o

CUAHSI

universities allied for water research

HydroObjects

Version 1.1

July, 2008

by:

Tim Whiteaker
Center for Research in Water Resources
The University of Texas at Austin

Distribution

The HydroObjects software, source code, and documentation are provided free of charge under the Berkeley Software Distribution (BSD) license. Please see the following license information:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of The University of Texas at Austin nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

Thinktecture Copyright

The HydroObjects software utilizes thinktecture software. The thinktecture software and all other content of the distributed package, if not otherwise stated, is copyright 2003-2004 thinktecture (http://www.thinktecture.com/).

Terms of Use

Permission is hereby granted to use the thinktecture software, for both commercial and non-commercial purposes, free of charge. Permission is hereby granted to copy and distribute the software for non-commercial purposes. A commercial distribution is NOT allowed without prior written permission of the author(s). Further, redistribution and use in source and binary forms, with or without modification, are permitted only provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following warranty disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following warranty disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of thinktecture nor the names of its author(s) may be used to endorse or promote products derived from this software without specific prior written permission.

Funding

Funding for HydroObjects 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.

Table of Contents

Introduction and Software Description

HydroObjects is a .Net DLL with COM classes that support hydrology applications. The key class in the library is WebServiceWrapper, which provides a method for calling Web Services from a COM (e.g., Visual Basic for Applications (VBA)) environment. This class can be used to call WaterOneFlow Web Services for downloading hydrologic time series.

HydroObjects itself is not a standalone application. Rather, the user makes a reference to HydroObjects in another application, enabling that application to call Web Services using the WebServiceWrapper.

The WebServiceWrapper creates proxy classes for Web Services at run time. This makes HydroObjects a useful library even for programs that can already communicate with Web Services, because often in those programs the desired Web Service must be interrogated initially at design time to generate proxy classes. In other words, with the WebServiceWrapper, the program does not have to know about a Web Service until it is ready to use the Web Service.

This software manual includes the following sections:

1. Introduction and Software Description
2. Installation Information
3. Class Reference
4. Example Application

Installation Information

Installation Prerequisites

Prior to running the HydroObjects installation, you must first install the Microsoft .Net Framework Version 2.0. The .Net Framework Version 2.0 is free, and is required to run software applications developed in Microsoft's Visual Studio .Net 2005. Instructions for downloading and installing the .Net Framework Version 2.0 can be obtained from the Microsoft website via the following URL:

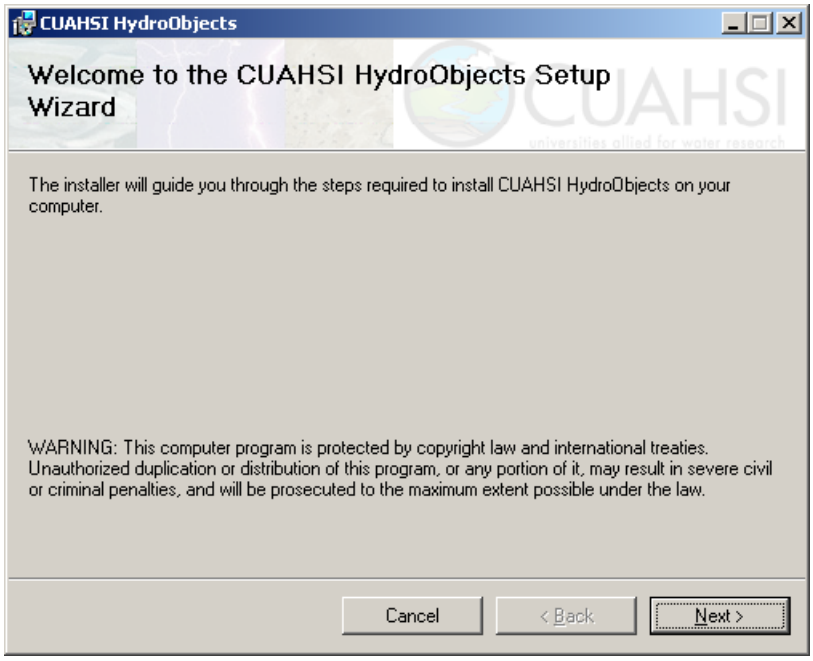
http://www.microsoft.com/downloads/details.aspx?FamilyID=0856EACB-4362-4B0D-8EDD-AAB15C5E04F5&displaylang=en

