



HIS document 5

CUAHSI WaterOneFlow Workbook (version 1.0)

**A guide to using CUAHSI's WaterOneFlow web services
to retrieve hydrologic time series data**

December 2007

Prepared by:

Tim Whiteaker

**Center for Research in Water Resources
University of Texas at Austin**

Distribution

Copyright © 2007 University of Texas at Austin

CUAHSI's WaterOneFlow web services are documented at the following URL:
<http://water.sdsc.edu/waterOneFlow/>

Disclaimers

Although much effort has been expended in the development and testing of the WaterOneFlow and HydroObjects, errors and inadequacies may still occur. Users must make the final evaluation as to the usefulness of WaterOneFlow and HydroObjects for his or her application.

Acknowledgements

The team of engineers, scientists and research assistants that contributed to this document includes:

Tim Whiteaker (editor), Research Associate, Center for Research in Water Resources, University of Texas, Austin, TX.

David Tarboton, Professor, Civil and Environmental Engineering, Utah State University, Logan, UT.

Jon Goodall, Assistant Professor, Nicholas School of the Environment and Earth Sciences, Duke University, Durham, NC.

David Valentine, GIS Programmer, San Diego Supercomputer Center, University of California at San Diego, La Jolla, CA.

Ernest To, Doctoral Candidate, Center for Research in Water Resources, University of Texas, Austin, TX.

Bora Beran, Doctoral Candidate, Computational Hydraulics Lab, Drexel University, Philadelphia, PA.

Thiha Min, Research Assistant, Center for Research in Water Resources, University of Texas, Austin, TX.

Funding

Funding for the work described in this document was provided by the Consortium of Universities for the Advancement of Hydrologic Science, Inc. (CUAHSI) under NSF Grant No. EAR-0413265. In addition, much input and feedback has been received from the CUAHSI Hydrologic Information System development team. Their contribution is acknowledged here.

Technical Support

If you find errors, or have feedback, please contact:

Tim Whiteaker
Center for Research in Water Resources
University of Texas at Austin
10100 Burnet Rd., Bldg 119
Austin, TX 78758
twhit@mail.utexas.edu

Table of Contents

Distribution.....	i
Disclaimers.....	i
Acknowledgements.....	i
Technical Support	ii
1.0 Introduction	1
1.1 WaterOneFlow Web Services.....	1
1.2 WaterOneFlow Web Service Methods and Output	2
1.2.1 GetSiteInfo/GetSiteInfoObject	3
1.2.2 GetVariableInfo/GetVariableInfoObject	4
1.2.3 GetValues/GetValuesObject	4
1.3 Document Outline	5
1.4 Obtaining This Workbook	6
2.0 Data Sources	7
2.1 USGS National Water Information System (NWIS).....	7
2.2 EPA STORAGE & RETRIEVAL SYSTEM (EPA STORET)	7
2.3 Moderate Resolution Imaging Spectroradiometer (MODIS)	7
2.4 Daymet.....	8
2.5 North American Mesoscale model (NAM).....	8
2.6 Susquehanna River Basin Hydrologic Observatory System (SRBHOS)	9
3.0 Ingesting Susquehanna River Basin Data into Excel with HydroObjects.....	10
3.1 Introduction.....	10
3.2 Computer Requirements	10
3.3 Installation.....	10
3.4 Downloading Susquehanna River Basin Data	12
3.4.1 Specifying the Data Source.....	13
3.4.2 Getting a List of Sites	14
3.4.3 Getting a List of Variables	14
3.4.4 Obtaining Site and Variable Information.....	15
3.4.5 Getting the Site Catalog	18
3.4.6 Downloading Time Series Data	19
3.5 Extending this Example	20

4.0	Ingesting STORET Data into Excel with HydroObjects	22
4.1	Introduction.....	22
4.2	Computer Requirements	22
4.3	Installation.....	22
4.4	Downloading EPA STORET Data.....	22
4.4.1	Obtaining Site and Variable Information.....	23
4.4.2	Downloading Time Series Data.....	25
5.0	Ingesting Weather and Streamflow Data into ArcGIS	27
5.1	Introduction.....	27
5.2	Computer and Skill Requirements	28
5.3	Installation.....	28
5.4	Retrieving Data with Weather Downloader.....	30
5.4.1	Opening the Map.....	30
5.4.2	Adding Weather Downloader to ArcMap.....	31
5.4.3	Downloading Weather Data.....	33
5.5	Downloading Data from Other Web Services	41
6.0	Plotting MODIS Data with Matlab.....	44
6.1	Introduction.....	44
6.2	Computer and Skill Requirements	44
6.3	Procedure	44
6.3.1	Setting up the XML Parser	44
6.3.2	Retrieving MODIS Data	45
7.0	Ingesting NWIS Data using VB.Net.....	51
7.1	Introduction.....	51
7.2	Computer and Skill Requirements	51
7.3	Accessing NWIS Data with a VB.Net Windows Application.....	51
7.3.1	Setting up the Project.....	51
7.3.2	Creating the Web Reference	52
7.3.3	Building the User Interface	54
7.3.4	Writing the Code.....	58
7.3.5	Running the Code	60
8.0	Ingesting NWIS Data Using Java	62
8.1	Computer and Skill Requirements	62
8.2	Procedure	62
8.2.1	Creating a New Project	62
8.2.2	Creating a Web Service Client.....	63
8.2.3	Creating a Class to Consume the Web Service.....	65

Appendix A: Source Code for parse_xml.m.....	73
Appendix B: Source Code for MODISPlot_xml.m.....	79
Appendix C: Source Code for nwis.java Class	81

1.0 Introduction

One of the key programs of the Consortium of Universities for the Advancement of Hydrologic Science (CUAHSI) is the development of Hydrologic Information Systems (HIS), which facilitate the integration of data and software to support hydrologic science. A main component of CUAHSI HIS is WaterOneFlow web services, which provide programmatic access to a growing collection of national, state, and individual investigator hydrologic observation repositories. This document describes how to use WaterOneFlow web services and methods, with tutorials providing examples of data access in a variety of software environments.

1.1 WaterOneFlow Web Services

Wikipedia gives the following definition for a web service:

According to the W3C a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface that is described in a machine-processable format such as WSDL. Other systems interact with the Web service in a manner prescribed by its interface using messages, which may be enclosed in a SOAP envelope, or follow a RESTful approach.

When a web service is published on the Internet, a computer with an Internet connection can call upon the web service to perform useful work.

CUAHSI WaterOneFlow web services facilitate the retrieval of hydrologic observations data. While several repositories of national hydrologic data are already available online, each data provider uses its own methodology for querying data, and its own output format for returning data. WaterOneFlow web services provide a common methodology and output format for these data sources, and permit data access directly from within the user's preferred software environment, rather than requiring the user to navigate to the data provider's web page, query data, and save the data locally.

WaterOneFlow web services have been developed for the following national networks:

USGS National Water Information System (NWIS) – national database of streamflow, water quality, and groundwater data

<http://water.sdsc.edu/waterOneFlow/NWIS/DailyValues.aspx>

<http://water.sdsc.edu/waterOneFlow/NWIS/UnitValues.aspx>

<http://water.sdsc.edu/waterOneFlow/NWIS/Data.aspx>

<http://water.sdsc.edu/waterOneFlow/NWIS/Groundwater.aspx>

EPA STORET – national database of water quality data

http://water.sdsc.edu/waterOneFlow/EPA/cuahsi_1_0.aspx

Daymet – daily surfaces of temperature, precipitation, humidity, and radiation for the contiguous United States

<http://water.sdsc.edu/waterOneFlow/DAYMET/Service.aspx>

MODIS – remotely sensed meteorological, oceanographic, and hydrologic data for the world
<http://water.sdsc.edu/waterOneFlow/MODIS/Service.aspx>

North American Mesoscale model (NAM) – prediction of climate variables for North America
<http://water.sdsc.edu/waterOneFlow/NAM12k/Service.aspx>

In addition to these national datasets, several academic investigator datasets are also now available as WaterOneFlow web services, such as the Susquehanna River Basin Hydrologic Observatory System.

These data sources are further described in Chapter 2.

To access a catalog of national networks currently available through WaterOneFlow, use the service at:

<http://water.sdsc.edu/waterOneFlow/NETWORKS/Service.aspx>

1.2 WaterOneFlow Web Service Methods and Output

To standardize access to these data sources, WaterOneFlow web services implement the following core methods regardless of data provider:

- GetSiteInfo
- GetSiteInfoObject
- GetVariableInfo
- GetVariableInfoObject
- GetValues
- GetValuesObject

In addition to consistent method names, WaterOneFlow services use consistent method signatures, and provide output in a consistent format, regardless of data provider. Thus, if you learn how to use the WaterOneFlow services for NWIS, you should easily be able to make the jump to using the WaterOneFlow service for EPA STORET.

For the methods with the suffix “Object” in the method name, data are returned in object format. For the other methods, data are returned in XML format. Both Object and XML formats are provided to suit the developer’s preference, or to accommodate the application programming environment of the developer.

NOTE: All methods include an authorization token parameter (authToken). This token permits CUAHSI to restrict access to web services. For the services described in this workbook, any authorization token will work, as all of these services are publicly available.

General documentation about CUAHSI web services can be found at

<http://water.sdsc.edu/waterOneFlow/>

Below is a brief summary of the core WaterOneFlow methods.

1.2.1 GetSiteInfo/GetSiteInfoObject

This method returns basic information about a site, such as its name, location, and a list of variables available at the site for time series retrieval. The method has the following signature:

```
GetSiteInfo(String location, String authToken)
```

Input Parameters:

location

For networks which measure data at discrete sites, such as NWIS, this location parameter should be written as “NetworkName:SiteCode”. For example, to specify the NWIS stream gage at the Colorado River at Austin (site code 08158000), use the following value for the location parameter:

```
"NWIS:08158000"
```

For networks that return a time series for any point in space (such as Daymet), use the convention “GEOM:POINT(Longitude Latitude)”. For example, to specify a point at 113 degrees West Longitude and 45 degrees North Latitude, use the following value for the location parameter:

```
"GEOM:POINT(-113 45)"
```

For networks that return a time series for a box defined by Latitude and Longitude coordinates (such as MODIS), use the convention “GEOM:BOX(WestLongitude SouthLatitude, EastLongitude NorthLatitude)”. For example, to specify a box that covers the whole earth, use the following value for the location parameter:

```
"GEOM:BOX(-180 -90,180 90)"
```

authToken

This parameter allows a data provider to restrict or monitor access to its web services, by requiring a password or some other means of identification in order to use a given web method. In many cases, this parameter may be left as an empty string, “”.

Example VB.Net Code

```
Dim ws As New [WsReference]  
Dim result As String = ws.GetSiteInfo("NWIS:08158000", "")  
Debug.Write(result)
```

1.2.2 *GetVariableInfo/GetVariableInfoObject*

This method returns information about a time series variable, such as name and units. The method has the following signature:

```
GetVariableInfo(String variable, String authToken)
```

Input Parameters:

variable

To specify a variable, you must include both the network name and the variable code within the network, in the following format: “NetworkName:VariableCode”. For example, to query the NWIS website for information about variable code 00010 (which happens to be water temperature), you would use the following value for the variable parameter:

```
"NWIS:00010"
```

authToken

This parameter allows a data provider to restrict or monitor access to its web services, by requiring a password or some other means of identification in order to use a given web method. In many cases, this parameter may be left as an empty string, “”.

Example VB.Net Code

```
Dim ws As New [WsReference]  
Dim result As String = ws.GetVariableInfo("NWIS:00010", "")  
Debug.Write(result)
```

1.2.3 *GetValues/GetValuesObject*

This method returns a time series for a given variable at a given location. The method has the following signature:

```
GetValues(String location, String variable, String startDate, String endDate,  
String authToken)
```

Input Parameters:

location

The location parameter is the same as for the GetSiteInfo method.

variable

The variable parameter is the same as for the GetVariableInfo method, except that this parameter may also include options for data retrieval. The parameter should be specified as follows:

```
"NetworkName:VariableCode/Option=Value"
```

In most cases, no option is required, and the “NetworkName:VariableCode” format may be used. Some networks, such as MODIS, do require an option to be set. As an example, to retrieve data for Cloud Optical Thickness in the Water Phase from MODIS, for a spatial average that includes both the land surface and the ocean, you would use the following value for the **variable** parameter:

```
"MODIS:11/plotarea=landocean"
```

startDate

This parameter specifies the start datetime for which time series records are desired. For networks that return time series with a temporal precision of one day or longer, use the following format for the startDate:

```
"yyyy-mm-dd"
```

For example, to specify the last day of 2003 as the start date for time series retrieval, use the following value for the startDate parameter:

```
"2003-12-31"
```

For networks with a temporal precision shorter than one day, you may specify the hours and minutes and so on with the format below:

```
"yyyy-mm-ddThh:mm:ss"
```

For example, to specify 6:30 AM on the last day of 2003, use the following value:

```
"2003-12-31T06:30"
```

endDate

This parameter specifies the end datetime for which time series records are desired. The format is the same as for the startDate parameter.

authToken

This parameter allows a data provider to restrict or monitor access to its web services, by requiring a password or some other means of identification in order to use a given web method. In many cases, this paramter may be left as an empty string, "".

Example VB.Net Code

```
Dim ws As New [WsReference]
Dim result As String = ws.GetValues("NWIS:08158000", _
                                     "NWIS:00010", _
                                     "2003-01-01", _
                                     "2003-12-31", _
                                     "")
Debug.Write(result)
```

1.3 Document Outline

The document is created in the form of a series of tutorials. Except for the first two chapters (Introduction and Data Sources), each chapter in the document is a tutorial on how to use different software or programming environments to access different WaterOneFlow web services and methods. The links for obtaining installation and data files for each tutorial are provided in the tutorial. In the current version of the document, you will learn how to access data from USGS NWIS, EPA STORET, MODIS, Daymet, NAM and an academic investigator database, from

Excel, ArcGIS, Matlab, VB.Net and Java. Each chapter/tutorial covers one software/programming environment dealing with one or more of the five data sources. The outline of this document is as follows:

Chapter 1 – Introduction

Chapter 2 – Data Sources

Chapter 3 – Ingesting data into Excel (Susquehanna River Basin example)

Chapter 4 – Ingesting data into Excel (STORET example)

Chapter 5 – Ingesting data into ArcGIS (Daymet, NAM and NWIS example)

Chapter 6 – Ingesting data into Matlab (MODIS example)

Chapter 7 – Ingesting data using VB.Net (NWIS Unit Values example)

Chapter 8 – Ingesting data using Java (NWIS Daily Values example)

1.4 Obtaining This Workbook

This workbook is available at the following location, with the title “CUAHSI WaterOneFlow Workbook”:

<http://www.cuahsi.org/his/documentation.html>

In the following location, you will also find a folder for installation files required by some of the exercises in this document, as well as a folder containing some solution files.

ftp://ftp.crwr.utexas.edu/pub/outgoing/CUAHSI/HIS_workbook/20070720/

2.0 Data Sources

This chapter describes some of the data sources for which WaterOneFlow web services have already been created.

2.1 USGS National Water Information System (NWIS)

Data providing organization: United States Geological Survey (USGS)

Website: <http://waterdata.usgs.gov/nwis>

The USGS NWIS is a comprehensive and distributed program that supports acquisition, processing and storage of water data. Most of the data stored in NWIS is available through NWIS website provided above (NWIS Web). The data available via NWIS web mainly include information on quantity and quality of surface and ground water. NWIS web serves both historical and real time data. The real time data, however, is not available for all sites.

Data provided by NWISWeb are regularly updated from NWIS. Real-time data are generally updated upon receipt at local Water Science Centers. NWISWeb provides access to data by category, such as surface water, ground water, or water quality, and by geographic area. NWIS data are available for all 50 states, plus border and territorial sites, and include data from as early as 1899 (at few stations) to present. Of the over 1.5 million sites with NWIS data, the vast majority (about 800,000) are for groundwater wells, about 25,000 sites are for streamflow data, and about 9,800 of the sites provide real-time data. In addition there are many sites with atmospheric data such as precipitation, and there are nearly 70 million water-quality results from about 4 million water samples collected at hundreds of thousands of sites.

2.2 EPA STORAGE & RETRIEVAL SYSTEM (EPA STORET)

Data providing organization: Environmental Protection Agency (EPA)

Website: <http://www.epa.gov/storet/>

EPA STORET is a repository for water quality, biological, and physical data and is used by state environmental agencies, federal agencies and universities. Most of the data is available through the STORET website which can be accessed using the link above. STORET maintains two systems; ‘legacy STORET’ and its successor ‘modernized STORET’. Legacy STORET has been static since January 1, 1999 while modernized STORET gets updated on a monthly basis. Currently EPA STORET receives data from 279 organizations adding up to about 275,000 stations.

2.3 Moderate Resolution Imaging Spectroradiometer (MODIS)

Data providing organization: National Aeronautics and Space Administration (NASA)

Website: http://g0dup05u.ecs.nasa.gov/Giovanni/modis.MOD08_M3.shtml

The Goddard Earth Sciences Data and Information Services Center (GES DISC) has created the GES DISC Interactive Online Visualization and Analysis Infrastructure (Giovanni) to enable Web-based visualization and analysis of satellite remotely sensed meteorological, oceanographic, and hydrologic data. The MODIS data are available through one of the Giovanni interfaces called the MODIS Online Visualization and Analysis (MOVAS). The MOVAS system, operational since September 2003, provides access to download, visualize and analyze MODIS Level-3 atmospheric monthly products. The MODIS Level-3 data includes monthly 1 x 1 degree grid average values of atmospheric parameters related to atmospheric aerosol particle properties, total ozone burden, atmospheric water vapor, cloud optical and physical properties, and atmospheric stability indices.

The data are available for the entire globe from March 1, 2000, to typically six months to a year prior to the current date. The time series returned by MOVAS is spatially averaged over the extent specified by a bounding box of lat-long coordinates. The data are temporally averaged with a monthly time step.

2.4 Daymet

Data providing organization: Numerical Terradynamic Simulation Group (NTSG) at University of Montana

Website: <http://www.DAYMET.org/dataSelection.jsp>

Daymet is a numerical model that provides daily surfaces of temperature, precipitation, humidity, and radiation over large regions of complex terrain. Daymet was developed to create fine resolution daily meteorological and climatological data necessary for plant growth model inputs. The input to Daymet includes digital elevation model and observations of maximum temperature, minimum temperature and precipitation from ground-based meteorological stations.

The data are available as surfaces or as numerical estimates at single points for the contiguous United States at a daily time interval. Data before 01/01/1980 or after 12/31/2003 may not be available.

2.5 North American Mesoscale model (NAM)

Data providing organization: Unidata program at the University Cooperation for Atmospheric Research (UCAR)

Website: <http://www.nco.ncep.noaa.gov/pmb/nwprod/analysis/>

The NAM model makes predictions of climate variables four times daily (0:00 UTC, 6:00 UTC, 12:00 UTC and 18:00 UTC), with the predictions extending 84 hours into the future, at three hour intervals. The spatial extent of the model is limited to North America. The spatial resolution of the model grid is 12.19 km, and the grid dimensions are 614 x 428. The east and west longitudinal extents of the grid (in decimal degrees) are -49.30897 and -133.49621. The north and south latitude extents of the grid (in decimal degrees) are 57.35624 and 12.12367.

2.6 Susquehanna River Basin Hydrologic Observatory System (SRBHOS)

Data providing organization: SRBHOS

Website: <http://www.srbhos.org/>

Susquehanna River Basin and Chesapeake Bay make up a CUAHSI WATERS combined test bed project. The Susquehanna River Basin (SRB) is the largest tributary to the Chesapeake Bay and the goal of these two projects (PIs are: Bill Ball, Kevin Dressler, Chris Duffy, Michael Piasecki and Pat Reed) is to expand the ability to represent and predict the physical processes associated with the nutrient loads reaching the Bay from the SRB. The SRBHOS group was formed to provide an umbrella for scientists and researchers in the basin that are interested in better understanding the environmental dynamics (including problems) of the basin and who would benefit from the existence of a hydrologic observatory. The SRBHOS group to this date has been extraordinarily successful in attracting funding from a variety of sources in order to conduct work much in line with the goals of SRBHOS, the latest being the funding for a Critical Zone observatory that will be situated within the SRBHOS basin.

The data provided comes from the "Real-Time Hydrologic Monitoring Data Network" site established by researchers (Pat Reed, Chris Duffy and others) from Penn State University. The RTH_Net field facility is investigating the dynamics of the terrestrial water and energy balance, focusing on closing the water and energy budget across the Shaver's Creek watershed. The research will attempt to find fundamental relationships of atmosphere-land-subsurface water and energy dynamics. RTH_Net will extend the current sensor systems within the Penn State Experimental Forest to resolve the roles of soil moisture and groundwater within the water cycle through the use of Evaporation- Transpiration-Recharge (E-T-R) sensor arrays to fully capture the essential space-time scales of terrestrial hydrology. RTH_Net will help identify how thresholds, feedbacks, and nonlinearities in atmosphere-soil-stream-groundwater systems serve to amplify low-frequency modes in runoff.

At the time of this writing, SRBHOS data are available for two test sites. The data cover nearly 30 surface and atmospheric variables, from about 1997 to the present, with about a 10-minute time step.

3.0 Ingesting Susquehanna River Basin Data into Excel with HydroObjects

by Tim Whiteaker

3.1 Introduction

This chapter demonstrates the use of the HydroObjects Application Programming Interface (API) to give users direct access to an academic investigator database from within Excel. This database of Susquehanna River Basin observations is made available online via WaterOneFlow web services, which the investigator has implemented. By conforming to WaterML and WaterOneFlow standards, the investigator has made it possible for a number of applications, such as HydroObjects, to directly work with the data.

The HydroObjects API is a .Net DLL that is compiled to also be COM compliant. This allows the API to supplement other COM compliant software systems (Word, R, Python, etc.). The example here shows an Excel spreadsheet that has been extended through the development of Visual Basic for Applications (VBA) macros that use the HydroObjects API as a resource, in order to download data from a certain type of WaterOneFlow web services.

These web services are built upon Observations Data Model (ODM) databases, which academic investigators use to store and publish their data. CUAHSI provides a generic web service that is meant to read from an ODM database, and so if an investigator organization uses this format, then that organization can simply plug in their database to the generic ODM web service, and then suddenly their data is accessible to the world in a standard manner.

