

27. Access Piping Point

a) Purpose

To access a Piping Point.

b) Problem Statement

Place a Valve. Write a standalone application to access PipingPoint belongs to this Valve.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMPipingComp objPipingComp = datasource.GetPipingComp(CONST_SPID_PipingComp);

Console.WriteLine("\tTotal PipingComp PipingPoints = " + pipingComp.PipingPoints.Count);

// loop through the piping point attributes

foreach (LMPipingPoint objPipingPoint in objPipingComp.PipingPoints)
{
       Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for
       PipingComp.PipingPoint");

       foreach (LMAAttribute objAttr in objPipingPoint.Attributes)
              Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
}

objPipingComp = null;
datasource = null;
```

28. Access Signal Point

a) Purpose

To access a Signal Point.

b) Problem Statement

Place an offline Instrument. Write a standalone application to access PipingPoint belongs to this offline Instrument.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMInstrument objInstrument = datasource.GetInstrument(CONST_SPID_OfflineInstrument);

Console.WriteLine("\tTotal Instrument SignalPoints = " +
       objInstrument.SignalPoints.Count);

// loop through the signal points' attributes
foreach (LMSignalPoint objSignalPoint in objInstrument.SignalPoints)
{
       Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for
              Instrument.SignalPoint");

       foreach (LMAAttribute objAttr in objSignalPoint.Attributes)
              Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
 }

 objInstrument = null;
 datasource = null;
```

29. ImpliedItem

a) Purpose

To navigate the relationship between Implied item and its parent item.

b) Problem Statement

Place a Instrument off-line with implied item. Write a standalone application to obtain any items in the database that are Implied Item

c) Solution

1. Dim LMPlantItem, LMACriterion and LMAFilter
2. Implied item would have property “PartOfType” is equal to “Implied”, which has the index number is 2.

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "PartOfType";
criterion.Operator = "=";
criterion.ValueAttribute = "2"; // implied item

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PlantItem";
objFilter.Criteria.Add(criterion);

LMPlantItems objPlantItems = new LMPlantItems();
objPlantItems.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal ImpliedItems filtered = " + objPlantItems.Count);

foreach (LMPlantItem objPlantItem in objPlantItems)
{
     Console.WriteLine("PlantItem PartOfType  ==>  " +
            objPlantItem.Attributes["PartOfType"].Value);
     Console.WriteLine("PlantItem.PartOfPlantItemObject ItemType  ==>  " +
            objPlantItem.PartOfPlantItemObject.Attributes["ItemTypeName"].Value);
}

criterion = null;
objFilter = null;
objPlantItems = null;
datasource = null;
```

30. PARTOFPLANTITEM RELATIONSHIPS

a) Purpose

To navigate the relationship between item and its parent item.

b) Problem Statement

Place a Instrument off-line with implied item, two nozzles, two trays, TEMA ends Write a standalone application to find all the items that have parent item in the database

c) Solution

1. Dim LMPlantItem, LMACriterion and LMAFilter
2. Implied item would have property “SP_PartOfID” is not NULL

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "SP_PartOfID";
criterion.Operator = "!=";
criterion.ValueAttribute = null;

// add the criterion to a filter
LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PlantItem";
objFilter.Criteria.Add(criterion);

LMPlantItems objPlantItems = new LMPlantItems();
objPlantItems.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal PlantItems filtered = " + objPlantItems.Count);

foreach (LMPlantItem objPlantItem in objPlantItems)
{
       Console.WriteLine("PlantItem PartOfType  ==>  " +
              objPlantItem.Attributes["PartOfType"].Value);
       Console.WriteLine("PlantItem.PartOfPlantItemObject ItemTypeName  ==>  " +
              objPlantItem.PartOfPlantItemObject.Attributes["ItemTypeName"].Value);
}

objFilter = null;
criterion = null;
objPlantItems = null;
datasource = null;
```

31. Access Instrument Loop

a) Purpose

To get familiar with relationship between PlantItemGroup and PlantItem

c) Problem Statement

Use LMAFilter to search for Instrument Loops, then check how many PlantItems are associated with the Instrument Loop. At the end, try to associate an Instrument with this Instrument Loop.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "ItemTag";
criterion.Operator = "=";
criterion.ValueAttribute = "L-100L";

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "InstrLoop";
objFilter.Criteria.Add(criterion);

LMInstrLoops objInstrLoops = new LMInstrLoops();
objInstrLoops.Collect(datasource, null, "", objFilter);

LMInstrLoop objInstrLoop = null;
if (objInstrLoops.Count > 0)
     objInstrLoop = objInstrLoops.Nth(1);
else
{
     criterion = null;
     objFilter = null;
     objInstrLoops = null;
     objInstrLoop = null;
     datasource = null;
     return;
 }

 LMPlantItems objPlantItems = objInstrLoop.PlantItems;
 Console.WriteLine("\tTotal PlantItems in the InstrLoop = " + objPlantItems.Count);

 int i = 1;
 foreach (LMPlantItem objPlantItem in objPlantItems)
 {
     Console.WriteLine("ItemTypeName # " + i.ToString() + " = " +
            objPlantItem.ItemTypeName.ToString() + ",  PlantItemID = " +
            objPlantItem.Id);
     i++;
 }

 // add an instrument to the instrument loop
 LMInstrument objInstr = datasource.GetInstrument(CONST_SPID_OfflineInstrument);
```

```
Console.WriteLine("Total PlantItemGroups associated with Instrument before adding to
   InstrLoop = " + objInstr.PlantItemGroups.Count);


   objInstr.PlantItemGroups.Add(objInstrLoop.AsLMPlantItemGroup().AsLMAItem());

Console.WriteLine("Total PlantItemGroups associated with Instrument after adding to
   InstrLoop = " + objInstr.PlantItemGroups.Count);

criterion = null;
objFilter = null;
objInstr = null;
objInstrLoops = null;
objInstrLoop = null;
objPlantItems = null;
datasource = null;
```

32. LoadInstruments

a) Purpose

To navigate the relationship between Instrloop and Instrument through LoadInstruments method.

b) Problem Statement

Place couple instrloops, and couple instruments, then make association between them. Write a standalone application to find instruments associated with instrloops through LoadInstruments method.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMInstrLoops objInstrLoops = new LMInstrLoops();
objInstrLoops.Collect(datasource);

Console.WriteLine("\tTotal InstrumentLoops = " + objInstrLoops.Count);

// load the instruments and loop through them
LMInstruments objInstruments = objInstrLoops.LoadInstruments();
Console.WriteLine("\tTotal Instruments in all Loops = " + objInstruments.Count);

foreach (LMInstrument objInstrument in objInstruments)
{
     Console.WriteLine("Instrument ItemTag  ==>  " +
            objInstrument.Attributes["ItemTag"].Value);
     Console.WriteLine("Instrument.PlantItemGroups.Nth[1] ItemTag  ==>  " +
            objInstrument.PlantItemGroups.Nth(1).Attributes["ItemTag"].Value);
}

// loop through the instrument loops
foreach (LMInstrLoop objInstrLoop in objInstrLoops)
{
     Console.WriteLine("InstumentLoop ItemTag  ==>  " +
            objInstrLoop.Attributes["ItemTag"].Value);

     if (objInstrLoop.Instruments != null)
            Console.WriteLine("\tTotal InstrumentLoop.Instruments = " +
                   objInstrLoop.Instruments.Count);
 }

 objInstruments = null;
 objInstrLoops = null;
 datasource = null;
```

33. Identify Nozzle and Equipment

a) Purpose

To access nozzles on a vessel by navigating the relationship between nozzles and vessels.

b) Problem Statement

Place a vessel. Place two different nozzles on the vessel. Write a standalone application to retrieve the following properties of the nozzle:
SP_ID, aabbcc code, ID of equipment that the nozzle is connected to, Flowdirection, Nozzle type.

c) Solution

1. Obtain Vessel object by SP_ID
2. Use LMAVessel.nozzles to get a collection of nozzle belong to this Vessel

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);
LMNozzles objNozzles = objVessel.Nozzles;

Console.WriteLine("\tTotal Vessel.Nozzles = " + objNozzles.Count);

foreach (LMNozzle objNozzle in objNozzles)
{
     Console.WriteLine("Nozzle.Id = " + objNozzle.Id);
     Console.WriteLine("Nozzle.EquipmentID = " + objNozzle.EquipmentID);
     Console.WriteLine("Nozzle aabbcc_code  ==>  " +
            objNozzle.Attributes["aabbcc_code"].Value);
     Console.WriteLine("Nozzle FlowDirection  ==>  " +
            objNozzle.Attributes["FlowDirection"].Value);
     Console.WriteLine("Nozzle NozzleType  ==>  " +
            objNozzle.Attributes["NozzleType"].Value);
 }

 objVessel = null;
 objNozzles = null;
 datasource = null;
```

34. PipingComp and InlineComp

a) Purpose

To navigate the relationship between PipingComp and InlineComp.

b) Problem Statement

Place a Valve. Write a standalone application to navigate from pipingcomp to piperun and from piperun to pipingcomp through InlineComp.

c) Solution

1. Use LMADataSource.GetPipingComp
2. Loop LMPipingComp.InlineComps
3. Use LMInlinecomp.PipeRunObject

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMPipeRun objPiperun = null;
LMPipingComp objPipingComp = datasource.GetPipingComp(CONST_SPID_PipingComp);

if (objPipingComp.InlineComps.Count == 1)
{
     objPiperun = objPipingComp.InlineComps.Nth(1).PipeRunObject;

     Console.WriteLine("PipeRun ItemTag  ==>  " +
            objPiperun.Attributes["ItemTag"].Value);

     foreach (LMInlineComp objInlineComp in objPiperun.InlineComps)
     {
            objPipingComp = null;
            objPipingComp = objInlineComp.PipingCompObject;

            if (objPipingComp != null)
                   Console.WriteLine("PipingComp PipingCompType  ==>  " +
                          objPipingComp.Attributes["PipingCompType"].Value);
     }
  }

  objPipingComp = null;
  objPiperun = null;
```

35. INSTURMENT AND INLINECOMP

a) Purpose

To navigate the relationship between Inline-Instrument and InlineComp.

b) Problem Statement

Place a Instrument Valve. Write a standalone application to navigate from inline-instrument to piperun and from piperun to inline-instrument through inlinecomp.

c) Solution

1. Use LMADataSource.GetInstrument
2. Loop LMInstrument.InlineComps
3. Use LMInlinecomp.InstrumentObject

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMInstrument objInstrument = datasource.GetInstrument(CONST_SPID_InlineInstrument);

if (objInstrument.Attributes["IsInline"].Value.Equals("True"))
{
     LMPipeRun objPiperun = objInstrument.InlineComps.Nth(1).PipeRunObject;

     Console.WriteLine("PipeRun ItemTag  ==>  " +
            objPiperun.Attributes["ItemTag"].Value);

     foreach (LMInlineComp objInlineComp in objPiperun.InlineComps)
     {
            objInstrument = null;
            objInstrument = objInlineComp.InstrumentObject;

            if (objInstrument != null)
                   Console.WriteLine("Instrument InstrumentType  ==>  " +
                          objInstrument.Attributes["InstrumentType"].Value);
      }

       objPiperun = null;
  }

  objInstrument = null;
  datasource = null;
```

36. OFFLINE INSTRUMENT AND SIGNALRUN

a) Purpose

Explore the relationship between offline instrument and SignalRun.

b) Problem Statement

Place an offline instrument, and then place couple singalruns connected with it. Write a standalone application to navigate from offline-instrument to signalrun.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMInstrument objInstrument = datasource.GetInstrument(CONST_SPID_OfflineInstrument);
LMSignalRun objSignalRun = objInstrument.SignalRunObject;

Console.WriteLine("SignalRun SignalType  ==>  " +
       objSignalRun.Attributes["SignalType"].Value);
Console.WriteLine("\tTotal SignalRun.Instruments = " + objSignalRun.Instruments.Count);

foreach (LMInstrument objInstr in objSignalRun.Instruments)
       Console.WriteLine("Instrument InstrumentType  ==>  " +
              objInstr.Attributes["InstrumentType"].Value);

objInstrument = null;
objSignalRun = null;
datasource = null;
```

37. Access Instrumetn Functions

a) Purpose

Explore the relationship between instrument and its functions

b) Problem Statement

Place an instrument, and populate properties for its functions. Write a standalone application to access instrument functions.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMInstrument objInstrument = datasource.GetInstrument(CONST_SPID_OfflineInstrument);

Console.WriteLine("\tTotal Instrument Fail Modes = " +
       objInstrument.InstrFailModes.Count);

Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for
       Instrument.InstrFailMode");

foreach (LMInstrFailMode objInstrFailMode in objInstrument.InstrFailModes)
       foreach (LMAAttribute objAttr in objInstrFailMode.Attributes)
              Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
              Console.WriteLine("\tTotal Instrument Functions = " +
                     objInstrument.InstrFunctions.Count);

Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for
       Instrument.InstrFunction");

foreach (LMInstrFunction objInstrFunction in objInstrument.InstrFunctions)
       foreach (LMAAttribute objAttr in objInstrFunction.Attributes)
              Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
              Console.WriteLine("\tTotal Instrument Options = " +
                     objInstrument.InstrOptions.Count);

Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for Instrument.InstrOption");

foreach (LMInstrOption objInstrOption in objInstrument.InstrOptions)
       foreach (LMAAttribute objAttr in objInstrOption.Attributes)
              Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);

objInstrument = null;
datasource = null;
```

38. IDENTIFY CONNECTORS OF A PIPERUN

Purpose

To traverse the relationships from the Model DataModel to the Drawing DataModel

b) Problem Statement

Place a piperun between two nozzles and place two valves on it. Populate the ItemTag of the piperun with a value (eg. `01110-GCD`). Retrieve the piperun by filtering for the piperun's ItemTag and locate all of its representations and connector representations.

c) Solution

1. Dim LMPipeRun, LMConnector, LMRepresentation
2. LMConnector is subclass of LMRepresentation, and its RepresentationType is "Connector".

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();


LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "ItemTag";
criterion.Operator = "=";
criterion.ValueAttribute = "01100-GCD";

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PipeRun";
objFilter.Criteria.Add(criterion);

LMPipeRuns objPipeRuns = new LMPipeRuns();
objPipeRuns.Collect(datasource, null, "", objFilter);

Console.WriteLine("pipeRuns Number  ==>  " + objPipeRuns.Count);

LMConnector objConnector;
foreach (LMPipeRun objPiperun in objPipeRuns)
     foreach (LMRepresentation objRep in objPiperun.Representations)
            if (objRep.RepresentationType.ToString().Equals("Connector"))
            {
                   objConnector = datasource.GetConnector(representation.Id);
                   Console.WriteLine("Connector ItemStatus  ==>  " +
                          objConnector.Attributes["ItemStatus"].Value);
                   Console.WriteLine("Connector.ModelItemObject ItemTypeName  ==>  " +
                          objConnector.ModelItemObject.Attributes["ItemTypeName"].Value);
             }

 criterion = null;
 objFilter = null;
 objPipeRuns = null;
 objConnector = null;
 datasource = null;
```

39. Find File Nmae of a Symbol

a) Purpose

Find file name of a symbol

b) Problem Statement

Place a vessel, then navigate from vessel to symbol, and get the file name of the vessel from the symbol object.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

                LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);
                LMSymbol objSymbol = 
datasource.GetSymbol(objVessel.Representations.Nth(1).Id);

                Console.WriteLine("Vessel Symbol FileName  ==>  " + 
objSymbol.Attributes["FileName"].Value);

                objVessel = null;
                objSymbol = null;
                datasource = null;
```

40. Find X, Y Coordinates of Symbol

a) Purpose

Find X, Y Coordinates of symbol

b) Problem Statement

Place a vessel, then navigate from vessel to symbol, and get the X, Y Coordinates of the vessel from the symbol object.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);
LMSymbol objSymbol = datasource.GetSymbol(objVessel.Representations.Nth(1).Id);

Console.WriteLine("Vessel XCoordinate  ==>  " +
       objSymbol.Attributes["XCoordinate"].Value);
Console.WriteLine("Vessel YCoordinate  ==>  " +
       objSymbol.Attributes["YCoordinate"].Value);

objVessel = null;
datasource = null;
```

41. FIND X, Y COORDINATES OF PIPERUN

a) Purpose

Find X, Y Coordinates of Piperun

b) Problem Statement

Place a Piperun, then navigate from Piperun to Connector, and get the X, Y Coordinates of the Piperun from Connector object.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMConnector objConnector = null;
LMPipeRun objPiperun = datasource.GetPipeRun(CONST_SPID_PipeRun);

foreach (LMRepresentation objRep in objPiperun.Representations)
     if (objRep.Attributes["RepresentationType"].Value.Equals("Connector") &&
            objRep.Attributes["ItemStatus"].Value.Equals("Active"))
     {
            objConnector = datasource.GetConnector(objRep.Id);

            foreach (LMConnectorVertex objConnectorVertex in
                   objConnector.ConnectorVertices)
            {
                   Console.WriteLine("PipeRun ConnectorVertex XCoordinate  ==>  " +
                          objConnectorVertex.Attributes["XCoordinate"].Value);
                   Console.WriteLine("PipeRun ConnectorVertex YCoordinate  ==>  " +
                          objConnectorVertex.Attributes["YCoordinate"].Value);
            }
      }

 objConnector = null;
 objPiperun = null;;
 datasource = null;
```

42. Find Labels of a Symbol

a) Purpose

Find labels on a symbol

b) Problem Statement

Place a vessel, then place couple labels on it, then navigate from Vessel to Representation, then find labels on the Vessel.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);
LMSymbol objSymbol = datasource.GetSymbol(objVessel.Representations.Nth(1).Id);

Console.WriteLine("\tTotal LabelPersists found on this Symbol = " +
     objSymbol.LabelPersists.Count);

foreach (LMLabelPersist objLabelPersist in objSymbol.LabelPersists)
{
     Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for LabelPersist");

     foreach (LMAAttribute objAttr in objLabelPersist.Attributes)
            Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);

     foreach (LMLeaderVertex objLeaderVertex in objLabelPersist.LeaderVertices)
     {
            Console.WriteLine("LabelPersist LeaderVertex XCoordinate  ==>  " +
                   objLeaderVertex.Attributes["XCoordinate"].Value);
            Console.WriteLine("LabelPersist LeaderVertex YCoordinate  ==>  " +
                   objLeaderVertex.Attributes["YCoordinate"].Value);
      }
  }

  objVessel = null;
  objSymbol = null;
  datasource = null;
```

43. FIND PARENT REPRESENTATION OF A LABEL

a) Purpose

Find Parent Representation of a label.

b) Problem Statement

Place a label on to a vessel, get label object first, then navigate from label to find Representation it labels, then navigate from the Representation to ModelItem.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMLabelPersist objLabelPersist = datasource.GetLabelPersist(CONST_SPID_LabelPersist);
LMRepresentation objRep = objLabelPersist.RepresentationObject;

Console.WriteLine("\tTotal LabelPersists found on this Parent Representation = " +
     objRep.LabelPersists.Count);

LMModelItem objModelItem = objRep.ModelItemObject;

Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for ModelItem");

foreach (LMAAttribute objAttr in objModelItem.Attributes)
     Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);

objLabelPersist = null;
objRep = null;
objModelItem = null;
datasource = null;
```

44. FIND PARENT DRAWING FOR A SYMBOL

a) Purpose

Find parent drawing of a symbol.

b) Problem Statement

Place a vessel on a drawing. Write a standalone application to find drawing this vessel is on.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);
LMSymbol objSymbol = datasource.GetSymbol(objVessel.Representations.Nth(1).Id);

LMDrawing objDrawing = objSymbol.DrawingObject;
Console.WriteLine("Drawing Name  ==>  " + objDrawing.Attributes["Name"].Value);

objVessel = null;
objSymbol = null;
objDrawing = null;
datasource = null;
```

45. FIND DRAWING AND PLANTITEMS IN IT

a) Purpose

Directly find the drawing not through an item first, then find all PlantItems in it.

Problem Statement

Open a drawing. Write a standalone application to find what drawing is and how many PlantItems in it.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMDrawing objDrawing = datasource.GetDrawing(CONST_SPID_Drawing);
Console.WriteLine("Drawing Name  ==>  " + objDrawing.Attributes["Name"].Value);

LMACriterions criteria = new LMACriterions();
criteria.AddNew("1stOne");
criteria["1stOne"].SourceAttributeName = "Representation.Drawing.Name";
criteria["1stOne"].Operator = "=";
criteria["1stOne"].ValueAttribute = objDrawing.Attributes["Name"].Value;
objDrawing = null;
criteria.AddNew("2ndOne");
criteria["2ndOne"].SourceAttributeName = "ItemStatus";
criteria["2ndOne"].Operator = "=";
criteria["2ndOne"].ValueAttribute =1;
criteria["2ndOne"].Conjunctive = true;

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PlantItem";
objFilter.Criteria.Add(criteria["1stOne"]);
objFilter.Criteria.Add(criteria["2ndOne"]);

LMPlantItems objPlantItems = new LMPlantItems();
objPlantItems.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal filtered PlantItems in the drawing = " +
       objPlantItems.Count);

criteria = null;
objFilter = null;
objPlantItems = null;
datasource = null;
```

46. FILTER FOR ITEMS IN PLANT STOCKPILE

a) Purpose

To filter for all items in plant stockpile.

b) Problem Statement

Write a standalone application to get all items in plant stockpile.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterions criteria = new LMACriterions();
criteria.AddNew("1stOne");
criteria["1stOne"].SourceAttributeName = "ItemStatus";
criteria["1stOne"].Operator = "=";
criteria["1stOne"].ValueAttribute = 1;
criteria.AddNew("2ndOne");
criteria["2ndOne"].SourceAttributeName = "Representation.InStockpile";
criteria["2ndOne"].Operator = "=";
criteria["2ndOne"].ValueAttribute = 2; // 2 is an index, it stands for True
criteria["2ndOne"].Conjunctive = true;
criteria.AddNew("3rdOne");
criteria["3rdOne"].SourceAttributeName = "Representation.SP_DrawingId";
criteria["3rdOne"].Operator = "=";
criteria["3rdOne"].ValueAttribute = 0; // zero stands for Plant Stockpile
criteria["3rdOne"].Conjunctive = true;

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "Vessel";
objFilter.Criteria.Add(criteria["1stOne"]);
objFilter.Criteria.Add(criteria["2ndOne"]);
objFilter.Criteria.Add(criteria["3rdOne"]);

LMVessels objVessels = new LMVessels();
objVessels.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal Vessels filtered = " + objVessels.Count);

foreach (LMVessel objVessel in objVessels)
{
    Console.WriteLine("Vessel ItemTag  ==>  " + objVessel.Attributes["ItemTag"].Value);
    Console.WriteLine("Vessel.Representations.Nth[1] InStockpile  ==>  " +
            objVessel.Representations.Nth(1).Attributes["InStockpile"].Value);
}

objFilter = null;
criteria = null;
objVessels = null;
datasource = null;
```

47. IDENTIFY ITEMS CONNECTED TO A PIPERUN

Purpose

To traverse the relationships from LMConnector to LMSymbol

b) Problem Statement

Place a piperun between two nozzles and place two valves on it. Populate the ItemTag of the piperun with a value (eg. unit1100-GCD). Retrieve the piperun by filtering for the piperun's ItemTag. Identify all of the items connected to the ends of the connectors of the piperun.

c) Solution

1. Dim LMPipeRun, LMConnector, LMRepresentation
2. LMConnector has properties "`ConnectItem1SymbolObject`" and "`ConnectItem2SymbolObject`", that returns the symbol object connected to the Connector

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "ItemTag";
criterion.Operator = "=";
criterion.ValueAttribute = "01110-GCD"

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PipeRun";
objFilter.Criteria.Add(criterion);

LMPipeRuns objPipeRuns = new LMPipeRuns();
objPipeRuns.Collect(datasource, null, "", objFilter);

LMConnector objConnector = null;
foreach (LMPipeRun objPiperun in objPipeRuns)
     foreach (LMRepresentation objRep in objPiperun.Representations)
            if (objRep.RepresentationType.Equals("Connector"))
            {
                   objConnector = datasource.GetConnector(objRep.Id);
                   if (objConnector.ConnectItem1SymbolObject != null)
                   {
                          Console.WriteLine("Connector.ConnectItem1SymbolObject
                          .ModelItemObject.ItemTypeName = " +
                          objConnector.ConnectItem1SymbolObject.ModelItemObject
                          .ItemTypeName);

                          Console.WriteLine("Connector.ConnectItem1SymbolObject
                          .ModelItemID = " +
                          objConnector.ConnectItem1SymbolObject.ModelItemID);
                    }
                    if (objConnector.ConnectItem2SymbolObject != null)
                    {

                          Console.WriteLine("Connector.ConnectItem2SymbolObject
                          .ModelItemObject.ItemTypeName = " +
                          objConnector.ConnectItem2SymbolObject.ModelItemObject
                          .ItemTypeName);

                          Console.WriteLine("Connector.ConnectItem2SymbolObject
                          .ModelItemID = " +
                          objConnector.ConnectItem2SymbolObject.ModelItemID);
                     }
               }

criterion = null;
objFilter = null;
objPipeRuns = null;
objConnector = null;
datasource = null;
```

48. Identify the PipeRun associated with the PipingComp

a) Purpose

To traverse the relationships from LMPipingComp to LMPipeRun

b) Problem Statement

Place a piperun, then place a valve in the middle of the piperun. Assume you only know the SP_ID of the valve. Write a standalone application to obtain the PipeRun on which the valve is sitting, then read properties (ID and Name) of the piperun.

Solution

1. Dim LMPipingComp, LMSymbol, LMPipeRun
2. LMSymbol has a property "Connect1Connectors", that returns the collection of LMConnector object connected to the Symbol, then from LMConnector.ModelItemID, returns the ModelItemID of the Connector, which is the SP_ID of the PipeRun.
3. Alternate, LMPipingComp has method "InlineComps", which returns the collection of LMInlineComps associated with the PipingComp, then, LMInlineComp has a property "PipeRunID", which returns the SP_ID of the PipeRun, on which the PipingComp is sitting.

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMPipingComp objPipingComp = datasource.GetPipingComp(CONST_SPID_PipingComp);
LMSymbol objPipingCompSym =
     datasource.GetSymbol(objPipingComp.Representations.Nth(1).Id);

LMPipeRun objPiperun =
     datasource.GetPipeRun(objPipingCompSym.Connect1Connectors.Nth(1).ModelItemID);

Console.WriteLine("PipeRun.Id = " + objPiperun.Id);
Console.WriteLine("PipeRun ItemTag  ==>  " + objPiperun.Attributes["ItemTag"].Value);

Console.WriteLine("PipingComp.Id = " + objPipingComp.Id);
Console.WriteLine("PipeRun.InlineComps.Nth(1).Id = " +
     objPiperun.InlineComps.Nth(1).Id);

objPipingComp = null;
objPipingCompSym = null;
objPiperun = null;
datasource = null;
```

