CoilStore

North Star BlueScope

NSBS System for Storing Coil Data

This system provides the means to store and retrieve coil data in a SQL2012 (or better) database.

Dan Houck (AS&E)

18 Oct 2014

10/18/2014

COILSTORE

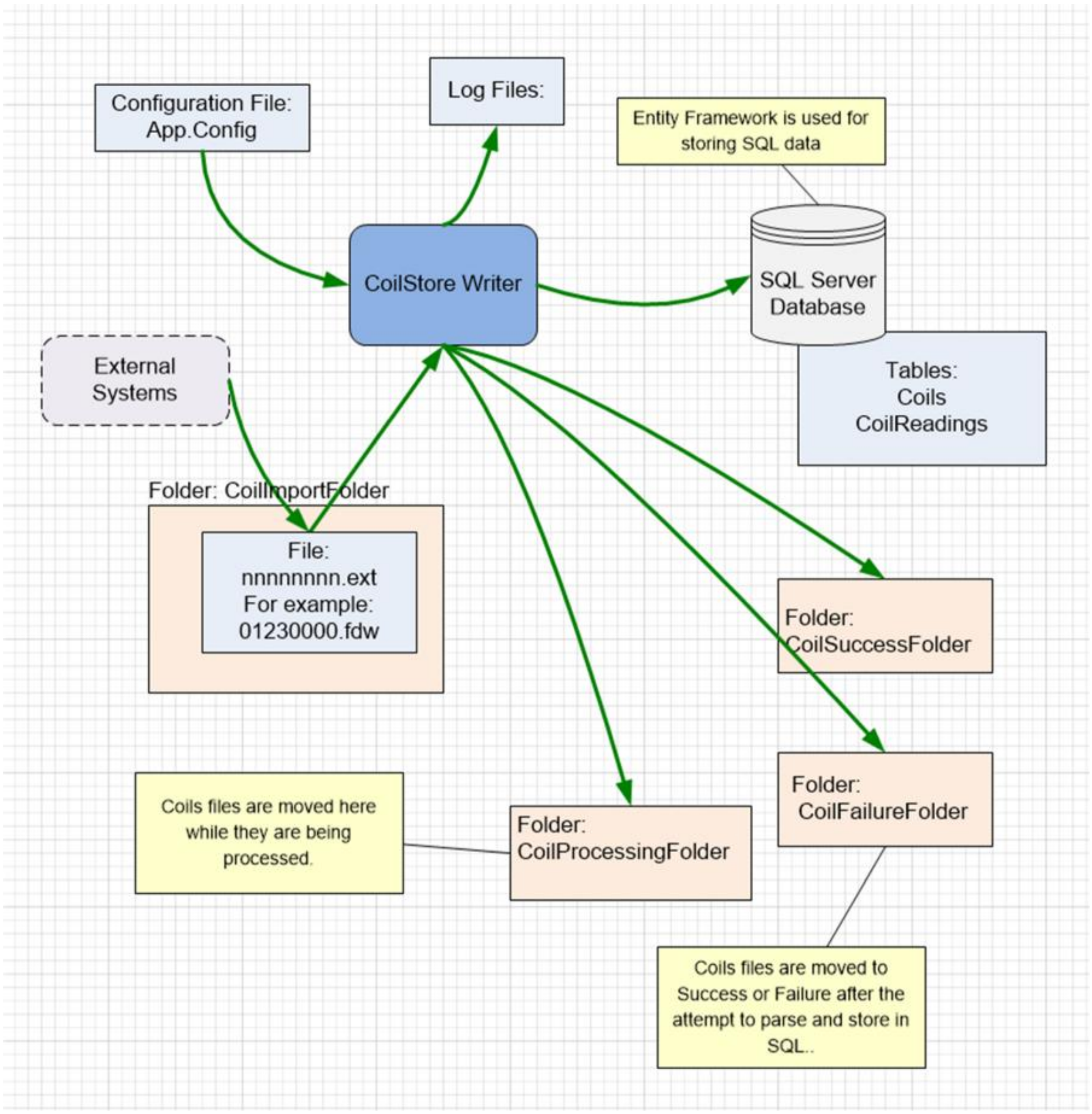
NSBS System for Storing Coil Data

Contents

Overview

CoilStore is designed to take raw coil data reading files and place them into a SQL Server relational database.

The following diagram illustrates the major points, which are discussed in more detail below.


