AUTODESK® AEC PLANT SOLUTIONS
AUTOCAD P&ID
AUTOCAD PLANT 3D

External Database Reference Sample Application

Autodesk®

Revision 1.0

Contents

External Database Reference Model Sample

Overview

The AutoCAD P&ID project database model is an extensible model which can be configured to reference data tables of an external database using either a Virtual Mapping, or an Existing Mapping:

- Virtual Mapping: the column in the external database table does not become a property in the project hierarchy; it will appear as a new property, and can have a name that's different from the external database column name.
- Existing Mapping: the column in the external database is mapped to an existing property in the project table.

The project database interacts with external databases through the OLE DB .NET provider, and can connect to most database systems with an installation of the corresponding OLE DB providers on the client system.

Additional external table columns can be referenced from the project table by joining the project table with an external database table; rows in the project and external database tables are bound by the specification of a foreign key (column) for the project table, and a primary key (column) for the external database table.

This document describes a sample database extension where columns from an external database (`autodesk.accdb`) are added to the project database.

Illustration

External database : `autodesk.accdb`
External table : `Table1`

Key column : `Field1`
Extension columns : `Field2, Field3, Field4`

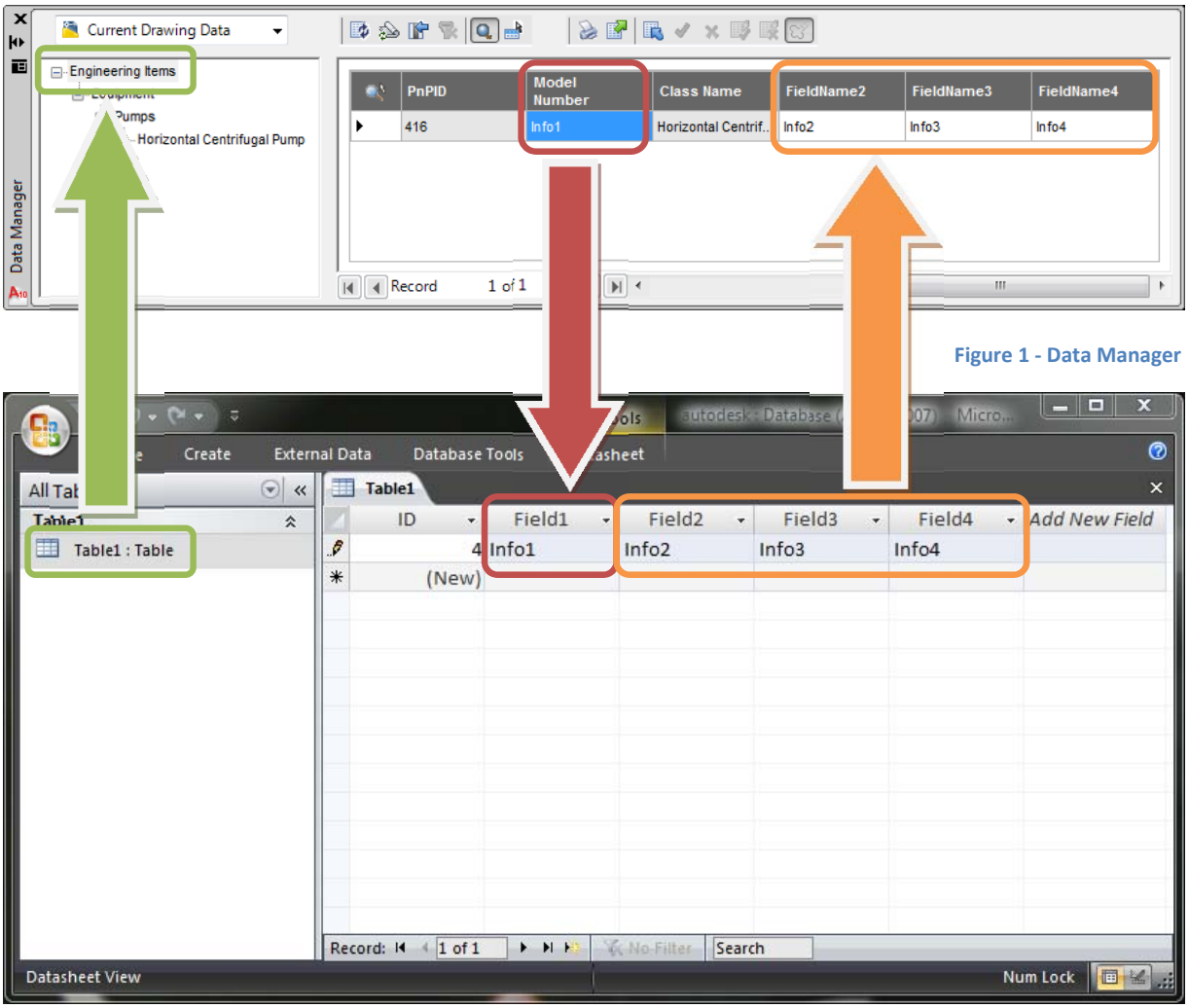

Figure 1 - Data Manager

Figure 2 - External Data Source in Access 2007

Using the External Reference Manager (XDbReferenceManagerArx.dll)

To use the tool, start AutoCAD P&ID 2010, and 'NETLOAD' the `XDbReferenceManagerArx.dll` .NET ObjectARX application. In the Project Manager, open the project that you want to manage external references for.

Creating a Data Source

Step 1 – Invoke External Reference Manager

Invoke the External Reference Manager dialog with the 'PLANTXDBMANAGER' command.

Step 2 – Create an External Data Source

Create a data source named "autodesk_datasource" by clicking the "Add" button in the "Data Sources" tab; multiple data sources can be defined.

