XDataSource Reference Guide

Bhaskar

Table of Contents

Introduction

XDataSource is a SQL query wrapped dll to access Smartplant Database. XPID is a Class within the Class Library which accesses Hexagon's SPPID Database independently. The library doesn't need SPPID application to be installed on the Development or Deployment Machine. This Library is entirely different than conventional Llama Automation. Please note that the delivered is still a better reference as it contains all the Classes developed by Hexagon. The XDataSource has just the required classes which can be used by any Developer or Administrator to develop any application to read data from the Application Database.

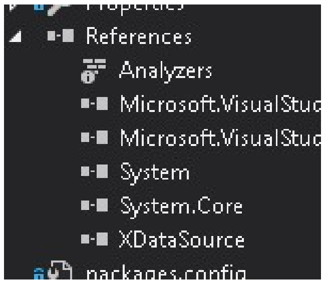
Prerequisite

- The user needs access to Database to run any application built on top of it.
- 6 Information required to connect: Database Server, Database Name, Plant Schema, Plant DD Schema, PID Schema, PID DD Schema.
- It is customized to work only for Hierarchy 7, that is Plant -> Area -> Unit.
- It works only with MS SQL Server projects as of now.
- This is a Work in Progress. Till now it can get Plant, Area, Unit, Enum, Codelist, Attribution, PipeRun and Drawing Information.

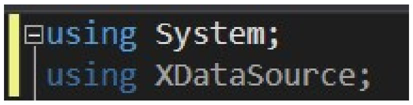
Understanding the Code

Accessing PBS

Add reference of XdataSource to your C#.NET project.



Use the reference



```
using System;
using XDataSource;
```

The XDataSource is developed as an interface to first validate, then connect to SPPID Plant. To connect to a plant, create a text file with following information in same order without misspelled words. In case the reference does not connect correctly, this file should be checked for any misspelled names.