Configuration File: App.Config
Log Files:
Entity Framework is used for storing SQL data
CoilStore Writer
SQL Server Database
Tables: Coils CoilReadings
External Systems
Folder: CoilImportFolder
File: nnnnnnnn.ext For example: 01230000.fdw
Folder: CoilSuccessFolder
Folder: CoilFailureFolder
Folder: CoilProcessingFolder
Coils files are moved here while they are being processed.
Coils files are moved to Success or Failure after the attempt to parse and store in SQL..

Overview Diagram

OPERATION

External systems place data for a coil as ASCII files into the coilImport folder. These files are expected to have a file name as an 8 digit number, and have coil collection types identified by their three character extension of either “fdh” or “fdw”. Examples are 01350000.fdh and 01350000.fdw.

More information on these files is available in the appendix.

The CoilStore process watches for files in this folder, and moves them to the file named by the setting CoilProcessingFolder (for example d:\(data)\CoilImport\Processing) so the files can be processed by the CoilStoreEngine. Upon processing the file is placed into one of the two subfolders:

Setting	Example
CoilFailureFolder	d:\(data\CoilImport\Failure
CoilSuccessFolder	d:\(data\CoilImport\Success

Files are watched using the system FileWatcher, but for insurance there is also a setting FilePollSeconds (example value: 15) that will look for the files periodically.

Any logging – informational or otherwise – is placed in a date-stamped file according to the setting LogFolder. An example value is d:\logs.

The data is parse and – if successful – placed in the SQL database. If a duplicate coil is found, then the existing coil is deleted and the new data replaces it.

The data is inserted with a constructed Coil Key (BigInt) that has the structure:

Yymmddhhmmssd,

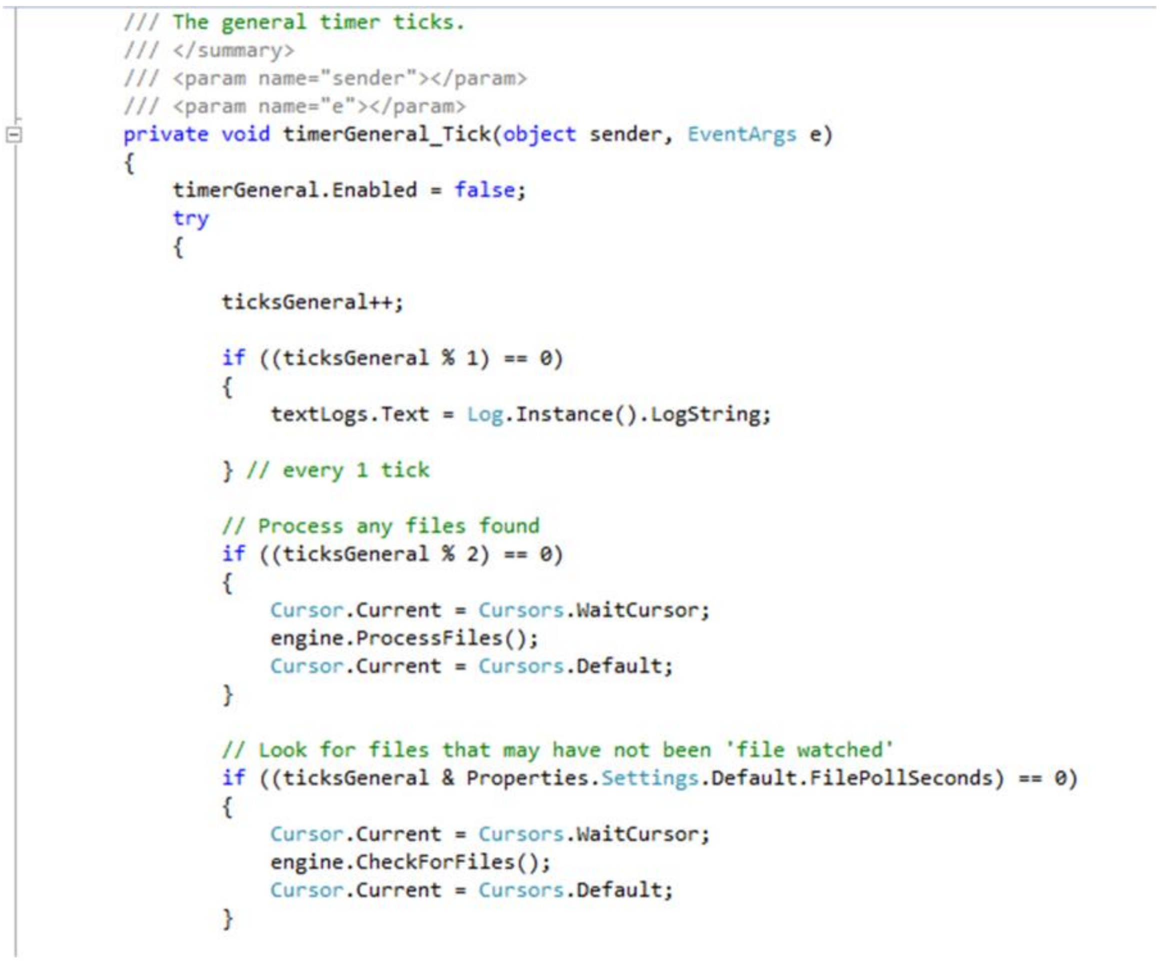
Where the time is the time the coil was processed (received in the ASCII file), and the “d” on the end is 1 if the file extension was “fdh” and 2 if it was “fdw”.