The Excel macros in the spreadsheet that you are about to work with allow the user to get metadata (e.g. site location, number of variables measured and their description) and time series data for any variable measured at site by simply clicking buttons within Excel. The HydroObjects provides a layer of abstraction between Excel and the web services, which allows the steps demonstrated in this chapter to also be applicable for the other data sources within WaterOneFlow.

3.2 Computer Requirements

- Working internet connection
- MS Excel
- HydroObjects
- ODMws.xls spreadsheet (included with HydroObjects installation)

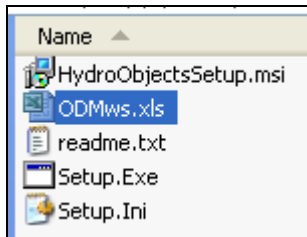
3.3 Installation

Install HydroObjects and the Excel document using the following steps:

1. Download the HydroObjects setup file from

<http://www.cuahsi.org/his/toolkit.html>

The zip file contains several files used for setup, as well as a readme.txt file and an example spreadsheet which makes use of HydroObjects.

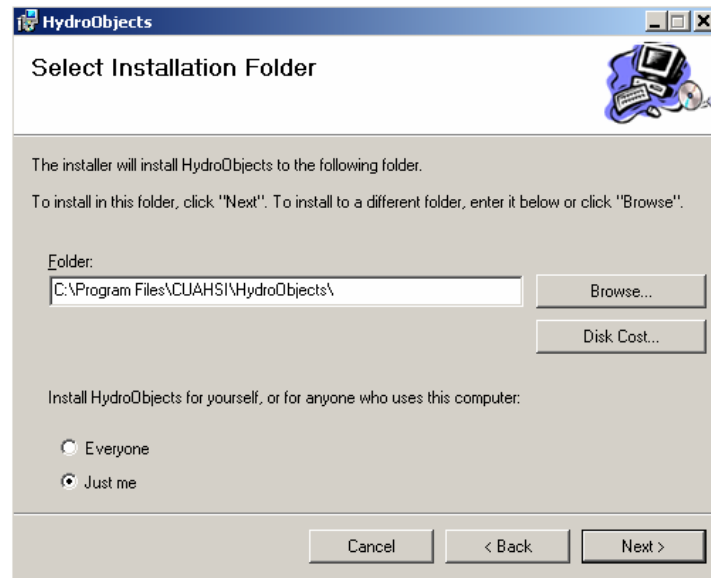


You will use the spreadsheet in this exercise.

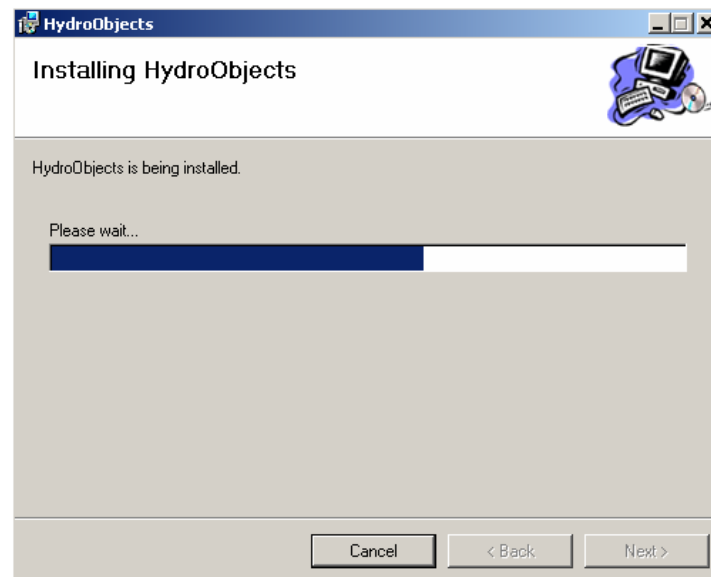
2. Unzip all the contents of the zip file, and double click on Setup.exe to run the setup file.
3. In the HydroObjects Setup Wizard, click Next to start the installation process.



4. Use the default installation folder for HydroObjects or choose your preferred location, specify to install the program for “Everyone” or “Just me”, and click Next.



5. In the Confirm Installation window, click Next to start the installation process. You should see the progress bar as shown below:

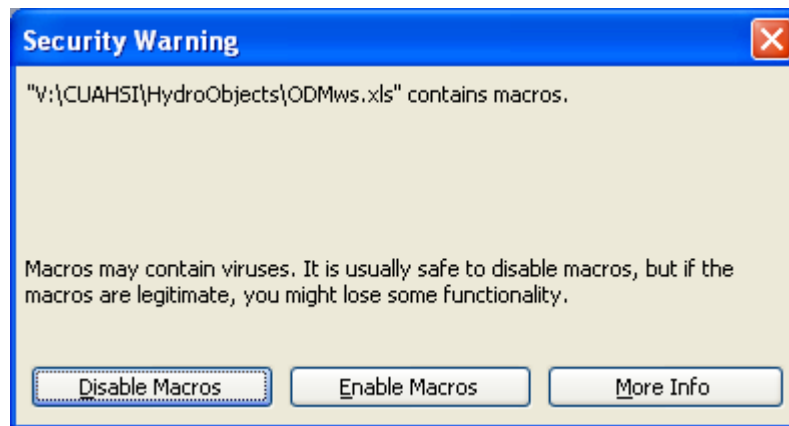


6. After HydroObjects is successfully installed, click Close to finish the installation process.
7. You may now delete the setup files if you wish. But don't delete ODMws.xls, as you will use it in this exercise.

3.4 Downloading Susquehanna River Basin Data

To download Susquehanna River Basin data, double click on ODMws.xls to open the file in Excel.

ODMws.xls contains macros that use WaterOneFlow web services to download observations data. (You may need to change your security settings in Excel to Medium in order to run macros.) If prompted about enabling macros when opening the file, make sure you enable the macros by choosing the “Enable Macros” option as shown below:



ODMws.xls contains six worksheets: Data Source, Sites, Variables, Site and Variable Info, Site Catalog, and Time Series. These worksheets are described through the exercises below.

3.4.1 Specifying the Data Source

The Data Source worksheet provides general information about the spreadsheet, and lists some sample WSDLs where academic investigators have already provided access to their data using ODM and WaterOneFlow web services. To hook the spreadsheet up to a particular data source, you provide the address to the WSDL for the web service which gives access to the data, the name of the web service, and the name of the observation network within the web service.

Tip: A WSDL is an XML document that tells a client application what a web service can do.

Here you will tell the spreadsheet to get data from the Susquehanna River Basin project.

1. On the Data Source worksheet, find the WSDL locations for example data sources at the bottom of the visible data on the worksheet. Copy the WSDL for SRBHOS, and paste it into the yellow cell entitled “WSDL of OD Web Service”.

WSDL of OD Web Services	
http://ccbay.tamucc.edu/CCBayODWS/cuahsi_1_0.asmx?WSDL	This is a URL pointing to the location for the WSDL file.
<p>Note: When running the macros in this worksheet, if you get a "License information for this component not found" or "Active-X can't create object" error, then you need to reset the reference to HydroObjects in VBA (Tools-References).</p>	
Example Data Sources	
WSDL	Description
http://ccbay.tamucc.edu/CCBayODWS/cuahsi_1_0.asmx?WSDL	Corpus Christi Bay (Texas A&M Corpus Christi)
http://his09.umbc.edu/BaltOD/cuahsi_1_0.asmx?WSDL	Baltimore Urban Observatory (University of Maryland)
http://cbe.cae.drexel.edu/wateroneflow/CIMS.asmx?WSDL	Chesapeake Information Management System
http://ees-his06.ad.ufl.edu/santafe-ysi/cuahsi_1_0.asmx?WSDL	Santa Fe watershed -- nitrate (YSI) (University of Florida)
http://ees-his06.ad.ufl.edu/santafe-isus/cuahsi_1_0.asmx?WSDL	Santa Fe watershed -- nitrate (ISUS) (University of Florida)
http://ees-his06.ad.ufl.edu/santafe-gwl/cuahsi_1_0.asmx?WSDL	Santa Fe watershed -- groundwater (University of Florida)
http://ees-his06.ad.ufl.edu/santafe-microwavecitra/cuahsi_1_0.asmx?WSDL	Santa Fe watershed -- Microwave Citra (University of Florida)
http://his08.iuhr.uiowa.edu/cuahsi_1_0.asmx?WSDL	Clear Creek -- water temperature (University of Iowa)
http://his08.iuhr.uiowa.edu/cuahsi_1_0.asmx?WSDL	Clear Creek -- water quality (University of Iowa)
http://his.safll.umn.edu/SAFLMC/cuahsi_1_0.asmx?WSDL	St Anthony Falls Laboratory Minnehaha Creek
http://his03.geol.umn.edu/COTCsnow/cuahsi_1_0.asmx?WSDL	Crown of the Continent Observatory (University of Minnesota)
http://ferry.mon.pitt.edu/cuahsi_1_0.asmx?WSDL	Perry Monmouth Sound (University of Pittsburgh)
http://cbe.cae.drexel.edu/wateroneflow/SRBHOS.asmx?WSDL	Susquehanna River Basin Hydrologic Observatory

3.4.2 Getting a List of Sites

The Sites worksheet retrieves and lists all of the observations sites within the network.

1. Click the Sites worksheet to activate it.
2. Press the Click to Get Sites button. After a moment, a list of sites is returned, along with some metadata about the sites.

Click to Get Sites			
After running GetSites, results will appear below.			
Site Code	Site Name	Latitude	Longitude
SRBHOS:RTHNet	Shale Hills of Penn State University	40.665817	-77.9040146
SRBHOS:BURDRUN	The Burd Run Interdisciplinary Water	40.064861	-77.5188599
SRBHOS:RTHNet2	Shale Hills of Penn State University	40.665218	-77.9031525
SRBHOS:RTHNet3	Shale Hills of Penn State University	40.664516	-77.9055176
SRBHOS:RTHNet4	Shale Hills of Penn State University	40.664963	-77.907341

This particular network only had five sites at the time of this writing. It is an experimental network, but actually has quite a bit of observations data for each site.

3.4.3 Getting a List of Variables

Now you will get a list of variables measured within the observation network.

1. Click the Variables worksheet to activate it.

2. Press the Click to Get Variables button. After a moment, a list of variables is returned, along with some metadata about them.

Click to Get Variables		
After running GetVariables, results will appear below.		
Variable Code	Variable Name	Units
SRBHOS:508	Nitrogen, nitrate (NO3) nitroger	milligrams per liter
SRBHOS:509	Water temperature	degree celcius
SRBHOS:510	Oxygen, dissolved	milligrams per liter
SRBHOS:511	Water depth	meter
SRBHOS:512	pH	dimensionless

Now that you know the sites and variables in the observation network, you are ready to query information about specific sites and variables.

3.4.4 Obtaining Site and Variable Information

The Site and Variable Info worksheet provides buttons for discovering what variables are measured at each site and brief metadata for each site and variable (eg. name, units, etc.). Each station is identified by a station number assigned by the observation network. To get the information on the variables measured at any station, you must enter the correct station number in the yellow cell (G3) next to “Site Code”. Likewise, unique variable codes have been assigned to each variable in the network.

The following steps illustrate how to download site and variable information.

1. On the Site and Variable Info worksheet, enter “SRBHOS:RTHNet” as the Site Code in cell G3, and then click the Click to Get Site Info button.

In a moment, the spreadsheet will be automatically populated with information about the site and variables measured at the site.

GetSiteInfo					
Site Code -->	SRBHOS:RTHNet				
Click to Get Site Info					
After running GetSiteInfo, results will appear below.					
Name	Shale Hills of Penn State University				
Latitude	40.66581726				
Longitude	-77.904				
Data Series:					
Variable Code	Variable Name	Units	Value Count	Start Date	End Date
SRBHOS:516	Radiation, net 10-minu	watts per squar	53565	5/1/2006 13:30	5/8/2007 23:50
SRBHOS:519	Precipitation, 10-minut	millimeter	53565	5/1/2006 13:30	5/8/2007 23:50
SRBHOS:519	Precipitation, 10-minut	millimeter	53565	5/1/2006 13:30	5/8/2007 23:50
SRBHOS:524	Wind direction, 10-min	degree north	53565	5/1/2006 13:30	5/8/2007 23:50

When the Get Site Info button is clicked, an Excel macro uses HydroObjects to call the WaterOneFlow web service located at the WSDL that you specified earlier to get the information about the site. This information is then brought back to the Excel application (and this all happens in just a few seconds!). The data flow sequence is the same for all the buttons that you will use in ODMws.xls to get the data.

Once the list of variables measured at a particular station is available in the form of codes, you may be interested in knowing what each code means and the measurement units for the variable. To get information on a specific variable, enter the variable code in the yellow cell (K3) next to Variable Code, press Enter and click the “Click to Get Variable Info” button.

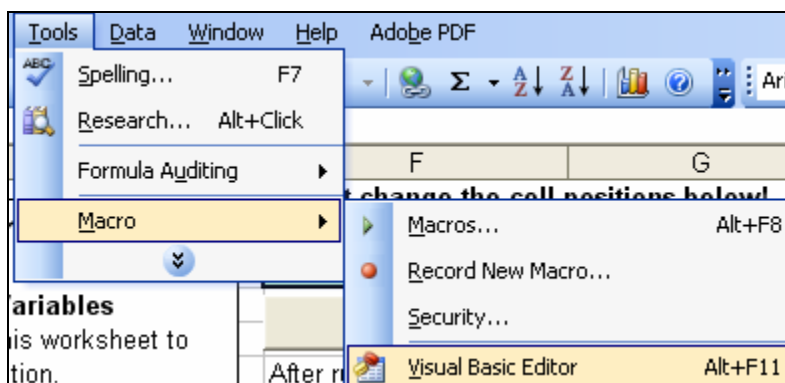
2. On the Site and Variable Info worksheet, enter “SRBHOS:516” as the Variable Code in cell K3, and then click the Click to Get Variable Info button.

After a moment, information about the variable appears in the spreadsheet.

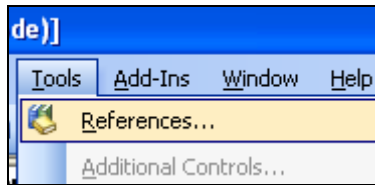
Do not change the cell positions below.	
GetVariableInfo	
Variable Code -->	SRBHOS:516
Click to Get Variable Info	
After running GetVariableInfo, results will appear below.	
Name	Radiation, net
Units	watts per square meter

Wow! Fast and easy access to remote data, right from my spreadsheet! But how does the magic happen? Let’s now take a look at the code behind the button.

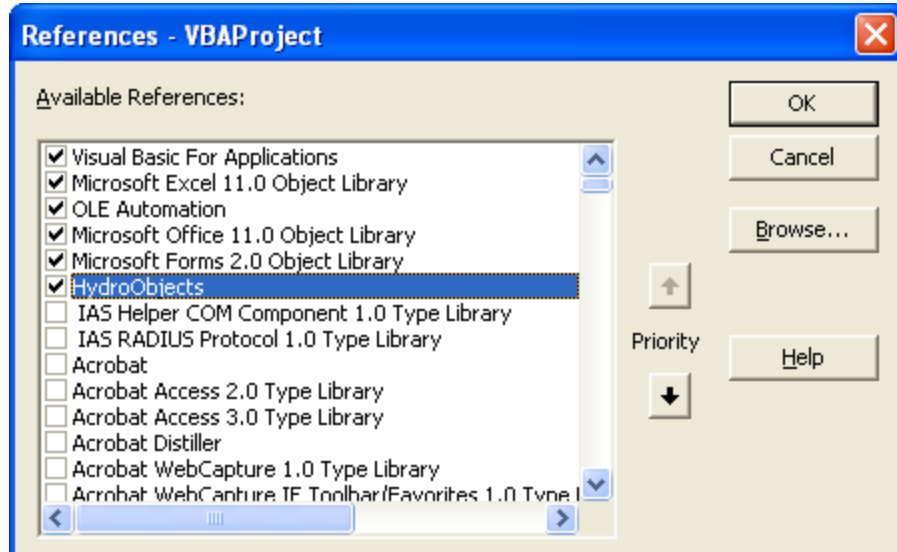
3. Click the Tools menu, then point to Macro, then click Visual Basic Editor.



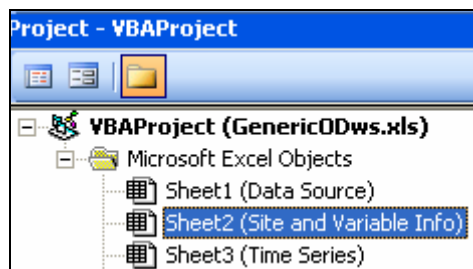
4. In VBA, click the Tools menu, and then click References.



Notice that a reference has already been made to HydroObjects in this spreadsheet. This enables the spreadsheet to communicate with WaterOneFlow web services.



5. Close the References window.
6. In the Project window of VBA, double click the Site and Variable Info sheet to open the code for that worksheet.



7. Scroll down to the procedure called GetVariableInfo and examine the code. This is the procedure that's called when you click the Click to Get Variable Info button.

This code is performing three majors tasks:

- a. Read inputs from the worksheet (in this case, the Variable Code).
- b. Use HydroObjects to call GetVariableInfo from WaterOneFlow.
- c. Write the result to the spreadsheet.

The key method being called in the code is `InvokeCall`, which requests information about the variable of interest from `WaterOneFlow`. To use the method, you provide the WSDL for the web service that you want to access, the name of the web service, the name of the method on the service that you want to call, and an array of parameters that the method requires. In this case, the WSDL is being read from the spreadsheet, in the text box that you filled out at the beginning of this exercise.

The `InvokeCall` method returns whatever object is defined by the web method. In order to allow any type of object to be returned, `InvokeCall` does not try to predict the properties and methods of the returned objects. This means that the programmer must already know the properties of `WaterOneFlow` objects before writing code (e.g., a variable object has a property called `variableName`, which provides the name of the variable). The programmer new to `WaterOneFlow` can use the code in this spreadsheet as a guide to accessing the properties of `WaterOneFlow` objects.

8. Close the VBA window.

3.4.5 Getting the Site Catalog

The Site Catalog worksheet retrieves all sites in the observation network, as well as a catalog of time series data for each site. Basically, the underlying macro calls `GetSites` to get a list of sites, and then for each site, calls `GetSiteInfo` to get a list of variables measured at each site.

1. Click the Site Catalog worksheet to activate it.
2. Press the Click to Get Site Catalog button. After a moment, the site catalog appears in the worksheet, and a message box indicates that the tool is finished.



3. Click OK to dismiss the message box.

Click to Get Site Catalog		Max Sites to Get	100
After running GetSites, results will appear below.			
Progress is shown in bottom left. To cancel, press Ctrl-Break and click End. You may need to run the Reset			
Site Code	Site Name	Latitude	Longitude
SRBHOS:RTHNet	Shale Hills of Penn State Universit	40.66581726	-77.90401459
Variable Code	Variable Name	Units	Count
SRBHOS:516	Radiation, net 10-minute	watts per square met	53565
SRBHOS:519	Precipitation, 10-minute increment	millimeter	53565
SRBHOS:519	Precipitation, 10-minute increment	millimeter	53565
SRBHOS:524	Wind direction, 10-minute continu	degree north	53565
SRBHOS:526	Wind speed, 10-minute continuous	meters per second	53565
SRBHOS:545	Air Temperature, 10-minute contin	degree celcius	39221
SRBHOS:597	Wind speed, 5-minute continuous	meters per second	63878
SRBHOS:598	Wind direction, 5-minute continuo	degree north	63878
SRBHOS:599	Radiation, net 5-minute continuous	watts per square met	63878
SRBHOS:601	Precipitation, 5-minute increments	millimeter	63878
SRBHOS:601	Precipitation, 5-minute increments	millimeter	63878
SRBHOS:517	Precipitation, daily	millimeter	558
SRBHOS:520	Relative humidity	percent	39221
SRBHOS:522	Snow Temperature	degree celcius	27442
SRBHOS:525	Wind speed, daily maximum	meters per second	558
SRBHOS:528	Gage height, instantaneous	meter	2052
SRBHOS:529	Snow depth	centimeter	27439
Site Code	Site Name	Latitude	Longitude
SRBHOS:BURDRUN	The Burd Run Interdisciplinary Wa	40.0648613	-77.51885986
Variable Code	Variable Name	Units	Count
SRBHOS:509	Water temperature	degree celcius	170967
SRBHOS:510	Oxygen, dissolved	milligrams per liter	166233
SRBHOS:511	Water depth	meter	89381

This can be a great place to start if you don't know much about a given observation network. Just click to download the site catalog, go get a cup of coffee, and when you come back, you can peruse what's available for each site in the network.

NOTE: Because so many calls to the web service are made, this can be a lengthy process, especially for observation networks with a lot of sites. You can limit the number of sites returned by inputting a number in the yellow cell next to "Max Sites to Get".

3.4.6 Downloading Time Series Data

The Time Series worksheet provides access to the time series data. It requires four input parameters: the site code, the variable code, a start data, and an end date. Enter the site code into the yellow cell (G3) next to "Site Code". Enter a variable code from the list in Site and Variable Info worksheet in the yellow cell (G4) next to "Variable Code", then enter a valid start date and end date.

Now let's use the spreadsheet to download net radiation values from the SRBHOS database, for a few days in January, 2007.

1. On the Time Series worksheet, enter "SRBHOS:RTHNet" as the Site Code.
2. Enter "SRBHOS:516" as the Variable Code. This is the code for net radiation.
3. Enter "1/1/2007" as the Start Date.
4. Enter "1/5/2007" as the End Date. These data occur at 10-minute intervals or less. Therefore, the time series will be returned up to the beginning of 1/5/2007, which means we'll be getting back about four days' worth of data (January 1st all the way through the 4th).

The cells should be filled in as shown below.

GetValues	
Site Code	SRBHOS:RTHNet
Variable Code	SRBHOS:516
Start Date	1/1/2007
End Date	1/5/2007

5. Click the Click to Get Values button.

After a moment, the spreadsheet is populated with the time series of net radiation.

DateTime	Value
1/1/2007 0:10	-1.031000018
1/1/2007 0:20	0.455000013
1/1/2007 0:30	1.103000045
1/1/2007 0:40	-0.263999999
1/1/2007 0:50	-2.085000038
1/1/2007 1:00	-2.803999901
1/1/2007 1:10	-3.763000011
1/1/2007 1:20	-2.803999901
1/1/2007 1:30	-2.348999977

After you push the Click to Get Values button, an Excel macro populates a WaterOneFlow time series object from the ODM web service by using HydroObjects. The macro then reads the datetimes and values from the object to fill in the spreadsheet as shown in the figure above.