Database Server Name

Database Name

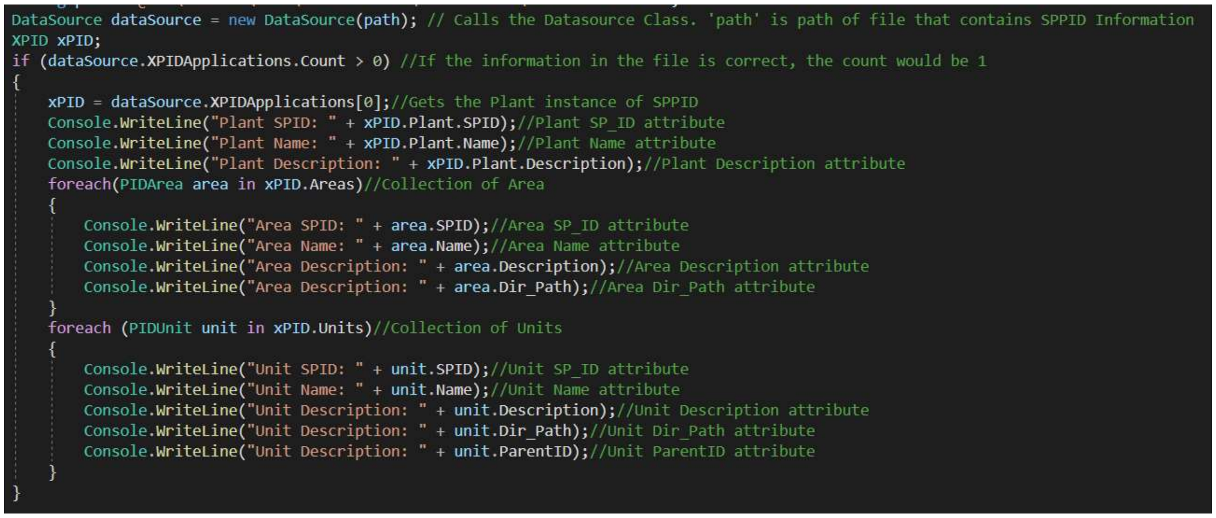
Plant Schema Name

Plant DD Schema Name

SPPID Schema Name

SPPID DD Schema Name

Save the text file at a path. This full path with file name will be used by XDataSource in following snippet.



```
DataSource dataSource = new DataSource(path); // Calls the Datasource Class. 'path' is path of file that contains SPPID Information
XPID xPID;
if (dataSource.XPIDApplications.Count > 0) //If the information in the file is correct, the count would be 1
{
    xPID = dataSource.XPIDApplications[0];//Gets the Plant instance of SPPID
    Console.WriteLine("Plant SPID: " + xPID.Plant.SPID);//Plant SP_ID attribute
    Console.WriteLine("Plant Name: " + xPID.Plant.Name);//Plant Name attribute
    Console.WriteLine("Plant Description: " + xPID.Plant.Description);//Plant Description attribute
    foreach(PIDArea area in xPID.Areas)//Collection of Area
    {
        Console.WriteLine("Area SPID: " + area.SPID);//Area SP_ID attribute
        Console.WriteLine("Area Name: " + area.Name);//Area Name attribute
        Console.WriteLine("Area Description: " + area.Description);//Area Description attribute
        Console.WriteLine("Area Description: " + area.Dir_Path);//Area Dir_Path attribute
    }
    foreach (PIDUnit unit in xPID.Units)//Collection of Units
    {
        Console.WriteLine("Unit SPID: " + unit.SPID);//Unit SP_ID attribute
        Console.WriteLine("Unit Name: " + unit.Name);//Unit Name attribute
        Console.WriteLine("Unit Description: " + unit.Description);//Unit Description attribute
        Console.WriteLine("Unit Description: " + unit.Dir_Path);//Unit Dir_Path attribute
        Console.WriteLine("Unit Description: " + unit.ParentID);//Unit ParentID attribute
    }
}
```

In the above example, following has been achieved:

```
DataSource dataSource = new DataSource(path); // Calls the Datasource Class. 'path' is path
of file that contains SPPID Information
```

This command connects to the base DataSource class, which is an entry point in the Automation. The ‘path’ variable is the same path of above text file.

Once DataSource is initialised, it validates if the provided information is valid. If valid, a collection of SPPID Application is returned, which contains one instance of XPID.

```
if (dataSource.XPIDApplications.Count > 0) //If the information in the file is correct, the
count would be 1
```

Now the application is collected in XPID class.

```
xPID = dataSource.XPIDApplications[0];//Gets the Plant instance of SPPID
```

Once it a valid class is returned, your program is set to dive in for all information from SPPID Database.

To check if everything is working, you can fetch the Plant, Area, Unit Information from the Database.

```
Console.WriteLine("Plant SPID: " + xPID.Plant.SPID);//Plant SP_ID attribute
Console.WriteLine("Plant Name: " + xPID.Plant.Name);//Plant Name attribute
Console.WriteLine("Plant Description: " + xPID.Plant.Description);//Plant Description
attribute
foreach(PIDArea area in xPID.Areas)//Collection of Area
{
    Console.WriteLine("Area SPID: " + area.SPID);//Area SP_ID attribute
    Console.WriteLine("Area Name: " + area.Name);//Area Name attribute
    Console.WriteLine("Area Description: " + area.Description);//Area Description attribute
    Console.WriteLine("Area Description: " + area.Dir_Path);//Area Dir_Path attribute
}
foreach (PIDUnit unit in xPID.Units)//Collection of Units
{
    Console.WriteLine("Unit SPID: " + unit.SPID);//Unit SP_ID attribute
    Console.WriteLine("Unit Name: " + unit.Name);//Unit Name attribute
    Console.WriteLine("Unit Description: " + unit.Description);//Unit Description attribute
    Console.WriteLine("Unit Description: " + unit.Dir_Path);//Unit Dir_Path attribute
    Console.WriteLine("Unit Description: " + unit.ParentID);//Unit ParentID attribute
}
```

Following properties are available for each of these classes:

PIDPlant:

- SPID
- Name
- Description

PIDArea

- SPID
- Name
- Description
- Dir_Path

PIDUnit

- SPID
- Name
- Description
- Dir_Path
- ParentID

Accessing Drawings

There're 2 ways to access Drawings in XPID.

```
PIDDrawings drawings;
drawings =  xPID.GetDrawings();//Get all the Active Drawings from the Plant
drawings = xPID.GetDrawings(ColumnValue,ColumnName);//Gets specific Active Drawings from the
Plant
```

The first Method gets all the Drawings from Active status from the database.

The second Method is an overload Method, where you can filter your search based on column name and value.

Note: XPID collects the Drawings only in Active Status

The PIDDrawing class has following attributes:

SPID

Name

Description

DrawingNumber

Title

PlantGroup

Template

Attributes

PIDAttributes

PIDAttribute is an important class. Since all attributes are not available for each class, every information from each table is stores in PIDAttributes class. This information can be further accessed by looping through each PIDAttribute.

```
PIDDrawing drawing = drawings[0];
foreach(PIDAttribute attribute in drawing.Attributes)
{
   Console.WriteLine(attribute.Name);
   Console.WriteLine(attribute.Value);
}
```

PIDAttribute has following properties:

Name

Value

PIDFilter And PIDCriterias

Before moving further to PlantItems, let's look at Filter and Criteria.

