

HOW TO ANALYZE DATA FROM WEB SERVICES IN MATLAB

INTRODUCTION

MATLAB users can take advantage of web service methods by using the built in function *createClassFromWsdL*. 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 function then creates a MATLAB class that can be used to access web services from MATLAB. This tutorial demonstrates how to call a CUASHI HIS Water Data web service from MATLAB, parse the result, and plot a time series graph. In the exercise, you will use *GetSites* and *GetSiteInfo* to get the sites and then site information available from a Water Data web service to build up a table of available data. The MATLAB commands to then parse the XML return from a *GetValues* call are illustrated so as to create a plot of one of the time series available from the web service.

COMPUTER AND SKILL REQUIREMENTS

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

- Working Internet connection
- MATLAB version 7 (or greater) software
- Geodise XML Toolbox for MATLAB. This toolbox is available from http://www.geodise.org/toolboxes/generic/xml_toolbox.htm. This is used to parse the XML text that the web services returns into MATLAB objects.

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

- MATLAB version 7

Note: The source code for the *Reynolds.m* M-file used is located in Appendix A.

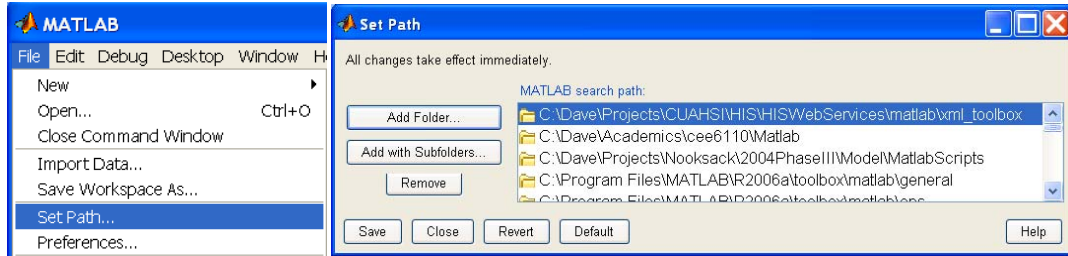
PROCEDURE

The information returned from CUAHSI web services is formatted either in XML or directly in object format accessible by MATLAB. 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. To parse the XML we will use functions from the free open source XML Toolbox from Geodise.

SETTING UP THE XML PARSER

1. **Download** the file `xml_toolbox-3.1.2_MATLAB-7.0.zip` from http://www.geodise.org/toolboxes/generic/xml_toolbox.htm. Unzip the file into a convenient location. A folder `xml_toolbox` should be obtained from this zip file.
2. **Start** MATLAB.

3. **Set** the MATLAB path to include the xml_toolbox folder with MATLAB xml tools

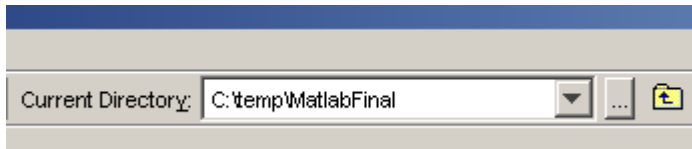


CREATE CLASS AND GET SITES

The first things that need to be done are to create the matlab class then use the GetSites function to identify sites from a Water Data web service.

Note: Instead of entering code as shown below, you could copy the code from Appendix A, or simply refer to Reynolds.m if you were provided a copy of the finished file with this workbook.

1. **Open** MATLAB and set the current directory to the location where you wish to perform the work.



2. Open a matlab file to record your work (e.g. Reynolds.m), or use the existing file. The File -> Open or File -> New commands do this.
3. In the MATLAB editor, **enter** the following lines of code. Highlight this code and press F9 to execute the code. This code creates an instance of the MODIS class from the WSDL.

```
% create NWIS Class and an instance of the class
createClassFromWsd1('http://idahowaters.uidaho.edu/RCEW_ODWS/cuahsi_1_0
.asmx?WSDL');
WS = WaterOneFlow;
```

This creates an instance of the class named WaterOneFlow, then assigns it to the variable WS. The URL above is a URL of one of the CUAHSI Water Data web services, in this case the web service for Reynolds Creek Experimental Watershed.