Once the .Net Framework Version 2.0 has been installed, you can continue with the HydroObjects installation.

Installing HydroObjects

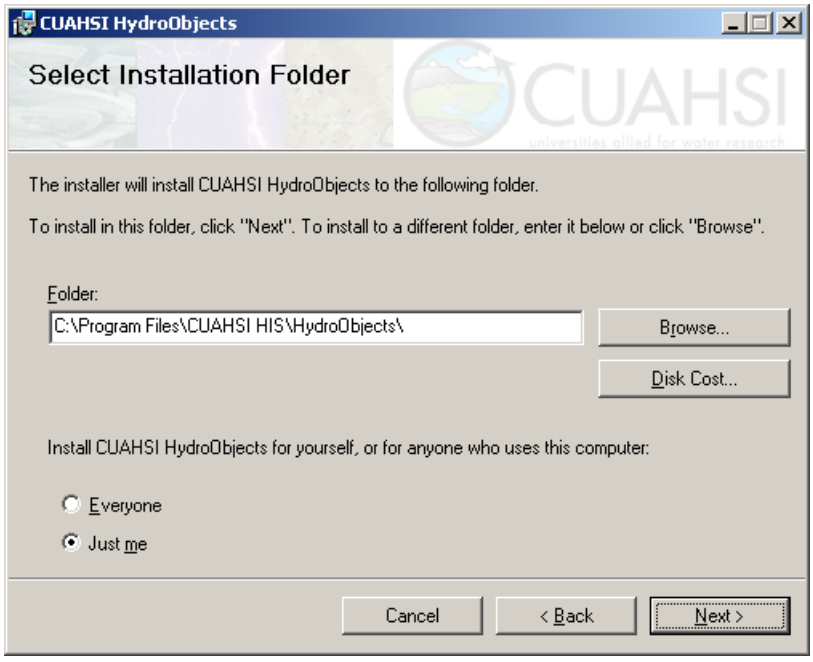
Install HydroObjects using the following steps:

1. First, ensure that you have installed the Microsoft .Net Framework Version 2.0. See the previous section if you have not done so.
2. Double click on the setup.exe installation file. This will begin the installation of HydroObjects. After a few moments, the following window will appear:

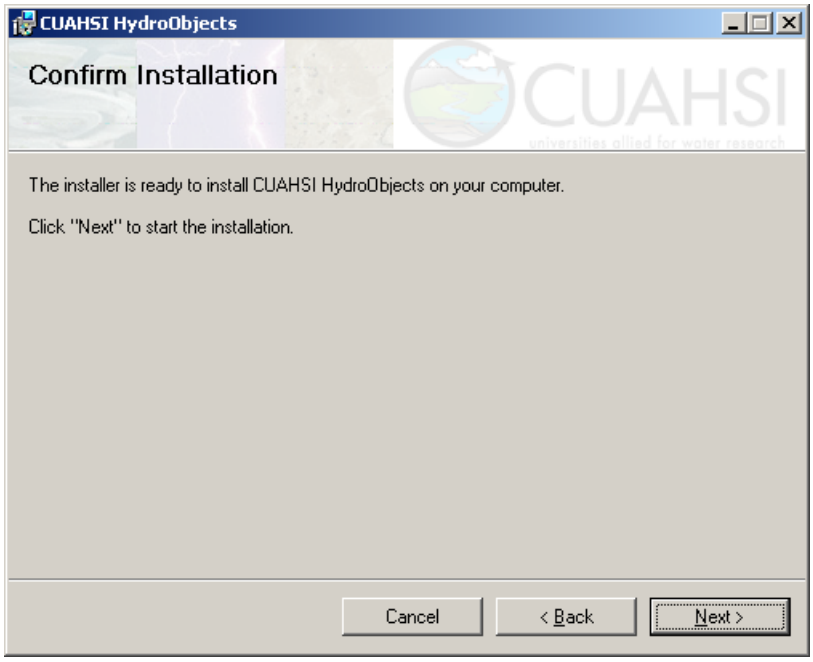


3. Click Next to continue with the installation. You will see the following window:

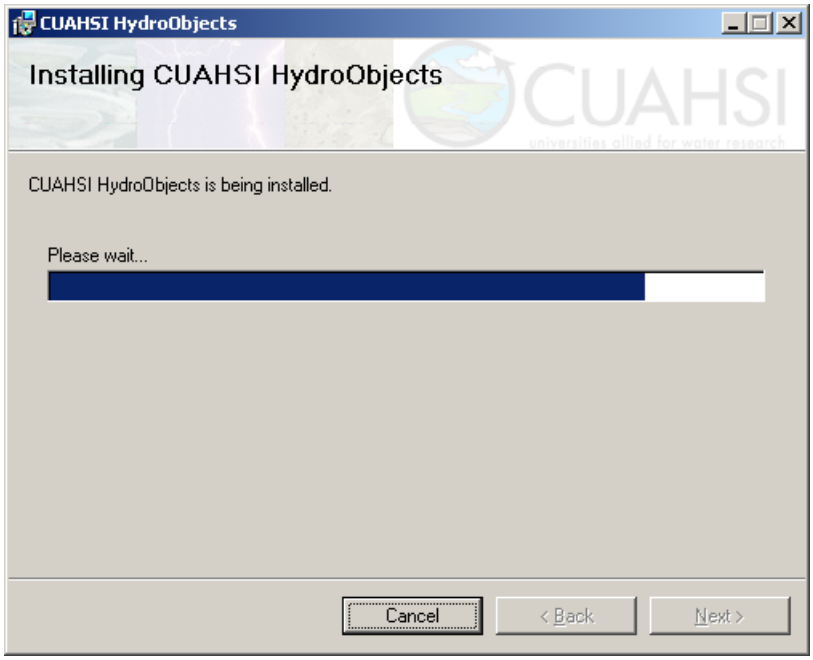
4. Read the license and then click on the radio button next to “I Agree” to accept the license. Click Next to continue. The following window will appear:



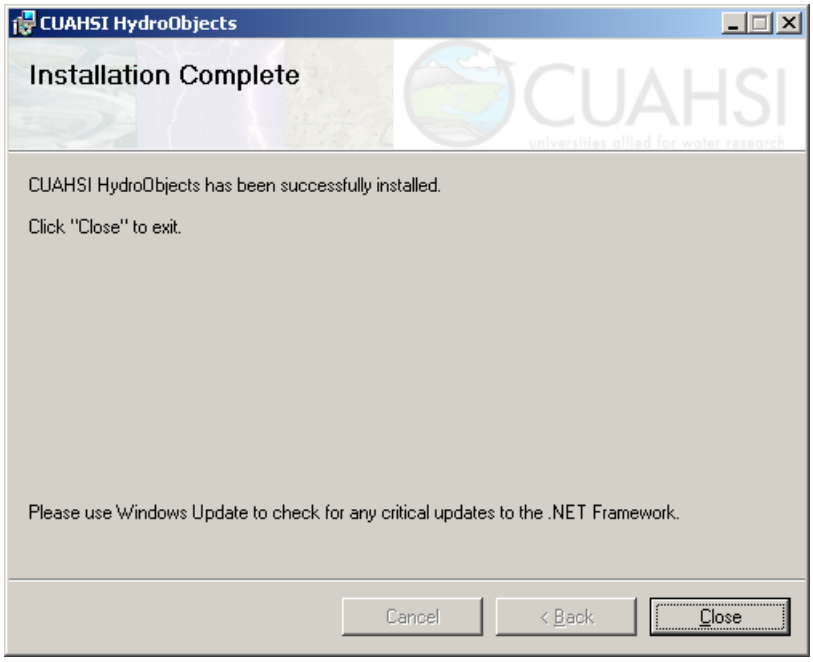
5. Choose where to install HydroObjects, and for whom to install HydroObjects, and click Next. The following window will appear:



6. Click Next to start the installation. As installation progresses, you will see a progress bar.



7. When installation has completed, click Close.



8. When installation has finished, view documentation or an example application by clicking Start | All Programs | CUAHSI HIS | HydroObjects.

UNINSTALLING HYDROOBJECTS

HydroObjects can be uninstalled from the Add or Remove Programs item under Control Panel.

Class Reference

HydroObjects currently contains one class: WebServiceWrapper. In the future, more classes related to hydrologic processing may be added.

WebServiceWrapper Class

This class provides a method for calling Web Services from a COM (e.g., VBA) environment.

Description

This class can be used to call WaterOneFlow Web Services for downloading hydrologic time series. The proxy classes for Web Services are created at run time.

Members

ClearAssemblyCaches Method

Clears local assembly caches for a given Web Service.

Syntax

object.**ClearAssemblyCaches (***wsdl***)**

The **ClearAssemblyCaches** method syntax has the following argument:

Part	Description
wsdl	Required. A **String** providing the URI to the WSDL for the web service.

Remarks

In order to call a Web Service, the WebServiceWrapper object must first build a cache of proxy classes for the Web Service. These classes define what the Web Service can do. If a Web Service has changed since the last time the cache was built, the cache will be out of date. The **ClearAssemblyCaches** method clears the cache, which forces the WebServiceWrapper object to rebuild the cache from the latest version of the Web Service.

Use this method when the Web Service that you are calling tends to be updated frequently, or whenever you want to make certain that you are using caches that reflect the most up-to-date version of a Web Service.

DownloadWebASCII Method

Returns ASCII text from a given resource.

Syntax

variable = object.**DownloadWebASCII (***address***)**

The **DownloadWebASCII** method syntax has the following object qualifier and argument:

Part	Description
variable	A **String** variable.
address	Required. A **String** providing the URI to the resource from which to download ASCII data.

Remarks

This method is handy if you want to download the content from a web page as text.

ENUMSTRING METHOD

Converts a dynamic proxy enum to a text value.

Syntax

variable = object.**EnumString (***enumName, value***)**

The **EnumString** method syntax has the following object qualifier and arguments:

Part	Description
variable	A **String** variable.
enumName	Required. A **String** providing the name of the enum.
value	Required. A reference to an **Object** that is the Parent object for the enum.

Remarks

Environments and languages such as VBA may not be able to convert enumerations returned from Web Services. Instead of getting the text value for the enumeration, a numerical value indicating the order of the item in the enumeration may be returned. The **EnumString** method converts the enumeration to text, and returns the text to the client.

Note that the *value* parameter is not a reference to the enum object itself, but to the enum object's parent object.

INVOKECALL METHOD

Make a dynamic connection to and call a method from a Web Service.