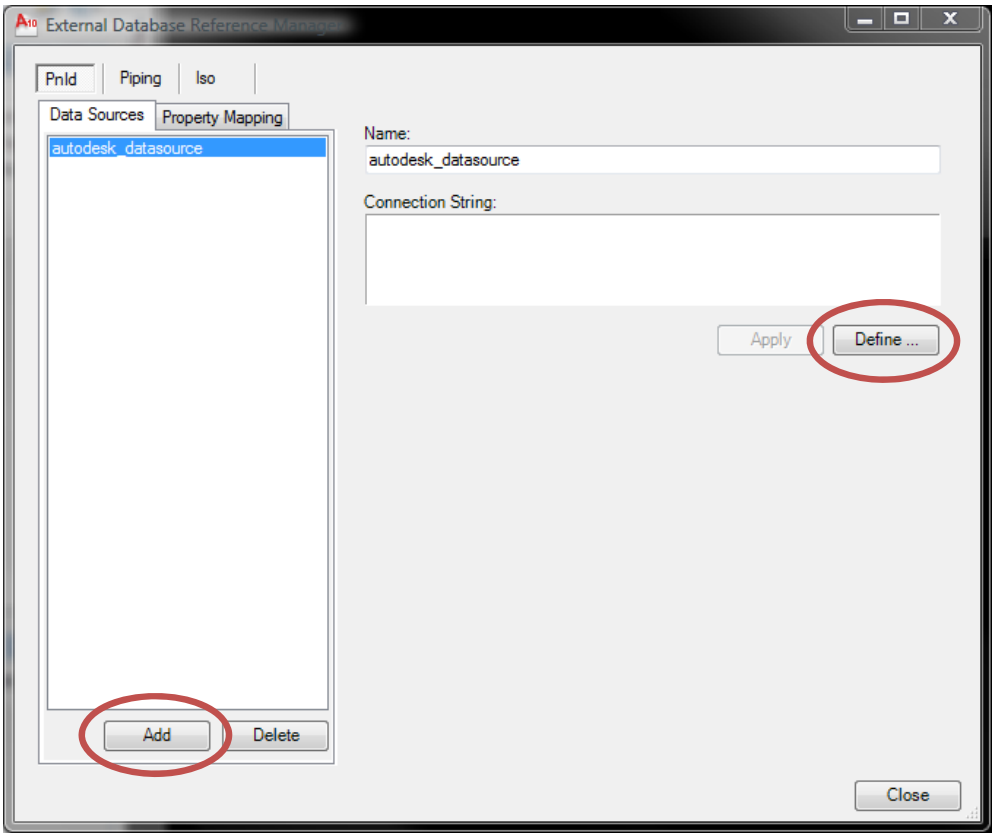

Figure 3 - Defining a Data Source

Step 3 – Define the Data Source

Click the "Define..." button to specify the connection properties in the "Data Link Properties" dialog. In this example, we're using the "Microsoft Office 12.0 Access Database Engine" to connect to an Microsoft Access database at `I:\catalogs\autodesk.accdb`.

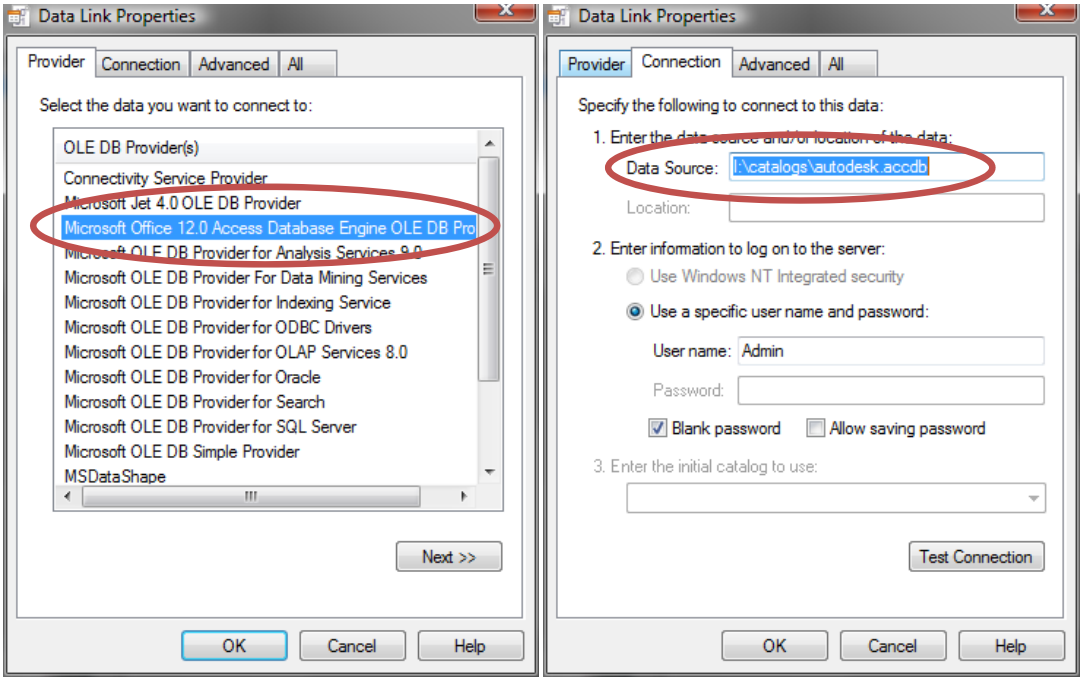

Figure 4 - Defining a Database Connection

Step 4 – Create Property Mappings

Once the data source is created, switch to the "Property Mapping" tab to define a property mapping. Toggle the "Hide Unused" button to show all available classes in the project hierarchy. Select the "EngineeringItems" table (i.e. the table that you want to extend), and click the "Map Virtual Property" button to create a new property mapping set; multiple property mapping sets can be defined. Alternatively, you can also click the "Map Existing Property" button to create mappings to existing columns in the project table.

Select "autodesk_datasource" as the Referenced data source, and "Table1" as the Referenced table [see BLUE box in Figure 5 below].

Once the referenced table is selected, map the "ModelNumber" column in the project database to the "Field1" column in the external database as shown [see RED box in Figure 5].

The last thing to do in this step, is to specify the columns from the external database table and how they will display when extending the "EngineeringItems" table [see ORANGE box in Figure 5].