All of the data is placed into the SQL Server database into two tables Coils or CoilReading. More information on these tables is available in the appendix.

The Program

The program is a WinForms application. It sets up a FileWatcher to look for files being received, and also has a timer to check for files the FileWatcher might miss, and also checks the processing queue according to file FilePollSeconds setting.

Here is a snippet for the timer method:

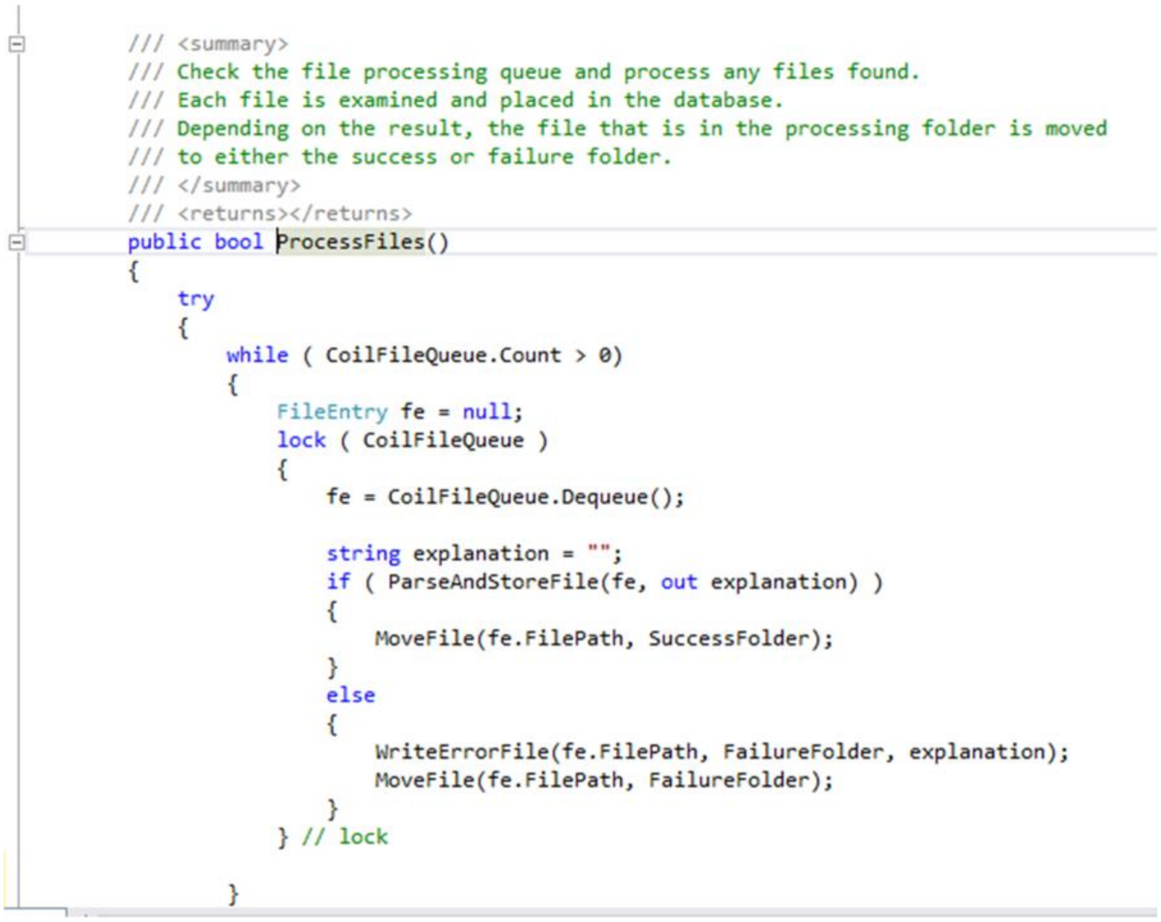


```
        /// The general timer ticks.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void timerGeneral_Tick(object sender, EventArgs e)
        {
            timerGeneral.Enabled = false;
            try
            {

                ticksGeneral++;

                if ((ticksGeneral % 1) == 0)
                {
                    textLogs.Text = Log.Instance().LogString;

                } // every 1 tick

                // Process any files found
                if ((ticksGeneral % 2) == 0)
                {
                    Cursor.Current = Cursors.WaitCursor;
                    engine.ProcessFiles();
                    Cursor.Current = Cursors.Default;
                }

                // Look for files that may have not been 'file watched'
                if ((ticksGeneral & Properties.Settings.Default.FilePollSeconds) == 0)
                {
                    Cursor.Current = Cursors.WaitCursor;
                    engine.CheckForFiles();
                    Cursor.Current = Cursors.Default;
                }
```