49. Navigate Items to Get Parent Item

Purpose

To navigate items such as PipeRun, Connectors, nozzles, to get the parent item of nozzle – Equipment.

b) Problem Statement

Place a vessel with a nozzle on it, then place a piperun connected to the nozzle, then place a valve in the piperun. Write a standalone application to navigate from the piperun, through the piperun's connectors and the nozzle to arrive at the vessel. Print out some properties of the vessel.

c) Solution

1. Dim LMAFilter, LMACriterion, LMPipeRuns
2. From LMPipeRun.Representations, obtain LMConnector, whose RepresentationType is "Connector", then, from LMConnector.ConnnectItem1SymbolObject or LMConnector.ConnecItem2SymbolObject find the Symbol object connect to the Connector, then, from LMSymbol.ModelItemID find the SP_ID of the symbol, then the Nozzle object is located, and LMNozzle has a property "EquipmentObject", which returns the LMEquipment object, which is connected to the Nozzle

◊ **Example code**

Dim datasource As LMADataSource

```
LMADataSource datasource = new LMADataSource();
LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "ItemTag";
criterion.Operator = "=";
criterion.ValueAttribute = "01110-GCD";

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "PipeRun";
objFilter.Criteria.Add(criterion);

LMPipeRuns objPipeRuns = new LMPipeRuns();
objPipeRuns.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal PipeRuns filtered = " + objPipeRuns.Count);

LMConnector objConnector = null;
LMNozzle objNozzle = null;
foreach (LMPipeRun objPiperun in objPipeRuns)
     foreach (LMRepresentation objRep in objPiperun.Representations)
            if (objRep.RepresentationType.Equals("Connector"))
            {
                   objConnector = datasource.GetConnector(objRep.Id);
                   if (objConnector.ConnectItem1SymbolObject != null)
                          if (objConnector.ConnectItem1SymbolObject
                          .ModelItemObject.ItemTypeName.Equals("Nozzle"))
                          {
                                 objNozzle = datasource.GetNozzle(objConnector
                                 .ConnectItem1SymbolObject.ModelItemID);
                                   break;
                          }
```

```
                         if (objConnector.ConnectItem2SymbolObject != null)
                               if (objConnector.ConnectItem2SymbolObject
                                  .ModelItemObject.ItemTypeName.Equals("Nozzle"))
                              {
                                objNozzle = datasource.GetNozzle(objConnector
                               .ConnectItem2SymbolObject.ModelItemID);
                                break;
                              }
             }

   // arrive at the parent object
   LMEquipment objEquipment = objNozzle.EquipmentObject;

   Console.WriteLine("Equipment.Id = " + objEquipment.Id);
   Console.WriteLine("Equipment.EquipmentType = " + objEquipment.EquipmentType);
   Console.WriteLine("Equipment ItemTag  ==>  " +
       objEquipment.Attributes["ItemTag"].Value);
   Console.WriteLine("Equipment.Nozzles.Count = " +
       objEquipment.Nozzles.Count);

   criterion = null;
   objFilter = null;
   objConnector = null;
   objPipeRuns = null;
   objNozzle = null;
   objEquipment = null;
   datasource = null;
```

50. Navigate through BranchPoint

Purpose

To navigate through branch point on a piperun.

b) Problem Statement

Place a vessel with two nozzles on it with ItemTags N10 and N20 respectively. Generate the itemtag for the vessel by assigning a TagPrefix. Place a straight piperun starting from the nozzle (N10) and end it in space with no connection. Start a branch piperun from some point on the first piperun, and extend it to the second nozzle (N20). Write a standalone application to navigate from the first nozzle (N10), through the piperun connectors and the second nozzle to arrive back at the vessel.

c) Solution

1. Dim LMAFileter, LMACriterion, LMSymbol, LMPipeRun
2. BranchPoint is a Symbol Representation of the PipeRun on which it is sitting, BranchPoint's RepresentationType is "Branch"

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "ItemTag";
criterion.Operator = "=";
criterion.ValueAttribute = "N10";

// add the criterion to the filter
LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "Nozzle";
objFilter.Criteria.Add(criterion);

LMNozzles objNozzles = new LMNozzles();
objNozzles.Collect(datasource, null, "", objFilter);

LMNozzle objNozzle = null;
if (objNozzles.Count == 1)
     objNozzle = objNozzles.Nth(1);
else
{
     objFilter = null;
     objNozzles = null;
     objNozzle = null;
     datasource = null;
     return;
 }

 LMSymbol objSymbol = datasource.GetSymbol(objNozzle.Representations.Nth(1).Id);
```

```
  // check nozzle symbol's connect1connectors & connect2connectors information to find a
  connector connected to the nozzle
  LMConnector objConnector = null;
       if (objSymbol.Connect1Connectors.Count >= 1)
              foreach (LMConnector objTconnector in objSymbol.Connect1Connectors)
                     if (objTconnector.ItemStatus.Equals("Active"))
                            if (objTconnector.ModelItemObject
                                   .ItemTypeName.Equals("PipeRun"))
                                objConnector = objTconnector;
       if (objConnector == null && objSymbol.Connect2Connectors.Count >= 1)
              foreach (LMConnector objTconnector in objSymbol.Connect2Connectors)
                     if (objTconnector.ItemStatus.Equals("Active"))
                            if (objTconnector.ModelItemObject
                                   .ItemTypeName.Equals("PipeRun"))
                                objConnector = objTconnector;

// once the connector is found, check connectitem1symbolobject and
connectitem2symbolobject information to find the BranchPoint// the modelitem for the
BranchPoint symbol is the piperun, but the representation type is "Branch"
LMSymbol objBranchSym = null;
if (objConnector.ConnectItem1SymbolObject != null)
       if (objConnector.ConnectItem1SymbolObject
              .ModelItemObject.ItemTypeName.Equals("PipeRun"))
              if (objConnector.ConnectItem1SymbolObject
                     .AsLMRepresentation().RepresentationType.Equals("Branch"))
                            objBranchSym = objConnector.ConnectItem1SymbolObject;

if (objBranchSym == null && objConnector.ConnectItem2SymbolObject != null)
       if (objConnector.ConnectItem2SymbolObject
              .ModelItemObject.ItemTypeName.Equals("PipeRun"))
              if (objConnector.ConnectItem2SymbolObject
                     .AsLMRepresentation().RepresentationType.Equals("Branch"))
                            objBranchSym = objConnector.ConnectItem2SymbolObject;

// after the BranchPoint is located, again use the connect1connectors &
connect2connectors method to locate the connector connected to the BranchPoint
// make sure this connector is pointing back to the new piperun
LMConnector objConnector3 = null;
if (objBranchSym.Connect1Connectors.Count >= 1)
       foreach (LMConnector objConnector2 in objBranchSym.Connect1Connectors)
              if (!objConnector2.ModelItemID.Equals(objConnector.ModelItemID))
              {
                     objConnector3 = objConnector2;
                     break;
              }
 if (objConnector3 == null && objBranchSym.Connect2Connectors.Count >= 1)
       foreach (LMConnector objConnector2 in objBranchSym.Connect2Connectors)
              if (!objConnector2.ModelItemID.Equals(objConnector.ModelItemID))
              {
                     objConnector3 = objConnector2;
                     break;
  }
```

```
// after second connector is located, check connectitem1symbolobject &
connectitem2symbolobject to find the second nozzle
LMNozzle objNozzle2 = null;
if (objConnector3.ConnectItem1SymbolObject != null)
     if (objConnector3.ConnectItem1SymbolObject
            .ModelItemObject.ItemTypeName.Equals("Nozzle"))
            objNozzle2 = datasource.GetNozzle(objConnector3
                          .ConnectItem1SymbolObject.ModelItemID);
            if (objNozzle2 == null && objConnector3.ConnectItem2SymbolObject != null)
                  if (objConnector3.ConnectItem2SymbolObject
                          .ModelItemObject.ItemTypeName.Equals("Nozzle"))
                      objNozzle2 = datasource.GetNozzle(objConnector3
                                        .ConnectItem2SymbolObject.ModelItemID);

 // print out two nozzles' name and ItemTag of the vessel to which they are attached
 Console.WriteLine("Nozzle2 ItemTag  ==>  " + objNozzle2.Attributes["ItemTag"].Value);
 Console.WriteLine("Nozzle ItemTag  ==>  " + objNozzle.Attributes["ItemTag"].Value);
 Console.WriteLine("Nozzle2.EquipmentObject ItemTag  ==>  " +
     objNozzle2.EquipmentObject.Attributes["ItemTag"].Value);
 Console.WriteLine("Nozzle.EquipmentObject ItemTag  ==>  " +
     objNozzle.EquipmentObject.Attributes["ItemTag"].Value);

 criterion = null;
 objFilter = null;
 objSymbol = null;
 objBranchSym = null;
 objConnector = null;
 objNozzles = null;
 objNozzle = null;
 objNozzle2 = null;
 objConnector3 = null;
 datasource = null;
```

51. Navigate Through OPC

Purpose

To get familiar with navigation through OPC

b) Problem Statement

Place an OPC, then place its pair OPC into another drawing, and connected the pair OPC to a piperun with itemtag populated. Then write a standalone application to navigate for OPC to its pairOPC, and print out the itemtag of piperun that the pair OPC is connected with.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "Representation.Drawing.Name";
criterion.Operator = "=";
criterion.ValueAttribute = "OPC";

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "OPC";
objFilter.Criteria.Add(criterion);

LMOPCs objOPCs = new LMOPCs();
objOPCs.Collect(datasource, null, "", objFilter);

Console.WriteLine("\tTotal OPCs filtered = " + objOPCs.Count);

LMOPC objpairOPC = null;
LMPipeRun objPiperun = null;
LMSymbol objSym = null;

foreach (LMOPC objOPC in objOPCs)
{
       objpairOPC = objOPC.pairedWithOPCObject;
       foreach (LMRepresentation objRep in objpairOPC.Representations)
       {
              if (objRep.DrawingID.GetHashCode() > 0)
                     Console.WriteLine("pairedOPC Representation.DrawingObject Name  ==>
                            " + objRep.DrawingObject.Attributes["Name"].Value);
                     objSym = datasource.GetSymbol(objRep.Id);

              foreach (LMConnector objConnector in objSym.Connect1Connectors)
              {
                     objPiperun = datasource.GetPipeRun(objConnector.ModelItemID);
                     Console.WriteLine("pairedOPC PipeRun ItemTag  ==>
                            " + objPiperun.Attributes["ItemTag"].Value);
              }

              foreach (LMConnector objConnector in objSym.Connect2Connectors)
              {
                     objPiperun = datasource.GetPipeRun(objConnector.ModelItemID);
```

```
                    Console.WriteLine("pairedOPC PipeRun ItemTag  ==>
                           " + objPiperun.Attributes["ItemTag"].Value);
              }
       }
}

criterion = null;
objFilter = null;
objOPCs = null;
objpairOPC = null;
objPiperun = null;
objSym = null;
datasource = null;
```

52. ACCESS RELATIONSHIP FROM REPRESENTATION

a) Purpose

To access relationship object from representation object.

b) Problem Statement

Place piperun, then place a valve on the piperun. Write a standalone application to obtain the valve, then get the relationship objects belong to this valve.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMPipingComp objPipingComp = datasource.GetPipingComp(CONST_SPID_PipingComp);

LMRepresentation objRep = objPipingComp.Representations.Nth(1);

foreach (LMRelationship objRelationship in objRep.Relation1Relationships)
{
       if (objRelationship.Item1RepresentationObject != null)
              Console.WriteLine("\tPipingComp end 1:  Item 1 ItemType = " +
       objRelationship.Item1RepresentationObject.ModelItemObject.AsLMAItem().ItemType);
       if (objRelationship.Item2RepresentationObject != null)
              Console.WriteLine("\tPipingComp end 1:  Item 2 ItemType = " +
       objRelationship.Item2RepresentationObject.ModelItemObject.AsLMAItem().ItemType);
              Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for
       Relationship on end 1 of PipingComp");

       foreach (LMAAttribute attrib in objRelationship.Attributes)
              Console.WriteLine(attrib.Name + "  ==>  " + attrib.Value);
}

foreach (LMRelationship objRelationship in objRep.Relation2Relationships)
{
       if (objRelationship.Item1RepresentationObject != null)
              Console.WriteLine("\tPipingComp end 2:  Item 1 ItemType = " +
       objRelationship.Item1RepresentationObject.ModelItemObject.AsLMAItem().ItemType);
       if (objRelationship.Item2RepresentationObject != null)
              Console.WriteLine("\tPipingComp end 2:  Item 2 ItemType = " +
       objRelationship.Item2RepresentationObject.ModelItemObject.AsLMAItem().ItemType);
              Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for
       Relationship on end 2 of PipingComp");

        foreach (LMAAttribute attrib in objRelationship.Attributes)
              Console.WriteLine(attrib.Name + "  ==>  " + attrib.Value);
}

objPipingComp = null;
objRep = null;
datasource = null;
pidAutoDwg = null;
pidAutoApp = null;
```

53. Access Inconsistency

a) Purpose

To access the Inconsistency.

b) Problem Statement

Write a standalone application to get all relationship objects belong to a drawing, then access the Inconsistency from relationship.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "Name";
criterion.Operator = "=";
criterion.ValueAttribute = "automation";

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "Drawing";
objFilter.Criteria.Add(criterion);

LMDrawings objDrawings = new LMDrawings();
objDrawings.Collect(datasource, null, "", objFilter);

foreach (LMDrawing objDrawing in objDrawings)
       foreach (LMRelationship objRelationship in objDrawing.Relationships)
              foreach (LMInconsistency objInconsistency in
                            objRelationship.Inconsistencies)
              {
                     Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'
                                          for Relationship.Inconsistency");
                     foreach (LMAAttribute objAttr in objInconsistency.Attributes)
                            Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
              }

criterion = null;
objFilter = null;
objDrawings = null;
datasource = null;
```