4. In the MATLAB editor, **enter** and execute the following lines of code. This code to run the GetSites function. The first argument specifies the web service. The second object is to specify which sites to "get". By leaving this blank, all sites are

retrieved. The third argument is an authentication token, not used at present, so it is left blank.

```
% Get all the sites
sites=GetSites(WS, '', '');
```

The GetSites function returns an object that is directly readable in MATLAB. The following lines expose some of the content of the sites object that is returned.

```
% Look at information about the first site
sites.site(1,1).siteInfo.geoLocation.geogLocation.latitude
sites.site(1,1).siteInfo.geoLocation.geogLocation.longitude
sites.site(1,1).siteInfo.siteName
nsites=length(sites.site)
```

The return from these lines is:

```
ans =43.2949782665
ans =-116.827040735
ans =Station 012x29
nsites = 85
```

These indicate the latitude, longitude and sitename of the first of these sites, together with the number of sites

5. In the MATLAB editor, **enter** and execute the following lines of code.

```
% plot the site locations
for is=1:nsites
    lat(is)=str2num(sites.site(is,1).siteInfo.geoLocation.geogLocation.latitude);
    lon(is)=str2num(sites.site(is,1).siteInfo.geoLocation.geogLocation.longitude);
    snames{is}=sites.site(is,1).siteInfo.siteName;
end
plot(lon,lat, '.')
```

This plots a point at the latitude and longitude of each site.

GET SITE INFORMATION

6. In the MATLAB editor, **enter** and execute the following lines of code.

```
% Get detailed information for Site of Interest
scode=sites.site(2,1).siteInfo.siteCode
siteid=strcat('RCEW:',scode)
strSite=GetSiteInfoObject(WS,siteid, '');
```

This calls the GetSiteInfo function (object version) for a specific site, the second site on the list. The calling argument "siteid" is the siteCode from the sites object.

In the MATLAB editor, **enter** and execute the following lines of code to look at the strSite object that was returned.

```
% Look at what was returned
seriesAtSite=strSite.site.seriesCatalog
seriesAtSite.series(1).variable.variableName
seriesAtSite.series(1).variable.variableCode
seriesAtSite.series(1).variable.units
seriesAtSite.series(1).variableTimeInterval.beginDateTime
seriesAtSite.series(1).variableTimeInterval.endDateTime
seriesAtSite.series(1).valueCount
```

The return from these lines is

```
ans =Precipitation breakpoint
ans =V002
ans =millimeter
ans =1962-01-06T01:00:00
ans =1976-12-23T12:31:00
ans =6276
```

This shows series catalog information for the first series from the catalog for the site that was just retrieved.

7. In the MATLAB editor, **enter** and execute the following lines of code

```
% Get this information for all series and all sites
ns=0
clear varName varUnits beginDt endDt varCode siteNames siteCodes
valCounts
for isite=1:nsites
    scode=sites.site(isite,1).siteInfo.siteCode;
    siteid=strcat('RCEW:',scode);
    strSite=GetSiteInfoObject(WS,siteid,'');
    seriesAtSite=strSite.site.seriesCatalog;
    nsc=length(seriesAtSite); % Number of series at the site
    for is=1:nsc
        nseries=length(seriesAtSite(is).series);
        for iss=1:nseries
            ns=ns+1
            % Variable name, units, begin and end date and time
            siteNames{ns,1}=strSite.site.siteInfo.siteName;
            siteCodes{ns,1}=strSite.site.siteInfo.siteCode;
            varName{ns,1}=seriesAtSite(is).series(iss).variable.variableName;
            varCode{ns,1}=seriesAtSite(is).series(iss).variable.variableCode;
            varUnits{ns,1}=seriesAtSite(is).series(iss).variable.units;
```

```

        beginDt{ns,1}=seriesAtSite(is).series(iss).variableTimeInte
        rval.beginDateTime;
        endDt{ns,1}=seriesAtSite(is).series(iss).variableTimeInterv
        al.endDateTime;
        valCounts{ns,1}=seriesAtSite.series(1).valueCount;
    end
end
end
table=[siteNames siteCodes varName varCode varUnits beginDt endDt
valCounts];

```

This loops over all sites, calling `GetSiteInfo` for each and building up a table of the catalog information. Following are excerpts from this table.