Syntax

variable = object.**InvokeCall (***Wsdl, ServiceName, MethodName, ParameterArray, ClearAssemblyCaches***)**

The **InvokeCall** method syntax has the following object qualifier and argument:

Part	Description
variable	A **Object** that will receive the outputs from the Web Service method. The object should be dimensioned appropriately to handle the expected return from the Web Service method. For example, when the method returns a **String**, dimension *variable* as a **String**. If the return type is unknown, dimension *variable* as an **Object** (aka **Variant**).
Wsdl	Required. A **String** providing the URI to the WSDL for the web service.
ServiceName	Required. A **String** providing the name of the Web Service to access.
MethodName	Required. A **String** providing the name of the method to call from the Web Service.
ParameterArray	Required. An **Object** array with parameters required by the method from the Web Service.
ClearAssemblyCaches	Optional. A **Boolean** which forces the object to clear and rebuild the assembly caches for the Web Service.

Remarks

Because WebServiceWrapper makes dynamic connections to Web Services, it does not know about parameters required for each web method at design time. Thus, when sending parameters to a web method, an array is used,

where each item in the array represents a web method parameter. The order of items in the array corresponds to the order of parameters in the web method signature. For example, suppose a web method required two parameters: StartDate and EndDate. The *ParameterArray* would be dimensioned (0 to 1), where item (0) would be the StartDate, and item (1) would be the EndDate.

The *ClearAssemblyCaches* parameter indicates if the assembly cache for the Web Service should be cleared and rebuilt before accessing the web method. This task is performed the same as in the separate **ClearAssemblyCaches** method, except that if an error occurs while clearing the cache while calling **InvokeCall**, the error is trapped and cleared and no notification is sent to the calling procedure. If this behavior is undesirable, call **ClearAssemblyCaches** before calling **InvokeCall**, and then set the *ClearAssemblyCaches* parameter to False.

The data type for *variable* is determined at run time, when the method populates *variable* with whatever is returned from the web method. As a result, the properties and methods of *variable* may not be available at design time, which makes programming a bit more difficult if the web method does not return a basic data type such as a String. When programming against complex classes returned from web methods, a recommended approach is to call **InvokeCall** in debug mode, and interrogate the returned object while debugging with tools from your IDE.

MSGBOXONERROR PROPERTY (READ/WRITE)

Determines if message box is shown on error during **InvokeCall** method.

Syntax

object.**MsgBoxOnError** = [*value*]

The **MsgBoxOnError** property syntax has the following object qualifier:

Part	Description
Value	A **Boolean** that determines the MsgBoxOnError value.

Remarks

Applications that are not Web-Service-enabled may not handle Web Service errors elegantly. If the **MsgBoxOnError** property is set to True, the **InvokeCall** method will display an error message as a message box if an error occurs, and then the method will pass the error on to the calling procedure. This insures that the user is made aware of web errors should one occur.

BASIC WORKFLOW FOR ACCESSING WEB SERVICES WITH HYDROOBJECTS

The basic workflow for accessing Web Services with HydroObjects is:

1. Determine information about the Web Service that you want to access. This can usually be accomplished by opening a web browser and navigating to the WSDL (or corresponding .asmx page if one is available) for the Web Service.
    a. What is the URI to its WSDL?
    b. What is the Service Name?
    c. What is the name of the method you wish to call?
    d. What are the parameters for the method?
2. Program with WebServiceWrapper to access the Web Service.
    a. Create a parameter array with one item for each parameter expected by the web method.

    b. If you have not yet cleared assembly caches since starting the current instance of your program, do so by calling ClearAssemblyCaches on a WebServiceWrapper object.
    c. Call InvokeCall on a WebServiceWrapper object and assign the result to an object.
3. Use the object populated by WebServiceWrapper to send output to your program.

EXAMPLE APPLICATION

An Excel spreadsheet called Streamflow.xls is included with HydroObjects, which demonstrates how to use the WebServiceWrapper class to call Web Services and send the result to cells in a spreadsheet. The use case for the spreadsheet is to click a button and see a graph of the latest daily average streamflow values for a chosen stream gage within the USGS National Water Information System (NWIS) network.

The spreadsheet includes a reference to HydroObjects made via its VBA environment. Macros in the spreadsheet use HydroObjects to give Excel access to a Web Service for USGS NWIS daily values. The macros are well-commented, and serve as a learning tool for understanding how to use HydroObjects to access Web Services.