3.5 Extending this Example

The HydroObjects API can be used in a COM compliant or .Net software package to perform data access tasks like the ones demonstrated here. You can view the macro by selecting Tools -> Macros -> Visual Basic Editor from within Excel. The routines within the macros provide examples for how to implement HydroObjects within other software packages.

Another way to extend this example is to add additional data sources to the spreadsheet. Because the WaterOneFlow services follow a standard format, acquiring data from different sources (from NWIS to EPA Storet, for example) requires little more than changing the URL to the WSDL for the particular web service of interest. If you are familiar with Visual Basic and

would like to customize this example, there are a number of resources on <http://water.sdsc.edu/wateroneflow> to help in the process.

4.0 Ingesting STORET Data into Excel with HydroObjects

by Bora Beran

4.1 Introduction

This chapter demonstrates the use of the HydroObjects Application Programming Interface (API) to allow users direct access to EPA's STORET database of water quality information from within Excel. The HydroObjects API is a .Net DLL that is compiled to also be COM compliant. This allows the API to supplement other COM compliant software systems (Word, R, Python, etc.). The example here shows an Excel spreadsheet that has been extended through the development of Visual Basic for Applications (VBA) macros that use the HydroObjects API as a resource, in order to download data from STORET. With these Excel macros, the user is able to get metadata (e.g. site location, number of variables measured and their description) and time series data for any variable measured at a STORET site by simply clicking buttons within Excel. HydroObjects provides a layer of abstraction between Excel and the web services, which allows the steps demonstrated in this chapter for STORET to also be applicable for the other data sources within WaterOneFlow.

4.2 Computer Requirements

- Working internet connection
- MS Excel
- HydroObjects
- EPA.xls spreadsheet

4.3 Installation

If you haven't already installed HydroObjects (as required for Chapter 3), follow the procedure as in Chapter 3.

To download the EPA.xls spreadsheet, open a web browser and go to

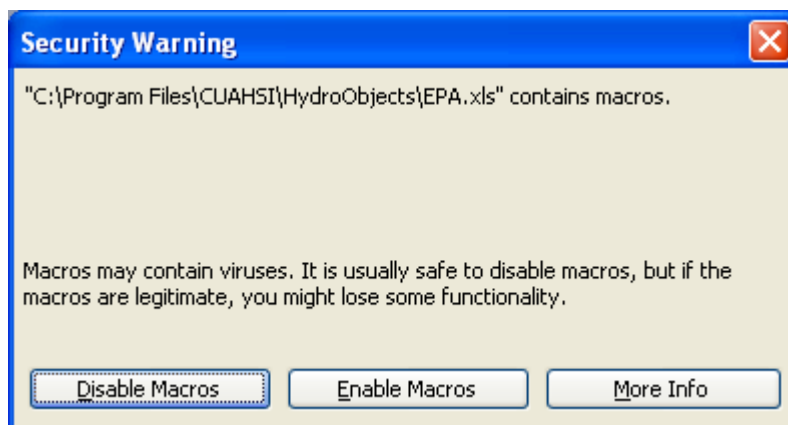
<ftp://ftp.crwr.utexas.edu/pub/outgoing/CUAHSI/HydroObjects>

and click the EPA.xls link.

4.4 Downloading EPA STORET Data

To download EPA STORET data, double click on EPA.xls to open the file in Excel.

EPA.xls contains macros that use WaterOneFlow web services to download EPA STORET data. (You may need to change your security settings in Excel to Medium in order to run macros.) If prompted about enabling macros when opening the file, make sure you enable the macros by choosing the "Enable Macros" option as shown below:

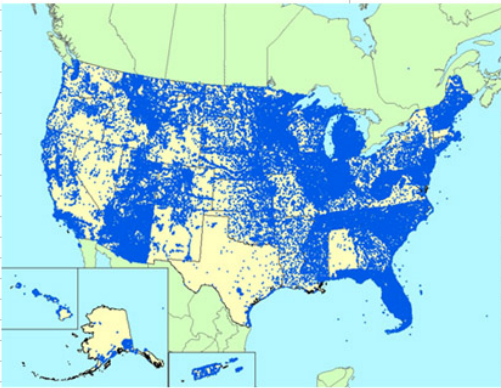


EPA.xls contains five worksheets: Data Source, Sites, Variable, Site and Variable Info, and Time Series. The Data Source worksheet provides general information about the data, web site for accessing the data and location of web services.

4.4.1 Obtaining Site and Variable Information

Sites

The Sites worksheet (shown below) provides an image of the site file (location of stations) for the data, and information on the sites. There are about 275,000 stations for downloading EPA data. For demonstration purposes, information on only eight random stations are included in the Sites worksheet as shown below (To download data for sites not listed in the excel file, the user must know the EPA STORET organization and station codes for the site of interest). Users can use this worksheet to keep a list of stations they regularly visit.

Sites						
This worksheet lists the names and site codes for a small subset of EPA sites, as well as latitude, longitude, HUC, station type and the state that the site is in. You can use this worksheet to store stations of interest. You can use the area next to the map to store the organization acronyms.						
	Organization Code		Organization Name			
	21FLPNS		Florida Department of Environmental Protection			
	MNPCA1		Minnesota Pollution Control Agency			
	UTAHDWQ		Utah Department Of Environmental Quality			
	21NJDEP1		NJ Department of Environmental Protection			
	11NPSPWRD		National Park Service			
	21SC60WQ		SC Department of Health & Environmental Control			
	21FLSFWM		South Florida Water Management District			
	21FLLOX		Loxahatchee River District (Florida)			
	21FLPOLK		Polk County Water Resources (Florida)			
	21IOWA		Iowa Dept. of Natural Resources			
	21MABCH		Massachusetts Department of Public Health			
	21MICH		Michigan Department of Environmental Quality			
	21RIBCH		Rhode Island Department of Health			
	21PABCH		Pennsylvania Department of Health			
Name		State	Site Code	HUC	Latitude	Longitude
BIG COLDWATER CR @ OLD IRON BRIDGE	Florida	21FLPNS:33030069	03140104	30.743084	-86.984	River/Stream
LAKE LARSON 7 MI NW OF RED WING	Minnesota	MNPCA1:25-0016	07040001	44.6288872	-92.66714	Lake
2 CULVERTS 1.65 MILES SO OF 1253	Utah	UTAHDWQ:4990460	16020102	40.93111	-111.898613	River/Stream
SHARK RIVER	New Jersey	21NJDEP1:5655200060	02030104	40.1958351	-74.03389	Estuary
EYDH032	Nevada	11NPSPWRD:GRBA_NURE_169	16020301	39.1489449	-114.136841	Spring
SAVANNAH RIVER AT LOCK AND DAM	South Carolina	21SC60WQ:SV-323	03060106	33.3712769	-81.94458	River/Stream
BOYNTON FARMS BASIN, AMESTOY FARMS S	Florida	21FLSFWM:BFBFAFSP	03090202	26.51562	-80.21293	Transport Canal
STONE HARBOR 108TH STREET BEACH	New Jersey	21NJDEP1:L7080928136	02040302	39.0438881	-74.76833	Ocean

Variables

The Variables worksheet contains a list of EPA STORET variable codes, medium codes and their descriptions.

Site and Variable Info

The Site and Variable Info worksheet (shown below) provides buttons for discovering what variables are measured at each site and brief metadata for each variable (eg. name, units, etc.). Each EPA station is identified by the collecting organization's code followed by a station code. To get the information on the variables measured at any station, you must enter the correct station identifier in the yellow cell (G3) next to Site Code, separating organization code and station code with a colon, as in MNPCA1:25-0016. The following screenshot shows an example using a station managed by Florida Department of Environmental Protection (21FLPNS) with station code 33030069.

GetSiteInfo			GetVariableInfo		
Site Code -->		21FLPNS:33030069	Variable Code -->		1473-1
<div>Click to Get Site Info</div>			<div>Click to Get Variable Info</div>		
After running GetSiteInfo, results will appear below.			After running GetVariableInfo, results will appear below.		
NameBIG COLDWATER CR @ OLD IRON BRIDGE			NamePhosphorus as P		
Latitude30.74308395			Unitsug/l		
Longitude-86.984			DescriptionPhosphorus as P (Sample Medium: Water)		
Variable Codes	Start Date	End Date	Value Count	Variable Description	
11810-1	9/15/1998	3/22/2005	166	Chlorophyll a, corrected for pheophytin (Sample Medium: Water)	
120-1	1/5/1999	3/22/2005	174	Chlorophyll a, uncorrected for pheophytin (Sample Medium: Water)	
131-1	1/26/1999	3/22/2005	86	Cloud cover (Sample Medium: Water)	
138-1	1/13/1998	3/15/2005	184	Color, True (Sample Medium: Water)	
201-1	1/6/1998	3/22/2005	361	Dissolved oxygen (DO) (Sample Medium: Water)	
11739-1	7/25/2000	3/22/2005	229	Enterococcus Group Bacteria (Sample Medium: Water)	
137-1	1/13/1998	3/22/2005	373	Fecal Coliform (Sample Medium: Water)	
330-1	3/16/1999	7/9/2002	21	Nitrogen, ammonia (NH3) as NH3 (Sample Medium: Water)	
333-1	3/16/1999	7/9/2002	21	Nitrogen, Kjeldahl (Sample Medium: Water)	
336-1	3/16/1999	7/9/2002	21	Nitrogen, Nitrite (NO2) + Nitrate (NO3) as N (Sample Medium: Water)	
1-1	1/13/1998	3/22/2005	327	pH (Sample Medium: Water)	
1473-1	3/16/1999	7/9/2002	21	Phosphorus as P (Sample Medium: Water)	
395-1	5/16/2000	6/10/2003	158	Precipitation (Sample Medium: Water)	
434-1	1/6/1998	3/22/2005	355	Salinity (Sample Medium: Water)	
456-1	1/27/1998	3/15/2005	141	Solids, Total Suspended (TSS) (Sample Medium: Water)	
139-1	1/13/1998	3/22/2005	335	Specific conductance (Sample Medium: Water)	
605-1	1/5/1999	3/22/2005	242	Temperature, air (Sample Medium: Water)	
481-1	1/6/1998	3/22/2005	368	Temperature, water (Sample Medium: Water)	
11737-1	1/26/1999	3/22/2005	366	Total Coliform (Sample Medium: Water)	

Once you click the yellow cell next to Site Code, a drop-down menu appears with a list of EPA STORET stations provided in the Sites worksheet. You can either choose one of the numbers from this list or you can enter any other station number of interest.

The following steps illustrate how to download site and variable information.

1. On the Site and Variable Info worksheet, enter "21NC02WQ:J9930000" as the Site Code in cell G3, and then click the Click to Get Site Info button. This site code is for a station along the Neuse River in North Carolina.

After a few seconds, a list parameter codes for each variable measured at the station and the period for which the data are available appears. To the right you can also see the number of records for each parameter and a description of parameter codes. Above this list you can see the

name of the station and its coordinates (Latitude and Longitude). Variable Codes include measured parameter and measurement medium information separated by a dash. Codes and their descriptions can be found in the Variables worksheet.

Name	NEUSE RIV AT CM NR AT MOUTH NR PAMLICO		
Latitude	35.1100		
Longitude	-76.4761		
Variable Codes	Start Date	End Date	Value Count
21-1	2/26/1997	12/7/2005	48
29-1	2/26/1997	12/7/2005	48
34-1	9/27/1999	10/14/1999	2
74-1	2/26/1997	12/7/2005	48

- On the Site and Variable Info worksheet, enter “1-1” as the Variable Code in cell L3, and then click the Click to Get Variable Info button. Make sure medium information is not missing. In this example “-1” indicates that medium is “Water”.

After a moment, information about the variable appears in the spreadsheet.

Name	pH		
Units	None		
Description	pH (Sample Medium: Water)		

4.4.2 Downloading Time Series Data

The Time Series worksheet provides access to the time series data. It requires four input parameters: the site code, the variable code, a start data, and an end date. Enter the site code into the yellow cell (G3) next to “Site Code”. You have the option of choosing the site code from the drop-down menu or entering a valid EPA organization/station code. Enter a variable code from the list in Site and Variable Info worksheet in the yellow cell (G4) next to “Variable Code”, then enter a valid start date and end date.

Now let’s use the spreadsheet to download pH values for the site that we queried above.

- On the Time Series worksheet, enter “21NC02WQ:J9930000” as the Site Code.
- Enter “1-1” as the Variable Code.
- Enter “1/1/1995” as the Start Date.
- Enter “1/30/2006” as the End Date.

The cells should be filled in as shown below.

GetValues	
Site Code	21NC02WQ:J9930000
Variable Code	1-1
Start Date	1/1/1995
End Date	1/30/2006

5. Click the Click to Get Values button.

After a moment, the spreadsheet is populated with the time series of pH values.

DateTime	Value
8/17/2004 11:00	8.2
4/27/2005 11:00	7.5
10/17/2002 13:35	7.9
7/24/2002 10:10	7.5
6/25/2002 9:44	7.8
11/12/2003 11:10	7.8
10/12/2004 11:16	7.8
6/26/2001 10:29	7.8
8/27/2000 11:38	8.4

In addition to the time series data, to the right of the data you will find some information about the time series, including units and date/time you downloaded it.

NOTE: The time series values are not in chronological order, which reflects how they are returned from the EPA website. WaterOneFlow services attempt to modify the data they retrieve as little as possible, in order to preserve the integrity of the data.

Congratulations! You have just used Excel and HydroObjects to download water quality data from EPA's STORET database.

5.0 Ingesting Weather and Streamflow Data into ArcGIS

by Ernest To

5.1 Introduction

This chapter demonstrates the ingestion of meteorological and streamflow data from Daymet, NAM and NWIS in ArcGIS using a custom ArcMap tool called *Weather Downloader*.

Weather Downloader is a very useful tool for downloading time series data needed to describe the hydrology of a given geographical area. It utilizes CUAHSI web services to access meteorological and streamflow data stored in public data repositories. Weather Downloader's user interface allows the user to extract data by specifying the following inputs:

- 1) a point feature class in ArcGIS that contains locations of interest;
- 2) variables of interest (e.g. precipitation, temperature), and;
- 3) time periods of interest.

The screenshot shows the 'Weather Downloader' dialog box with the following settings:

- Please select the point featureclass:** USGS_Gages_SanMarcos
- Please select identifier field in featureclass:** HydrolD
- Please type in the path and filename of the geodatabase that contains the TimeSeries table:** \\yellow\\c_drive\\Ernest\\WeatherDownloader\\analysis\\GIS\\projects\\for_HIS_work
- Historical Data**
 - Source: Daymet - Data available from 1/1/1980 to 12/31/2003.
 - Variables selected: 1 - Daily maximum temperature (deg C), 2 - Daily minimum temperature (deg C), 3 - Average temperature during day (deg C), 4 - Precipitation (cm), 5 - Vapor Pressure Deficit (Pa), 6 - Solar Radiation (SRAD) (W m⁻²), 7 - Day length (s).
 - Start date: 1/1/1990, End date: 12/31/2003
- Forecasted Values**
 - Source: UNIDATA North American Model 12KM - at 3-hour intervals for the next 3.5
 - Variable selected: 8 - Total Precipitation (kg m⁻² or mm rainfall)
 - * Note that forecasts are in zulu time (Greenwich Mean Time).
- Streamflow Data**
 - Source: USGS National Water Information System (NWIS)
 - Variable selected: 9 - Daily Streamflow Data (cfs)
 - Please select field that contains USGS gage number: OBJECTID
 - Start date: 1/1/1990, End date: 12/31/2003
- Other Web Services (downloads time series for one location and one variable only)**
 - NOTE: Works independently from the selected featured class.
 - Variable selected: 10 - Other (Please click on "Advanced" to specify source, method and parameters")
 - Advanced button
- Output Options**
 - ☒ Replace contents of TimeSeries table.
 - ☐ Append to contents of TimeSeries table.
 - OK, Cancel buttons

When executed, Weather Downloader cycles through each location in the point feature class, downloads the desired data through CUAHSI's web services and writes them to the TimeSeries table of an Arc Hydro geodatabase specified by the user.

Arc Hydro is an ArcGIS data model which facilitates preprocessing and integration of hydrological geospatial and temporal data with hydrologic and hydraulic simulation models. For more information on Arc Hydro, see:

<http://www.crwr.utexas.edu/giswr/hydro/index.html>

5.2 Computer and Skill Requirements

To complete this exercise, your computer must meet the following requirements:

- Windows 2000 or above
- ESRI ArcGIS 9.1 or above
- Microsoft .NET Framework version 2.0 (x86) (this is available at:
<http://www.microsoft.com/downloads/details.aspx?FamilyID=0856eacb-4362-4b0d-8edd-aab15c5e04f5&DisplayLang=en>)
- Internet connection

This exercise assumes that you have some familiarity with the ArcGIS 9.1 software environment.

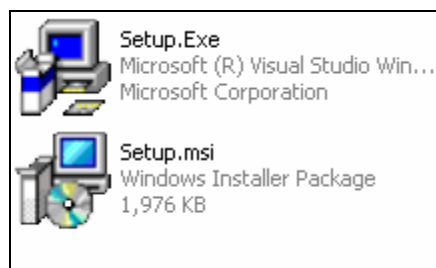
5.3 Installation

To install the Weather Downloader tool:

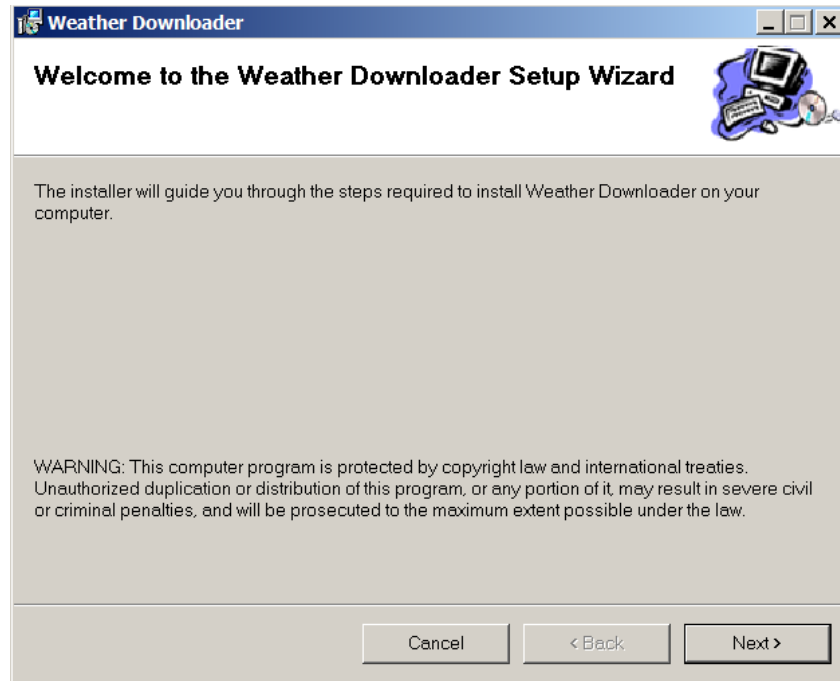
1. Download the Setup file from the following link:

ftp://ftp.crwr.utexas.edu/pub/outgoing/CUAHSI\HIS_workbook/20070720/InstallationFiles/WeatherDownloaderSetup_20070629.zip

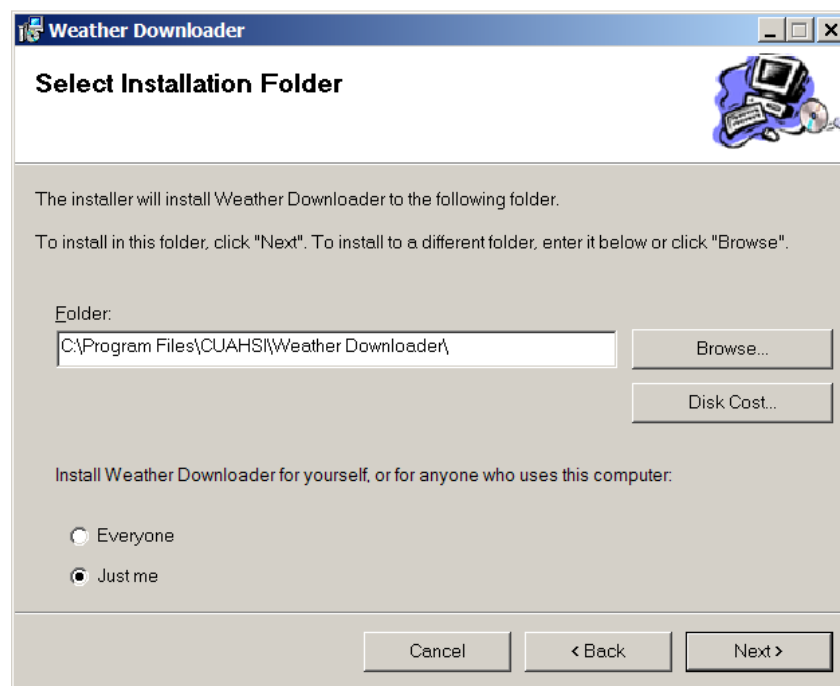
2. After download, unzip the file on a local drive (contents shown below), and double click on Setup.Exe to run the setup file.