54. ACCESS RULEREFERENCE

a) Purpose

To access the RuleReference.

b) Problem Statement

Write a standalone application to get all relationship objects belong to a drawing, then access the RuleReference from relationship.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "Name";
criterion.Operator = "=";
criterion.ValueAttribute = "automation";

LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "Drawing";
objFilter.Criteria.Add(criterion);

LMDrawings objDrawings = new LMDrawings();
objDrawings.Collect(datasource, null, "", objFilter);

foreach (LMDrawing objDrawing in objDrawings)
       foreach (LMRelationship objRelationship in objDrawing.Relationships)
              foreach (LMRuleReference objRuleReference in objRelationship.RuleReferences)
              {
                     Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for
                            Relationship.RuleReference");
                     foreach (LMAAttribute objAttr in objRuleReference.Attributes)
                            Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
              }

criterion = null;
objFilter = null;
objDrawings = null;
datasource = null;
```

55. Access PlantGroup from PlantItem

Purpose

To access the PlantGroup to which the PlantItem belongs.

b) Problem Statement

Place a vessel. Write a standalone application to get the plantgroup to which the PlantItem is associated.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);

LMPlantGroup objPlantGroup = objVessel.PlantGroupObject;

Console.WriteLine("PlantGroup Name  ==>  " + objPlantGroup.Attributes["Name"].Value);

String strParentID = objPlantGroup.Attributes["ParentID"].Value as String;

LMPlantGroup objParentPlantGroup = datasource.GetPlantGroup(strParentID);

Console.WriteLine("ParentPlantGroup Name  ==>  " +
objParentPlantGroup.Attributes["Name"].Value);

objVessel = null;
objPlantGroup = null;
objParentPlantGroup = null;
```

56. Access PlantGroup from Drawing

a) Purpose

To access the PlantGroup to which the Drawing belongs.

b) Problem Statement

Place a vessel. Write a standalone application to obtain the drawing associated with the vessel. Get the plantgroup to which the drawing belongs.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);

LMPlantGroup objPlantGroup =
objVessel.Representations.Nth(1).DrawingObject.PlantGroupObject;

Console.WriteLine("PlantGroup Name  ==>  " + objPlantGroup.Attributes["Name"].Value);

LMPlantGroup objParentPlantGroup =
datasource.GetPlantGroup(objPlantGroup.Attributes["ParentID"].Value.ToString());

Console.WriteLine("ParentPlantGroup Name  ==>  " +
objParentPlantGroup.Attributes["Name"].Value);

objVessel = null;
objPlantGroup = null;
objParentPlantGroup = null;
datasource = null;
```

57. Access Customized PlantGroup

a) Purpose

To access the customized property of a user defined PlantGroup type.

b) Problem Statement

Create a new PlantGroup type, “SubArea”, in Smart Engineering Manager, then create a new Hierarcy template using this new PlantGroup. Then create a new plant using this new Hierarcy template, after creation of new plant, add a new property “T1” to the new PlantGroup. Write a standalone application to read this new property “T1”.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);
string SubArea = "SPMSubArea";
Console.WriteLine("Vessel.PlantGroupObject 'T1'  ==>  " +
objVessel.PlantGroupObject.Attributes["T1"].Value);
Console.WriteLine("DataSource.GetItem 'SPMSubArea' of Vessel.PlantGroupID 'T1'  ==>  "
+ datasource.GetItem(ref SubArea, objVessel.PlantGroupID).Attributes["T1"].Value);

objVessel = null;
datasource = null;
datasource = null;
```

58. ACCESS WORKSHARE STIE

a) Purpose

To get familiar with the workshare site object in LLAMA.

b) Problem Statement

Place a Vessel, find the workshare site to which this vessel belongs. Print out properties of the workshare site. Browser relateionship between workshare site and other entities, such as PlantGroup, PlantItemGroup, OPC, and DrawingSite.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);
LMDrawing objDrawing = datasource.GetDrawing(objVessel.Representations.Nth(1).DrawingID);

// get the PlantGroup just above the drawing, in our case, should be the Unit
LMPlantGroup objPlantGroup = objDrawing.PlantGroupObject;

LMWSSite objWSSite = objPlantGroup.WSSiteObject;

Console.WriteLine("\tTotal WsSite attributes = " + objWSSite.Attributes.Count);
// loop through the workshare site attributes
Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for WsSite");
foreach (LMAAttribute objAttr in objWSSite.Attributes)
Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
Console.WriteLine("\tTotal WsSite.OPCs = " + objWSSite.OPCs.Count);
// loop through the workshare site OPCs
foreach (LMOPC objOPC in objWSSite.OPCs)
{
       Console.WriteLine("OPC Tag  ==>  " + objOPC.Attributes["OPCTag"].Value);
       Console.WriteLine("OPC.WsSiteObject Name  ==>  " +
              objOPC.WSSiteObject.Attributes["Name"].Value);
}

Console.WriteLine("\tTotal WsSite.PlantItemGroups = " + objWSSite.PlantItemGroups.Count);
// loop through the workshare site plant item groups
foreach (LMPlantItemGroup objPlantItemGroup in objWSSite.PlantItemGroups)
{
       Console.WriteLine("PlantItemGroupType  ==>  " +
              objPlantItemGroup.Attributes["PlantItemGroupType"].Value);
       Console.WriteLine("PlantItemGroup.WsSiteObject Name  ==>  " +
              objPlantItemGroup.WSSiteObject.Attributes["Name"].Value);
}

Console.WriteLine("\tTotal WsSite.PlantGroups = " + objWSSite.PlantGroups.Count);// loop
through the workshare site plant groups
foreach (LMPlantGroup oPlantGroup in objWSSite.PlantGroups)
{
       Console.WriteLine("PlantGroupType  ==>  " +
              oPlantGroup.Attributes["PlantGroupType"].Value);
       Console.WriteLine("PlantGroup Name  ==>  " + oPlantGroup.Attributes["Name"].Value);
       Console.WriteLine("PlantGroup.WsSiteObject Name  ==>  " +
              oPlantGroup.WSSiteObject.Attributes["Name"].Value);
}

Console.WriteLine("\tTotal WsSite.DrawingSites = " + objWSSite.DrawingSites.Count);
// loop through the workshare site drawing sites
foreach (LMDrawingSite objDrawingSite in objWSSite.DrawingSites)
{
       Console.WriteLine("DrawingSite name  ==>  " +
              objDrawingSite.Attributes["Name"].Value);
       Console.WriteLine("DrawingSite.WsSiteObject Name  ==>  " +
              objDrawingSite.WSSiteObject.Attributes["Name"].Value);
}

objVessel = null;
objDrawing = null;
objPlantGroup = null;
objWSSite = null
```

59. ACCESS DRAWINGSITE

a) *Purpose*

To get familiar with the drawingsite object in LLAMA.

b) *Problem Statement*

Get a drawingsite object, the print out properties of the drawingsite.

c) *Solution*

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);
LMDrawing objDrawing = datasource.GetDrawing(objVessel.Representations.Nth(1).DrawingID);

LMDrawingSite objDrawingSite = objDrawing.DrawingSites.Nth(1);

Console.WriteLine("\tTotal DrawingSite attributes = " + objDrawingSite.Attributes.Count);

Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for DrawingSite");
foreach (LMAAttribute objAttr in objDrawingSite.Attributes)
       Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
       Console.WriteLine("\tTotal DrawingSubscribers = " +
              objDrawingSite.DrawingSubscribers.Count);

foreach (LMDrawingSubscriber objDrawingSubscriber in objDrawingSite.DrawingSubscribers)
{
       Console.WriteLine("DrawingSubscriber.DrawingSiteObject Name  ==>  " +
              objDrawingSubscriber.DrawingSiteObject.Attributes["Name"].Value);
       Console.WriteLine("DrawingSubscriber.WsSiteObject Name  ==>  " +
              objDrawingSubscriber.WSSiteObject.Attributes["Name"].Value);
}

Console.WriteLine("DrawingSite.DrawingObject Name  ==>  " +
       objDrawingSite.DrawingObject.Attributes["Name"].Value);
Console.WriteLine("DrawingSite.WsSiteObject Name  ==>  " +
       objDrawingSite.WSSiteObject.Attributes["Name"].Value);

if (objDrawingSite.ToWSSiteWSSiteObject != null)
       Console.WriteLine("DrawingSite.ToWsSiteWsSiteObject Name  ==>  " +
              objDrawingSite.ToWSSiteWSSiteObject.Attributes["Name"].Value);
Console.WriteLine("DrawingSite.PlantGroupObject Name  ==>  " +
       objDrawingSite.PlantGroupObject.Attributes["Name"].Value);

objDrawing = null;
objVessel = null;
objDrawingSite = null;
```

60. Workshare Awareness in LLAMA

a) *Purpose*

To check out the workshare awareness in LLAMA.

b) *Problem Statement*

Set a satellite site as active project, then access a vessel in a drawing which is read-only for this satellite site, try to modify the property of the vessel and commit to database. See what happens?

c) *Solution*

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMADataSource datasource = new LMADataSource();

LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);

try
{
       datasource.BeginTransaction();
       Console.WriteLine("Vessel Name  ==>  " + objVessel.Attributes["Name"].Value);

       objVessel.Attributes["Name"].Value = "InWorkshare";
       objVessel.Commit();
       datasource.CommitTransaction();
}

catch (Exception ex)
{
MessageBox.Show(ex.Message.ToString());
}

objVessel = null;
datasource = null;
```

61. Access Active Project

a) Purpose

To access the active project

b) Problem Statement

Set The Plant or one of projects as active project, then use LMADatasource.GetActiveProject to obtain the active project. Then, print out all attributions of the active project, pay attention to the Project Status.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMActiveProject objActiveProject = datasource.GetActiveProject();

Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for ActiveProject");
foreach (LMAAttribute objAttr in objActiveProject.Attributes)
       Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);

objActiveProject = null;
datasource = null;
```

62. How to Access Plant from Project

a) Purpose

When user is in a project, how to find the project belongs to which The Plant?

b) Problem Statement

Set one of the projects as active project, then try to find The Plant.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMPlantGroups objPlantGroups = new LMPlantGroups();
objPlantGroups.Collect(datasource);

Console.WriteLine("\tTotal PlantGroups = " + objPlantGroups.Count);

foreach (LMPlantGroup objPlantGroup in objPlantGroups)
{
       Console.WriteLine("'ATTRIBUTE NAME'  ==>  'ATTRIBUTE VALUE'  for PlantGroup");
       foreach (LMAAttribute objAttr in objPlantGroup.Attributes)
       {
              Console.WriteLine(objAttr.Name + "  ==>  " + objAttr.Value);
              if (objAttr.Name.Equals("Depth") && objAttr.Value.Equals(0))
                     MessageBox.Show("\tPlantGroup ThePlant Name  ==>  " +
                            objPlantGroup.Attributes["Name"].Value);
       }
}

objPlantGroups = null;
datasource = null;
```

63. Access Claim Status of Items

a) Purpose

To access the items’ claim status.

b) Problem Statement

Set items in different claim status, and access claim status by using function LMADatasource.GetModelItemClaimStatus

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMVessel objVessel = datasource.GetVessel(CONST_SPID_Vessel);

Console.WriteLine("DataSource.GetModelItemClaimStatus of Vessel = " +
datasource.GetModelItemClaimStatus(objVessel.AsLMAItem()));

objVessel = null;
datasource = null;
```

64. ACCESS OPTIONSETTINGS

Purpose

To access the OptionSetting by Filter, and read value of OptionSetting

b) Problem Statement

Write a standalone application to obtain optionsetting (Default Assembly Path) by filter and read the value of the optionsetting.

c) Solution

1. Dim LMAFilter, LMACriterion, LMOptionSetting
2. LMOptionSetting is a independent object, which does not has any relationship with other objects in Data Model. To access LMOptionSetting, users need to know exactly what they are looking for, for example, in optionsettings, where is the “Default Assembly Path” ?