The important processing logic is contained within the CoilStoreEngine class, and the two most important methods are:

Method	Description
ProcessFiles()	Examine the processing queue. Process the file and update the database.
CheckForFiles()	Look for files in the folders and place them in the processing queue.

Here is a portion of the ProcessFiles Method:



```
/// <summary>
/// Check the file processing queue and process any files found.
/// Each file is examined and placed in the database.
/// Depending on the result, the file that is in the processing folder is moved
/// to either the success or failure folder.
/// </summary>
/// <returns></returns>
public bool ProcessFiles()
{
    try
    {
        while ( CoilFileQueue.Count > 0)
        {
            FileEntry fe = null;
            lock ( CoilFileQueue )
            {
                fe = CoilFileQueue.Dequeue();

                string explanation = "";
                if ( ParseAndStoreFile(fe, out explanation) )
                {
                    MoveFile(fe.FilePath, SuccessFolder);
                }
                else
                {
                    WriteErrorFile(fe.FilePath, FailureFolder, explanation);
                    MoveFile(fe.FilePath, FailureFolder);
                }
            } // lock

        }
```

To separate the logic and make sure there are no threading problems, the file processing requests are placed into a FIFO queue (CoilFileQueue) by CheckForFiles, and pulled from the queue by ProcessFiles. This is all done within a lock on the queue.

For each coil file processed there is a single master record created in Coils and a number of child records – one for each reading – in CoilReadings.

The readings are actually stored in two different ways:

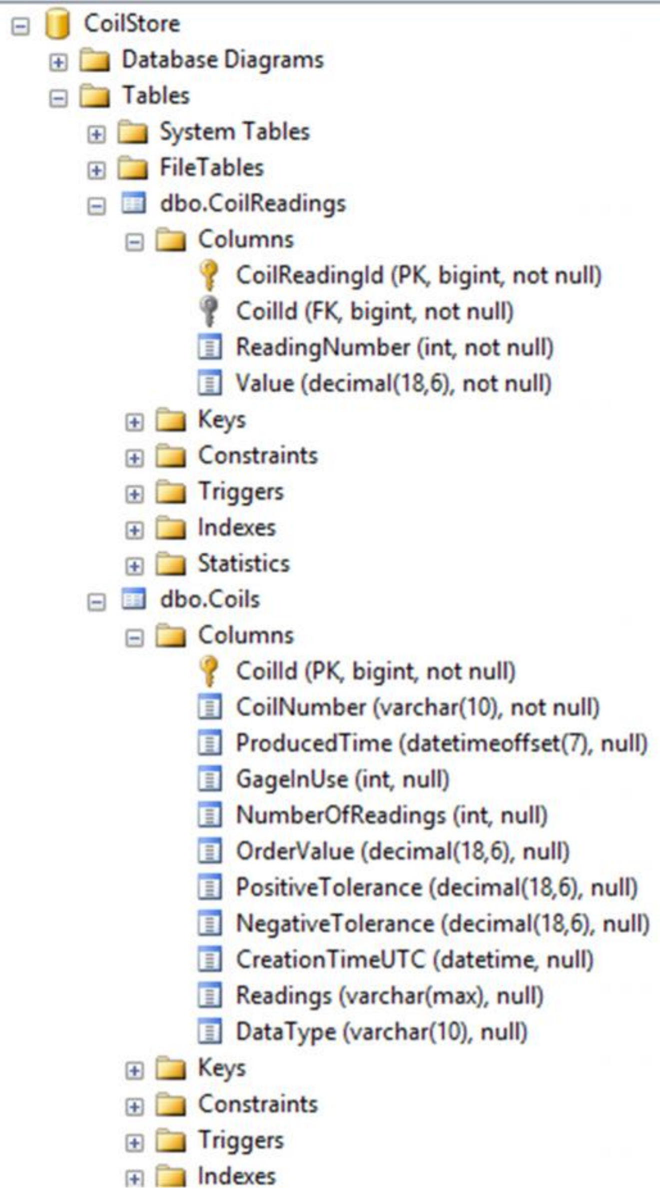
1. Relationally, with one record per reading, and
2. In a comma list within the master record.