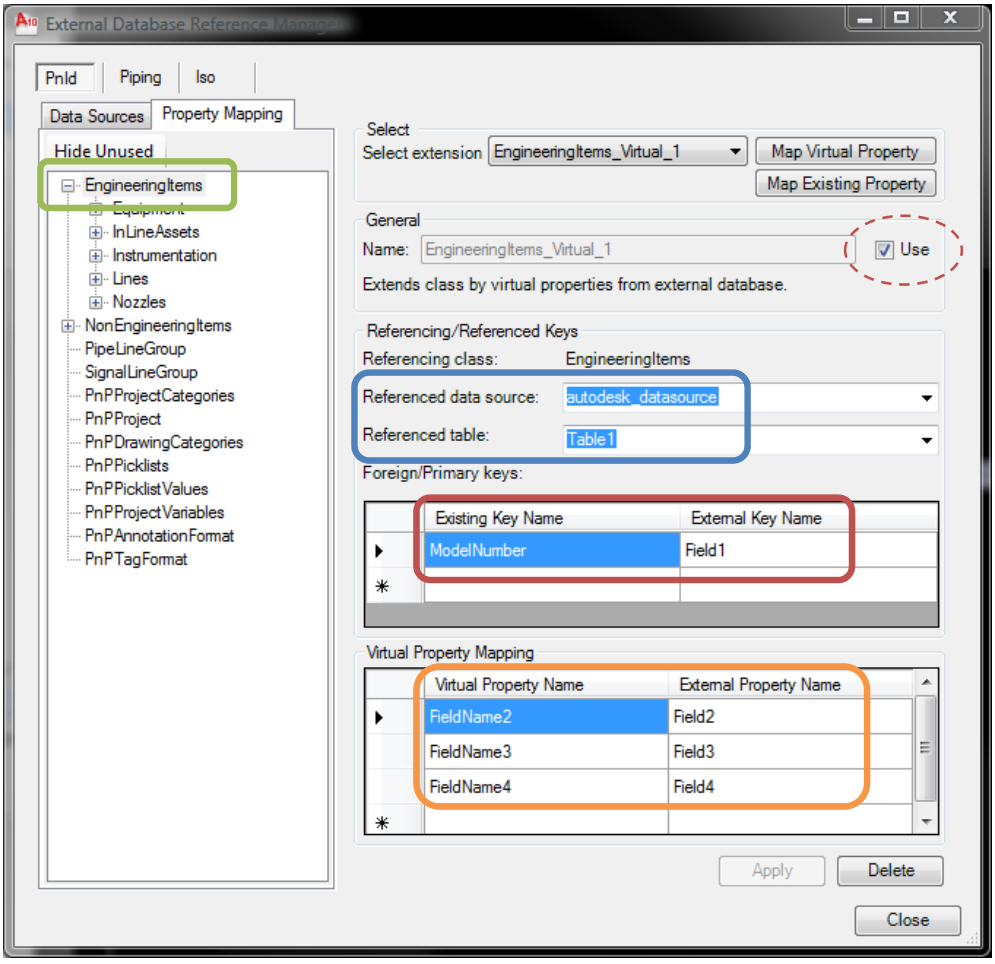

Figure 5 - Defining a Virtual Property Mapping

If you're creating mappings to existing project table columns, specify the columns from the external database table and the columns in the project table that they should map to [see ORANGE box in Figure 6 below].