In order to access any Items, filters can be created using multiple criteria. Following is a sample of using PIDFilter, PIDCriterias to get the collection of PIDPipeRuns:

```
PIDCriterias criterias = new PIDCriterias();
PIDCriteria criteria = new PIDCriteria();
PIDFilter filter = new PIDFilter();
criteria.AttributeName = "ItemTag";
criteria.Operator = "Is Not";
criteria.AttributeValue = "NULL";
criterias.Add(criteria);
filter.ItemType = "PipeRun";
filter.Criterias = criterias;

PIDPipeRuns pipeRuns = xPID.GetPipeRuns(filter);
Console.WriteLine(pipeRuns.Count);
```

Following are the properties of respective classes

PIDFilter:

- ItemType
- Criterias

PIDCriteria

- AttributeName
- AttributeValue
- Operator
- Conjunctive

PIDPipeRuns

In the above example, we've seen how we can get a collection of PIDPipeRuns by applying a filter. Once we get the collection, we can loop through each of the PIDPipeRun and read its properties using direct properties, or PIDAttributes. Please note the properties Representations and Cases. We will discuss them further

```
foreach(PIDPipeRun pipeRun in pipeRuns)
{
	Console.WriteLine(pipeRun.ItemTag);
}
```

Following are the properties of PIDPipeRun class:

- SPID
- ItemTag
- TagSequenceNo
- NominalDiameter
- InsulationType
- InsulationPurpose
- InsulationThickness
- Description
- Name

PlantGroupID

ConstructionStatus

ConstructionBy

InsulationSpec

HeatTraceMedium

FlowDirection

FluidSystem

PipeRunClass

CorrosionAllowance

Attributes

Representations

Cases

PIDRepresentation

PIDRepresentation class consists all information from T_Representation. As one PlantItem may have more than one representation in some cases, a separate Class serves the purpose. PIDPipeRun has Representations property which can be accessed as below:

```
PIDRepresentations representations = pipeRun.Representations;
foreach(PIDRepresentation representation in representations)
{
    Console.WriteLine(representation.SPID);
}
```

PIDRepresentation has following properties:

SPID

RepresentationType

FileName

ModelItemID

RadLayer

ExportLayer

RepresentationClass

DrawingID

Attributes

PIDCase, PIDCaseProcess and PIDCaseControl

In order to display Cases, CaseProcess, and CaseControl, following code can be used:

```
PIDCases cases = pipeRun.Cases;
foreach(PIDCase @case in cases)
{
    foreach(PIDCaseControl control in xPID.GetCaseControls(@case.SPID))
    {

    }
    foreach (PIDCaseProcess process in xPID.GetCaseProcesses(@case.SPID))
    {

    }
}
```

Following are the properties of respective Classes:

PIDCase

SPID

Description

UpdateCount

Name

FluidType

CaseType

ModelItemID

CaseClass

FluidSystem

Corrosive

Erosive

Toxic

Attributes

PIDCaseProcess

SPID

UpdateCount

Temperature

TemperatureSI

Pressure

PressureSI

SpecificGraivity

SpecificGraivitySI

Quality

CaseID

Viscosity

ViscositySI

FlowRate

FlowRateSI

FluidState

VaporPressure

VaporPressureSI

Comppressibility

ComppressibilitySI

CpCvRatio

CpCvRatioSI

CriticalPressure

CriticalPressureSI

MassDensity

MassDensitySI

MassFlowRate

MassFlowRateSI

MolecularWeight

MolecularWeightSI

PourPintTemp

PourPintTempSI

Velocity

VelocitySI

Attributes

PIDCaseControl

SPID

UpdateCount

Pressure

PressureSI

Quality

LiquidLevel

LiquidLevelSI

LevelReference

LevelReferenceSI

CaseID

ElectCurrent

ElectCurrentSI

Voltage

VoltageSI

VaccuumPressure

VaccuumPressureSI

Attributes

Disclaimer

1. This is a free to use Class Library and is not for sale. No License is ever required to use or deploy this Library.

2. There's no Training or Session cost demanded ever by the Developer. Please refrain to any fee demand for training on this Library, as all the required information is posted on Internet.

3. This Class library is solely built on information or knowledge gained personally.

4. This library is Not a replacement of Hexagon's original Automation reference. Please use it on projects after proper assesment.

5. The Library is developed only to read the information from Database. This application does not have any DDL or DML query incorporated.

6. Any issue/concern can be communicated to sparks.bhaskar@gmail.com

7. There're more to come in future. Please stay tuned on LinkedIn, Twitter, Insta @bhaskarsrivatsa