There are several reasons for doing this, here are a few:

1. By making it relational, it will be easier to use with some off-the-shelf packages.
2. By storing them in the master record, it will be more efficient when a high-volume/capacity application is needed.
3. Since they are within the master, the child records can always be re-created.

APPENDIX – DATABASE STRUCTURE

The database consists of two tables with a foreign key on the CoilId field.



```
CoilStore
  Database Diagrams
  Tables
    System Tables
    FileTables
    dbo.CoilReadings
      Columns
        CoilReadingId (PK, bigint, not null)
        CoilId (FK, bigint, not null)
        ReadingNumber (int, not null)
        Value (decimal(18,6), not null)
      Keys
      Constraints
      Triggers
      Indexes
      Statistics
    dbo.Coils
      Columns
        CoilId (PK, bigint, not null)
        CoilNumber (varchar(10), not null)
        ProducedTime (datetimeoffset(7), null)
        GageInUse (int, null)
        NumberOfReadings (int, null)
        OrderValue (decimal(18,6), null)
        PositiveTolerance (decimal(18,6), null)
        NegativeTolerance (decimal(18,6), null)
        CreationTimeUTC (datetime, null)
        Readings (varchar(max), null)
        DataType (varchar(10), null)
      Keys
      Constraints
      Triggers
      Indexes
```

Tables and Fields

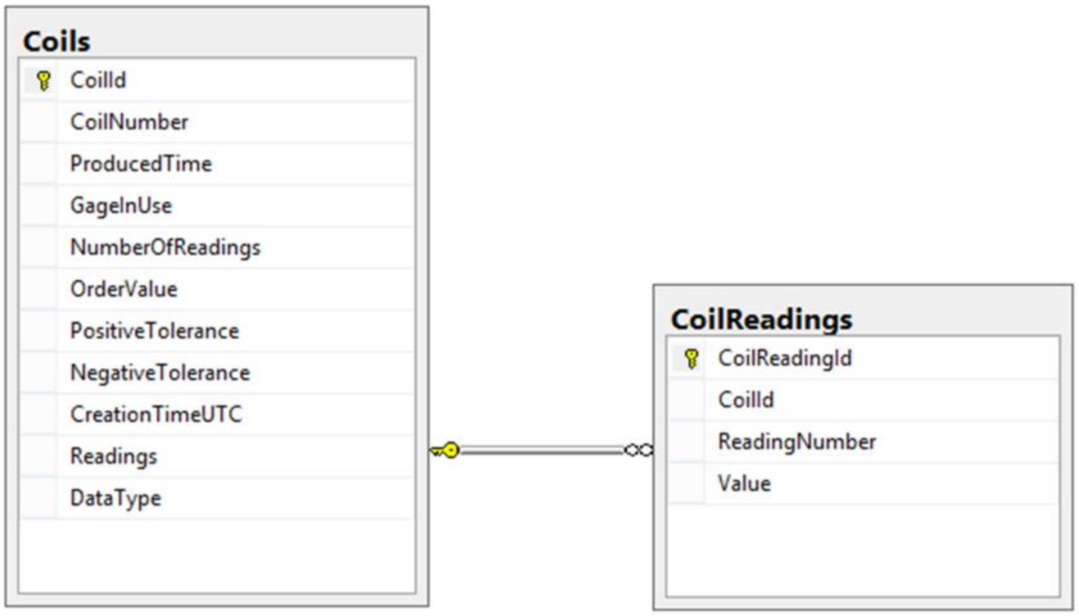
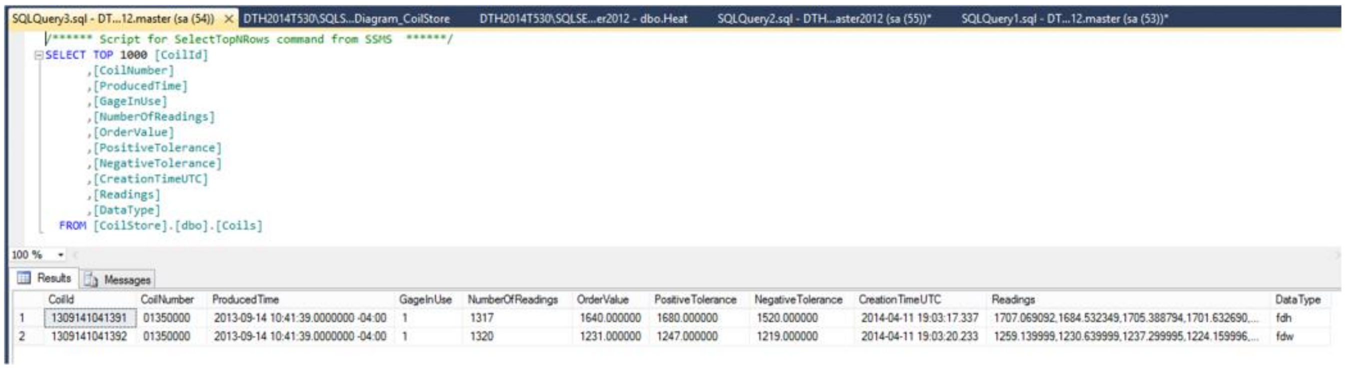