Before using the spreadsheet, make sure that:

1. Your computer has Internet access (including access through firewalls, if present, in order to download data).
2. MS Excel 2003 Service Pack 3, or MS Excel 2007 or greater is installed.
3. Macros are enabled in Excel.

To use the spreadsheet:

1. Open Streamflow.xls. (Remember to enable macros.)
2. Input the site code for a USGS stream gage next to the box marked Site Code. An example of a valid site code is **NWIS:08158000**, which is for the Colorado River at Austin, TX.
3. Click the Get Values button. The time series from the web service will appear in the worksheet. A chart in the worksheet will also be updated with the new data.

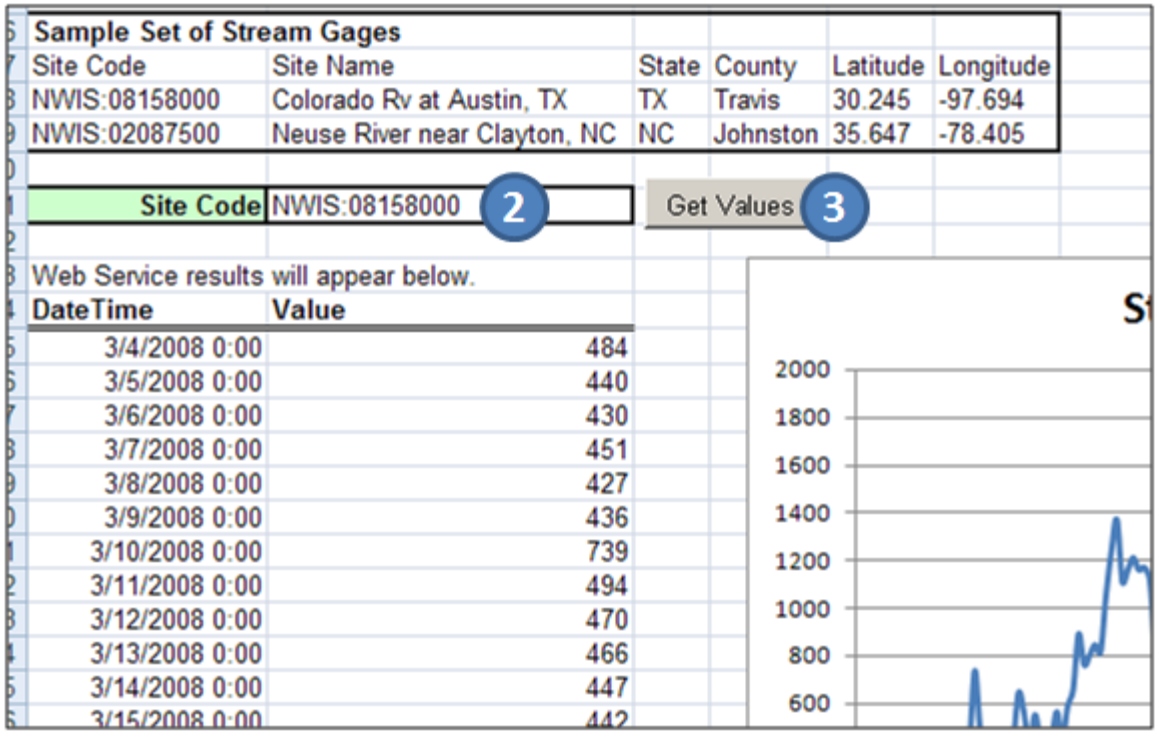


Figure 1 Steps 2 (input site code) and 3 (click Get Values) for using the spreadsheet.

4. Once you are familiar with the user experience from the spreadsheet, open the VBA environment to view the macros that make the spreadsheet work. Try modifying the macros to serve your own needs.