XML Data Adapter Proof of Concept

Installation Guide

Glen Oliff
Stephanie Liu-Barnes
Alexis Karnauskas

August 31, 2007

Table of Contents

Downloading the Software

The proof of concept XML data adapter requires two downloadable components (listed below) - the java development kit (JDK) and JUnit (for unit testing). The following section describes how to install this software properly.

Download the Java 1.6 SDK (JDK 6 as referenced on java.sun.com)

1. Go to the following website: http://java.sun.com
2. Once on this website, you will see a “Popular Downloads” section on the right side of the screen.
3. Under the “Popular Downloads” section, click on “Java SE”.
4. Once on the Java SE Downloads page, click the Download link for the JDK 6. There are a few options listed on the page, but you only need to download JDK 6.
5. Download the “Offline” installation that pertains to your operating system.
6. Once the download has completed, launch the installer and choose an appropriate location for installation of the Java Development Kit (ex: C:\Java).

Download JUnit 4

1. Go to the following website: http://junit.org/index.htm
2. To download the latest version of JUnit, click on the “Download” link in the middle of the page.
3. Save the .zip file to your machine.
4. Unzip the JUnit .zip file to a specific location on your machine (ex: C:\Java). It is recommended that you unzip it in the vicinity of the recently installed JDK.

Setting Up the Development / Execution Environment

For this example setup, the environment creation will be done using the Windows environment. Certainly, this could be achieved by writing a "profile" on a UNIX/LINUX system as well.

Steps to set up the Environment

1. Create a command file (ex: hydroxc_env.cmd) and open it for editing.
2. Update the command file with the template listed below, replacing the items in angle brackets (in italics) with the paths specific to your machine:

```
@set JAVA_HOME=<path to jdk directory>
@set JUNIT_HOME=<path to Junit directory>

@set CLASSPATH=%JAVA_HOME%\lib;%JUNIT_HOME%\<junit jar file>;.
@set PATH=%JAVA_HOME%\bin;%PATH%
```

   Below is an example of what a completed script might look like:

```
@set JAVA_HOME=E:\Java\jdk1.6.0
@set JUNIT_HOME=E:\Java\junit4.1

@set CLASSPATH=%JAVA_HOME%\lib;%JUNIT_HOME%\junit-4.1.jar;.
@set PATH=%JAVA_HOME%\bin;%PATH%
```

3. Once the script is completed, run it from the command prompt.
4. If there were no errors after running the script from the command prompt, verify that the path to java is correct by typing the following:

```
java –version
```

5. If the path is correct, you will see a message like the one listed below. Please keep in mind that the version number may have changed since these instructions were written.

```
java version "1.6.0"
Java(TM) SE Runtime Environment (build 1.6.0-b105)
Java HotSpot(TM) Client VM (build 1.6.0-b105, mixed mode, sharing)
```

6. In the case that schema changes are made to HydroXC.xsd, LocationMapping.xsd, or ParameterMapping.xsd (as a result of further development), you will need to rerun the regeneration scripts for the XML binding classes. In such a case, you will need to use the "xjc" utility provided as a part of the JDK 6. Please type the following in the same command prompt and hit Enter:

```
xjc
```

7. Verify that you end up with a usage message and not an error message that the command cannot be found. Here's an example of the error message that windows displays when the command cannot be found:

```
'xjc' is not recognized as an internal or external command,
operable program or batch file.
```

Running the Proof-of-Concept Data Adapter Examples

Now that the appropriate components are downloaded and the environment is set up, the proof-of-concept data adapter can be executed in several different scenarios. This section provides an explanation of how to run examples that are included with the HydroXC XML Data Adapter Proof-of-Concept and were demonstrated as part of the HydroXC Phase 3 workshop.

Notes for Running the Demo Examples

For each of the following examples, you will need to open a few files. Given that this is a Java utility, often times the text is not properly formatted when opening in a limited functionality browser (i.e. notepad). To open the files, please use one of the following browsers listed below:

- XML: Internet Explorer, Firefox, Visual Studio .NET, or other tool capable of rendering the XML in a readable format
- shefin_hydroxc: Wordpad, MS Word, Textpad, or other program that is good at handling multiple formats with line breaks

Please run all commands in the examples below from the environment created in the previous section of this document. All of the commands (shef_read, shef_write) to run are located in the highest level of the HydroXC installation folder.

Each example from the list below consists of one direction that the proof-of-concept data adapter handles. Examples 1 & 3 perform the shefout -> HydroXC conversion. Examples 2, 4, 5, & 6 perform the HydroXC -> shefin conversion. When performing each example, it will be helpful to make a copy of the resultant output file for comparison afterwards. If you are working with Example #1, please copy the shef.xml file to shef_ex01.xml for comparison after running all the examples.