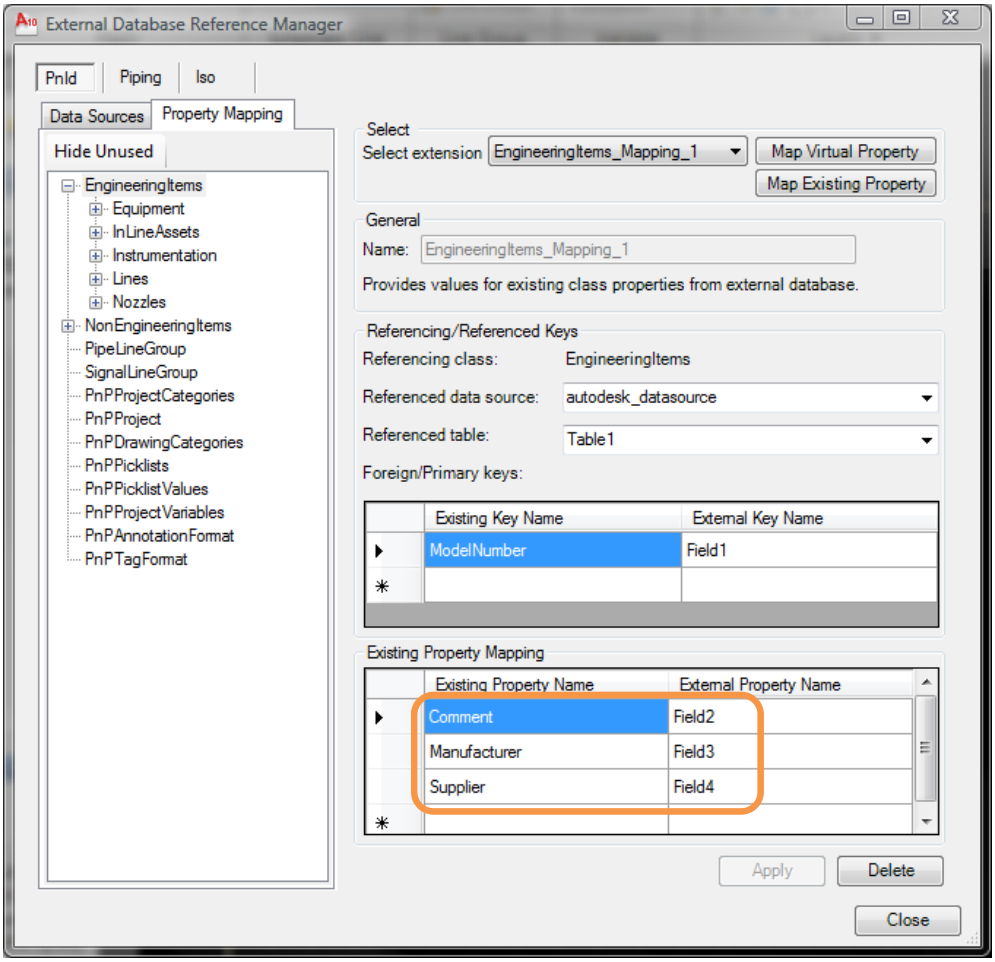

Figure 6 - Defining an Existing Property Mapping

Step 5 – Turn on and Apply

Check the “Use” checkbox in the General group box [see RED oval in Figure 5 above], this turns on the reference so that it is used. To finish and exit the dialog box, click the “Apply” button followed by the “Close” button.

Done!

Invoke the Data Manager, and scroll to the right. Notice that three new columns now exist for the “EngineeringItems” table. If you type in “Info1” for the “ModelNumber” column of a record, “FieldName2” through “FieldName4” are automatically updated.

If you've created mappings to existing project columns, after you've typed in "Info1" for the "ModelNumber" column of a record, select and right-click the "Engineering Items" node in the class tree and select the "Reload Mapped Properties..." menu item[1] to populate the mapped property values in the selected table.