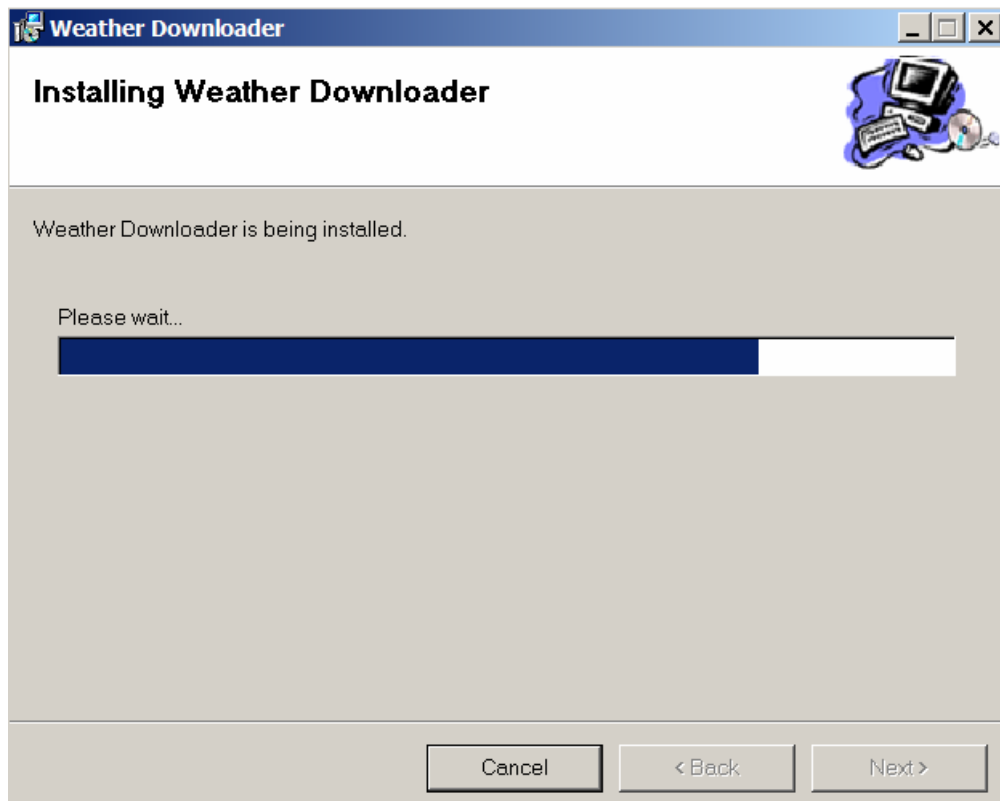
3. In the Weather Downloader Setup Wizard, click Next to start the installation process.



4. Use the default installation folder for Weather Downloader or choose your preferred location, specify to install the program for “Everyone” or “Just Me”, and click Next.



5. Click Next to get the Confirm Installation Window, and then click Next to start the installation process. You should see the progress bar as shown below:



6. After the Weather Downloader is successfully installed, click Close to finish the weather downloader installation process.

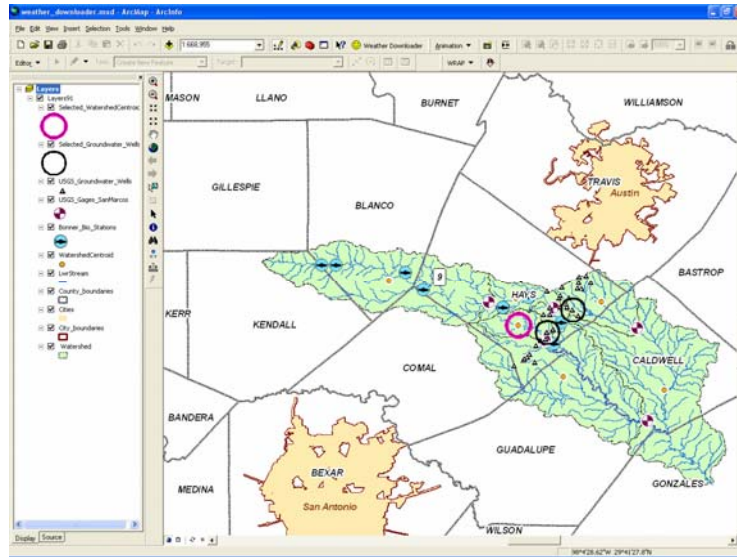
Besides installing the weather downloader, the installation process also adds two files (weather_downloader.mxd and weather_downloader.mdb) in C:\Program Files\CUAHSI\Weather Downloader (or any other location specified during the set up) that you will use in this exercise.

5.4 Retrieving Data with Weather Downloader

In this exercise, you will use Weather Downloader to retrieve climate and streamflow time series data for the San Marcos basin in Texas.

5.4.1 Opening the Map

1. Double click on weather_downloader.mxd to open the ArcMap document as shown below. (The file is located in the directory where you installed Weather Downloader.)



This map shows the San Marcos watershed which is located between the cities of Austin and San Antonio in Texas. The green polygons are the catchments and the orange points are their centroids. The purple and white symbols are USGS stream gages. The black triangles are USGS groundwater well stations.

2. Open the attribute table for USGS_Gages_SanMarcos. The table is shown below.

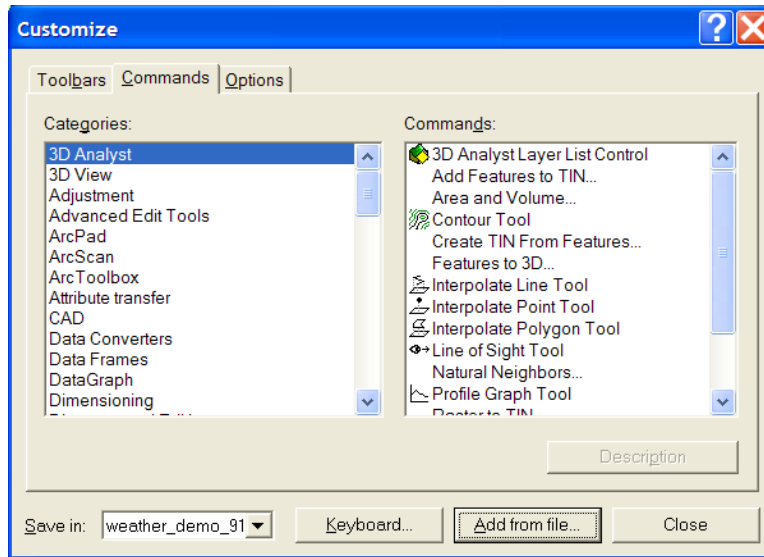
OBJECTID*	Shape*	HydroID	HydroCode	FType	Name	JunctionID
1	Point	1000001	08171000	0	Blanco River At Wimberley Tx	640
2	Point	1000002	08171300	0	Blanco River Nr Kyle Tx	641
3	Point	1000003	08172400	0	Plum Creek At Lockhart Tx	639
4	Point	1000004	08170500	0	San Marcos R At San Marcos Tx	642
5	Point	1000005	08172000	0	San Marcos River At Luling Tx	638

Record: 1 Show: All Selected Records (0 out of 5 Selected.) Options

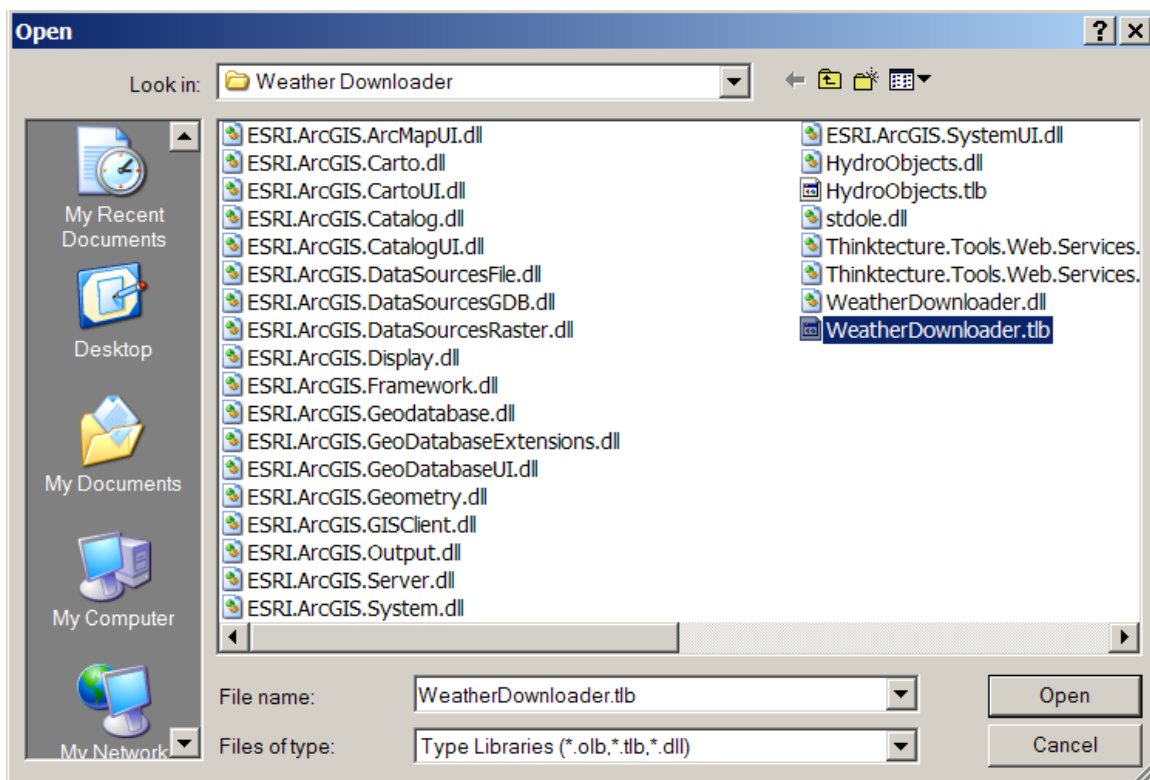
From the table, we see that this feature class contains five points representing USGS stream gages. The values in the field HydroCode are the unique stream gage IDs assigned to each gage by the USGS. These are public identifiers by which these features may be identified in any system. The field HydroID stores the unique ID of these features in the geodatabase. The HydroID links each stream gage to other features in the geodatabase, such as time series records which you will download in this exercise.

5.4.2 Adding Weather Downloader to ArcMap

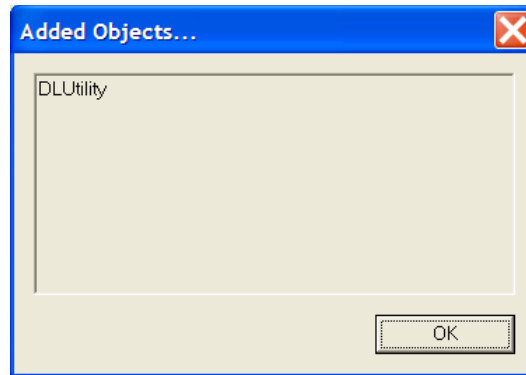
1. From ArcMap's standard menu bar, click Tools, and then click Customize....
2. Click on the Commands tab, and then click on Add from file....



3. Navigate to the directory: “C:\Program Files\CUAHSI\Weather Downloader” (or wherever you installed Weather Downloader on the local drive).

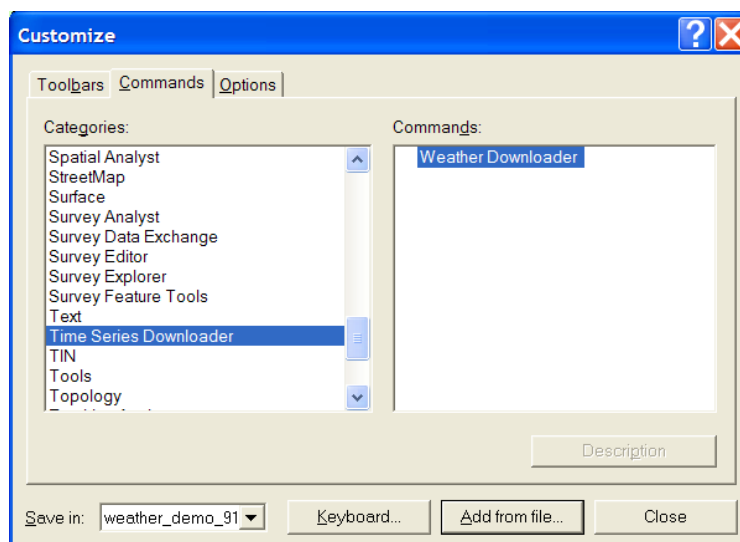


4. Select WeatherDownloader.tlb and click Open. If the installation is successful, the dialog box shown below should appear to confirm that the tool has been successfully added.



(If you see a message saying “No new objects” or something similar, check to see that you have administrator privileges on your computer. You must be able to register DLLs in order to add Weather Downloader to ArcMap.)

5. Click OK, and then drag Weather Downloader tool (see below) from the customize window to anywhere in the ArcMap toolbar to create a new button named Weather Downloader. Then close the customize window.



The weather downloader tool will appear as a button in the ArcMap toolbar as shown below:



5.4.3 Downloading Weather Data

1. Click on the Weather Downloader button to get the form shown below.

Weather Downloader

Please select the point featureclass: Selected_Groundwater_Wells

Please select identifier field in featureclass: HydroID

Please type in the path and filename of the geodatabase that contains the TimeSeries table.
C:\Program Files\CUAHSI\Weather Downloader\weather_downloader.mdb Browse

Atmospheric | Surface | Subsurface | Custom

Historical Data
Source: Daymet - Data available from 1/1/1980 to 12/31/2003.

☐ 1 - Daily maximum temperature (deg C) ☐ 5 - Vapor Pressure Deficit (Pa)
☐ 2 - Daily minimum temperature (deg C) ☐ 6 - Solar Radiation (SRAD) (W m -2)
☐ 3 - Average temperature during day (deg C) ☐ 7 - Day length (s)
☐ 4 - Precipitation (cm)



Start date: 1/1/1990 End date: 12/31/2003

Forecasted Values
Source: UNIDATA North American Model 12KM - at 3-hour intervals for the next 3.5

☐ 8 - Total Precipitation (kg m-2 or mm rainfall)

* Note that forecasts are in zulu time (Greenwich Mean Time).

*Created by Ernest Sin Chit To
University of Texas at Austin*

 
photo from www.free-pictures-photos.com

☒ Replace contents of TimeSeries table.
☐ Append to contents of TimeSeries table.

OK Cancel

The form contains several input options, including four tabs for retrieving different types of data: Atmospheric, Surface, Subsurface, and Custom. The first three tabs are pre-programmed to work with a specific data provider and web service. The last tab, Custom, allows you to enter the WSDL of any other WaterOneFlow web service that wasn't already included with Weather Downloader. This is useful when a new service is created after the Weather Downloader program was last compiled.

Selecting a point layer

2. In the top left combo box (located below “Please select the point feature class”), select the feature class that contains the point locations where you want weather data. For this exercise, select USGS_gages_SanMarcos.

Selecting identifier field in the point layer

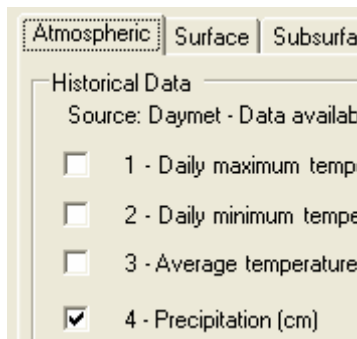
3. In the second combo box, select the identifier field that will link the feature class to the time series data. By default, the tool selects HydroID in the attribute table of the input layer as the identifier field. If the HydroID field is unavailable, the combo box defaults to the first integer field in the attribute table. For this exercise, make sure HydroID is chosen in this combo box.

Selecting data output location

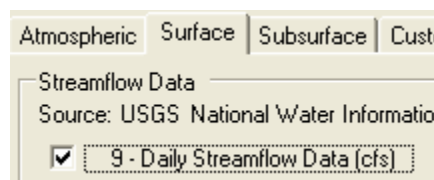
4. By default, the data output location is set to look for a geodatabase named “weather_downloader.mdb” that resides in the same directory as the ArcMap document. If a different geodatabase is desired, the user can click on the Browse button and navigate to the location of the desired geodatabase. For this exercise, leave the default of C:\Program Files\CUAHSI\Weather Downloader\weather_downloader.mdb, or wherever you installed Weather Downloader.

Selecting variables

5. Select the variables that you want to download by clicking on the check boxes next to the variable descriptions. First let’s get some atmospheric data. Make sure the Atmospheric tab is selected.
6. In the frame for Historical Data, check the box for “4 - Precipitation (cm)”.

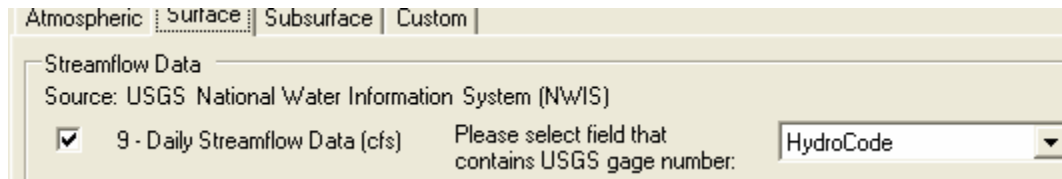


7. Now click the Surface tab. In the frame for Streamflow Data, check the box for “9 – Daily Streamflow Data (cfs)”



Please note that for Streamflow data, NWIS uses the USGS gage number (instead of latitude and longitude) to locate the point of interest. Recall that these gage numbers are stored in the HydroCode field of our USGS_Gages_SanMarcos feature class.

8. In the associated combo box on the Weather Downloader form, in the frame for Streamflow Data (see below), select HydroCode as the USGS gage number field.



Atmospheric | **Surface** | Subsurface | Custom

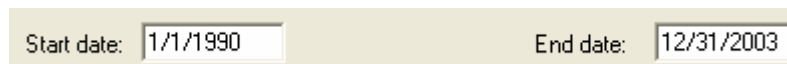
Streamflow Data

Source: USGS National Water Information System (NWIS)

☒ 9 - Daily Streamflow Data (cfs) Please select field that contains USGS gage number: **HydroCode**

Specifying period of interest

You can specify the start dates and end dates for the Daymet and USGS stations by typing into the Start date and End date boxes,



Start date: 1/1/1990 End date: 12/31/2003

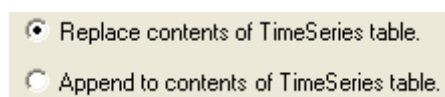
You can type in any date range. However, the web services will only return data that are within the period of record for the location of interest.

9. For this exercise, leave the default start and end dates unchanged (start date = 01/01/1990 and end date = 12/31/2003) for Daymet and NWIS.

For UNIDATA, the tool is set to download the most recent forecast results from the NCEP North American Mesoscale Model (12km). Because forecasts are generated for a fixed period of time into the future (i.e. 84 hours), the web service is “hard-wired” to download all the data in this period. The North American Mesoscale model is run every six hours at 00:00, 06:00, 12:00, 18:00 Greenwich Mean Time with forecasts being made at 3 hour-intervals into the next 84 hours (i.e. 3.5 days).

Replace/Append to Timeseries table

10. The last two radio buttons in the form let you choose between overwriting existing data in the TimeSeries table (Replace contents of TimeSeries table), or keeping the existing data in the TimeSeries table (Append to contents of TimeSeries table). For this exercise, select the Replace contents of TimeSeries table option.



☒ Replace contents of TimeSeries table.

☐ Append to contents of TimeSeries table.

The completed form should look like the one below for the Atmospheric tab.

The screenshot shows the 'Weather Downloader' dialog box. The title bar is blue with the text 'Weather Downloader' and a close button. The main area has a light beige background. At the top, there are two labels: 'Please select the point featureclass:' and 'Please select identifier field in featureclass:'. Below these are two dropdown menus. The first dropdown menu is set to 'USGS_Gages_SanMarcos' and the second is set to 'HydroID'. Below these is a label: 'Please type in the path and filename of the geodatabase that contains the TimeSeries table.' followed by a text box containing 'C:\Program Files\CUAHSI\Weather Downloader\weather_downloader.mdb' and a 'Browse' button. Below the text box are four tabs: 'Atmospheric', 'Surface', 'Subsurface', and 'Custom'. The 'Atmospheric' tab is selected. Below the tabs is a section titled 'Historical Data' with a sub-label 'Source: Daymet - Data available from 1/1/1980 to 12/31/2003.' Below this are seven checkboxes arranged in two columns. The first column has checkboxes for '1 - Daily maximum temperature (deg C)', '2 - Daily minimum temperature (deg C)', '3 - Average temperature during day (deg C)', and '4 - Precipitation (cm)'. The second column has checkboxes for '5 - Vapor Pressure Deficit (Pa)', '6 - Solar Radiation (SRAD) (W m⁻²)', and '7 - Day length (s)'. The checkbox for '4 - Precipitation (cm)' is checked. Below the checkboxes are two date fields: 'Start date: 1/1/1990' and 'End date: 12/31/2003'. Below the date fields is a section titled 'Forecasted Values' with a sub-label 'Source: UNIDATA North American Model 12KM - at 3-hour intervals for the next 3.5'. Below this is a checkbox for '8 - Total Precipitation (kg m⁻² or mm rainfall)'. Below the checkbox is a note: '* Note that forecasts are in zulu time (Greenwich Mean Time)'. At the bottom of the dialog box, there are two radio buttons: 'Replace contents of TimeSeries table.' (which is selected) and 'Append to contents of TimeSeries table.'. To the right of the radio buttons are 'OK' and 'Cancel' buttons. At the bottom of the dialog box, there is a small image of a cloudy sky, a logo for 'CRWR' (Center for Research in Water Resources), and text: 'Created by Ernest Sin Chit To University of Texas at Austin' and 'photo from www.free-pictures-photos.com'.

Weather Downloader

Please select the point featureclass: USGS_Gages_SanMarcos

Please select identifier field in featureclass: HydroID

Please type in the path and filename of the geodatabase that contains the TimeSeries table.
C:\Program Files\CUAHSI\Weather Downloader\weather_downloader.mdb

Atmospheric Surface Subsurface Custom

Historical Data
Source: Daymet - Data available from 1/1/1980 to 12/31/2003.

☐ 1 - Daily maximum temperature (deg C) ☐ 5 - Vapor Pressure Deficit (Pa)

☐ 2 - Daily minimum temperature (deg C) ☐ 6 - Solar Radiation (SRAD) (W m⁻²)

☐ 3 - Average temperature during day (deg C) ☐ 7 - Day length (s)

☒ 4 - Precipitation (cm)

Start date: 1/1/1990 End date: 12/31/2003

Forecasted Values
Source: UNIDATA North American Model 12KM - at 3-hour intervals for the next 3.5

☐ 8 - Total Precipitation (kg m⁻² or mm rainfall)

* Note that forecasts are in zulu time (Greenwich Mean Time).

Replace contents of TimeSeries table.
Append to contents of TimeSeries table.

OK Cancel

Created by Ernest Sin Chit To
University of Texas at Austin

photo from www.free-pictures-photos.com

CRWR

For the Surface tab, the form should look like the one below.

Weather Downloader



Please select the point featureclass: Please select identifier field in featureclass:

Please type in the path and filename of the geodatabase that contains the TimeSeries table.