Table Relationship

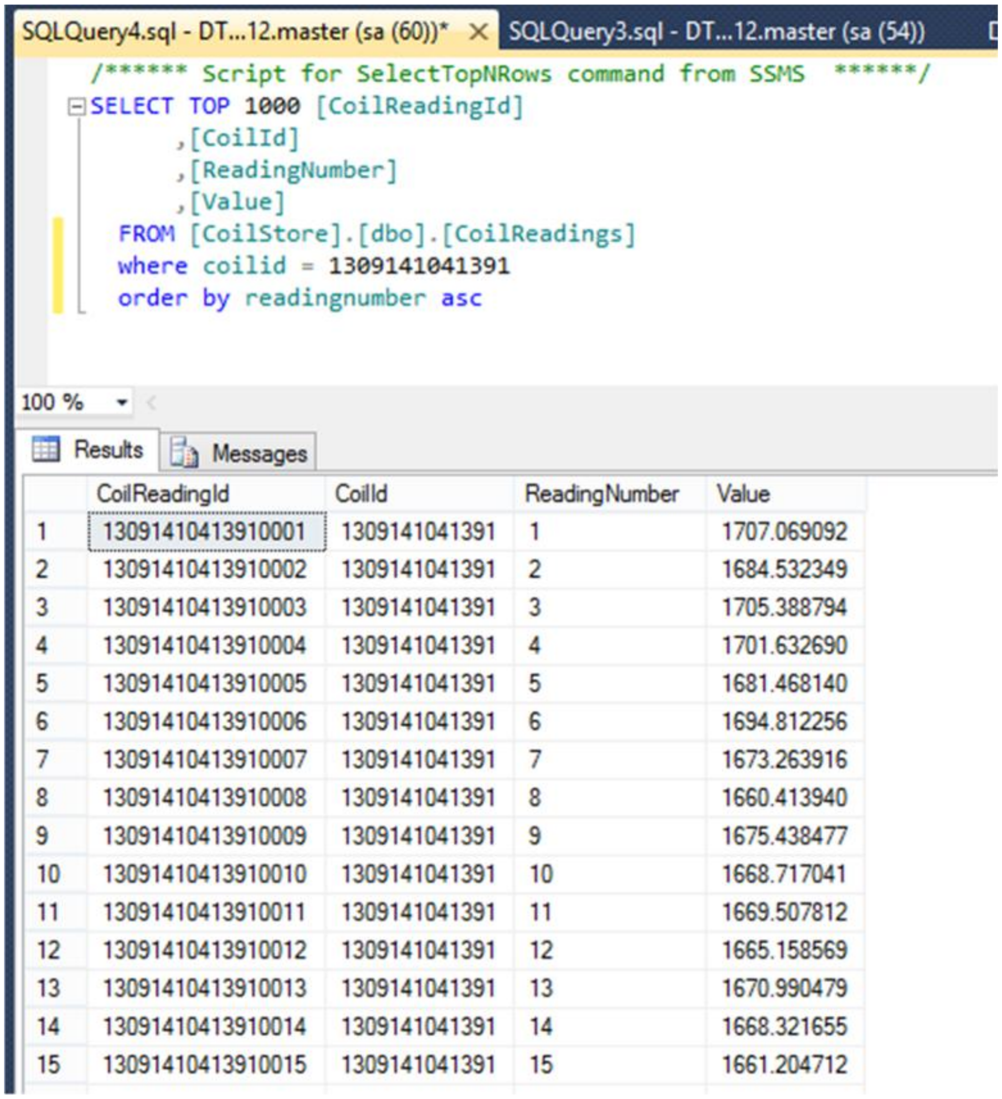


```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP 1000 [CoilId]
      ,[CoilNumber]
      ,[ProducedTime]
      ,[GageInUse]
      ,[NumberOfReadings]
      ,[OrderValue]
      ,[PositiveTolerance]
      ,[NegativeTolerance]
      ,[CreationTimeUTC]
      ,[Readings]
      ,[DataType]
  FROM [CoilStore].[dbo].[Coils]
```