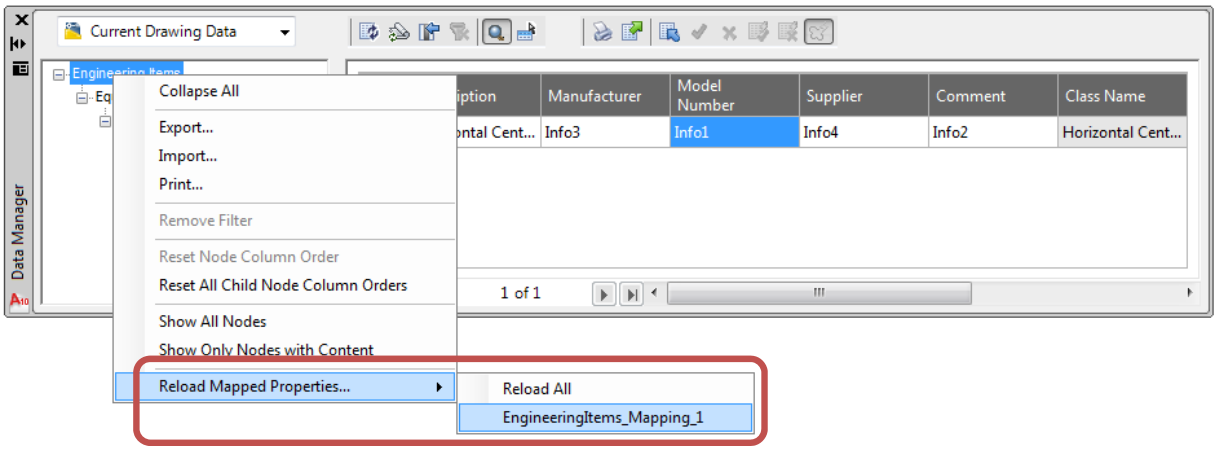

Figure 7 - Reloading Existing Property Mappings

[1] The "Reload Mapped Properties..." context menu item is only available in AutoCAD Plant 3D 2010, and will not be shown in AutoCAD P&ID 2010.

Additional Notes

Configuration Files

The data source connection parameters and property mapping definitions are stored in ".edf" configuration files in the project folder; a configuration file is created for and named after each project part (i.e. `PnId.edf`, `Piping.edf`, `Iso.edf`, etc.).

The use-status settings for virtual property mappings are user and project specific, and are saved as XML configuration files in the user profile application data folder for each project (e.g. `C:\Users\<User_Name>\AppData\Roaming\Autodesk\AutoCAD Plant 3D 2010\R18.0\enu\Support\PnPProjects\<Project_Name>`).

Module List

- XDbReferenceUI.dll
  - Common User Interface (UI) controls assembly.
- XDbReferenceManagerArx.dll
  - Contains the AutoCAD ObjectARX .NET application and the 'PLANTXDBMANAGER' command implementation; uses controls from the XDbReferenceUI assembly.
- XDbReferenceManager.exe
  - Stand-alone External Database Reference Manager application; uses controls from the XDbReferenceUI assembly.

Building the Sample Application

This sample application is built on Visual Studio 2008 and with the PnPSDK. The PnPSDK requires AutoCAD 2010 ObjectARX. You can find the AutoCAD 2010 ObjectARX at **www.autodesk.com/objectarx**.

Microsoft Visual Studio requirements and setup are described in *ObjectARX Introductory Concepts > Overview of ObjectARX > Getting Started from the ObjectARX Developer's Guide* (**arxdev.chm**).

PnPSDK files are installed with AutoCAD P&ID or AutoCAD Plant 3D into the `\<AutoCAD P&ID|Plant 3D>\PnPSDK` folder. You may overlay the PnPSDK onto ObjectARX by copying the contents of the PnPSDK folder to the ObjectARX folder, which is `C:\ObjectARX 2010` by default. The SDK files share existing ObjectARX folders, but no files are overwritten.

To build the sample application, copy the XDbReference folder to the ObjectARX folder, and build the `XDbReference.sln` solution in Visual Studio.

Known Issues

- The "Reload Mapped Properties..." Data Manager context menu item is not available in AutoCAD P&ID 2010, and any previous release of AutoCAD P&ID.

  An additional command or reactor implementation is required to trigger a refresh of the mapped project column values when the foreign key value of a row is changed. A sample implementation can be found in Appendix A.

- The XDbReferenceManager.exe stand-alone application has dependencies on two additional assemblies:
  - PnPSQLiteEngine.dll
  - System.Data.SQLite.dll

  These assemblies can be obtained from the AutoCAD P&ID or AutoCAD Plant 3D program folder, and need to be manually copied to the XDbReferenceManager.exe application folder, or added to the system path.