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "Name";
criterion.Operator = "=";
criterion.ValueAttribute = "Default Assembly Path";
LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "OptionSetting";
objFilter.Criteria.Add(criterion);

LMOptionSettings objOptionSettings = new LMOptionSettings();
objOptionSettings.Collect(datasource, null, "", objFilter);

LMOptionSetting objOptionSetting = objOptionSettings.Nth(1);

Console.WriteLine("Default Assembly Path Name  ==>  " +
                     objOptionSetting.Attributes["Name"].Value);
Console.WriteLine("Default Assembly Path Value  ==>  " +
                     objOptionSetting.Attributes["Value"].Value);

criterion = null;
objFilter = null;
objOptionSettings = null;
objOptionSetting = null;
```

65. Create a Vessel and Place into Stockpile

Purpose

Use PIDCreateItem method to create a vessel in the stockpile

b) Problem Statement

Write a standard executable to create a vessel and place it in the stockpile.

c) Solution

◊ **Open the Smart P&ID drawing.**

1. Create a drawing through SPManager.
2. Double-click on the drawing to open up Smart P&ID

◊ **Create a standard executable VB project**

3. Select a standard exe project
4. Reference the “Logical Model Automation” and “Placement Automation” libraries

◊ **Add code to place a vessel into stockpile**

5. Use the Function Function **PIDCreateItem**(DefinitionFile As String) As **LMAItem**
6. Provide the DefinitionFile string indicating the location of the symbol on a server
7. Use the return value to future reference.

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement;
objPlacement = new Placement();

//create a vessel into stockpile;
String strVesselDef = @"\Equipment\Vessels\Horizontal Drums\Horz Surge w-Horiz Dea.sym";

LMAItem objItem = objPlacement.PIDCreateItem(strVesselDef);

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

pidAutoApp = null;
pidAutoDwg = null;
objPlacement = null;
objItem = null;
```

66. Place a Vessel on a Drawing

Purpose

Use the PidPlaceSymbol method to place a vessel on a drawing

b) Problem Statement

Write a standard executable to place a vessel on a drawing.

c) Solution

◊ **Open the Smart P&ID drawing.**

1. Create a drawing through SPManager.
2. Double-click on the drawing to open up Smart P&ID

◊ **Create a standard executable VB project**

3. Select a standard exe project
4. Reference the “Logical Model Automation” and “Placement Automation” libraries

◊ **Add code to place a vessel**

5. Use the method Function **PIDPlaceSymbol**(DefinitionFile As String, X As Double, Y As Double, [Mirror], [Rotation], [ExistingItem As LMAItem], [TargetItem]) As **LMSymbol**
6. Provide the DefinitionFile string indicating the location of the symbol on a server
7. Provide the X and Y coordinates of the placement on the drawing.
8. Use the return value to future reference.

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement;
objPlacement = new Placement();

String strVesselDef = @"\Equipment\Vessels\Horizontal Drums\Horz Surge w-Horiz Dea.sym";
LMSymbol objSymbol = objPlacement.PIDPlaceSymbol(strVesselDef, 0.3, 0.2);

if (objSymbol.Equals(null))
       MessageBox.Show("Placement unsuccessful");

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

pidAutoDwg = null;
pidAutoApp = null;
objPlacement = null;
objSymbol = null;
```

67. PLACE NOZZLES AND TRAYS ON A VESSEL

Purpose

Use PIDPlaceSymbol method to place equipment components on a vessel

b) Problem Statement

Write a standard executable to place nozzles and trays on a vessel.

c) Solution

◊ **Open the Smart P&ID drawing.**

1. Create a drawing through SPManager.
2. Double-click on the drawing to open up Smart P&ID

◊ **Create a standard executable VB project**

3. Select a standard exe project
4. Reference the “Logical Model Automation” and “Placement Automation” libraries

◊ **Add code to place a vessel**

5. Use the Function **PIDPlaceSymbol**(DefinitionFile As String, X As Double, Y As Double, [Mirror], [Rotation], [ExistingItem As LMAItem], [TargetItem]) As **LMSymbol**
6. Provide the DefinitionFile string indicating the location of the symbol on a server
7. Provide the X and Y coordinates of the placement on the drawing.
8. Provide the TargetItem as an LMAItem.
9. Use the return value to future reference.
10. Repeat place nozzles while set **PIDSnapToTarget** to TRUE

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();

String strVesselDef = @"\Equipment\Vessels\Vertical Drums\1D 1C 2to1.sym";
Double dblVesselX = 0.3;
Double dblVesselY = 0.2;
LMSymbol symVessel = objPlacement.PIDPlaceSymbol(strVesselDef, dblVesselX, dblVesselY);
LMVessel objVessel = objPlacement.PIDDataSource.GetVessel(symVessel.ModelItemID);

objVessel.Attributes["CleaningReqmts"].Value = "CC1";
objVessel.commit();

// place two nozzles on the vessel
String strNozzleDef = @"\Equipment Components\Nozzles\Flanged Nozzle with blind.sym";

LMSymbol symNozzle1 = objPlacement.PIDPlaceSymbol(strNozzleDef, dblVesselX - 0.2,
dblVesselY + 0.05, null, null, null, symVessel.AsLMRepresentation());
LMSymbol symNozzle2 = objPlacement.PIDPlaceSymbol(strNozzleDef, dblVesselX + 0.2,
dblVesselY + 0.07, null, null, null,symVessel.AsLMRepresentation());


// place two trays on the vessel
String strTrayDef = @"\Equipment Components\Trays\Bubble Cap Trays\2-Pass Bubl Side.sym";
```

```
LMSymbol symTray1 = objPlacement.PIDPlaceSymbol(strTrayDef, dblVesselX - 0.05,
dblVesselY + 0.05, null, null, null, symVessel.AsLMRepresentation());
LMSymbol symTray2 = objPlacement.PIDPlaceSymbol(strTrayDef, dblVesselX + 0.05,
dblVesselY + 0.1, null, null, null, symVessel.AsLMRepresentation());

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objPlacement = null;
symNozzle1 = null;
symNozzle2 = null;
symTray1 = null;
symTray2 = null;
symVessel = null;
objVessel = null;
pidAutoDwg = null;
pidAutoApp = null;
objPlacement = null;
```

68. PLACE LABELS ON A VESSEL

Purpose

Use PIDPlaceLabel method to place labels on equipment

b) Problem Statement

Write a standard executable to populate some properties of a vessel and then place labels to display them.

c) Solution

◊ **Open the Smart P&ID drawing.**

1. Create a drawing through SPManager.
2. Double-click on the drawing to open up Smart P&ID

◊ **Create a standard executable VB project**

3. Select a standard exe project
4. Reference the "Logical Model Automation" and "Placement Automation" libraries

◊ **Add code to place a vessel**

5. Use the Function **PIDPlaceSymbol**(DefinitionFile As String, X As Double, Y As Double, [Mirror], [Rotation], [ExistingItem As LMAItem], [TargetItem]) As **LMSymbol**
6. Provide the DefinitionFile string indicating the location of the symbol on a server
7. Provide the X and Y coordinates of the placement on the drawing.
8. Provide the TargetItem as an LMAItem when placing nozzles or trays.
9. Use the return value to future reference.

◊ **Add code to delete the vessel**

10. Use the Function **PIDPlaceLabel**(DefinitionFile As String, Points() As Double, [Mirror], [Rotation], [LabeledItem As LMRepresentation], [IsLeaderVisible As Boolean = False]) As **LMLabelPersist**
11. The Points array consists of the exact number of points (starting from index 1) necessary to place the label.
12. The LMRepresentation argument must be a representation of the parent item on the drawing.
13. The return object is the label object.

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();
// place the vessel
String strVesselDef = @"\Equipment\Vessels\Vertical Drums\1D 1C 2to1.sym";
Double dblVesselX = 0.2;
Double dblVesselY = 0.2;
LMSymbol symVessel = objPlacement.PIDPlaceSymbol(strVesselDef, dblVesselX, dblVesselY);
LMVessel objVessel = objPlacement.PIDDataSource.GetVessel(symVessel.ModelItemID);
objVessel.Name = "Vessel for Label Placement";
objVessel.InsulPurpose = "R15";
objVessel.HTraceMedium = "SS";
objVessel.HTraceMediumTemp = "300 F";
objVessel.HTraceReqmt = "ET";
objVessel.Commit();
```

```
// place 3 different labels for the vessel
String strLabelDef1 = @"\Equipment\Labels - Equipment\Equipment Name.sym";
Double[] dblPtsArray = new Double[4];
dblPtsArray[0] = dblVesselX;
dblPtsArray[1] = dblVesselY;
dblPtsArray[2] = dblVesselX; // + 0.02;
dblPtsArray[3] = dblVesselY + 0.1;

LMLabelPersist labelpersist = objPlacement.PIDPlaceLabel(strLabelDef1, dblPtsArray,
                                  null, null, symVessel.AsLMRepresentation());
String strLabelDef2 = @"\Equipment\Labels - Equipment\Insulation Purpose.sym";
dblPtsArray[2] = dblVesselX - 0.05;
dblPtsArray[3] = dblVesselY + 0.1;
labelpersist = objPlacement.PIDPlaceLabel(strLabelDef2, dblPtsArray,
                    null, null, symVessel.AsLMRepresentation(), true);
String strLabelDef3 = @"\Equipment\Labels - Equipment\Heat Tracing.sym";
dblPtsArray[2] = dblVesselX + 0.05;
dblPtsArray[3] = dblVesselY + 0.1;
labelpersist = objPlacement.PIDPlaceLabel(strLabelDef3, dblPtsArray,
                    null, null, symVessel.AsLMRepresentation(), true);

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objVessel = null;
labelpersist = null;
objPlacement = null;
symVessel = null;
pidAutoDwg = null;
pidAutoApp = null;
objPlacement = null;
```

69. PLACE OPC

a) *Purpose*

Use PIDPlaceOPC method to place an OPC into drawing.

b) *Problem Statement*

Write a standard executable to place an OPC into drawing.

c) *Solution*

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();

String strOPCDef = @"\Piping\Piping OPC's\Off-Drawing.sym";
LMSymbol objSymbol = objPlacement.PIDPlaceSymbol(strOPCDef, 0.1, 0.1);

if (objSymbol.Equals(null))
       MessageBox.Show("Placement unsuccessful");

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objPlacement = null;
objSymbol = null;
pidAutoDwg = null;
pidAutoApp = null; ; ;
```

70. Place OPC From StockPile

a) Purpose

Use PIDPlaceOPC method to place an OPC from StockPile into drawing.

b) Problem Statement

Place an OPC into a drawing, and place its pair OPC in plant stockpile, then open another drawing with a piperun placed, then write a standard executable to find the OPC, then find its pair OPC in StockPile, then place it pair OPC from StockPile into current drawing, and connect with the piperun.

c) Solution

◊ **Example code**

Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

```
Placement objPlacement = new Placement();
LMPipeRun objPiperun = objPlacement.PIDDataSource.GetPipeRun(CONST_SPID_PipeRun);
LMConnector objConnector = null;
foreach (LMRepresentation objRep in objPiperun.Representations)
       if (objRep.Attributes["RepresentationType"].Value.Equals("Connector"))
       {
              objConnector = objPlacement.PIDDataSource.GetConnector(objRep.Id);
              break;
       }
       // place the OPC from the stockpile into the active drawing
       Double dblOpcX = (Double)objConnector.ConnectorVertices.Nth(1)
                            .Attributes["XCoordinate"].Value;
       Double dblOpcY = (Double)objConnector.ConnectorVertices.Nth(1)
                            .Attributes["YCoordinate"].Value;

       LMOPC objOPC = objPlacement.PIDDataSource.GetOPC(CONST_SPID_OPC);
       LMOPC objpairOPC = objOPC.pairedWithOPCObject;

       String strOpcDef = @"\Piping\Piping OPC's\Off-Drawing.sym";

       LMSymbol objSymbol = objPlacement.PIDPlaceSymbol(strOpcDef, dblOpcX, dblOpcY,
                                   null, null, objpairOPC.AsLMAItem());

       if (objSymbol.Equals(null))
              MessageBox.Show("Placement unsuccessful");

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objPiperun = null;
objOPC = null;
objpairOPC = null;
objConnector = null;
objPlacement = null;
objSymbol = null;
pidAutoDwg = null;
pidAutoApp = null;
```

71. Place Piperun with PIDPlaceRun

a) Purpose

Use PIDPlaceRun method to place a Piperun from stockpile into active drawing

b) Problem Statement

Write a standalone application to create a piperun in stockpile, then place this piperun from stockpile into active drawing. Then place a valve, and place a piperun connects first piperun and the valve.

c) Solution

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();
String strPipeRunDef = @"\Piping\Routing\Process Lines\Primary Piping.sym";
LMAItem objItem = objPlacement.PIDCreateItem(strPipeRunDef) as LMAItem;
PlaceRunInputs objInputs = new PlaceRunInputs();
objInputs.AddPoint(0.1, 0.1);
objInputs.AddPoint(0.2, 0.1);
LMConnector objConnector = objPlacement.PIDPlaceRun(objItem, objInputs);

LMSymbol objSymbol = objPlacement.PIDPlaceSymbol(@"\Piping\Valves\2 Way Common
                            \Ball Valve.sym", 0.15, 0.3, null, 1.57);
objItem = objPlacement.PIDCreateItem(strPipeRunDef) as LMAItem;
objInputs = new PlaceRunInputs();
objInputs.AddConnectorTarget(objConnector, 0.15, 0.1);
objInputs.AddPoint(0.15, 0.15);
objInputs.AddPoint(0.12, 0.15);
objInputs.AddPoint(0.12, 0.2);
objInputs.AddPoint(0.15, 0.2);
objInputs.AddSymbolTarget(objSymbol, 0.15, 0.3);
objConnector = objPlacement.PIDPlaceRun(objItem, objInputs);