	CoilId	CoilNumber	ProducedTime	GageInUse	NumberOfReadings	OrderValue	PositiveTolerance	NegativeTolerance	CreationTimeUTC	Readings	DataType
1	1309141041391	01350000	2013-09-14 10:41:39.0000000 -04:00	1	1317	1640.000000	1680.000000	1520.000000	2014-04-11 19:03:17.337	1707.069092,1684.532349,1705.388794,1701.632690,...	fdh
2	1309141041392	01350000	2013-09-14 10:41:39.0000000 -04:00	1	1320	1231.000000	1247.000000	1219.000000	2014-04-11 19:03:20.233	1259.139999,1230.639999,1237.299995,1224.159996,...	fdw

Sample Coils Data

The child records, with examples:



SQLQuery4.sql - DT...12.master (sa (60))* SQLQuery3.sql - DT...12.master (sa (54))

```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT TOP 1000 [CoilReadingId]
      ,[CoilId]
      ,[ReadingNumber]
      ,[Value]
  FROM [CoilStore].[dbo].[CoilReadings]
  where coilid = 1309141041391
  order by readingnumber asc
```

	CoilReadingId	CoilId	ReadingNumber	Value
1	13091410413910001	1309141041391	1	1707.069092
2	13091410413910002	1309141041391	2	1684.532349
3	13091410413910003	1309141041391	3	1705.388794
4	13091410413910004	1309141041391	4	1701.632690
5	13091410413910005	1309141041391	5	1681.468140
6	13091410413910006	1309141041391	6	1694.812256
7	13091410413910007	1309141041391	7	1673.263916
8	13091410413910008	1309141041391	8	1660.413940
9	13091410413910009	1309141041391	9	1675.438477
10	13091410413910010	1309141041391	10	1668.717041
11	13091410413910011	1309141041391	11	1669.507812
12	13091410413910012	1309141041391	12	1665.158569
13	13091410413910013	1309141041391	13	1670.990479
14	13091410413910014	1309141041391	14	1668.321655
15	13091410413910015	1309141041391	15	1661.204712

Sample CoilReading Data

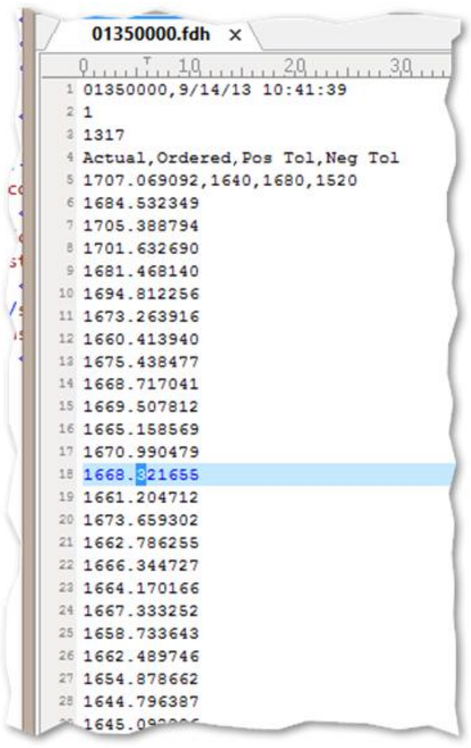
Appendix - Raw Coil File

The format of the FDH file is an 8 digit name, followed by the extension FDH.