Atmospheric ☒ Surface ☐ Subsurface ☐ Custom ☐

Streamflow Data
Source: USGS National Water Information System (NWIS)
☒ 9 - Daily Streamflow Data (cfs) Please select field that contains USGS gage number:
Start date: End date:

*Created by Ernest Sin Chit To
University of Texas at Austin*

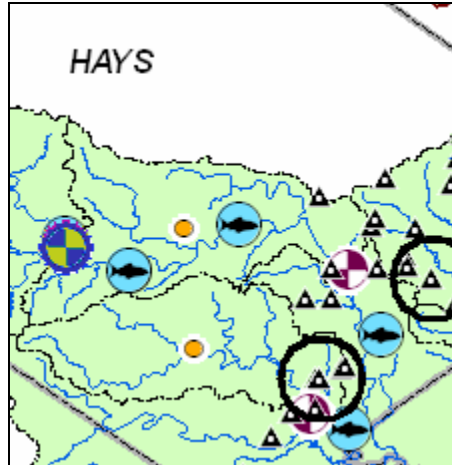
 
photo from www.usgs.gov

☒ Replace contents of TimeSeries table.
☐ Append to contents of TimeSeries table.

Running the tool

11. Once all the inputs are provided, click OK to run the tool.

The tool highlights the points for which the data are being downloaded as shown below. Since you are downloading two types of data, each point will be highlighted twice.



Besides highlighting the points, the progress of the tool is shown in the status bar in ArcMap at the bottom-left corner of the ArcMap window. Please note that the tool may take a few minutes to complete the data download.

Once the download is complete, the tool will display the time taken in a dialog box as shown below.

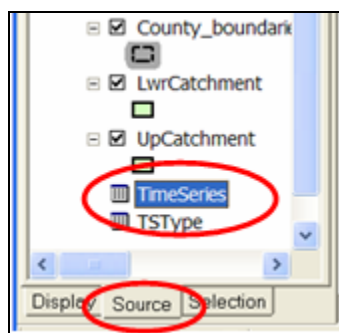


12. Click OK to continue.

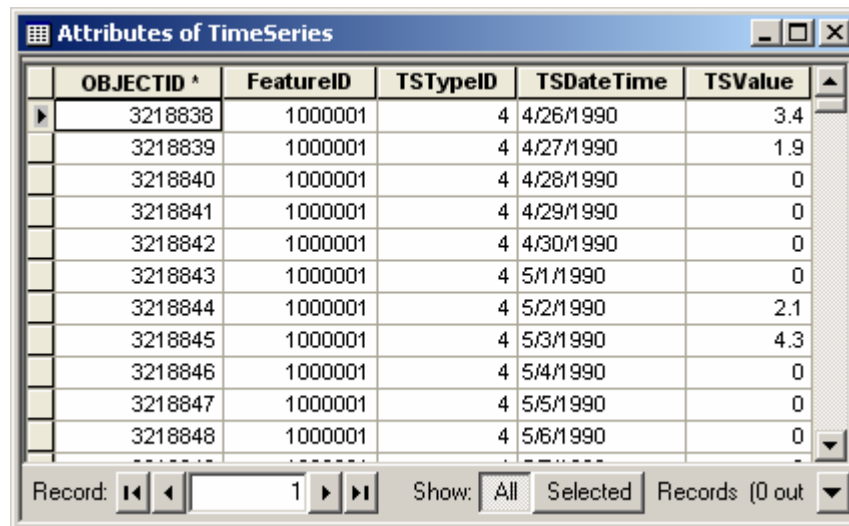
Inspecting the results

The weather downloader tool downloads the data in Arc Hydro time series format.

13. In the ArcMap table of contents (left window), select the source tab and then select the TimeSeries table as shown below.



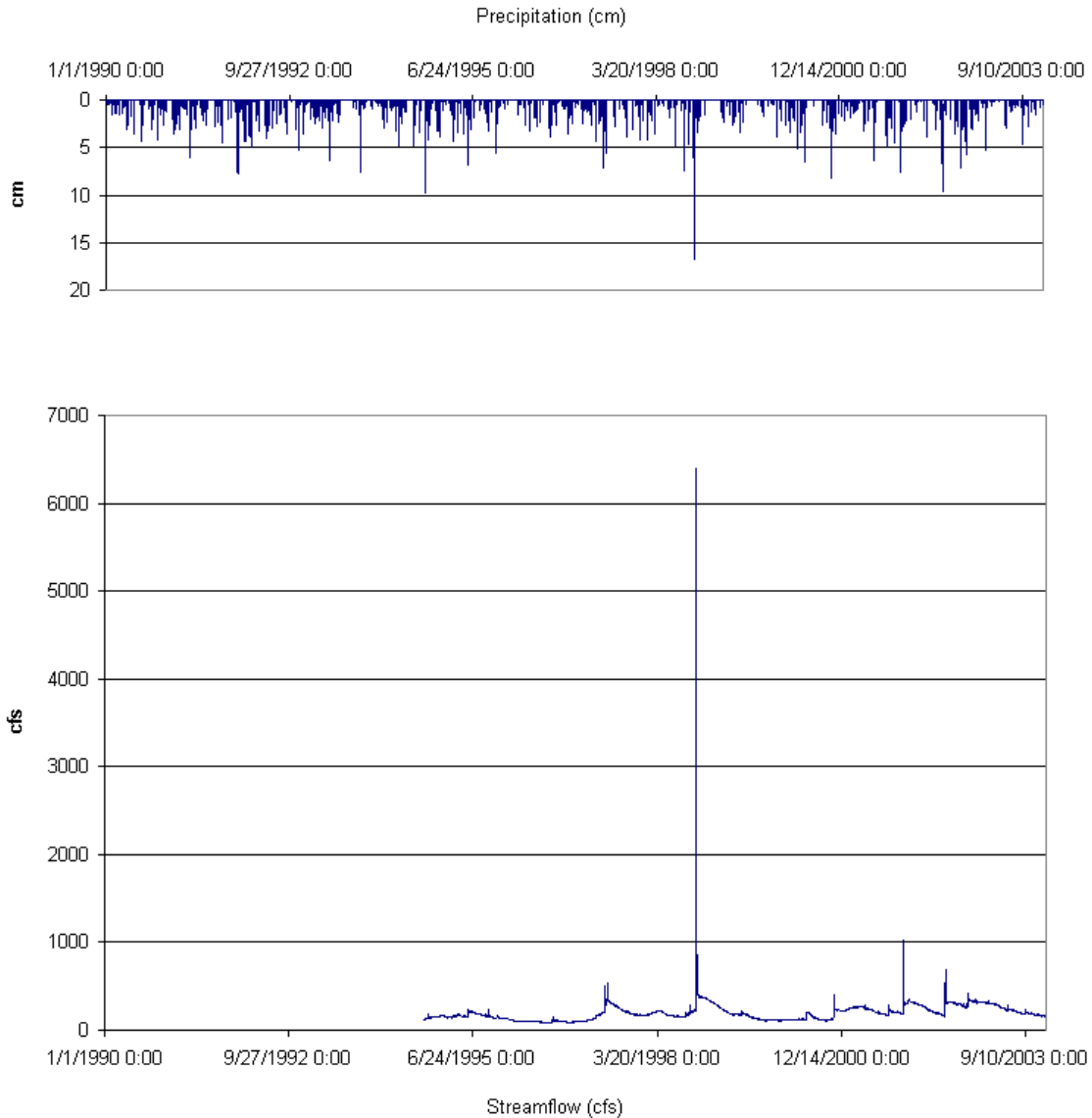
12. Right click on TimeSeries table, and then click Open to view the contents of the table as shown below:



OBJECTID *	FeatureID	TSTypeID	TSDateTime	TSValue
3218838	1000001	4	4/26/1990	3.4
3218839	1000001	4	4/27/1990	1.9
3218840	1000001	4	4/28/1990	0
3218841	1000001	4	4/29/1990	0
3218842	1000001	4	4/30/1990	0
3218843	1000001	4	5/1/1990	0
3218844	1000001	4	5/2/1990	2.1
3218845	1000001	4	5/3/1990	4.3
3218846	1000001	4	5/4/1990	0
3218847	1000001	4	5/5/1990	0
3218848	1000001	4	5/6/1990	0

The time values are stored in the TSDateTime field, and the corresponding data are stored in the TSValue field. The TSTypeID field stores variable codes which can be looked up in the TSType table for more information such as units, time interval, etc.

You can export the attribute table into a .dbf file and do some fancy plots with it. For instance the following are the hyetograph and hydrograph of USGS gage 08170500 (San Marcos River at San Marcos) that are plotted from the data you have just downloaded. Notice how well the major storm events in the graphs line-up with each other despite the different sources of data. Pretty amazing, isn't it?



5.5 Downloading Data from Other Web Services

Weather Downloader allows the user to utilize other web services in CUAHSI's arsenal through the "Other Web Services" feature. Some of these web services include:

- 1) NWIS Instantaneous Irregular Data (Field Measurements; Water Quality)
- 2) NWIS Unit Values (Real Time) Data
- 3) NWIS Ground Water Data
- 4) MODIS (Moderate Resolution Imaging Spectroradiometer) data

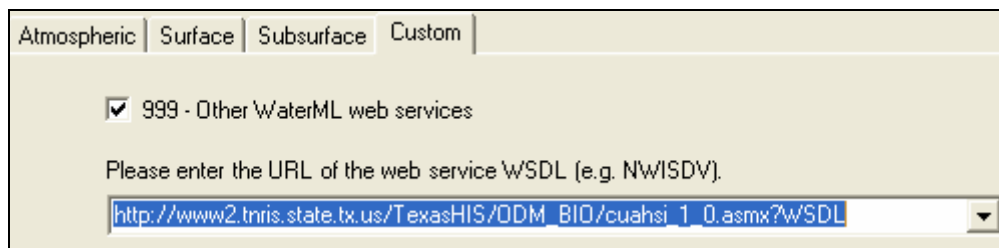
This feature allows the invocation of web services in a similar manner to programming in VB.NET. The user has to choose or type the WSDL (Web Service Description Language) of the

desired web service, the name of the desired web method, and the input parameters for the method into a form and then invoke the web service through Weather Downloader.

This basic form of input provides versatility and allows the use of web services not yet incorporated into the main interface of Weather Downloader. The limitation of this feature is that it works on one location at a time and does not cycle through a group of points as with Daymet, NAM and NWIS daily values web services.

1. To access the “Other Web Services” feature, open Weather Downloader, and click the Custom tab.
2. Place a check next to “999 – Other WaterML webservices”. Be sure to uncheck other data sources that you may have already downloaded on other tabs (i.e., “4” and “9”).
3. For the WSDL, select the one for ODM_BIO in Texas. This web service provides biological data from Texas State University.

http://www2.tnris.state.tx.us/TexasHIS/ODM_BIO/cuahsi_1_0.asmx?WSDL

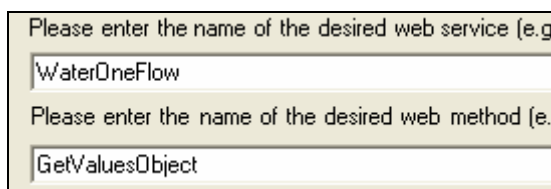


Atmospheric | Surface | Subsurface | Custom

☒ 999 - Other WaterML web services

Please enter the URL of the web service WSDL (e.g. NWISDV).

4. For the web service name, enter “WaterOneFlow”.
5. For the web method, enter “GetValuesObject”.



Please enter the name of the desired web service (e.g.

Please enter the name of the desired web method (e.

6. Next, you would specify the location (site code), variable, and period of record for the time series you want to retrieve. Some default were automatically placed in these text boxes when you selected the WSDL for ODM_BIO, so just accept the defaults.
7. In the Replace/Append option at the very bottom of the form, choose the Append to contents of TimeSeries table option, and then click OK.

Please enter the parameters for the web method (handles

Location:	00:9	
Variable:	00:Micropterus salmoides	
Start Date:	1900-08-01	
End Date:	2007-08-01	
Authentication Code:		

Please specify HydrolD and TSTypeID to identify the downr

HydrolD	-999	TSTypeID	999
---------	------	----------	-----

☒ Replace contents of TimeSeries table.
☐ Append to contents of TimeSeries table.

8. Once download is complete, open up the TimeSeries table in the geodatabase and inspect the values. What you'll see are counts for the selected species at each time step.

This concludes the Weather Downloader demonstration.

6.0 Plotting MODIS Data with Matlab

by David Tarboton

6.1 Introduction

Matlab users can take advantage of web service methods by using the *createClassFromWsd* function. This function creates a Matlab class based on a WSDL. The URL to the WSDL is provided to the function when the function is called. This chapter demonstrates how to call a CUASHI web service from Matlab, parse the result, and plot a time series graph. In the exercise, you will write an M-file that creates a plot of the Cloud Optical Thickness Water Phase variable from NASA's MODIS database of remote sensing data.

6.2 Computer and Skill Requirements

To complete this exercise, your computer must meet the following requirements:

- Working Internet connection
- Matlab version 7 (or greater) software

This exercise assumes that you have some familiarity with the following software environments:

- Matlab version 7

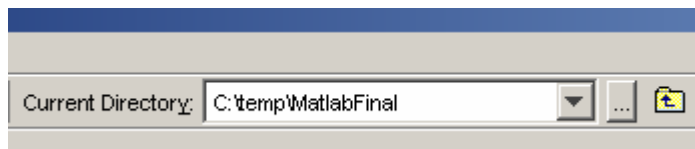
NOTE: The source code for the `parse_xml` M-file used in this exercise is located in Appendix A. The source code for the `MODISPlot_xml` M-file is located in Appendix B.

6.3 Procedure

WaterOneFlow web services return data either in XML or Object form. XML is a very useful format for web services, because it is platform independent and self-describing. Yet while Matlab can create a class from a WSDL, it does not contain inherent classes for working with XML. Therefore, we'll begin the exercise by creating an M-file called *parse_xml* that will serve as our Matlab XML parser.

6.3.1 Setting up the XML Parser

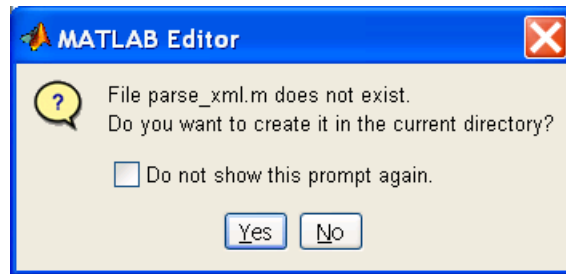
1. Start Matlab.
2. Set the current directory to the location where you wish to perform the work.



3. If you already have a copy of the `parse_xml.m` file, copy the file to the working directory and go to the section entitled Retrieving MODIS Data. Otherwise, continue to the next step.
4. In the Command Window, enter the command

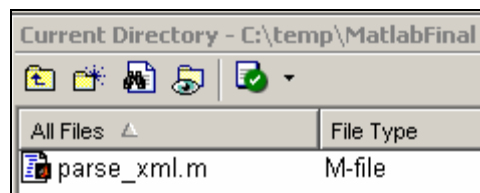
```
edit parse_xml
```

If you get the following prompt, click Yes to create the file parse_xml.



5. In the Matlab editor, enter the code for parse_xml as found in Appendix A. If you are viewing an electronic copy of this document, try copying and pasting the code.
6. In the Matlab editor, click the File menu, and then click Save.

You should now see the parse_xml.m file in Matlab's Current Directory window.



6.3.2 Retrieving MODIS Data

With the XML parser in place, the next step is to build an M-file for retrieving MODIS data.

NOTE: Instead of entering code as shown below, you could copy the code from Appendix B, or simply refer to MODISPlot_xml.m if you were provided a copy of the finished file with this document.

1. In the Command Window, enter the command

```
edit MODISPlot_xml
```

2. In the Matlab editor, enter the following lines of code. This code creates an instance of the MODIS class from the WSDL.

```
% Create class.  
wsdl='http://water.sdsc.edu/waterOneFlow/MODIS/Service.asmx?WSDL';  
createClassFromWsd1(wsdl);  
  
% This creates an instance of the class.  
svsMODIS = MODIS;
```

3. In the Matlab editor, enter the following lines of code. This code sets the parameters for calling the GetValues method, and then calls GetValues. In this case, the parameters instruct the MODIS class to retrieve Cloud Optical Thickness Water Phase values for 2004, spatially averaged over an area that roughly covers Travis County in Texas. The plotArea parameter indicates that the values should include areas both over the land surface and the oceans. You will find a list of valid codes and keywords for retrieving MODIS data at:

<http://water.sdsc.edu/waterOneFlow/MODIS/Service.asmx>

```
% Specify input parameters.
w='-98.2' % West longitude.
s='30' % South latitude.
e='-97.3' % East longitude.
n='30.7' % North latitude.
location=['GEOM:BOX(','w',' ','s',' ','e',' ','n','')]
% Variable Code 11 = Cloud Optical Thickness Water Phase.
variableCode='MODIS:11/plotarea=land'
startDate='2004-01-01'
endDate='2004-12-01'

% Call the GetValues function to get the time series data.
xmlValues=GetValues(svsMODIS,location,variableCode, ...
    startDate,endDate, '')
```

4. In the Matlab editor, click the File menu, and then click Save.
5. In the Matlab Command Window, enter the command

MODISPlot_xml

This command runs the code in the MODISPlot_xml.m file. When the code runs, you will see the values that have been set for the parameters to the GetValues call, followed by the XML String returned from the web service that is saved in the variable called xmlValues.

```
endDate =

2004-12-01

xmlValues =

<timeSeriesResponse xmlns:gml="http://www.opengis.net/gml"
```

Alternatively you can run each of the commands above individually in sequence by highlighting them and pressing F9 in the Matlab editing environment.

To get an understanding of the structure of the XML output we will copy it to a file and view it using an XML viewer such as Internet Explorer.

6. Copy the XML output (beginning with `<timeSeriesResponse` and ending with `</timeSeriesResponse>`) and paste the text into a new text document using a text editor.
7. Save the document as `MODISExample.xml`, and close the text editor.
8. Open `MODISExample.xml` with an XML viewer.

Below is a screenshot of the document, as viewed in Internet Explorer. Some of the XML elements have been collapsed for readability.

```
<timeSeriesResponse xmlns:gml="http://www.opengis.net/gml" xmlns:xlink="http://www
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:wtr="http://ww
<queryInfo>
- <timeSeries name="MODIS">
+ <sourceInfo xsi:type="DataSetInfoType">
- <variable>
  <variableCode vocabulary="MODIS">11</variableCode>
  <variableName>Cloud Optical Thickness Water Phase (QA WT)</variableName>
  <units unitsAbbreviation="um" unitsCode="55" unitsType="Length">micron</units>
</variable>
- <values unitsAbbreviation="um" unitsCode="55" count="12">
  <value dateTime="2004-01-01T00:00:00">18.9011</value>
  <value dateTime="2004-02-01T00:00:00">31.9133</value>
  <value dateTime="2004-03-01T00:00:00">21.5478</value>
  <value dateTime="2004-04-01T00:00:00">18.8422</value>
  <value dateTime="2004-05-01T00:00:00">14.2878</value>
  <value dateTime="2004-06-01T00:00:00">9.4344</value>
  <value dateTime="2004-07-01T00:00:00">5.3655</value>
  <value dateTime="2004-08-01T00:00:00">7.3822</value>
  <value dateTime="2004-09-01T00:00:00">7.9600</value>
  <value dateTime="2004-10-01T00:00:00">10.7789</value>
  <value dateTime="2004-11-01T00:00:00">25.8700</value>
  <value dateTime="2004-12-01T00:00:00">19.5344</value>
</values>
</timeSeries>
</timeSeriesResponse>
```

Notice the hierarchy of data in the XML. Understanding the hierarchy is crucial to navigating the XML in Matlab. The `parse_xml` utility converts an XML string into a Matlab structure, the contents of which can be accessed through parent/child relationships. For example, in `MODISExample.xml`, the name of the time series variable is stored in the `variableName` tag. This tag is a child of the `variable` tag, which is a child of the `timeSeries` tag, which is a child of the `timeSeriesResponse` tag, which is a child of the XML document itself. In other words, the `variableName` tag is four levels down. The order of child tags in the same “generation” is also important. The `variable` tag is the second child of the `timeSeries` tag. Therefore, the complete path to the `variableName` tag can be summarized as follows:

- a) Get the first child (there is always only one) of the XML document. (This returns `timeSeriesResponse`)
- b) Get the second child of this element. (This returns `timeSeries`)
- c) Get the second child of this element. (This returns `variable`)
- d) Get the second child of this element. (This returns `variableName`)

This logic will be used to retrieve information from the Matlab structure created from this XML string.

9. Close the XML file.
10. In the Matlab editor for `MODISPlot_xml`, enter the following lines of code. This code calls the `parse_xml` function which feeds the XML string to the parser, and returns a Matlab structure object created from the XML string.

```
% Parse the XML string.
structValues=parse_xml(xmlValues);
```

Execute just this code by highlighting it and pressing F9. Examine the structure `structValues` that is returned, to see that it contains the complete content of the XML string in a series of nested structures. For example, if you type:

```
structValues
```

in the Matlab command window, you will see the following.

```
>> structValues

structValues =

    child: [1x1 struct]
```

This indicates that the structure returned contains one child that is itself a structure named `child`. To drill down further into this structure, the logic of the XML needs to be followed. For example

```
structValues.child.child(2).child(2).child(2)
```

returns the following

```
>> structValues.child.child(2).child(2).child(2)

ans =

    tag: 'VARIABLENAME'
   attribs: [1x1 struct]
    value: 'Cloud Optical Thickness Water Phase (QA WT)'
    child: []
```

This displays the `VARIABLENAME` tag which is the second child of the element `variable`, which is the second child of the element `timeSeries` which is the second child of the element `timeSeriesResponse` which is the only child element in the structure.

11. In the Matlab editor for `MODISPlot_xml`, enter then execute following lines of code.

The display functions write the name and units for the variable to the Matlab Command Window.

```
% Report the name and units of the chosen variable.
display(structValues.child.child(2).child(2).child(2).value)
display(structValues.child.child(2).child(2).child(3).value)
```

12. In the Matlab editor for `MODISPlot_xml`, enter the following lines of code. This code retrieves the time series records from the structure then loops through all the records and stores the datetimes and values in an array. The `datenum` function converts the datetimes to numeric format, which aids in plotting the data.

```
% Get the <value> tags.
Recs=structValues.child.child(2).child(3).child;
[d1,d2]=size(Recs)

% Build arrays of datetimes and values.
for i=1:d2
    % Reformat date to that Matlab can understand it.
    datetime=Recs(i).attribs(1).value;
    year=datetime(1:4);
    month=datetime(6:7);
    day=datetime(9:10);
    datetime=[month, '/', day, '/', year];
    dn(i)=datenum(datetime); % Convert to numeric date.
    % Read the time series value.
    values(i)=str2double(Recs(i).value);
end
```

13. In the Matlab editor for `MODISPlot_xml`, enter the following lines of code. This code sets up the axis for plotting, and then plots the data using the Matlab plot function.