Example 1: Running SHEF to HydroXC with no Location Mapping File

Required File: shefout.406

This example converts the binary “shefout” format to standard HydroXC compliant XML. Please note that viewing the shefout.406 file from most text editors is ***not*** readable to the human eye. If you would like to view the binary data, please view in a hex editor. This “shefout[1]” file is the product of running the SRUS54KMAF.16152010.406 source file through the SHEF parser[2].

1. Run the following command:

   ```
   shef_read shefout.406
   ```

2. After running the command, if there are errors, there is likely a problem with the environment. Please ensure that the steps have been followed to properly set up the environment.
3. If the environment is set up correctly, the execution should have produced a “shef.xml” file in the current directory. Open the file with a program that handles formatting XML, such as Internet Explorer.

[1] When transforming any new SHEF .B messages (i.e. one that is not provided already), you must first parse the message using the SHEF parser, and then feed the resulting data file into the proof-of-concept adapter. In keeping with example 1, you would call your recently parsed shefout file as the first argument to the shef_read command.

[2] If you would like to learn more about SHEF and the SHEF parser, please go to the SHEF homepage on the NOAA website: http://www.nws.noaa.gov/oh/hrl/shef/shefcode.htm

Example 2: Running HydroXC to SHEF with no Location Mapping File

Required Files: ShefToShef.xml, shef.xml (generated as part of Example 1)

This example uses the output from Example 1 as an input file. Example 1 should be run prior to running Example 2.

1. Run the following command:

   ```
   shef_write shef.xml ShefToShef.xml
   ```

2. The output from this invocation results in a “shefin_hydroxc” file. This output is in NOAA’s Standard Hydrologic Exchange Format (SHEF).

Example 3: Running SHEF to HydroXC with a Location Mapping File

Required Files: shefout.406, SL_406.xml

This example runs through the same data set as Example 1, but this time with the location mapping file called “SL_406.xml”.

1. Run the file from Example 1, “shefout.406” with the following command:

   ```
   shef_read shefout.406 SL_406.xml
   ```

2. Verify as per Example 1 that the “shef.xml” file has been created. Note in this case that the Location names have been replaced with Latitude/Longitude values from the Location mapping file specified at runtime (SL_406.xml).

Example 4: Running HydroXC to SHEF with a Location Mapping File

Required Files: shef.xml (generated as part of Example 3), ShefToShef.xml, SL_406.xml

This example runs through the same data set as Example 3, but this time with the location mapping file ”SL_406.xml”.

1. Run the file resulting from Example 3, “shef.xml” with the following command:

   ```
   shef_write shef.xml ShefToShef.xml SL_406.xml
   ```

2. Verify that the resulting “shefin_hydroxc” file is the same as the one generated as part of Example 2, by either comparing visually or by running with a diff utility.

Example 5: Basic Parameter Mapping for HydroXC to SHEF

Required Files: shef_garbledparams.xml, GarbledToShef.xml

This example demonstrates a scenario where the HydroXC file includes the same kind of data as the SHEF file represents, but the data is identified by different names. If you open the “shef_garbledparams.xml” file, it should look quite familiar in structure. However, the parameter names are not recognizable from the parameters generated from the “shefout” context. In such a case, we will need to specify a different Parameter Mapping file to match together the two naming conventions.

1. Run the following command:

   ```
   shef_write shef_garbledparams.xml GarbledToShef.xml
   ```

When the resulting “shefin_hydroxc” file is created, notice that it looks quite similar to the other “shefin” source files from the previous examples.

Example 6: Advanced Parameter Mapping for HydroXC to SHEF

Required Files: DataMappingSource.xml, DataMappingToShef.xml

This example covers more advanced topics for parameter mapping. Review the “DataMappingToShef.xml” file prior to running this example. You will notice that there are multiple mapping items for the “Physical Element Code” section. In addition, the “ignore” flag is set for the duration, and there are instances of algorithm invocation.

1. Run the following command:

   ```
   shef_write DataMappingSource.xml DataMappingToShef.xml
   ```

2. In the resulting “shefin_hydroxc” file, you will see where the transformations have taken place from the data mapping file. Compare the “datapoint” parameter from the “DataMappingSource.xml” file to the data section from the resulting “shefin_hydroxc” file.

Conclusion

The installation instructions in this document create the environment for you to fully run the SHEF/HydroXC proof-of-concept data adapter. With this in place, there are a multitude of areas to extend this adapter or begin creating new adapters. In general, this Java environment should be replicated for new adapters, so that it can eventually transform into a standard HydroXC execution environment. Please refer to the HydroXC website, under the Phase 3 activities, to learn more about the data adapter work.