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objConnector = null;
objInputs = null;
objItem = null;
objPlacement = null;
objSymbol = null;
pidAutoDwg = null;
pidAutoApp = null;
```

72. JOIN TWO PIPERUNS

a) Purpose

Use PIDAutoJoin to auto join two piperuns

b) Problem Statement

Write a standalone application to createa piperun in stockpile, then place this piperun from stockpile into active drawing. Then place another piperun from middle of first piperun to have an end open, then place a vessel with a nozzle, then place a new piperun connects nozzle and second piperun, then use PIDAutoJoin to join second and third piperuns.

c) Solution

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();

// place the first piperun
String strPipeRunDef = @"\Piping\Routing\Process Lines\Primary Piping.sym";
LMAItem objItem = objPlacement.PIDCreateItem(strPipeRunDef) as LMAItem;
PlaceRunInputs objInputs = new PlaceRunInputs();
objInputs.AddPoint(0.1, 0.1);
objInputs.AddPoint(0.2, 0.1);
LMConnector objConnector = objPlacement.PIDPlaceRun(objItem, objInputs);

// place the second piperun
objItem = objPlacement.PIDCreateItem(strPipeRunDef) as LMAItem;
objInputs = new PlaceRunInputs();
objInputs.AddLocatedTarget(0.15, 0.1);
objInputs.AddPoint(0.15, 0.3);
objConnector = objPlacement.PIDPlaceRun(objItem, objInputs);
LMPipeRuns objPiperuns = new LMPipeRuns();

objPiperuns.Add(objPlacement.PIDDataSource.GetPipeRun(objConnector.ModelItemID)
       .AsLMAItem());

// place vessel and nozzle
LMSymbol objSymbol = objPlacement.PIDPlaceSymbol(@"\Equipment\Vessels\Vertical Drums
                             \1D 1C 2to1.sym", 0.15, 0.5);
String strNozzleDef = @"\Equipment Components\Nozzles\Flanged Nozzle with blind.sym";

objSymbol = objPlacement.PIDPlaceSymbol(strNozzleDef, 0.15, 0.5 - 0.1, null, null, null,
              objSymbol.AsLMRepresentation());

// place the third piperun
objItem = objPlacement.PIDCreateItem(strPipeRunDef) as LMAItem;
objInputs = new PlaceRunInputs();
objInputs.AddConnectorTarget(objConnector, 0.15, 0.3);
objInputs.AddSymbolTarget(objSymbol, (Double)objSymbol.Attributes["XCoordinate"].Value,
(Double)objSymbol.Attributes["YCoordinate"].Value);
```

```
objConnector = objPlacement.PIDPlaceRun(objItem, objInputs);

objPiperuns.Add(objPlacement.PIDDataSource.GetPipeRun(objConnector.ModelItemID)
       .AsLMAItem());

// auto join piperuns
LMAItem objSurvivorItem = null;
foreach (LMPipeRun objPiperun in objPiperuns)
objPlacement.PIDAutoJoin(objPiperun.AsLMAItem(), AutoJoinEndConstants.autoJoin_Both,
       ref objSurvivorItem);

MessageBox.Show("PipeRun Join Done!");

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objConnector = null;
objInputs = null;
objItem = null;
objPiperuns = null;
objPlacement = null;
objSurvivorItem = null;
objSymbol = null;
pidAutoDwg = null;
pidAutoApp = null;
```

73. Place Gap

Purpose

Use PIDPlaceGap method to place a Gap.

b) Problem Statement

Write a standalone application to place Connector, then place a Gap in the middle of the connector

c) Solution

1. PIDPlaceGap returns a LMSymbol object, whose RepresentationType is “GAP”

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();

String strPipeRunDef = @"\Piping\Routing\Process Lines\Primary Piping.sym";
String strGapDef = @"\Piping\Gaps\gap-lines.sym";

Double[] dblPtsArray = new Double[4];
dblPtsArray[0] = 0.2;
dblPtsArray[1] = 0.2;
dblPtsArray[2] = 0.4;
dblPtsArray[3] = 0.2;
LMConnector objConnector = objPlacement.PIDPlaceConnector(strPipeRunDef, dblPtsArray);
LMSymbol objSymbol = objPlacement.PIDPlaceGap(strGapDef, 0.3, 0.2, 0.02, 0.02,
objConnector, -1.57);

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objConnector = null;
objPlacement = null;
objSymbol = null;
pidAutoDwg = null;
pidAutoApp = null;
```

74. PLACE BOUNDEDSHAPE

a) Purpose

Use PIDPlaceBoundedShape method to place a BoundedShape (AreaBreak).

b) Problem Statement

Write a standalone application to place a BoundedShape (AreaBreak) and a vessel with nozzle. Add a vessel and assign it to be a part of the AreaBreak

c) Solution

1. PIDPlaceBoundedShape places a visual BoundedShape aroung the items rather than estbilish the relationship between BoundedShape and items inside of it.

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();
LMADataSource datasource = objPlacement.PIDDataSource;

LMSymbol symVessel = objPlacement.PIDPlaceSymbol(@"\Equipment\Vessels\Vertical Drums
                           \1D 1C 2to1.sym", 0.25, 0.25);

LMSymbol symNozzle = objPlacement.PIDPlaceSymbol(@"\Equipment Components\Nozzles\Flanged
Nozzle with blind.sym", 0.0, 0.0, null, null, null, symVessel.AsLMRepresentation());


// place the BoundedShape as an AreaBreak
Double[] dblPtsArray = new Double[10];
dblPtsArray[0] = 0.1;
dblPtsArray[1] = 0.1;
dblPtsArray[2] = 0.4;
dblPtsArray[3] = 0.1;
dblPtsArray[4] = 0.4;
dblPtsArray[5] = 0.4;
dblPtsArray[6] = 0.1;
dblPtsArray[7] = 0.4;
dblPtsArray[8] = 0.1;
dblPtsArray[9] = 0.1;
LMBoundedShape objBoundedShape = objPlacement.PIDPlaceBoundedShape(@"\Design
                                  \Area Break.sym", dblPtsArray);

// get the BoundedShape (AreaBreak) as PlantItemGroup
LMPlantItemGroup objPlantItemGroup = datasource.
                     GetPlantItemGroup(objBoundedShape.ModelItemID);

LMVessel objVessel = datasource.GetVessel(symVessel.ModelItemID);

Console.WriteLine("\tTotal Vessel PlantItemGroups = " + objVessel.PlantItemGroups.Count);

objVessel.PlantItemGroups.Add(objPlantItemGroup.AsLMAItem());
```

```
Console.WriteLine("\tTotal Vessel PlantItemGroups = " + objVessel.PlantItemGroups.Count);

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

symNozzle = null;
objVessel = null;
objBoundedShape = null;
datasource = null;
symVessel = null;
objPlantItemGroup = null;
objPlacement = null;
```

75. Place Assembly

a) Purpose

Use PIDPlaceAssembly method to place assembly into drawing

b) Problem Statement

Create an assembly using the Smart P&ID modeler. Write a standalone application to place an assembly into drawing.

c) Solution

If the Assembly's source is in a location that is not accessible, change the source to current machine first

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();

String strAssemblyDef = @"\Assemblies\Equipment\Pump01.pid";
LMAItems objItems = objPlacement.PIDPlaceAssembly(strAssemblyDef, 0.2, 0.2) as LMAItems;

if (objItems.Count != 0)
       MessageBox.Show("Place Assembly completed.");

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objItems = null;
objPlacement = null;
pidAutoApp = null;
objPIDADrawing = null;
```

76. Delete Vessel from Drawing

a) Purpose

Use PIDRemovePlacement method to delete vessel from drawing

b) Problem Statement

Write a standard executable to delete vessel from the drawing.

c) Solution

◊ **Open the Smart P&ID drawing.**

14. Create a drawing through SPManager.
15. Double-click on the drawing to open up Smart P&ID

◊ **Create a standard executable VB project**

16. Select a standard exe project
17. Reference the “Logical Model Automation” and “Placement Automation” libraries

◊ **Add code to delete the vessel from drawing**

18. Use the Function **PIDRemovePlacement**(Representation As LMRepresentation) As **Boolean**
19. The LMRepresentation argument must be a representation of the item on the drawing.
20. The boolean return value can be stored to determine success or failure.

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();

String strVesselDef = @"\Equipment\Vessels\Horizontal Drums\Horz Surge w-Horiz Dea.sym";
LMSymbol objSymbol = objPlacement.PIDPlaceSymbol(strVesselDef, 0.2, 0.2);
LMVessel objVessel = objPlacement.PIDDataSource.GetVessel(objSymbol.ModelItemID);

LMRepresentation objRep = objVessel.Representations.Nth(1);

// remove the vessel from drawing to the stockpile
if (objPlacement.PIDRemovePlacement(objRep))
       Console.WriteLine("Vessel symbol removed successfully!");
else
       MessageBox.Show("ERROR: RemovePlacement unsuccessful !");

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objSymbol = null;
objVessel = null;
objPlacement = null;
objRep = null;
pidAutoDwg = null;
pidAutoApp = null;
```

77. Delete Vessel from Model

a) Purpose

Use PIDDeleteItem method to delete vessal from the project

b) Problem Statement

Write a standard executable to delete a vessel from project.

c) Solution

◊ **Open the Smart P&ID drawing.**

1. Create a drawing through SPManager.
2. Double-click on the drawing to open up Smart P&ID

◊ **Create a standard executable VB project**

3. Select a standard exe project
4. Reference the "Logical Model Automation" and "Placement Automation" libraries

◊ **Add code to delete the vessel from model**

5. Use the Function **PIDDeleteItem**(Item As LMAItem) As **Boolean** to remove from model
6. The LMRepresentation argument must be a representation of the item on the drawing.
7. The boolean return value can be stored to determine success or failure.

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();
String strVesselDef = @"\Equipment\Vessels\Horizontal Drums\Horz Surge w-Horiz Dea.sym";
LMSymbol objSymbol = objPlacement.PIDPlaceSymbol(strVesselDef, 0.2, 0.2);
LMAItem objItem = objPlacement.PIDDataSource.GetVessel(objSymbol.ModelItemID) as LMAItem;

if (objPlacement.PIDDeleteItem(objItem))
       Console.WriteLine("Delete from drawing model successful");
else
       MessageBox.Show("ERROR: Delete from drawing model unsuccessful !");
       LMAItem objItem2 = objPlacement.PIDCreateItem(strVesselDef) as LMAItem;
       objItem = objPlacement.PIDDataSource.GetVessel(objItem2.Id) as LMAItem;

if (objPlacement.PIDDeleteItem(objItem))
       Console.WriteLine("Delete from stockpile successful");
else
       MessageBox.Show("ERROR: Delete from stokcpile unsuccessful !");

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();
objSymbol = null;
objItem = null;
objItem2 = null;
objPlacement = null;
pidAutoDwg = null;
pidAutoApp = null;
```

78. Replace Symbol

Purpose

Use PIDRePlaceSymbol method to replace a vessel.

b) Problem Statement

Write a standalone application to place a vessel with a nozzle on it. Replace the vessel with different vessel. Note that the vessel is replaced and the nozzle is now on the new vessel.

c) Solution

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();
// place a vessel
String strVesselDef = @"\Equipment\Vessels\Vertical Drums\1D 1C 2to1.sym";
Double dblVesselX = 0.3;
Double dblVesselY = 0.2;

LMSymbol symVessel = objPlacement.PIDPlaceSymbol(strVesselDef, dblVesselX, dblVesselY);
// place a nozzle on the vessel
String strNozzleDef = @"\Equipment Components\Nozzles\Flanged Nozzle with blind.sym";

LMSymbol symNozzle = objPlacement.PIDPlaceSymbol(strNozzleDef, dblVesselX - 0.2,
dblVesselY + 0.05, null, null, null, symVessel.AsLMRepresentation());

Console.WriteLine("Prior to Replace Vessel.  (Press a key to continue...)");
Console.ReadLine();
Console.WriteLine("After replacing the vessel.");

// replace the vessel, note the nozzle remains on the new vessel
String strReplaceVesselDef = @"\Equipment\Vessels\Vertical Drums
                                   \2to1Parametric V Drum.sym";
LMSymbol symReplace = objPlacement.PIDReplaceSymbol(ref strReplaceVesselDef,
                            ref symVessel);

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

symReplace = null;
symVessel = null;
symNozzle = null;
objPlacement = null;
pidAutoDwg = null;
pidAutoApp = null;
```

79. Replace Label

Purpose

Use PIDRePlaceLabel method to replace a label.

b) Problem Statement

Write a standalone application to place a vessel with a nozzle on it, then replace the vessel with different vessel. Note vessel is replaced with nozzle now is sitting on the new vessel.

c) Solution

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();
// place a vessel
String strVesselDef = @"\Equipment\Vessels\Vertical Drums\2to1Parametric V Drum.sym";
LMSymbol objSymbol = objPlacement.PIDPlaceSymbol(strVesselDef, 0.2, 0.2);
// get the vessel and set some properties
LMVessel objVessel = objPlacement.PIDDataSource.GetVessel(objSymbol.ModelItemID);
objVessel.Attributes["Name"].Value = "V1";
objVessel.Attributes["TagPrefix"].Value ="T";
objVessel.Commit();

// place a label on the vessel
Double[] dblPtsArray = new Double[4];
dblPtsArray[0] = 0.21;
dblPtsArray[1] = 0.25;
dblPtsArray[2] = 0.1;
dblPtsArray[3] = 0.1;
String strLabelDef1 = @"\Equipment\Labels - Equipment\Equipment Name.sym";
LMLabelPersist objLabelPersist1 = objPlacement.PIDPlaceLabel(strLabelDef1, dblPtsArray,
null, null, objSymbol.AsLMRepresentation(), true);