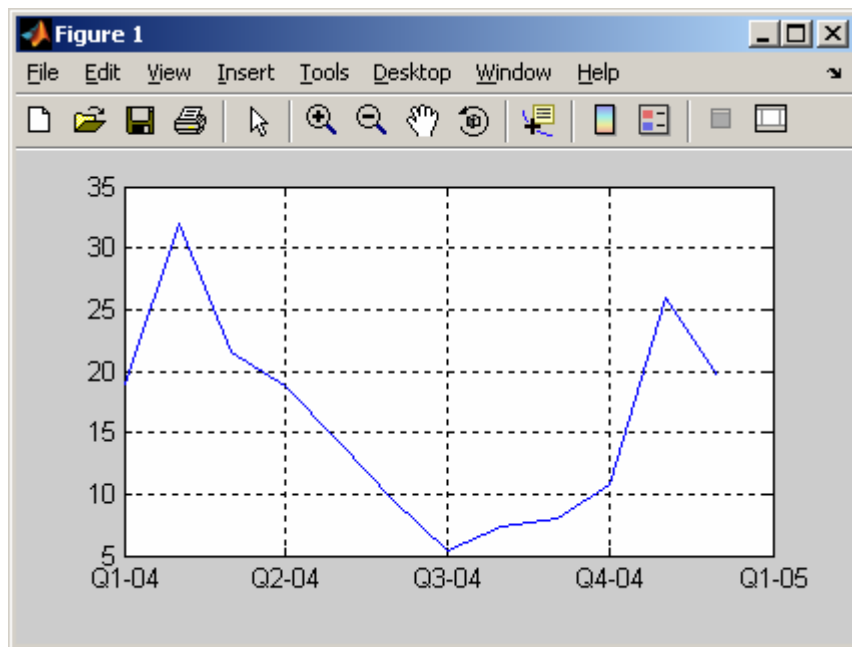
```
% Plot the graph.
plot(dn,values);datetick;
grid on % Turn on grid lines for this plot.
```

14. In the Matlab editor, click the File menu, and then click Save.

15. In the Matlab Command Window, enter the command

```
MODISPlot_xml
```

After a moment, you'll see the variable information appear in the Command Window, and then a graph will appear.



You can edit the plot, put legend and axes names by going to the Edit menu in Figure 1 plot shown above.

In this exercise, you have learned how to call web services from within Matlab and plot MODIS data. This concludes the exercise.

7.0 Ingesting NWIS Data using VB.Net

by Thiha Min and Tim Whiteaker

7.1 Introduction

Using web services is a breeze with Visual Studio. This chapter demonstrates how to call a web service with Visual Studio .Net 2005 and the Visual Basic .Net programming language.

In this exercise, you will create a VB.Net Windows application that uses the NWIS Unit Values web service to compute average streamflow over the past few days at the Colorado River at Austin, TX. The NWIS Unit Values web service returns real-time data for roughly the past 31 days. These data typically are recorded at 15-minute intervals.

7.2 Computer and Skill Requirements

To complete this exercise, your computer must meet the following requirements:

- Working Internet connection
- Visual Studio .Net 2005 software

This exercise assumes that you have some familiarity with the following software environments:

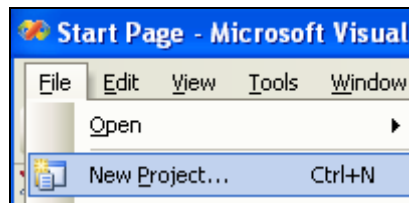
- Visual Studio .Net 2005

7.3 Accessing NWIS Data with a VB.Net Windows Application

In this exercise, you will create a windows application with one main window that allows the user to click to see what the average streamflow over the past few days is at the Colorado River at Austin, TX. The application lets the user specify the number of days for which data should be retrieved (up to 30 days back). The application then asks the NWIS Unit Values web service for streamflow values, and then computes the average of the returned values.

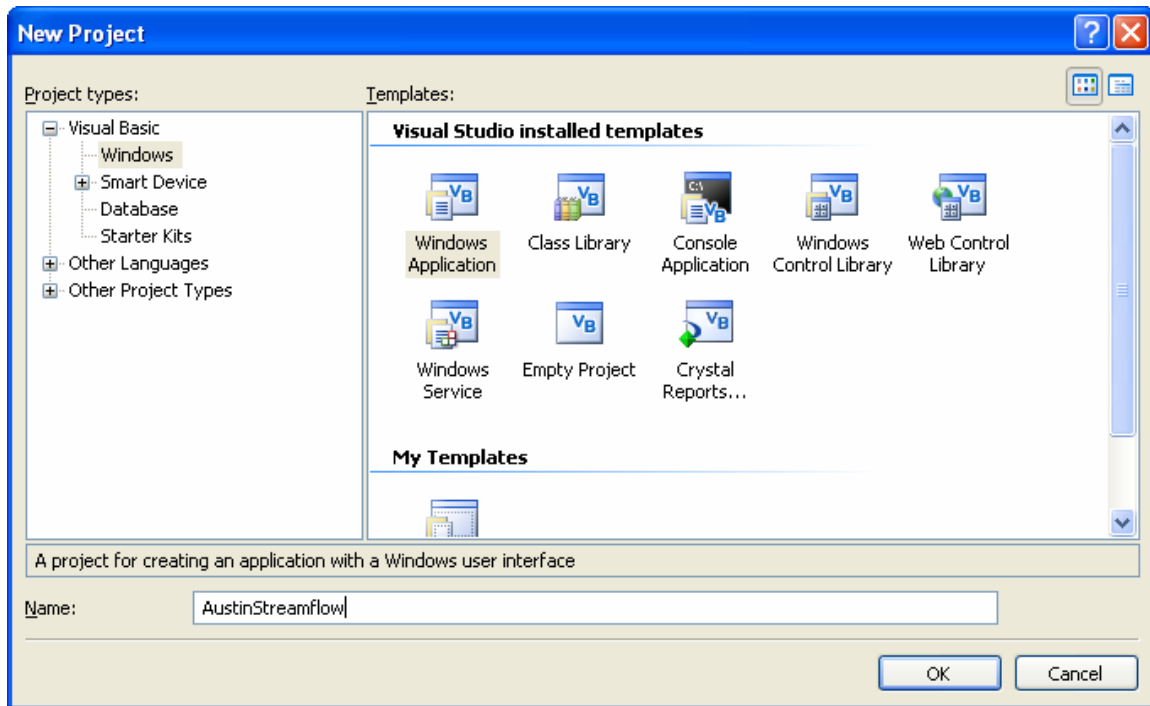
7.3.1 *Setting up the Project*

1. Start Visual Studio 2005 (Click on Start --- All Programs --- Microsoft Visual Studio 2005 --- Microsoft Visual Studio 2005).
2. Click the File menu, then click New Project....



3. In the New Project window, set the following properties:
 - a. Choose Visual Basic --- Windows from Project Types.
 - b. Select Windows Application from Templates.
 - c. Type "AustinStreamflow" as the Name.

- d. Click OK.

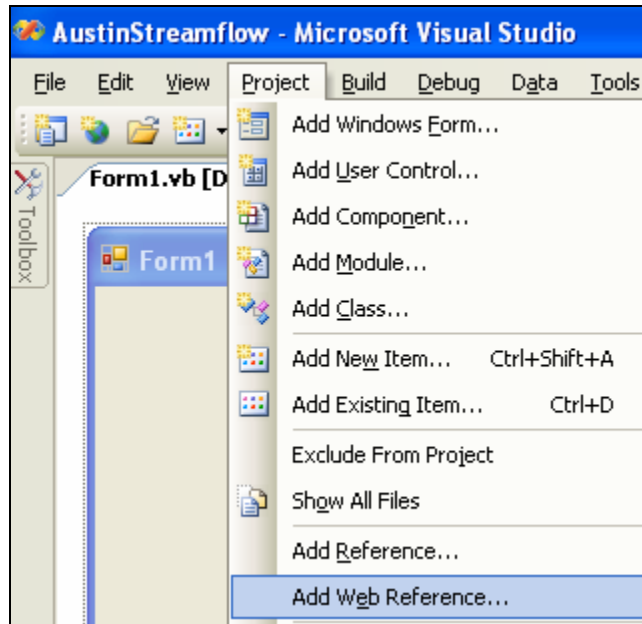


A new project will open with a default form called Form1.

7.3.2 *Creating the Web Reference*

This project will make use of the NWIS Unit Values web service to retrieve streamflow values from the USGS stream gage on the Colorado River at Austin. The web service becomes available to the project after making a web reference to the service.

1. Click the Project menu, then click Add Web Reference...

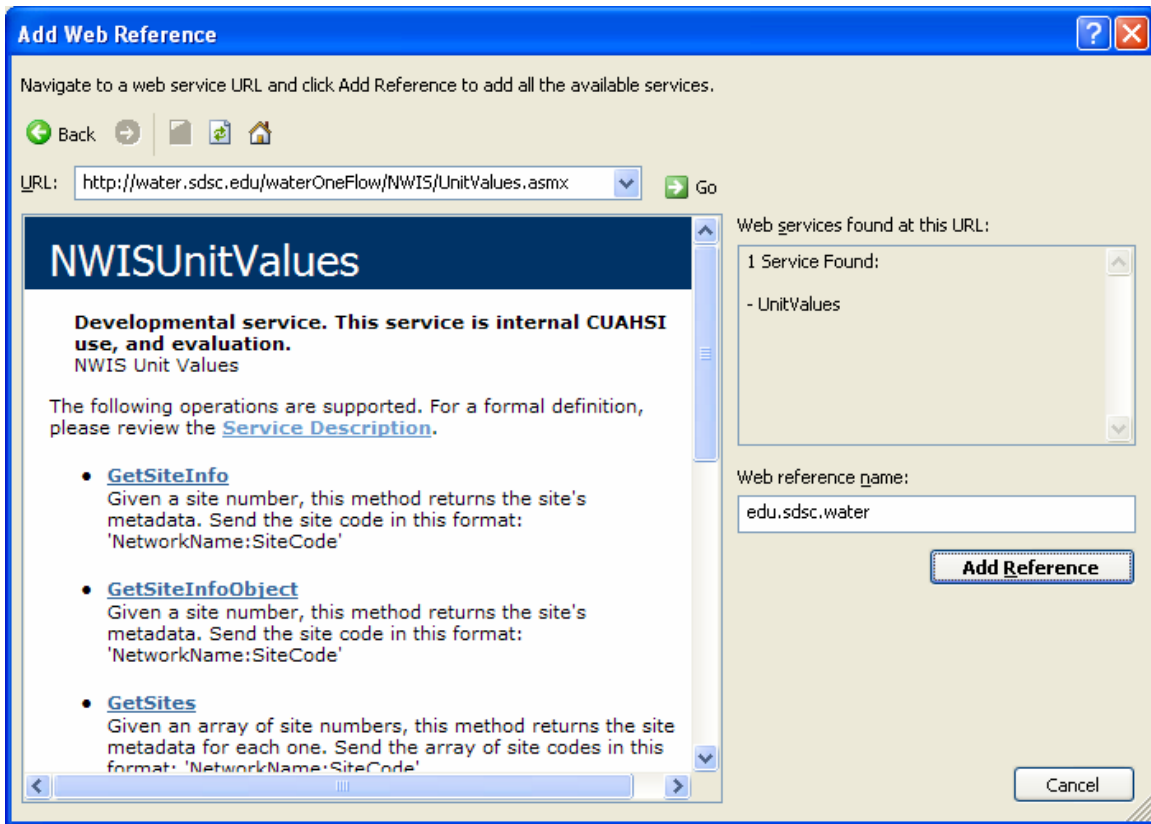


2. In the Add Web Reference window (next to URL:), type in the following URL:

`http://water.sdsc.edu/waterOneFlow/NWIS/UnitValues.asmx`

URL:	<input type="text" value="http://water.sdsc.edu/waterOneFlow/NWIS/UnitValues.asmx"/>	<input type="button" value="Go"/>
------	--	-----------------------------------

3. Click Go. Visual Studio will navigate to the URL and verify that a web service is present.



4. Change the Web reference name (from default edu.sdsc.water) to NWISUnitValues. This is the name by which you will reference the NWIS web service in your code.



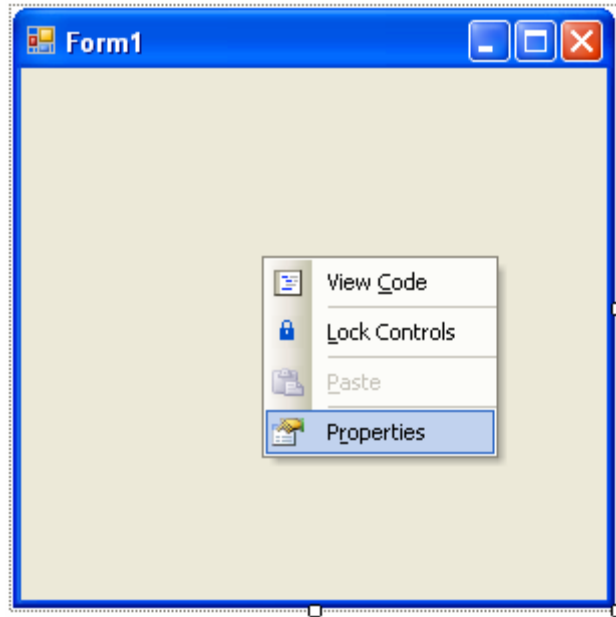
5. Click Add Reference.

The NWIS web service is now available for use within your project.

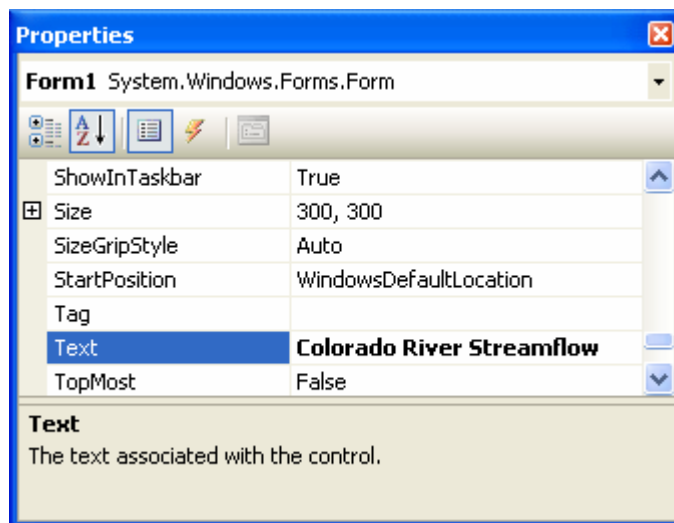
7.3.3 Building the User Interface

Now that you've set up the project, you'll build the user interface by adding controls to the form. Later, you'll add the code behind those controls which will perform the work.

1. Right click on Form1 and click Properties.



2. Change the Text property of the form to "Colorado River Streamflow". This changes the name that appears in the title bar of the form.



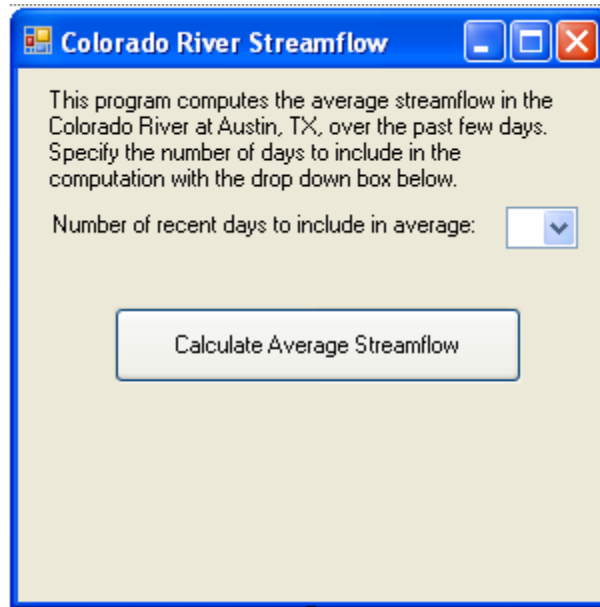
3. Add two labels, one combo box, and one button to the form, at roughly the same positions as shown in the figure below.



4. In a manner similar to setting the Text property of the form, set the properties of the controls as shown below.

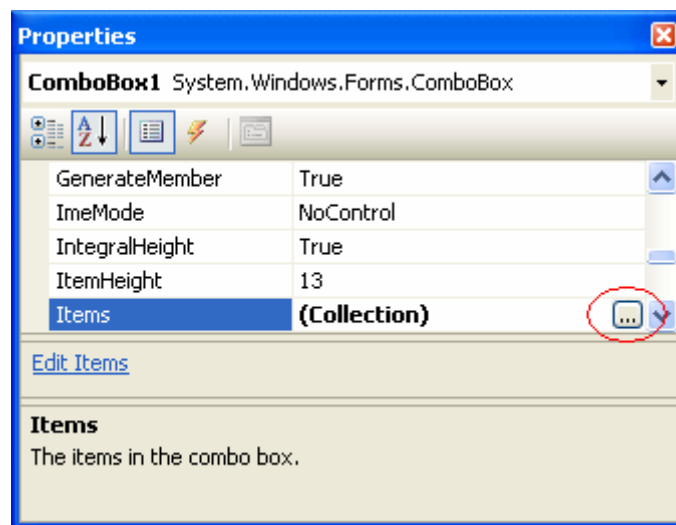
Control	Property	Value
Label1	Text	This program computes the average streamflow in the Colorado River at Austin, TX, over the past few days. Specify the number of days to include in the computation with the drop down box below.
	AutoSize	False
Label2	Text	Number of recent days to include in average:
ComboBox1	DropDownStyle	DropDownList
Button1	(Name)	btnCalculate
	Text	Calculate Average Streamflow

The form should now look similar to the one below.

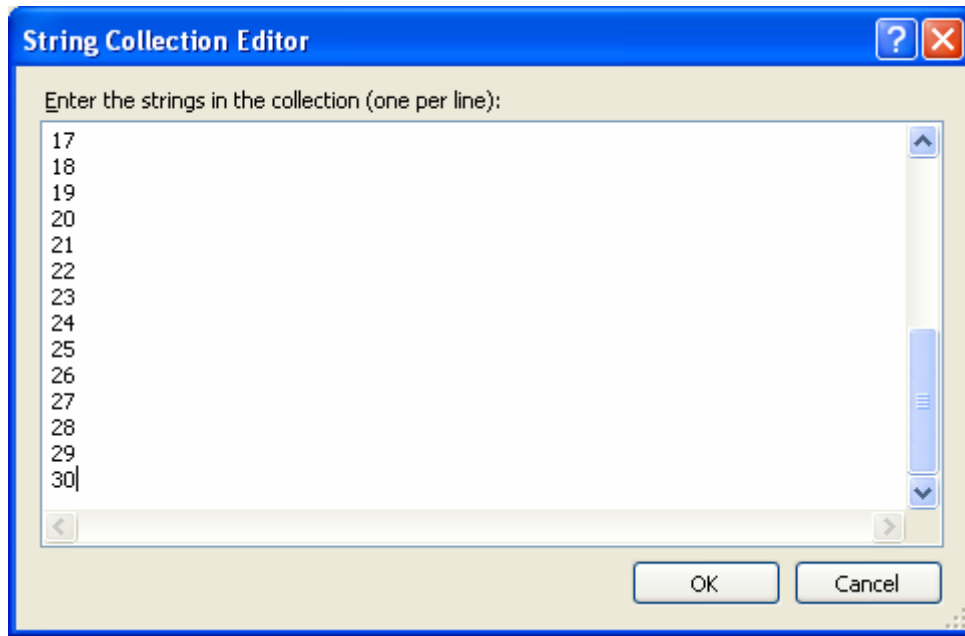


Now you will add the choice of 1 to 30 days to the combo box.

5. Click the properties for ComboBox1, and then select the Items property. Click the ellipsis next to (Collection).



6. Add the numbers 1 through 30 to the String Collection Editor window. This allows the user to select between 1 and 30 days to include in the computation of average streamflow.



7. Click OK to close the String Collection Editor window.

The design of the form is now complete. Next you will add code to make the form perform useful work.

7.3.4 Writing the Code

First you must set the default value of the drop down box. Let's use 30 as the default.

1. Double click the form (be sure and not to click on any of the controls that you have added to the form.) This opens the code editor, and creates stub code that will be run when the form opens.

```
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object,
    End Sub
End Class
```

2. Add the following code to the Form1_Load procedure.

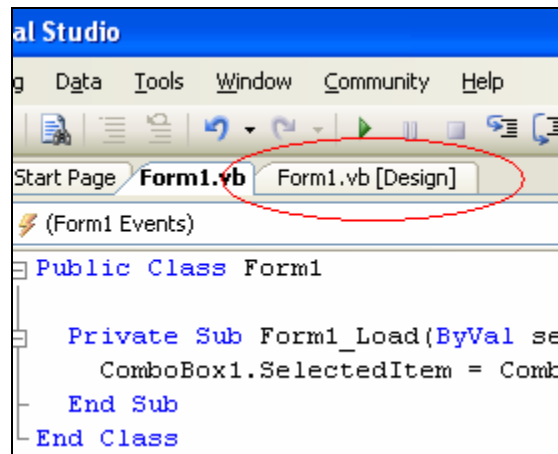
```
ComboBox1.SelectedItem = ComboBox1.Items.Item(29)
```

The result is shown in the screenshot below.

```
Private Sub Form1_Load(ByVal sender As System.Object,
    ComboBox1.SelectedItem = ComboBox1.Items.Item(29)
End Sub
```


In the code above, you are setting the selected item in the combo box to be the 30th item (which happens to be the number 30). Indices in VB.Net begin with zero, not one. So the first item in the combo box has an index of zero, while the last item has an index of 29 in this case.

3. At the top of the code editor, click the Form1.vb [Design] tab.



This shows the form and the controls that you have placed on it. This is a convenient view for choosing a specific control to write code for. Now you'll add code to the button to compute average streamflow.