Appendix A - Equivalent C# Code

The code below is equivalent to the interactive steps shown in the "Using the External Reference Manager (XDbReferenceManagerArx.dll)" section. It illustrates how a developer can use the APIs to extend the project database with virtual properties at run-time.

```
/* This is a sample of adding additional columns to a table that are
 * defined in an external database.
 */
[CommandMethod("xdb_sample", CommandFlags.Modal)]
public static void xdb()
{
    try
    {
        /* The first thing we have to do is ask for the current data
         * links manager. From this object, we can get to the
         * lower-level database layer API known as PnPDataObjects.
         */
        StringCollection names = DataLinksManager.GetLinkManagerNames();
        DataLinksManager dlm = DataLinksManager.GetManager(names[0]);

        /* The XDb reference manager tracks all the external data
         * sources and references. References can be set up to use the
         * "source" model that means to copy data from fields in the
         * external database into the data-cache or the "extension"
         * model where additional columns are added to a table in
         * the data-cache.
         */
        PnPDatabase db = dlm.GetPnPDatabase();
        PnPXDbReferenceManager mgr = db.XDbReferenceManager;

        /* Data source defines how to connect to a database. */
        PnPXDbDatasource ds = new PnPXDbDatasource("autodesk_datasource");

        /* Connect to an Access database */
        ds.ConnectionString     =     "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=I:\\catalogs\\autodesk.accdb;Persist Security Info=false;
        ds.Connect(string.Empty, string.Empty);

        /* Add the newly created data source to the XDb manager */
        mgr.Datasources.Add(ds);

        /* Create the reference to external data. This sample is
         * using the "extension model" where fields from the
         * external data source are displayed in the data manager.
         * The foreign key maps to the primary key in the external
         * database and this is how the two tables are kept
         * synchronized.
         */
        PnPXDbReference dbref_ext =
          new PnPXDbReference("EngineeringItems_Virtual_1");
        dbref_ext.IsExtension = true;

        /* Project database table and field */
        dbref_ext.Referencing.TableName = "EngineeringItems";
        dbref_ext.Referencing.ForeignKey.Add("ModelNumber");
```

```
        /* External database table and field */
        dbref_ext.Referenced.DatasourceName = "autodesk_datasource";
        dbref_ext.Referenced.TableName = "Table1";
        dbref_ext.Referenced.PrimaryKey.Add("Field1");

        /* The fields in the external database to extend the
         * project database with. The first parameter is the
         * local name in the project manager (displayed as column
         * header).
         */
        PnPXDbColumnMapCollection cm = dbref_ext.ColumnMap;
        cm.Add(new PnPXDbColumnMapEntry("FieldName2", "Field2"));
        cm.Add(new PnPXDbColumnMapEntry("FieldName3", "Field3"));
        cm.Add(new PnPXDbColumnMapEntry("FieldName4", "Field4"));

        /* Add the reference to the XDb manager */
        mgr.References.Add(dbref_ext);

        /* Once the XDb manager is tracking the data source and
         * the reference then we add the "extension" to a specific
         * table. In this example, the EngineeringItems table gets
         * the extension. All tables derived from EngineeringItems
         * will also display the fields.
         */
        PnPTable tbl = db.Tables["EngineeringItems"];
        tbl.AddRuntimeExtension("EngineeringItems_Virtual_1");
    }
    catch (System.Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message.ToString());
    }
    finally
    {
    }
}
```

The code below is equivalent to the "Reload Mapped Properties..." functionality shown in Figure 7. It illustrates how a developer can refresh Existing Mapping column values programmatically.

```
static void ReloadAllXDbReferences(DataLinksManager dlm, String tableName)
{
    PnPDatabase db = dlm.GetPnPDatabase();
    if (db == null || !db.Tables.Contains(tableName))
        return;

    PnPTable table=db.Tables[tableName];
    foreach (PnPXDbReference xRef in db.XDbReferenceManager.References)
    {
        if (!table.IsKindOf(xRef.Referencing.TableName))
            continue;

        if (!xRef.IsExtension)
            xRef.Fill();
    }
}
```

Appendix B - Integrated Plant Engineering Model (AU 2008)