// replace the label with a new label
String strLabelDef2 = @"\Equipment\Labels - Equipment\Equipment ID.sym";
LMLabelPersist objLabelPersist2 = objPlacement.PIDReplaceLabel(ref strLabelDef2,
                                          ref objLabelPersist1);

Console.WriteLine("Done.");

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objVessel = null;
objSymbol = null;
objLabelPersist1 = null;
objLabelPersist1 = null;
objPlacement = null;
pidAutoDwg = null;
pidAutoApp = null;
```

80. Replace OPC

a) Purpose

Use PIDRePlaceOPC method to replace an OPC.

b) Problem Statement

Write a standalone application to replace an OPC on draiwng.

c) Solution

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();
LMOPC objOPC = objPlacement.PIDDataSource.GetOPC(CONST_SPID_OPC);
LMSymbol objSymbol =
objPlacement.PIDDataSource.GetSymbol(objOPC.Representations.Nth(1).Id);

String strOPCDef = @"\Piping\Piping OPC's\Off-Unit.sym";
LMSymbol objSymbolReplace = objPlacement.PIDReplaceSymbol(ref strOPCDef, ref objSymbol);

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objOPC = null;
objSymbol = null;
objSymbolReplace = null;
objPlacement = null;
pidAutoDwg = null;
pidAutoApp = null;
```

81. Modify Parametric Symbol

a) Purpose

Use PIDApplyParameters method to modifty a Parametric Symbol

b) Problem Statement

Write a standalone application to place a parametric vessel symbol, and place a nozzle on it. Then, Modifies the parameters of the vessel.

c) Solution

1. Names() are the Variables defined in Catalog Manager for the parametric symbol

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();

String strVesselDef = @"\Equipment\Vessels\Vertical Drums\2to1Parametric V Drum.sym";
LMSymbol symVessel = objPlacement.PIDPlaceSymbol(strVesselDef, 0.2, 0.2, true, 1.57);

// place a nozzle on the vessel
String strNozzleDef = @"\Equipment Components\Nozzles\Flanged Nozzle with blind.sym";
LMSymbol symNozzle = objPlacement.PIDPlaceSymbol(strNozzleDef, 0.22, 0.4,
                            null, null, null, symVessel.AsLMRepresentation());

// change the parametric parameters of the vessel
String[] strNames = new String[2];
strNames[0] = "Top";
strNames[1] = "Right";
String[] strValues = new String[2];
strValues[0] = "0.38";
strValues[1] = "0.2";
objPlacement.PIDApplyParameters(symVessel.AsLMRepresentation(), strNames, strValues);

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

symNozzle = null;
symVessel = null;
objPlacement = null;
pidAutoDwg = null;
pidAutoApp = null;
```

82. Locate X, Y Coordinates of Signal Points on an Instrument

a) Purpose

Using PIDConnectPointLocation to locate X, Y coordinates of signal points on an instrument.

b) Problem Statement

Place an off-line instrument, connect a signal line to signal point on the instrument with index as 3.

c) Solution

Using PIDConnectPointLocation to find out X, Y coordinates first.

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();
String strInstrDef = @"\Instrumentation\Off-Line\With Implied Components
                           \Level\Discr Field Mounted LC.sym";
LMSymbol objSymbol = objPlacement.PIDPlaceSymbol(strInstrDef, 0.3, 0.3);
Double dblX = 0.0;
Double dblY = 0.0;
Boolean blnSuccess = objPlacement.PIDConnectPointLocation(objSymbol, 3,
                           ref dblX, ref dblY);
PlaceRunInputs objInputs = new PlaceRunInputs();
objInputs.AddPoint(0.2, 0.3);
objInputs.AddSymbolTarget(objSymbol, dblX, dblY);

String strSignalDef = @"\Instrumentation\Signal Line\Electric Binary.sym";
LMAItem objItem = objPlacement.PIDCreateItem(strSignalDef) as LMAItem;
LMConnector objConnector = objPlacement.PIDPlaceRun(objItem, objInputs);

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objSymbol = null;
objConnector = null;
objInputs = null;
objItem = null;
objPlacement = null;
pidAutoDwg = null;
pidAutoApp = null;
```

83. Place Instrument Loop

a) Purpose

Use PIDCreateItem method to place an Instrument Loop in stockpile.

b) Problem Statement

Write a standalone application to place an Instrument Loop in stockpile and place a Piperun into drawing, then associate the Instrument Loop with the Piperun.

c) Solution

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();
string strDwgName = "Automation.pid";
Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

Placement objPlacement = new Placement();
String strInstrDef = @"\Instrumentation\Off-Line\With Implied Components
                     \Pressure\Discr Field Mounted PC.sym";
LMSymbol objInstrSym = objPlacement.PIDPlaceSymbol(strInstrDef, 0.2, 0.2);
LMADataSource datasource = objPlacement.PIDDataSource;
LMInstrument objInstr = datasource.GetInstrument(objInstrSym.ModelItemID);

// place an instrument loop into the stockpile
String strInstrLoopDef = @"\Instrumentation\Loops\Pressure Loop.sym";
LMAItem objItem = objPlacement.PIDCreateItem(strInstrLoopDef) as LMAItem;

LMInstrLoop objInstrLoop = datasource.GetInstrLoop(objItem.Id);
objInstrLoop.Attributes["TagSuffix"].Value ="P";
objInstrLoop.Commit();
objInstrLoop = datasource.GetInstrLoop(objItem.Id);

Console.WriteLine("InstrumentLoop ItemTag  ==>  " +
                     objInstrLoop.Attributes["ItemTag"].Value);
Console.WriteLine("\tTotal Instrument.PlantItemGroups = " +
                     objInstr.PlantItemGroups.Count);

objInstr.PlantItemGroups.Add(objInstrLoop.AsLMPlantItemGroup().AsLMAItem());
objInstr.Commit();

Console.WriteLine("\tTotal Instrument.PlantItemGroups = " +
                     objInstr.PlantItemGroups.Count);

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

objInstrSym = null;
objItem = null;
objInstr = null;
objInstrLoop = null;
objPlacement = null;
datasource = null;
pidAutoDwg = null;
pidAutoApp = null;
```

84. Find and Replace Labels

a) Purpose

Comprehensive lab to practice filter for labels, and delete existing labels, then place new labels at the same X, Y Coordinates.

b) Problem Statement

Get collection of labels in the database, then loop through each label, and delete "\Piping\Segment Breaks\Construction Responsibility.sym" label, and place a new "\Piping\Segment Breaks\Construction Status.sym" label at the same X, Y Coordinate.

c) Solution

◊ **Example code**

```
Placement objPlacement = new Placement();

LMADataSource datasource = objPlacement.PIDDataSource;
// build multiple criterions, or criteria
LMACriterions criteria = new LMACriterions();
criteria.AddNew("1stOne");
criteria["1stOne"].SourceAttributeName = "ItemStatus";
criteria["1stOne"].Operator = "=";
criteria["1stOne"].ValueAttribute =1;
criteria.AddNew("2ndOne");
criteria["2ndOne"].SourceAttributeName = "SP_DrawingID";
criteria["2ndOne"].Operator = "=";
criteria["2ndOne"].ValueAttribute = objPlacement.PIDDataSource.PIDMgr.Drawing.ID;
criteria["2ndOne"].Conjunctive = true;
// add the multiple criteria to a filter
LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "LabelPersist";
objFilter.Criteria.Add(criteria["1stOne"]);
objFilter.Criteria.Add(criteria["2ndOne"]);

// apply the filter to collect label persists from the database
LMLabelPersists objLabelPersists = new LMLabelPersists();
objLabelPersists.Collect(datasource, null, "", objFilter);

Console.WriteLine("Labels found to replace = " + objLabelPersists.Count);
// loop through the filtered label persists
Double[] dblPtsArray = new Double[4];
LMLabelPersist objNewLabelPersist = null;
foreach (LMLabelPersist objLabelPersist in objLabelPersists)
       if (objLabelPersist.Attributes["FileName"].Value.Equals(@"\Piping\Segment
              Breaks\Construction Responsibility.sym"))
       {
              // casting array assignment values as Double here
              dblPtsArray[0] = (Double)objLabelPersist.LeaderVertices.Nth(1)
                                   .Attributes["XCoordinate"].Value;
              dblPtsArray[1] = (Double)objLabelPersist.LeaderVertices.Nth(1)
                                   .Attributes["YCoordinate"].Value;
              dblPtsArray[2] = (Double)objLabelPersist
                                   .Attributes["XCoordinate"].Value;
              dblPtsArray[3] = (Double)objLabelPersist
                                     .Attributes["YCoordinate"].Value;

              if (objPlacement.PIDRemovePlacement(objLabelPersist.AsLMRepresentation()))
                     objNewLabelPersist = objPlacement.PIDPlaceLabel(@"\Piping\Segment
              Breaks\Construction Status.sym", dblPtsArray, null, null, true);
       }

MessageBox.Show("Replacing Labels Done!");

pidAutoDwg.CloseDrawing(true);
pidAutoApp.Quit();

criteria = null;
datasource = null;
objFilter = null;
objLabelPersists = null;
objNewLabelPersist = null;
objPlacement = null;
pidAutoApp = null;
objPIDADrawing = null;
```

85. Open and Close an existing drawing

a) Purpose

Using PIDAutomation to open and close an existing drawing.

b) Problem Statement

Get collection of all drawings in the database, then loop through each drawing, open and close each drawing.

c) Solution

◊ **Example code**

```
LMADataSource datasource = new LMADataSource();
LMDrawings objDrawings = new LMDrawings();
objDrawings.Collect(datasource, null, "", null);

Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();

foreach (LMDrawing objDrawing in objDrawings)
     if (objDrawing.Attributes["ItemStatus"].Index.Equals(1)) // 1 stands for "Active"
     {
            string strDwgName = objDrawing.Attributes["Name"].Value.ToString();
            Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings
                   .OpenDrawing(ref strDwgName);

            if (pidAutoDwg != null)
            {
                   Console.WriteLine("Drawing " +
                   objDrawing.Attributes["Name"].Value.ToString() + " is open!");

                   pidAutoDwg.CloseDrawing(true); // save drawing when closing
            }
      }


   pidAutoApp.Quit();

   objDrawings = null;
   pidAutoApp = null;
   datasource = null;
```

86. Create , Open and Close a new drawing

a) Purpose

Using PIDAutomation to create, open and close a new drawing.

b) Problem Statement

Create, open and close a new drawing until one of your Units.

c) Solution

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();

String strPlantGroupName = Test";
String strTemplateFileName = @"\\SPID-TRN\Hexagon_Site\TSPL\
                                P&ID Reference Data\Template Files\C-Size.pid";
String strDrawingNumber = "TestCreateNewDrawing";
String strDrawingName = "TestCreateNewDrawing";
Drawing objPIDADrawing = pidAutoApp.Drawings.Add(strPlantGroupName,
                   ref strTemplateFileName, strDrawingNumber, ref strDrawingName);
if (objPIDADrawing != null)
{
     Console.WriteLine("Drawing " + objPIDADrawing.Name + " is open!");
     objPIDADrawing.CloseDrawing(true); // save before closing drawing
}

 pidAutoApp.Quit();
 pidAutoApp = null;
 objPIDADrawing = null;
```

87. Comprehensive Automation Lab

a) Purpose

To practice a comprehensive auatomation lab, including LLAMA, Placement and PIDAutomation.

b) Problem Statement

Write a standalone application to create a new drawing, then place an assembly into the drawing, then modify the piperuns placed by the assembly, set TagSequenceNo to 100. Then, close the drawing.

c) Solution

◊ **Example code**

```
Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
pidAutoApp.Activate();

LMACriterion criterion = new LMACriterion();
criterion.SourceAttributeName = "Name";
criterion.Operator = "=";
criterion.ValueAttribute = "PID Template Path";
LMAFilter objFilter = new LMAFilter();
objFilter.ItemType = "OptionSetting";
objFilter.Criteria.Add(criterion);

// apply the filter to collect the template files path
LMADataSource datasource = new LMADataSource();
LMOptionSettings optSettings = new LMOptionSettings();
optSettings.Collect(datasource, null, "", objFilter);

String strPlantGroupName ="Test";
String strTemplateFileName = optSettings.Nth(1).Attributes["Value"]
                                .Value.ToString() + @"\E-Size.pid";
String strDrawingNumber = "TestCreateNewDrawing1";
String strDrawingName = "TestCreateNewDrawing1";
Drawing objPIDADrawing = pidAutoApp.Drawings.Add(strPlantGroupName,
            ref strTemplateFileName, strDrawingNumber, ref strDrawingName);
if (objPIDADrawing != null)
{
     // place assembly
     Placement objPlacement = new Placement();
     datasource = objPlacement.PIDDataSource;
     LMAItems objItems = objPlacement.PIDPlaceAssembly(@"\Assemblies\Automation.pid",
                                0.2, 0.2) as LMAItems;

     // change TagSequenceNo
     LMConnector objConnector = null;
     LMPipeRun objPiperun = null;
     foreach (LMAItem objItem in objItems)
            if (objItem.ItemType.Equals("Connector"))
            {
                   objConnector = datasource.GetConnector(objItem.Id);
                   if (objConnector.ModelItemObject
                          .AsLMAItem().ItemType.Equals("PipeRun"))
                   {
                          objPiperun = datasource.GetPipeRun(objConnector.ModelItemID);
                              objPiperun.Attributes["TagSequenceNo"].Value = "100";
```

```
                         objPiperun.Commit();
                  }
            }

            objPlacement = null;
            objConnector = null;
            objItems = null;
            objPiperun = null;
            objPIDADrawing.CloseDrawing(true); // save before closing drawing
}

pidAutoApp.Quit();

optSettings = null;
objPIDADrawing = null;
criterion = null;
objFilter = null;
datasource = null;
pidAutoApp = null;
```