4. Double click the Calculate Average Streamflow button to open the code editor and automatically create stub code for the Click event for that button.
5. Add the following code to the btnCalculate_Click procedure.

```

.....
' Set initial parameters.
.....

' Set the siteCode for our gage of interest.
Dim location As String = "NWIS:08158000"
' Set the variableCode for streamflow.
Dim variable As String = "NWIS:00060"
' Set start and end date.
Dim startDate, endDate As String
Dim tmpDate As Date
endDate = Format(Now, "yyyy-MM-dd")
tmpDate = Now.AddDays(-1 * ComboBox1.SelectedItem + 1)
startDate = Format(tmpDate, "yyyy-MM-dd")

.....
' Call the web service.
.....

Dim ws As New NWISUnitValues.NWISUnitValues
Dim tsResponse As NWISUnitValues.TimeSeriesResponseType
tsResponse = ws.GetValuesObject(location, variable, _
                                startDate, endDate, "")

```

```

' Process the results.
'
' Process the results.
'

Dim vals As NWISUnitValues.TsValuesSingleVariableType
vals = tsResponse.timeSeries.values
If vals.count = 0 Then
    MsgBox("No values returned")
    Exit Sub
End If

Dim avg As Double = 0

For i As Integer = 0 To vals.count - 1
    avg += vals.value(i).Value
Next

avg = avg / vals.count
MsgBox("The average streamflow is " & _
    FormatNumber(avg, 1) & " cfs")

```

In the code above, you are first preparing the inputs to feed the web service. The tricky part of this is formatting the dates to “yyyy-MM-dd” format (e.g., 2006-12-31), which is what the web service is expecting. Another trick is calculating the start date by adding “negative” days to the current date in the line:

```
tmpDate = Now.AddDays(-1 * ComboBox1.SelectedItem + 1)
```

Next you are creating a new instance of the NWIS Unit Values web service, and calling the `GetValuesObject` method from the service with the date inputs from the user. This method returns an Object with the data retrieved from the web service.

Next, with the results from the `GetValuesObject` call, you are computing the average streamflow from the values returned, and then showing a message box to report the result.

7.3.5 *Running the Code*

The project is now ready to run.

1. Press F5 on your keyboard to run it.
2. Click the Calculate Average Streamflow button.

After a minute or two, a message box appears showing the average streamflow over the past 30 days. Note that your value may be different than the value in the screenshot below, since this exercise was created on another day than the current day.



3. **Close** the form when you are finished.

You have the exercise and have learned how to call a web service from Visual Studio .Net 2005. From this point, you could build the solution as an executable file by pressing Ctrl-Shift-B on your keyboard. See your Visual Studio help for more information about building solutions.

8.0 Ingesting NWIS Data Using Java

by David Valentine

In this chapter you will build a Java class that accesses the NWIS Daily Values web service to obtain daily streamflow values for Big Rock Creek near Valyermo, California, for the year 2001. The class will output the site code and site name for this location, as read from the web service, as well as the time series of streamflow values.

8.1 Computer and Skill Requirements

To complete this exercise, your computer must meet the following requirements:

- Working Internet connection
- Java SE 5: download from <http://java.sun.com/>
- NetBeans IDE 5.5: download from <http://www.netbeans.org>

This exercise assumes that you have some familiarity with general programming concepts and Java.

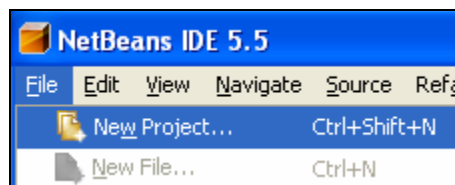
NOTE: The source code for the `nwis.java` class created in this exercise is located in Appendix C.

8.2 Procedure

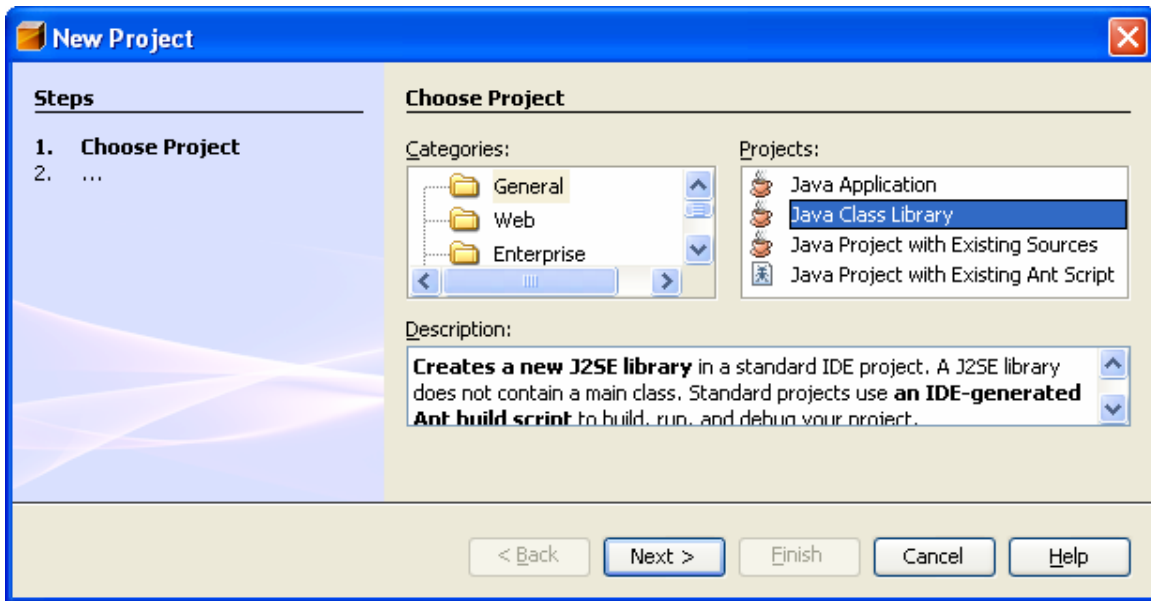
8.2.1 *Creating a New Project*

First, you will create a new Java project.

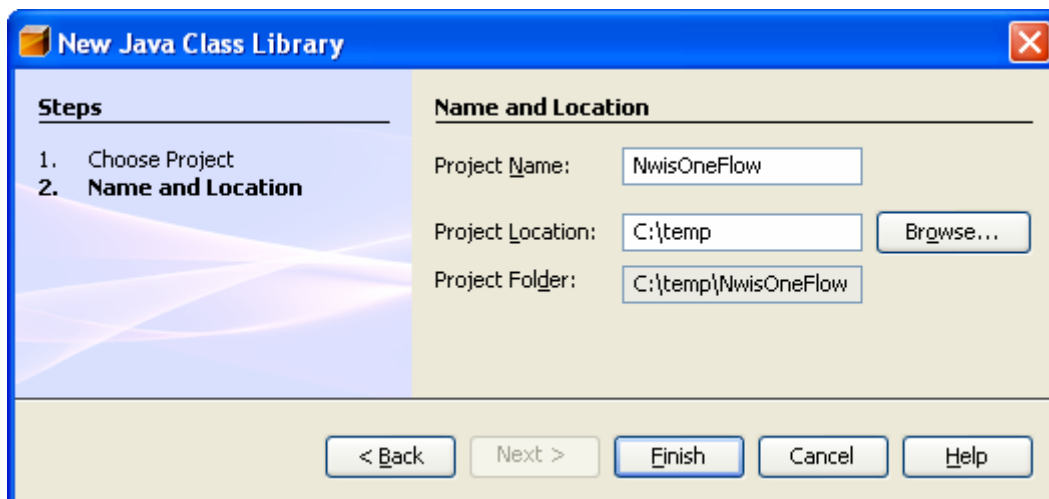
1. Start NetBeans IDE.
2. Click the File menu, and then click New Project...



3. In the New Project dialog, select General in the Categories pane. In the Projects pane, select Java Class Library.



4. Click Next.
5. In the New Java Class Library dialog, enter “NwisOneFlow” as the Project Name.
6. Enter the path in which you want the project folder to be created in Project Location. The IDE will create a subfolder at that location called “NwisOneFlow”, which will contain all files used in the project.

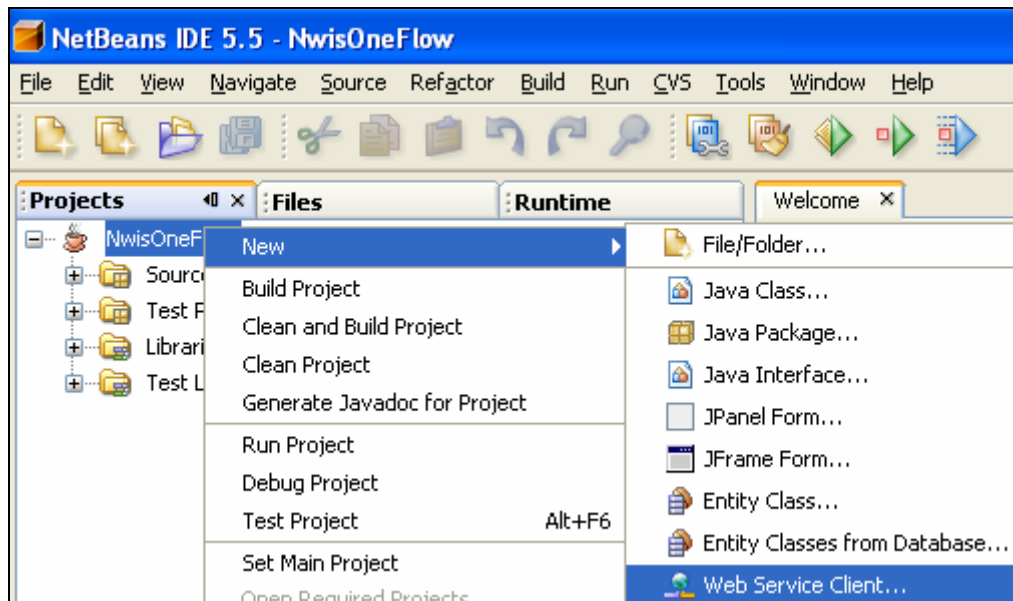


7. Click Finish.

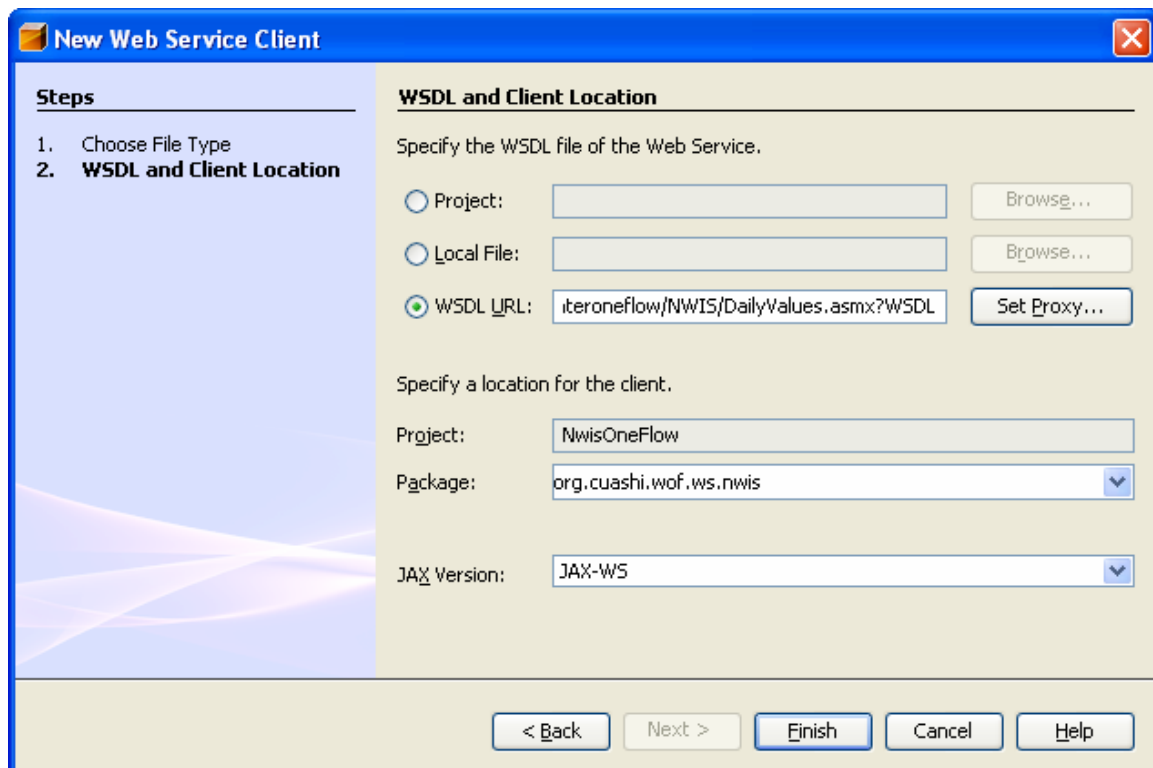
8.2.2 Creating a Web Service Client

With the project set up, you will now create a client for the NWIS web service.

1. In the Projects window, right click on NwisOneFlow, point to New, and then click Web Service Client...



2. In the New Web Service Client dialog, enter
 “http://water.sdsc.edu/wateroneflow/NWIS/DailyValues.asmx?WSDL” as the WSDL URL.
3. For the Package, enter “org.cuashi.wof.ws.nwis”
4. For the JAX Version, select JAX-WS.

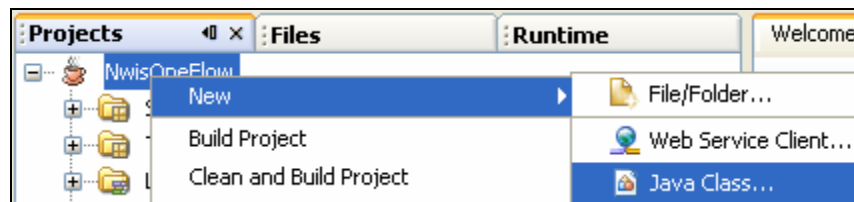


5. Click Finish to compile the web service client class.

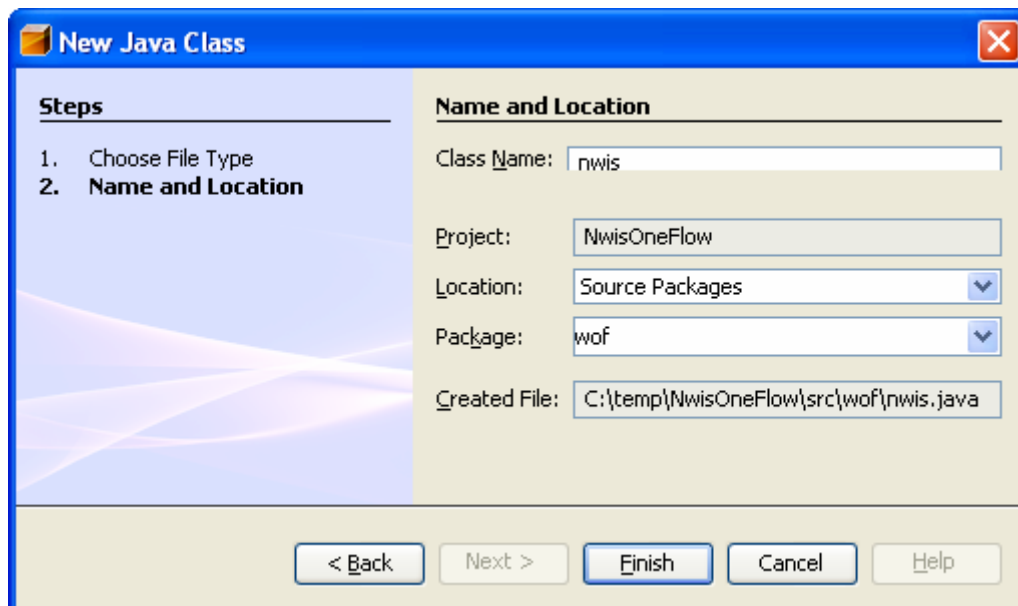
8.2.3 Creating a Class to Consume the Web Service

In this section you will create a new java class, `wof.nwis`, that accesses the web service client you just created. The class will download a time series of daily streamflow values at Big Rock Creek near Valyermo, California, for the year 2001. The site code for this location is 10263500, and the variable code for streamflow is 00060. You will hard code both of these values into the class. You will also hard code the time span (the year 2001). In a more robust application, you would let the user supply these parameters. The class will output the name of the site, its site code, and the time series of values.

1. In the Projects window, right click on the `NwisOneFlow` project, point to New, and then click Java Class...



2. In the New Java Class dialog, enter “`nwis`” as the Class Name, and “`wof`” as the *Package*.



3. Click Finish.

Now you will add a “main” procedure, which is the default procedure that will run when the class is invoked. It is within this procedure that you will eventually add the code to retrieve the time series values.

4. At the end of the source code for the `nwis` class, enter the following code.

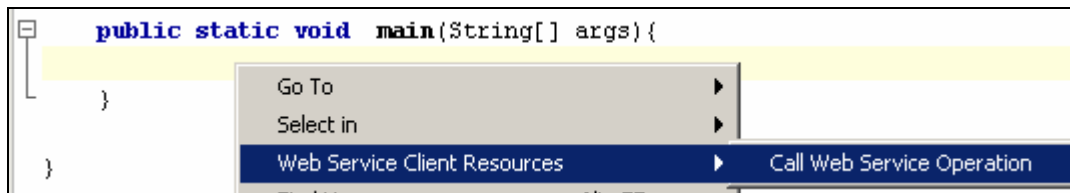
```
public static void main(String[] args){  
}
```

The screenshot below shows the result.

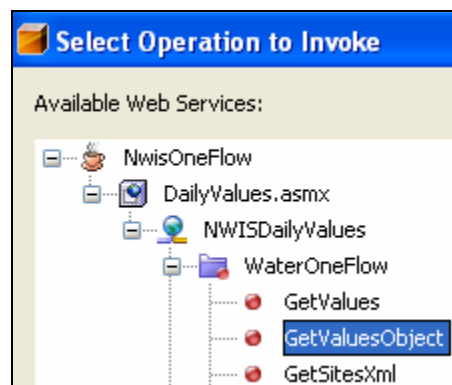
```
package wof;  
  
/**  
 *  
 * @author whiteaker  
 */  
public class nwis {  
  
    /** Creates a new instance of nwis */  
    public nwis() {  
    }  
  
    public static void main(String[] args){  
    }  
}
```

Now you will create the code for calling the web service. Fortunately, the IDE can create most of the code for you.

5. Right click within the code for the main method, point to Web Service Client Resources, and then click Call Web Service Operation.



6. In the Select Operations to Invoke dialog, select GetValuesObject, and click OK.



After you click OK, the IDE generates code for calling the `GetValuesObject` method from the NWIS web service. The IDE creates variables (e.g., location) for storing the parameters that will be sent to the web service, but leaves them empty for you to fill in later. A screenshot from the code editor is shown below.

```
public static void main(String[] args){
    try { // Call Web Service Operation
        org.cuashi.wof.ws.nwis.NWISDailyValues service =
            new org.cuashi.wof.ws.nwis.NWISDailyValues();
        org.cuashi.wof.ws.nwis.WaterOneFlow port =
            service.getWaterOneFlow();
        // TODO initialize WS operation arguments here
        java.lang.String location = "";
        java.lang.String variable = "";
        java.lang.String startDate = "";
        java.lang.String endDate = "";
        java.lang.String authToken = "";
        // TODO process result here
        org.cuashi.wof.ws.nwis.TimeSeriesResponseType result =
            port.getValuesObject(location, variable, startDate, endDate, authToken);
        System.out.println("Result = "+result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
}
```

When executed, the above code creates a web service, gets an instance, and then calls `GetValuesObject`. Now you will hard code the parameters for our site of interest. Remember, you are hard coding these parameters for this simple example application, but a more robust application would read these parameters as inputs from the user.

7. Fill in the parameters for calling the web method.

```
java.lang.String location = "NWIS:10263500";
java.lang.String variable = "NIWS:00060";
java.lang.String startDate = "2001-01-01";
java.lang.String endDate = "2001-12-31";
java.lang.String authToken = "";
```

A screenshot from the code editor is shown below.

```

public static void main(String[] args){
    String siteCode = null;
    String siteName = null;

    try { // Call Web Service Operation
        org.cuashi.wof.ws.nwis.NWISDailyValues service =
            new org.cuashi.wof.ws.nwis.NWISDailyValues();
        org.cuashi.wof.ws.nwis.WaterOneFlow port =
            service.getWaterOneFlow();
        // TODO initialize WS operation arguments here
        java.lang.String location = "NWIS:10263500";
        java.lang.String variable = "NIWS:00060";
        java.lang.String startDate = "2001-01-01";
        java.lang.String endDate = "2001-12-31";
        java.lang.String authToken = "";
        // TODO process result here
        org.cuashi.wof.ws.nwis.TimeSeriesResponseType result =
            port.getValuesObject(location, variable, startDate, endDate, authToken);
        System.out.println("Result = "+result);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
}

```

Now you will create variables to store the site code and name as read from the web service.

8. In the main procedure, above the try statement, add the following lines of code.

```

String siteCode = null;
String siteName = null;

```

A screenshot of the code editor is shown below.

```

public static void main(String[] args){
    String siteCode = null;
    String siteName = null;

    try { // Call Web Service Operation

```

To output the datetimes and values in the time series, you will use a List object.

9. Below the package declaration, add an import statement for the List library.

```

import java.util.List;

```

A screenshot from the code editor is shown below.

```
package wof;

import java.util.List;

/**
```

You will now tell the IDE to output the site code and site name to the Output window of the IDE.

- At the end of the try statement, replace the line that begins with “System.out.println” with the following lines of code, in order to output the site information:

```
org.cuashi.wof.ws.nwis.SiteInfoType sit =
    (org.cuashi.wof.ws.nwis.SiteInfoType)
        result.getTimeSeries().getSourceInfo();
siteCode = sit.getSiteCode().get(0).getValue();
siteName = sit.getSiteName();

System.out.println("siteCode = "+siteCode);
System.out.println("siteName = "+siteName);
```

A screenshot from the code editor is shown below.

```
// TODO process result here
org.cuashi.wof.ws.nwis.TimeSeriesResponseType result =
    port.getValuesObject(location, variable, startDate, endDate, authToken);

org.cuashi.wof.ws.nwis.SiteInfoType sit =
    (org.cuashi.wof.ws.nwis.SiteInfoType) result.getTimeSeries().getSourceInfo();
siteCode = sit.getSiteCode().get(0).getValue();
siteName = sit.getSiteName();

System.out.println("siteCode = "+siteCode);
System.out.println("siteName = "+siteName);

} catch (Exception ex) {
    // TODO handle custom exceptions here
}
```

Some notes on the above code logic: You are working with objects, so you need to do some type casting in order to get the correct object. The line below takes a sourceInfo type and casts it to a siteInfo type.

```
org.cuashi.wof.ws.nwis.SiteInfoType sit =
    (org.cuashi.wof.ws.nwis.SiteInfoType)
        result.getTimeSeries().getSourceInfo();
```

At present, there are two possible sourceInfo types: siteInfoType, and dataSetInfoType. If we were writing a more complete generic parser, we would use getClass().getName(), and cast based on the object type.