The format of the ASCII data in the file is:

CoilNumber,ProducedDateStamp
GageInUse – (permitted values are 1 and 2)
NumberofReadings – (permitted values are 1 – 2000)
Label1(Actual),Label2(Order),Label3(Pos Tol),Label4(Neg Tol) – (Define the next row of data)
ActualVal1,OrderVal,PosTolVal,NegTolVal
ActualVal2
ActualVal3
… ActualVal(n-2)
ActualVal(n-1)
ActualVal(n)

An example of the contents is:



```
01350000,9/14/13 10:41:39
1
1317
Actual,Ordered,Pos Tol,Neg Tol
1707.069092,1640,1680,1520
1684.532349
1705.388794
1701.632690
1681.468140
1694.812256
1673.263916
1660.413940
1675.438477
1668.717041
1669.507812
1665.158569
1670.990479
1668.321655
1661.204712
1673.659302
1662.786255
1666.344727
1664.170166
1667.333252
1658.733643
1662.489746
1654.878662
1644.796387
1645.09
```

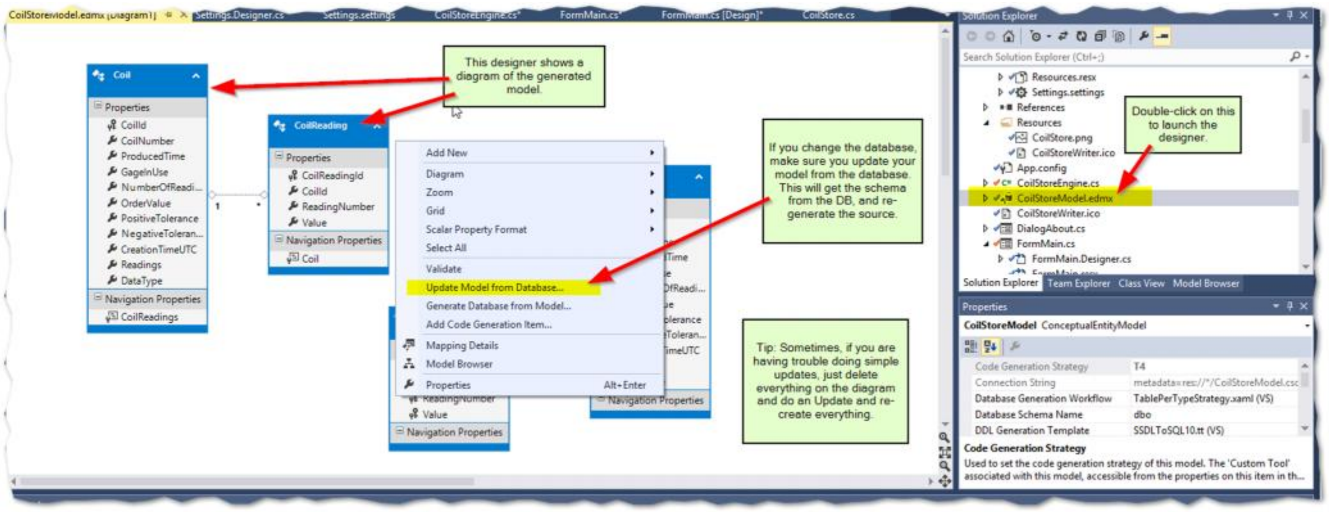
Example Received ASCII Data File

Appendix – Using Entity Framework and EDMX in Visual Studio

EF and the EDMX file is a large body of information that cannot be covered here. But perhaps a quick explanation on how to alter the database model used by CoilStore would be useful.

The whole idea of EF is to provide a model of the database that can be used by a programmer. This involves a lot of generated code and a very useful but large library of routines.

The place to look for much of this is the .EDMX file in the project folder. If you double-click on this, you are taken to a designer. From here you can right-click to get operations such as the ability to re-generate the model from the database.



This designer shows a diagram of the generated model.

If you change the database, make sure you update your model from the database. This will get the schema from the DB, and re-generate the source.

Tip: Sometimes, if you are having trouble doing simple updates, just delete everything on the diagram and do an Update and re-create everything.

Double-click on this to launch the designer.

Code Generation Strategy
Used to set the code generation strategy of this model. The 'Custom Tool' associated with this model, accessible from the properties on this item in th...