88. Create a Calculation Program

Purpose

Enable the Calculation button at the customized property “XYCoordinates” at ModelItem level to show X, Y coordinates of the symbol in format of X/Y.

b) Problem Statement

Write an Active-X dll implementing the DoCalculate method to read the X, Y coordinates of a Symbol to the customized property ““XYCoordinates” at ModelItem level. The customized property ““XYCoordinates” should be added at ModelItem level, with datatype is String, format is Variable Length, Maximum Length is 40, and Category is Accessories.

c) Solution

◊ **Example code**

```
bool ILMForeignCalc.ILMForeignCalc.DoCalculate(LMADataSource datasource, LMAItems items,
string strPropertyName, ref object objValue)
{
       Console.WriteLine("running\tLab 88 - Create A Calculation Program");
       return ShowXYCoordinates(datasource, items, strPropertyName, ref objValue);
}

private bool ShowXYCoordinates(LMADataSource datasource, LMAItems items,
       string strPropertyName, ref object objValue)
{
       Boolean blnRetVal = false; // return value
       LMEquipment objEquipment = null;
       LMSymbol objSymbol = null;

       foreach (LMAItem item in items)
              if (strPropertyName.Equals("XYCoordinates"))
              {
                      try
                     {
                         objEquipment = datasource.GetEquipment(item.Id);
                     }
                     catch (Exception ex)
                     {
                         MessageBox.Show(ex.GetBaseException().ToString());
                     }
                     if (equip != null)
                     {
                         objSymbol = datasource.GetSymbol(datasource.
                                    GetModelItem(item.Id).Representations.Nth(1).Id);
                        objValue = objSymbol.Attributes["XCoordinate"].Value + "/" +
                                           objSymbol.Attributes["YCoordinate"].Value;
                     }
                }
       blnRetVal = true;
       // clean up LLAMA variables and consumed memory
       objEquipment = null;
       objSymbol = null;
       return blnRetVal;
}
```

Save the Project and enter the ProgID in the Calculation ID field of the XYCoordinates Attribute in ModelItem through the DataDictionary Manager. Restart SPPID to find the button. Start the Project in Debug mode and then click on the button to step through your code. Then, compile the project, and click on the button again.

89. CREATE A VALIDATEPROPERTY PROGRAM

Purpose

Enable the Property validation at the ActuatorType attribute of InlineComp

b) Problem Statement

Write an Active-X dll implementing the DoValidateProperty method for placing corresponding actuator for an instrument valve when property ActuatorType is entered or changed.

c) Solution

◊ **Example code**

```
bool ILMForeignCalc.ILMForeignCalc.DoValidateProperty(LMADataSource datasource,
       LMAItems items, string strPropertyName, ref object objValue)
{
       Console.WriteLine("running\tLab 89 CREATE A VALIDATEPROPERTY PROGRAM");
       return AddActuator(datasource, items, strPropertyName, objValue);
}

private bool AddActuator(LMADataSource datasource, LMAItems items,
                            string strPropertyName, object objValue)
{
  Boolean blnRetVal = false; // return value
  LMInstrument objInstr = null;
  LMSymbol objSymbol = null;
  LMSymbol objSym = null;

  Ingr.SPPID.PIDAuto.Application pidAutoApp = new Ingr.SPPID.PIDAuto.Application();
  pidAutoApp.Activate();
  string strDwgName = "Automation.pid";
  Ingr.SPPID.PIDAuto.Drawing pidAutoDwg = pidAutoApp.Drawings.OpenDrawing(ref strDwgName);

  Placement objPlacement = new Placement();

  try
  {
    Boolean blnDelete = false;
    Boolean blnNeedAdd = false;
    Double dblX = 0.0;
    Double dblY = 0.0;
    String strDefFile = string.Empty;

    foreach (LMAItem item in items)
    {
      if (item.ItemType.Equals("Instrument") && strPropertyName.Equals("Actuator"))
      {
        if (item.Attributes["InstrumentClass"]
              .Value.Equals("Control valves and regulators"))
        {
          // get the instrument
          objInstr = datasource.GetInstrument(item.Id);
          if (objInstr.ChildPlantItemPlantItems.Count == 0)
            blnNeedAdd = true;
          elseif (objInstr.ChildPlantItemPlantItems.Count == 1)
                                {
```

```
              blnDelete = objPlacement.PIDDeleteItem(objInstr.ChildPlantItemPlantItems
                             .Nth(1).AsLMAItem());
              blnNeedAdd = true;
            }

            else
              MessageBox.Show("Wrong, there is more than 1 Child for this instrument!");

  if (blnNeedAdd)
  {
    switch (objValue.ToString())
    {
      case "Diaphragm":
        strDefFile = @"\Instrumentation\Actuators\Diaphragm Actuator.sym";
        break;
      case "Single acting cylinder":
        strDefFile = @"\Instrumentation\Actuators\Single Action Cyl Act.sym";
        break;
      case "Pilot operated cylinder":
        strDefFile = @"\Instrumentation\Actuators\Pilot Operated Cyl Act.sym";
        Break;
      case "Motor":
        strDefFile = @"\Instrumentation\Actuators\Motor Actuator.sym";
        break;
      case "Digital":
        strDefFile = @"\Instrumentation\Actuators\Digital Actuator.sym";
        break;
      case "Electro-hydraulic":
        strDefFile = @"\Instrumentation\Actuators\Electric-Hydraulic Act.sym";
        break;
      case "Single solenoid":
        strDefFile = @"\Instrumentation\Actuators\Solenoid Actuator.sym";
        break;
      case "Single solenoid w/reset":
        strDefFile = @"\Instrumentation\Actuators\Solenoid Act w-Man Reset.sym";
        break;
      case "Double solenoid":
        strDefFile = @"\Instrumentation\Actuators\Double Solenoid Act.sym";
        break;
      case "Pilot":
        strDefFile = @"\Instrumentation\Actuators\Pilot Actuator.sym";
        break;
      case "Weight":
        strDefFile = @"\Instrumentation\Actuators\Weight Actuator.sym";
        break;
      case "Manual":
        strDefFile = @"\Instrumentation\Actuators\Manual Actuator.sym";
        break;
      case "Spring":
        strDefFile = @"\Instrumentation\Actuators\Spring Actuator.sym";
        break;
      case "Capacitance sensor":
        strDefFile = @"\Instrumentation\Actuators\Capacitance Sensor Act.sym";
        break;
      case "Ball float":
        strDefFile = @"\Instrumentation\Actuators\Ball Float Actuator.sym";
        break;
      case "Displacement float":
        strDefFile = @"\Instrumentation\Actuators\Displacement Float Actuator.sym";
        break;
      case "Paddle wheel":
        strDefFile = @"\Instrumentation\Actuators\Paddle Wheel Actuator.sym";
        break;
      case "Diaphragm Rotary Actuator":
        strDefFile = @"\Instrumentation\Actuators\Diaph Actuator.sym";

        objSym = datasource.GetSymbol(objInstr.Representations.Nth(1).Id);
        dblX = (Double)objSym.Attributes["XCoordinate"].Value;
        dblY = (Double)objSym.Attributes["YCoordinate"].Value;
        break;

       default:
         // do nothing
         strDefFile = string.Empty;
         break;
      }

      if (strDefFile != string.Empty)
      {
         objSym = datasource.GetSymbol(objInstr.Representations.Nth(1).Id);
         dblX = (Double)objSym.Attributes["XCoordinate"].Value;
         dblY = (Double)objSym.Attributes["YCoordinate"].Value;
         objSymbol = objPlacement.PIDPlaceSymbol(strDefFile, dblX, dblY);
       }

       else
         MessageBox.Show("Couldn't find corresponding Actuator");
      }
       blnRetVal = true;
     }
    }
   }
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.GetBaseException().ToString());
  }
  finally
  {
   // clean up resources
   objInstr = null;
   objPlacement = null;
   objSymbol = null;
   objSym = null;
   }
   return blnRetVal;
  }
```

Save the Project and enter the ProgID in the Validation ID field of the ActuatorType Attribute in InlineComp through the DataDictionary Manager. Restart SPPID. Start the Project in Debug mode and then select different Actuators through ActuatorType property to step through your code. Then, compile the Project, and change the property again.

90. Create a ValidateItem Program

Purpose

Enable the Item validation when placing PipeRun.

b) Problem Statement

User added a new property “SystemCode” for Drawing, user want this property value to be copied to new Piperuns when placing them. Write an Active-X dll implementing the DoValidateItem method when placing PipeRun to make the copy from Drawing to PipeRun. You will notice a problem the a ProgID has existed for the PipeRun, learning how to call another validation program through your code.

c) Solution

◊ **Example code**

```
public const string PlantItemValidationProgID = "PlantItemValidation.Validate";

bool ILMForeignCalc.ILMForeignCalc.DoValidateItem(LMADataSource datasource, LMAItems
items, ENUM_LMAValidateContext context)
{
       var objLMForeignCalc = (ILMForeignCalc.ILMForeignCalc)datasource.
       CreateValidationCalculationObjectByProgId(PlantItemValidationProgID);

       if (objLMForeignCalc != null)
       {
              return objLMForeignCalc.DoValidateItem(datasource, items, context);
       }

       return CopySystemCode(datasource, items, context);
 }

 private bool CopySystemCode(LMADataSource datasource, LMAItems items,
       ENUM_LMAValidateContext context)
 {
       LMDrawing objDrawing = null;
       LMModelItem objModelItem = null;

       if (context.Equals(ENUM_LMAValidateContext.LMAValidateCreate))
       {
              foreach (LMAItem item in items)
              {
                     objModelItem = datasource.GetModelItem(item.Id);
                     objDrawing = objModelItem.Representations.Nth(1).DrawingObject;
                     item.Attributes["SystemCode"].Value =
                            objDrawing.Attributes["SystemCode"].Value;
                     item.Commit(true);
              }
       return true;
       }

       return false;
   }
```

Save the Project and enter the ProgID in the Validation Program field of the PipeRun through the DataDictionary Manager – DataBase Item Types. Restart SPPID. Start the Project in Debug mode and then place an PipeRun to step through your code. Then, compile the Project, and place PipeRun again.

91. Create a Drawing Validate Program

a) Purpose

Enable the Drawing validation when a drawing event is triggered.

b) Problem Statement

System admin wants to log the time user name when a drawing is opened, closed or printed. This example writes an Active-X dll (DrawingValidation.dll) implementing the DoValidateItem method when ac drawing event (Open, Close, Print, Create, Modify) is detected. The ProgID, **DrawingValidation.Validate** needs to be assigned to Drawing object in DataDictionary Manage -> DataBase Itemtypes table.

c) Solution

◊ **Example code**

```
bool ILMForeignCalc.ILMForeignCalc.DoValidateItem(LMADataSource datasource,
       LMAItems items, ENUM_LMAValidateContext context)
{
       String strFileName = String.Empty;
       try
       {
              switch (context)
              {
                     case ENUM_LMAValidateContext.LMAValidateClose:
                     strFileName = "Drawing Closed"; // context = 1
                        AutoGapAllCmd agaCmd = new AutoGapAllCmd();
                        RAD2D.Application radApp = datasource.PIDMgr.Application
                                                            .RadApplication;
                        agaCmd.GapAll(radApp);
                        // end AutoGap on active drawing
                        break;
                    case ENUM_LMAValidateContext.LMAValidateCreate:
                        strFileName = "Drawing Created"; // context = 3
                        break;
                    case ENUM_LMAValidateContext.LMAValidateDelete:
                        strFileName = "Drawing Deleted"; // context = 4
                        break;
                    case ENUM_LMAValidateContext.LMAValidateModify:
                        strFileName = "Drawing Modified"; // context = 6
                        break;
                    case ENUM_LMAValidateContext.LMAValidateOpen:
                        strFileName = "Drawing Opened"; // context = 8
                        break;
                }
                // create file
                if (strFileName.Length > 0)
                    CreateFile(strFileName, items);
                return true;
            }
            catch (System.Exception se)
            {
                MessageBox.Show(se.GetBaseException().ToString());
                return false;
            }
  }
```

```
private void CreateFile(String strFileName, LMAItems items = null)
{
       String strFolder = String.Empty;
       try
       {
              strFolder = Path.GetTempPath();
              if (!Directory.Exists(strFolder))
                     Directory.CreateDirectory(strFolder);
                     StreamWriter sw = new StreamWriter(strFolder + strFileName
                                         + ".txt", true);
                     sw.WriteLine(DateTime.Now);
                     sw.WriteLine("  Item Type: " + items.Nth(1).ItemType);
                     sw.WriteLine("  Name: " + items.Nth(1).Attributes["Name"]);
                     sw.WriteLine("  UserName: " + System.Environment.UserName + "\n");
                     sw.Close();
                     sw = null;
        }

        catch (System.Exception se)
        {
              MessageBox.Show(se.GetBaseException().ToString());
        }
  }
```