'Station 012x29'	'S001'	'Precipitation breakpoint'	'V002'	'millimeter'	'1962-01-06T05:22:00'	'1976-12-23T12:34:00'	'8758'
'Station 012x29'	'S001'	'Precipitation hourly'	'V001'	'millimeter'	'1962-01-06T06:00:00'	'1976-12-23T12:00:00'	'8758'
'Station 015x95'	'S002'	'Precipitation breakpoint'	'V002'	'millimeter'	'1962-01-06T01:00:00'	'1976-12-23T12:31:00'	'6276'
'Station 015x95'	'S002'	'Precipitation hourly'	'V001'	'millimeter'	'1962-01-06T02:00:00'	'1976-12-23T12:00:00'	'6276'
...							
'Station 045x04'	'S010'	'Precipitation hourly'	'V001'	'millimeter'	'1965-01-03T12:00:00'	'1975-12-31T10:00:00'	'4814'
'Station 046x17'	'S011'	'Streamflow breakpoint'	'V004'	'cubic meters per second'	'1964-01-29T00:00:00'	'1996-10-01T00:00:00'	'149213'
'Station 046x17'	'S011'	'Streamflow hourly'	'V003'	'cubic meters per second'	'1964-01-29T01:00:00'	'1996-10-01T00:00:00'	'149213'

GET VALUES

8. In the MATLAB editor, **enter** and execute the following lines of code

```

% GetValues to get the data
series=24 % This is the series in the table above
%begindate=beginDt(series)
begindate='1994-10-01T00:00:00'
enddate=endDt(series)
variable=strcat('RCEW:',varCode(series))
site=strcat('RCEW:',siteCodes(series))
valuesxml=GetValues(WS,site,variable,begindate,enddate,'');

```

This specifies parameters for one specific series and calls the `GetValues` function for this specific series, in this case the 24th series returned in the table built up earlier.

In the MATLAB editor, **enter** and execute the following lines of code

```

fid=fopen('temp.xml','w+')
n=fopen(fid,'%s',valuesxml)
fclose(fid)

```

This writes the XML `getvalues` return to a file that can be examined in a browser. The following is an excerpt from this file

```

- <timeSeriesResponse xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wtr="http://www.cuahsi.org/waterML/"
  xmlns="http://www.cuahsi.org/waterML/1.0/">
+ <queryInfo>
- <timeSeries>
  + <sourceInfo xsi:type="SiteInfoType">
  + <variable>
  - <values unitsAbbreviation="m3/s" unitsCode="36" unitsType="Flow" count="17545">
    <value sensorCode="nc" dateTime="1994-10-01T00:00:00"
      qualityControlLevel="Raw data" methodID="19" sourceID="1">0</value>
    <value sensorCode="nc" dateTime="1994-10-01T01:00:00"
      qualityControlLevel="Raw data" methodID="19" sourceID="1">0</value>
    <value sensorCode="nc" dateTime="1994-10-01T02:00:00"
      qualityControlLevel="Raw data" methodID="19" sourceID="1">0</value>

```

9. In the MATLAB editor, **enter** and execute the following lines of code

```

% Parse the XML into a Matlab object to work with.

% This is a function from Geodise XML toolbox
% http://www.geodise.org/toolboxes/generic/xml_toolbox.htm
valuesobj=xml_parseany(valuesxml);

```

This runs the function `xml_parseany` from the Geodise XML toolbox to create a matlab object from the XML returned by the `GetValues` function. This was needed here, because the object versions of the HIS web data service functions do not expose the attributes that include `dateTime` information.