Finally, you will add code to output the time series values.

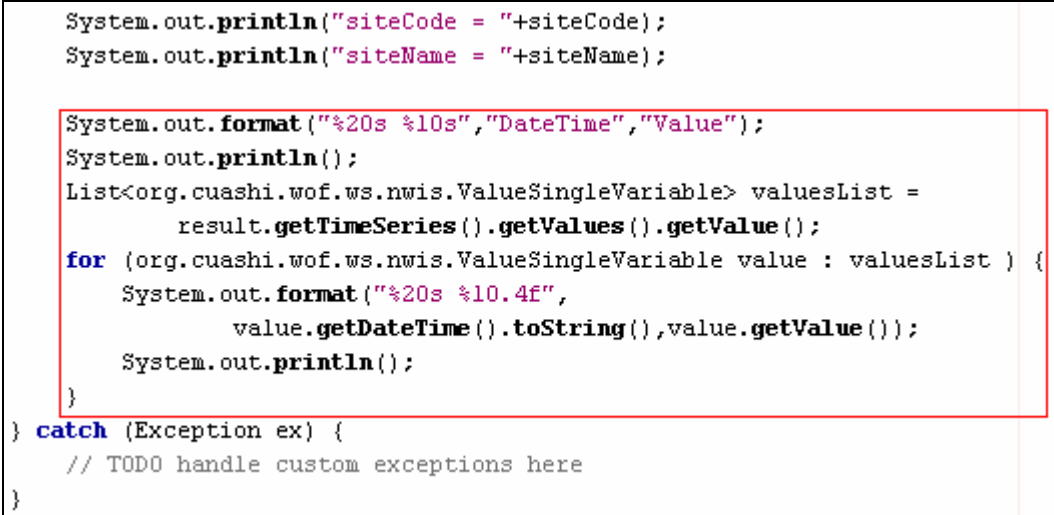
- Add the flowing code after the last “System.out.println” line that you just added.

```

System.out.format("%20s %10s", "DateTime", "Value");
System.out.println();
List<org.cuashi.wof.ws.nwis.ValueSingleVariable> valuesList =
    result.getTimeSeries().getValues().getValue();
for (org.cuashi.wof.ws.nwis.ValueSingleVariable value : valuesList ) {
    System.out.format("%20s %10.4f",
        value.getDateTime().toString(), value.getValue());
    System.out.println();
}

```

A screenshot from the code editor is shown below.



```

System.out.println("siteCode = "+siteCode);
System.out.println("siteName = "+siteName);

System.out.format("%20s %10s", "DateTime", "Value");
System.out.println();
List<org.cuashi.wof.ws.nwis.ValueSingleVariable> valuesList =
    result.getTimeSeries().getValues().getValue();
for (org.cuashi.wof.ws.nwis.ValueSingleVariable value : valuesList ) {
    System.out.format("%20s %10.4f",
        value.getDateTime().toString(), value.getValue());
    System.out.println();
}
} catch (Exception ex) {
    // TODO handle custom exceptions here
}

```

In the above code, we use a List and a for loop, which are features of java 1.5 and above. We loop through the set of values, and output formatted strings.

When finished, the code for the main method should look as follows. Note that text for long lines is wrapped.

```

public static void main(String[] args){
    String siteCode = null;
    String siteName = null;

    try { // Call Web Service Operation
        org.cuashi.wof.ws.nwis.NWISDailyValues service =
            new org.cuashi.wof.ws.nwis.NWISDailyValues();
        org.cuashi.wof.ws.nwis.WaterOneFlow port =
            service.getWaterOneFlow();
        // TODO initialize WS operation arguments here
        java.lang.String location = "NWIS:10263500";
        java.lang.String variable = "NIWS:00060";
        java.lang.String startDate = "2001-01-01";
        java.lang.String endDate = "2001-12-31";
        java.lang.String authToken = "";
        // TODO process result here
        org.cuashi.wof.ws.nwis.TimeSeriesResponseType result =

```

```

        port.getValuesObject(location, variable, startDate,
endDate, authToken);

        org.cuashi.wof.ws.nwis.SiteInfoType sit =
            (org.cuashi.wof.ws.nwis.SiteInfoType)
result.getTimeSeries().getSourceInfo();
        siteCode = sit.getSiteCode().get(0).getValue();
        siteName = sit.getSiteName();

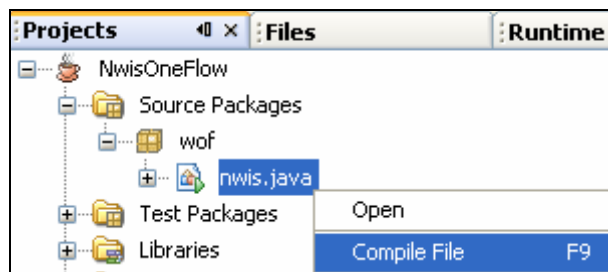
        System.out.println("siteCode = "+siteCode);
        System.out.println("siteName = "+siteName);

        System.out.format("%20s %10s", "DateTime", "Value");
        System.out.println();
        List<org.cuashi.wof.ws.nwis.ValueSingleVariable> valuesList =
            result.getTimeSeries().getValues().getValue();
        for (org.cuashi.wof.ws.nwis.ValueSingleVariable value :
valuesList ) {
            System.out.format("%20s %10.4f",
                value.getDateTime().toString(),value.getValue());
            System.out.println();
        }
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
}

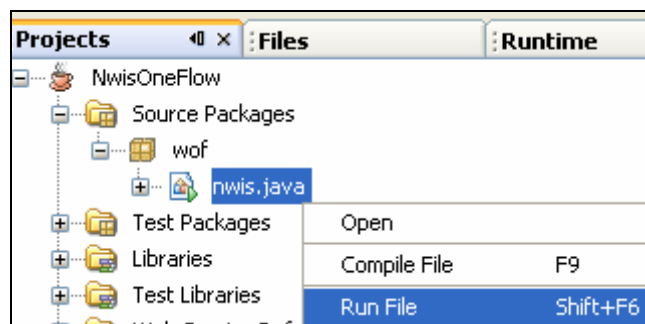
```

With the code finished, all that is left is to compile and run the file.

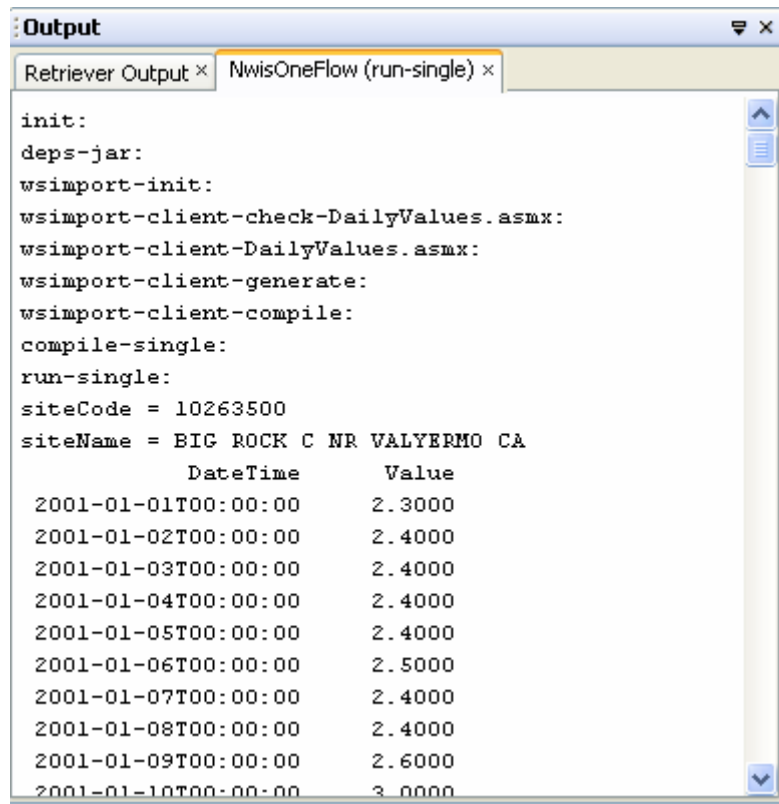
12. In the Projects window, right click on nwis.java and then click Compile File.



13. In the Projects window, right click on nwis.java and then click Run File.



In a moment, you will see the results of the `GetValuesObject` call as text in the Output window.



```
init:
deps-jar:
wsimport-init:
wsimport-client-check-DailyValues.asmx:
wsimport-client-DailyValues.asmx:
wsimport-client-generate:
wsimport-client-compile:
compile-single:
run-single:
siteCode = 10263500
siteName = BIG ROCK C NR VALYERMO CA
      DateTime      Value
2001-01-01T00:00:00    2.3000
2001-01-02T00:00:00    2.4000
2001-01-03T00:00:00    2.4000
2001-01-04T00:00:00    2.4000
2001-01-05T00:00:00    2.4000
2001-01-06T00:00:00    2.5000
2001-01-07T00:00:00    2.4000
2001-01-08T00:00:00    2.4000
2001-01-09T00:00:00    2.6000
2001-01-10T00:00:00    3.0000
```

Congratulations! You have created a Java class which calls the NWIS web service to retrieve time series data. This concludes the exercise.

Appendix A: Source Code for parse_xml.m

```
% Function extracted from xmltools.m so that it could be called directly
% David Tarboton 3/19/06
% xmltools.m originally Matlab File Exchange
%
http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=3074
%
function [z, str] = parse_xml( str, current_tag, current_value, attribs, idx)

next = 'child';

if nargin < 2
    current_tag    = '';
    current_value  = '';
    attribs        = '';
    idx            = 0;
end
z = [];

eot = 0;

while ~eot & ~isempty(udeblank(deblank(str)))

    f_end = strfind(str, '</');
    f_beg = strfind(str, '<');

    %< Si je n'ai plus de tag dans mon document
    if isempty(f_end) & isempty(f_beg)

        if ~strcmp(lower(current_tag), '?xml') & ~isempty(current_tag)
            error('xmltools:parse_xml', 'malformed xml string (current [%s])',
current_tag);
        else
            fprintf('end parsing at level %d\n',idx);
            eot = 1;
            return
        end
    end
    %>

    if isempty(f_end)
        f_end = length(str)
    else
        f_end = f_end(1);
    end
    if isempty(f_beg)
        f_beg = length(str)
    else
        f_beg = f_beg(1);
    end

    if f_end <= f_beg
        %< je rencontre une fermeture
```

```

new_tag = str((f_end+2):end);
str_t   = str(1:f_end-1);
f_end = strfind(new_tag, '>');
if isempty(f_end)
    error('xmltools:parse_xml', 'malformed xml string : never ending tag
[%s] encountered', current_tag);
end
f_end = f_end(1);
str    = new_tag(f_end+1:end); % reste
new_tag = new_tag(1:f_end-1);
if ~strcmp(upper(new_tag), upper(current_tag))
    error('xmltools:parse_xml', 'malformed xml string : [%s] not properly
closed (closing [%s] encountered)', current_tag, new_tag);
end
% fprintf('%sclose [%s]\n', repmat(' ', 2*(idx-1),1), current_tag);
z.tag      = upper(current_tag);
z.attrs    = parse_attrs(attrs);
z.value    = udeblank(deblank(sprintf('%s %s', current_value, str_t)));
eot        = 1;
%>
else
    %< je rencontre une ouverture
    % je vais appeler le même code sur ce qu'il y a après moi
    current_value = sprintf('%s %s', current_value, str(1:f_beg-1));
    new_tag      = str(f_beg+1:end);
    f_end = strfind(new_tag, '>');
    if isempty(f_end)
        error('xmltools:parse_xml', 'malformed xml string : never ending tag
encountered');
    end
    f_end = f_end(1);
    str_t = new_tag(f_end+1:end);
    new_tag = new_tag(1:f_end-1);
    if (new_tag(end) == '/')|(new_tag(end) == '?')
        %< Self closing tag
        % Je met (temporairement!) eot à 1, cela me permet de passer quelques
lignes
        % de code tranquillement
        eot = 1;
        %>
    end
    %< Attributs
    f_beg = strfind(new_tag, ' ');
    if isempty(f_beg)
        new_attrs = '';
        if eot
            new_tag = new_tag(1:end-1);
        end
    else
        new_attrs = new_tag(f_beg+1:end);
        if eot
            new_attrs = new_attrs(1:end-1);
        end
        new_tag      = new_tag(1:f_beg-1);
    end
    %>
    % fprintf('%sopen [%s]\n', repmat(' ', 2*idx,1), new_tag);

```



```

if eot
    %< If self-closing tag
    %   fprintf('%s<close [%s]\n', repmat(' ', 2*idx,1), new_tag);
    new_attribs = parse_attribs( new_attribs);
    if isfield(z, next)
        nxt = getfield(z, next);
        nxt(end+1) = struct( 'tag', new_tag, 'attribs', new_attribs, 'value',
'', next, []);
        z = setfield(z, next, nxt);
        %z.(next)(end+1) = struct( 'tag', new_tag, 'attribs', new_attribs,
'value', '', next, []);
    else
        z = setfield(z, next, struct( 'tag', new_tag, 'attribs', new_attribs,
'value', '', next, [] ));
        %z.(next) = struct( 'tag', new_tag, 'attribs', new_attribs, 'value', '',
next, []);
    end
    str = str_t;
    eot = 0;
    %>
else
    %< Appel du même code sur la suite

    % et stockage du resultat dans mes children.
    % Le code met aussi à jour le string courant |str|,
    % il en enlève la partie correspondant au string que je viens de
trouver.
    [t,str] = parse_xml(str_t, new_tag, '', new_attribs, 1+idx);
    if isfield(t, next)
        nx = getfield( t, next);
        %nx = t.(next);
    else
        nx = [];
    end
    if isfield(z, next)
        nxt = getfield(z, next);
        nxt(end+1) = struct( 'tag', t.tag, 'attribs', t.attribs, 'value',
t.value, next, nx);
        z = setfield(z, next, nxt);
        %z.(next)(end+1) = struct( 'tag', t.tag, 'attribs', t.attribs, 'value',
t.value, next, nx);
    else
        z = setfield(z, next, struct( 'tag', t.tag, 'attribs', t.attribs, 'value',
t.value, next, nx));
        %z.(next) = struct( 'tag', t.tag, 'attribs', t.attribs, 'value', t.value,
next, nx);
    end

    %>
end
end
%>
end
%>

```

```

%< Parse attribs
function z = parse_attribs( a)
if isempty(a)
    z = struct( 'name', '', 'value', '');
    return
end
b = tokens(a, ' ');
j = 1;
for i=1:length(b)
    if ~isempty(b{i})
        t = tokens(b{i}, '=');
        if length(t)==2
            u = t{2};
            if u(1)==' '
                u = u(2:end);
            end
            if u(end)==' '
                u = u(1:end-1);
            end
            z(j) = struct( 'name', upper(t{1}), 'value', u);
        else
            z(j) = struct( 'name', upper(a), 'value', '');
        end
        j = j +1;
    end
end
%>

%<* Ecriture d'une structure xml
function z = write_xml(fid, xml_struct, idx)

next = 'child';

if nargin < 3
    idx = 0;
end

margin = repmat(' ', 2*idx, 1);

closed_tag = 1;
%< Ouverture du tag
if isfield(xml_struct, 'tag')
    closed_tag = 0;
    fprintf(fid, '%s<%s', margin, xml_struct.tag);
    %< Ecriture des attributs
    if ~isfield(xml_struct, 'attribs')
        error('xmltools:write_xml', 'malformed MATLAB xml structure : tag without attribs');
    end
    for i=1:length(xml_struct.attribs)
        if ~isempty(xml_struct.attribs(i).name)
            fprintf(fid, ' %s="%s"', xml_struct.attribs(i).name,
xml_struct.attribs(i).value);
        end
    end
%>

```

```

%< Gestion des Auto closed tags
% Si le tag n'est pas auto fermé, alors |closed_tag| est à zéro
if ~isfield(xml_struct, next)
    error('xmltools:write_xml', 'malformed MATLAB xml structure : tag without
%s', next);
end
if ~isfield(xml_struct, 'value')
    error('xmltools:write_xml', 'malformed MATLAB xml structure : tag without
value');
end
if xml_struct.tag(1) == '?'
    fprintf(fid, '?>\n');
    closed_tag = 1;
elseif isempty(getfield(xml_struct, next)) & isempty(xml_struct.value)
%elseif isempty(xml_struct.(next)) & isempty(xml_struct.value)
    fprintf(fid, '/>\n');
    closed_tag = 1;
else
    fprintf(fid, '>\n');
end
%>
end
%>

%< Ecriture de la value
if isfield(xml_struct, 'value')
    if ~isempty(xml_struct.value)
        fprintf(fid, '%s%s\n', margin, xml_struct.value);
    end
end
%>

%< Ecriture des enfants
if ~isfield(xml_struct, next)
    error('xmltools:write_xml', 'malformed MATLAB xml structure : tag without
%s', next);
end
those_children = getfield(xml_struct, next);
%those_children = xml_struct.(next);
for i=1:length(those_children)
    write_xml(fid, those_children(i), idx+1);
end
%>

%< Fermeture du tag
if ~closed_tag
    fprintf(fid, '%s</%s>\n', margin, xml_struct.tag);
end
%>
%>*

%<* get childs with a specific tag name
function z = get_childs(z, next, tag_name);
u = getfield(z, next);

```

```

zo = [];
for i=1:length(u)
    v = u(i);
    if strcmp(upper(v.tag), upper(tag_name))
        if isempty(zo)
            zo.anext= v;
        else
            zo.anext(end+1) = v;
        end
    end
end
if ~isstruct( zo)
    if isfield(z, 'tag')
        tn = z.tag;
    else
        tn = 'root?';
    end
    error('XMLTOOLS:GET-TEG', 'problem in finding tag <%s> under one <%s>',
tag_name, tn);
end
z = [ zo.anext ];
%>*

%< udeblank
function s = udeblank(str)
s = deblank(str(end:-1:1));
s = s(end:-1:1);
if length(s)==0
    s = '';
end
%>

%< emptystruct
function z = emptystruct(next)
z = struct( 'tag', [], 'value', [], 'attribs', [], next, []);
%>

%< Tokens
function l = tokens(str,del)
l={} ;
% Boucle sur les tokens.
del = sprintf(del) ;
while ~isempty(str)
    [tok,str] = strtok(str,del) ;
    l{end+1} = tok ;
end
%>

```

Appendix B: Source Code for MODISPlot_xml.m

```
% Initialize variables.
clear values
clear dn

% Create class.
wsdl='http://water.sdsc.edu/waterOneFlow/MODIS/Service.asmx?WSDL';
createClassFromWsd1(wsdl);

% This creates an instance of the class.
svsMODIS = MODIS;

% Specify input parameters.
w='-98.2' % West longitude.
s='30' % South latitude.
e='-97.3' % East longitude.
n='30.7' % North latitude.
location=['GEOM:BOX(','w',' ','s',' ','e',' ','n','')']
% Variable Code 11 = Cloud Optical Thickness Water Phase.
variableCode='MODIS:11/plotarea=land'
startDate='2004-01-01'
endDate='2004-12-01'

% Call the GetValues function to get the time series data.
xmlValues=GetValues(svsMODIS,location,variableCode, ...
    startDate,endDate, '')

% Parse the XML string.
structValues=parse_xml(xmlValues);

% Report the name and units of the chosen variable.
display(structValues.child.child(2).child(2).child(2).value)
display(structValues.child.child(2).child(2).child(3).value)

% Get the <value> tags.
Recs=structValues.child.child(2).child(3).child;
[d1,d2]=size(Recs)

% Build arrays of datetimes and values.
for i=1:d2
    % Reformat date to that Matlab can understand it.
    datetime=Recs(i).attribs(1).value;
    year=datetime(1:4);
    month=datetime(6:7);
    day=datetime(9:10);
    datetime=[month, '/', day, '/', year];
    dn(i)=datenum(datetime); % Convert to numeric date.
    % Read the time series value.
    values(i)=str2double(Recs(i).value);
end
```

```
% Plot the graph.  
plot(dn,values);datetick;  
grid on % Turn on grid lines for this plot.
```

Appendix C: Source Code for nwis.java Class

```
/*
 * nwis.java
 *
 * Created on November 10, 2006, 1:15 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package wof;

import java.util.List;

public class nwis {

    /** Creates a new instance of nwis */
    public nwis() {
    }

    public static void main(String[] args){
        String siteCode = null;
        String siteName = null;

        try { // Call Web Service Operation
            org.cuashi.wof.ws.nwis.NWISDailyValues service =
                new org.cuashi.wof.ws.nwis.NWISDailyValues();
            org.cuashi.wof.ws.nwis.WaterOneFlow port =
                service.getWaterOneFlow();
            // TODO initialize WS operation arguments here
            java.lang.String location = "NWIS:10263500";
            java.lang.String variable = "NIWS:00060";
            java.lang.String startDate = "2001-01-01";
            java.lang.String endDate = "2001-12-31";
            java.lang.String authToken = "";
            // TODO process result here
            org.cuashi.wof.ws.nwis.TimeSeriesResponseType result =
                port.getValuesObject(location, variable, startDate,
endDate, authToken);

            org.cuashi.wof.ws.nwis.SiteInfoType sit =
                (org.cuashi.wof.ws.nwis.SiteInfoType)
result.getTimeSeries().getSourceInfo();
            siteCode = sit.getSiteCode().get(0).getValue();
            siteName = sit.getSiteName();

            System.out.println("siteCode = "+siteCode);
            System.out.println("siteName = "+siteName);

            System.out.format("%20s %10s", "DateTime", "Value");
            System.out.println();
            List<org.cuashi.wof.ws.nwis.ValueSingleVariable> valuesList =
                result.getTimeSeries().getValues().getValue();
            for (org.cuashi.wof.ws.nwis.ValueSingleVariable value :
valuesList ) {
```

```
        System.out.format("%20s %10.4f",
            value.getDateTime().toString(),value.getValue());
        System.out.println();
    }
} catch (Exception ex) {
    // TODO handle custom exceptions here
}
}
}
```