10. In the MATLAB editor, **enter** and execute the following lines of code

```

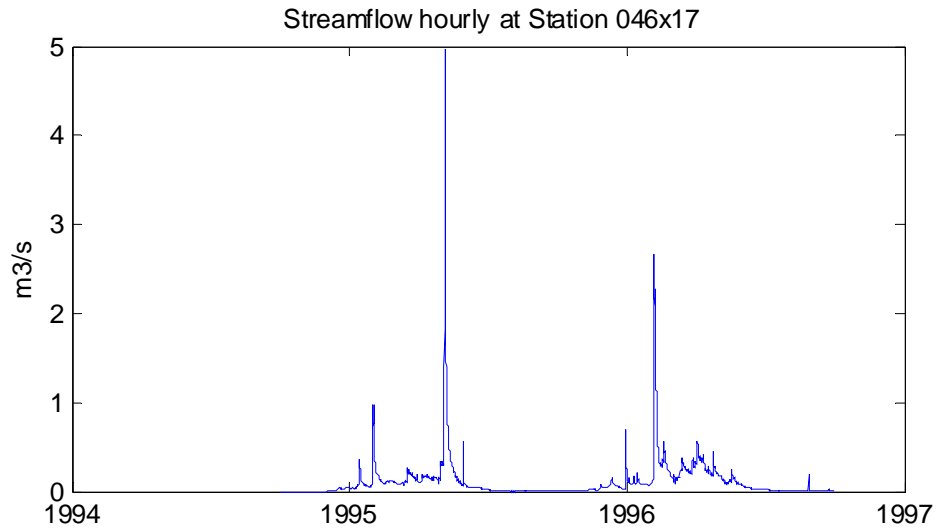
% Then plot the data
nvalues=str2num(valuesobj.timeSeries{1,1}.values{1,1}.ATTRIBUTE.count)
for i=1:nvalues
  datetime=valuesobj.timeSeries{1,1}.values{1,1}.value{1,i}.ATTRIBU
  TE.dateTime;
  year=str2num(datetime(1:4));
  month=str2num(datetime(6:7));
  day=str2num(datetime(9:10));
  hr=str2num(datetime(12:13));
  min=str2num(datetime(15:16));
  sec=str2num(datetime(18:19));
  date(i)=datenum(year,month,day,hr,min,sec);
  flowval(i)=str2num(valuesobj.timeSeries{1,1}.values{1,1}.value{1,
  i}.CONTENT);
end
plot(date,flowval);datetick;

% Label the plot with metadata from the returned object

```

```
ylabel(valuesobj.timeSeries{1,1}.values{1,1}.ATTRIBUTE.unitsAbbreviation)
title([valuesobj.timeSeries{1,1}.variable{1,1}.variableName{1,1}.CONTENT
T ' at ' ...
valuesobj.timeSeries{1,1}.sourceInfo{1,1}.siteName{1,1}.CONTENT])
```

This extracts the date time information from the GetValues response and formats it into a MATLAB datetime object, so that the data can be plotted. The following plot results. Note that the Y axis label and title are extracted from information in the GetValues response.



APPENDIX A: FULL SOURCE CODE FOR REYNOLDS.M

```
% This file presents an example of MATLAB commands to access CUAHSI HIS
web
% services
% David Tarboton
% July 15, 2008

% create NWIS Class and an instance of the class
createClassFromWsd1('http://idahowaters.uidaho.edu/RCEW_ODWS/cuahsi_1_0
.asmx?WSDL');
WS = WaterOneFlow;

% Get all the sites
sites=GetSites(WS, '', '');

% Look at information about the first site
sites.site(1,1).siteInfo.geoLocation.geogLocation.latitude
sites.site(1,1).siteInfo.geoLocation.geogLocation.longitude
sites.site(1,1).siteInfo.siteName
nsites=length(sites.site)

% plot the site locations
for is=1:nsites
    lat(is)=str2num(sites.site(is,1).siteInfo.geoLocation.geogLocatio
n.latitude);
    lon(is)=str2num(sites.site(is,1).siteInfo.geoLocation.geogLocatio
n.longitude);
    snames{is}=sites.site(is,1).siteInfo.siteName;
end
plot(lon,lat, '.')

% Get detailed information for Site of Interest
scode=sites.site(2,1).siteInfo.siteCode
siteid=strcat('RCEW:',scode)
strSite=GetSiteInfoObject(WS,siteid, '');

% Look at what was returned
seriesAtSite=strSite.site.seriesCatalog
seriesAtSite.series(1).variable.variableName
seriesAtSite.series(1).variable.variableCode
seriesAtSite.series(1).variable.units
seriesAtSite.series(1).variableTimeInterval.beginDateTime
seriesAtSite.series(1).variableTimeInterval.endDateTime
seriesAtSite.series(1).valueCount

% Get this information for all series and all sites
ns=0
clear varName varUnits beginDt endDt varCode siteNames siteCodes
valCounts
for isite=1:nsites
```

```

scode=sites.site(isite,1).siteInfo.siteCode;
siteid=strcat('RCEW:',scode);
strSite=GetSiteInfoObject(WS,siteid,'');
seriesAtSite=strSite.site.seriesCatalog;
nsc=length(seriesAtSite); % Number of series at the site
for is=1:nsc
    nseries=length(seriesAtSite(is).series);
    for iss=1:nseries
        ns=ns+1
        % Variable name, units, begin and end date and time
        siteNames{ns,1}=strSite.site.siteInfo.siteName;
        siteCodes{ns,1}=strSite.site.siteInfo.siteCode;
        varName{ns,1}=seriesAtSite(is).series(iss).variable.variableName;
        varCode{ns,1}=seriesAtSite(is).series(iss).variable.variableCode;
        varUnits{ns,1}=seriesAtSite(is).series(iss).variable.units;
        beginDt{ns,1}=seriesAtSite(is).series(iss).variableTimeInterval.beginDateTime;
        endDt{ns,1}=seriesAtSite(is).series(iss).variableTimeInterval.endDateTime;
        valCounts{ns,1}=seriesAtSite.series(1).valueCount;
    end
end
end
table=[siteNames siteCodes varName varCode varUnits beginDt endDt
valCounts];

% GetValues to get the data
series=24 % This is the series in the table above
%begindate=beginDt(series)
begindate='1994-10-01T00:00:00'
enddate=endDt(series)
variable=strcat('RCEW:',varCode(series))
site=strcat('RCEW:',siteCodes(series))
valuesxml=GetValues(WS,site,variable,begindate,enddate,'');

fid=fopen('temp.xml','w+')
n=fprintf(fid,'%s',valuesxml)
fclose(fid)

% Parse the XML into a Matlab object to work with.

% This is a function from Geodise XML toolbox
% http://www.geodise.org/toolboxes/generic/xml_toolbox.htm
valuesobj=xml_parseany(valuesxml);

% Then plot the data
nvalues=str2num(valuesobj.timeSeries{1,1}.values{1,1}.ATTRIBUTE.count)
for i=1:nvalues
    datetime=valuesobj.timeSeries{1,1}.values{1,1}.value{1,i}.ATTRIBUTE.dateTime;
    year=str2num(datetime(1:4));
    month=str2num(datetime(6:7));
    day=str2num(datetime(9:10));

```

```

    hr=str2num(datetime(12:13));
    min=str2num(datetime(15:16));
    sec=str2num(datetime(18:19));
    date(i)=datenum(year,month,day,hr,min,sec);
    flowval(i)=str2num(valuesobj.timeSeries{1,1}.values{1,1}.value{1,
i}.CONTENT);
end
plot(date,flowval);datetick;

% Label the plot with metadata from the returned object
ylabel(valuesobj.timeSeries{1,1}.values{1,1}.ATTRIBUTE.unitsAbbreviatio
n)
title([valuesobj.timeSeries{1,1}.variable{1,1}.variableName{1,1}.CONTEN
T ' at ' ...
    valuesobj.timeSeries{1,1}.sourceInfo{1,1}.siteName{1,1}.CONTENT])

```