AWDB Web Service User Guide

The AWDB (Air-Water Database) Web Service provides methods for programmatically accessing data, metadata (elements, units, stations, etc.), 30-Year Normals, and water supply forecasts.

Contents

Web Service Test Tool

Lets users run any methods of the Web Service, set SOAP requests, SOAP responses, and the actual data returned. This is a powerful tool that allows users to look at data to test the Web Service to ensure it returns what is wanted before implementing the calls in code.

WSDL File

The Web Services Description Language (WSDL) file fully describes the AWDB Web Service. It's an XML document that defines all the calls that can be made on the Web Service, the parameters and return values for each call, and the address of the service.

Tutorial

Helps new users get started using the Web Service. The tutorial contains an overview, details on how to generate Java client stubs, how to create an instance of the awdbWebService object, and several use cases.

AWDB Web Service Tutorial

Note: The following tutorial describes how to use a web service to access data in the Natural Resources Conservation Service (NRCS) National Water and Climate Center (NWCC) AWDB database. It is **not** meant to introduce general concepts and programming requirements relating to web services. Refer to the **AWDB Web Service Reference** for a complete listing of all Methods, Classes, Network Codes, and Element Codes.

Introduction

The National Water and Climate Center maintains a large database of soil, water, and climate data known as AWDB (air and water database). This database includes data from NWCC's extensive automated data collection system of snowpack and related climate data called SNOTEL (for SNOw TELemetry), its Soil Climate Analysis Network (SCAN) stations, as well as data from snow courses, streamflow stations, reservoirs, climate indices, National Weather Service COOP stations, 30-year normals, water supply forecasts, and more. All of this data can be accessed publicly through the AWDB Web Service.

What is a Web Service?

A **Web Service** is a way to communicate between two devices over the Internet. A user (client) of a Web Service invokes remote procedure calls on a Web Service that provides some data or other services. The user of a Web Service is typically another application. The application can be written in any language and on any platform.

A Web Service provider will provide a **WSDL** (Web Services Description Language) file that fully describes the Web Service. The WSDL file is an XML document that has all of the information needed to access and use a Web Service.

The WSDL file describes all of the calls that can be made on the Web Service, the parameters and return values for each call, and the address of the service. Most programming languages provide a way to translate a WSDL file into code that can be used to make calls to the Web Service.

AWDB Web Service Overview

The AWDB Web Service is a **SOAP** (Simple Object Access Protocol) Web Service. The AWDB Web Service provides methods for accessing metadata (about elements, units, stations, etc.), data, 30-year normals, and water supply forecasts.

To access the AWDB Web Service from a Java project, you must create the Java stubs from the Web Services Description Language (WSDL) and include the classes in your project.

This tutorial describes how to:

1. Generate the Java client stubs from the WSDL.
2. Create an instance of the AwdbWebService object.
3. Make calls to the AWDB database. Several “use cases” show examples of how to extract different types of data for different timeframes.

Web Service URL

The URL for the Web Service Description Language (WSDL) for the AWDB Web Service is: **https://wcc.sc.egov.usda.gov/awdbWebService/services?WSDL**

Conventions

The following conventions apply to the AWDB Web Service:

- Method parameters marked with an asterisk (*) are required.
- Several methods take a station triplet string or a list of triplets. These triplets identify a station and are composed of a three part string that contains the station id, the state code, and the network code of the station all delimited by a colon (‘:’), as follows:

[station id]:[state code]:[network code]

Example: "302:OR:SNTL"

Methods that request results for multiple stations will return an array holding the results in the same order the stations were given.

- Dates must be in the format YYYY-MM-DD. For example January 4, 2013 would be formatted as 2013-01-04.

Generate Java Client Stubs

wsimport is the utility to generate Java client stubs from a WSDL. It is supplied with the Java JDK and is located in the *bin* directory. If you do not have the JDK, you will need to **download** it first.

The *wsimport* utility simply needs to be given the desired package in which to put the new stubs and the WSDL address. From a command line, use the following command:

```
wsimport -p gov.usda.nrcs.wcc.awdbWebService -s .
‘https://www.wcc.nrcs.usda.gov/awdbWebService/services?WSDL’
```

This generates the stubs for the AWDB Web Service and places them in the package gov.usda.nrcs.wcc.awdbWebService. Once the client stubs are generated, they are imported and used just like any other Java package.

Create Instance of AwdbWebService Object

To make calls to the AWDB Web Service, you need an instance of the AwdbWebService class that is pointed to the Web Service. This can be done once prior to using the AWDB Web Service calls, such as on program start, and the instance can be reused repeatedly throughout the application.

To create this instance, use the following code:

```
import gov.usda.nrcs.wcc.awdbWebService.*; //The web service stubs
import javax.xml.namespace.QName;
import java.net.URL;

AwdbWebService m_webService = null;
try
```

```
{
    URL wsURL = newURL("https://www.wcc.nrcs.usda.gov/awdbWebService/services?wsdl");
    AwdbWebService_Service lookup = new AwdbWebService_Service(wsURL, new
    QName("https://www.wcc.nrcs.usda.gov/ns/awdbWebService", "AwdbWebService"));
        m_webService = lookup.getAwdbWebServiceImplPort();
}
catch (Exception e)
{
    //On failure do...
}
```

Now you can use m_webService to make calls to the AWDB database. For example: **m_webService.getElements()** returns all elements in the AWDB database.

Typical Use Cases

Following is an example java class that contains separate methods which demonstrate three common uses of the AWDB Web Service:

Use Case 1: Get an inventory of stations. The example gets an inventory of all SNOTEL stations in Oregon that have snow-water equivalent data (element code WTEQ) and returns a list of stations.

Use Case 2: Get period of record data. The example returns period of record data that are snow-water equivalent for a given station and date range.

Use Case 3: Get last seven days data. The example returns the last seven days of snow-water equivalent data, relative to today, for a given station.

The main method utilizes Use Case 1 and Use Case 3 by looping through the stations and returning each stationâ€™s metadata and values.

```
import gov.usda.nrcs.wcc.awdbWebService.*;
import javax.xml.namespace.QName;
import java.net.URL;
import java.math.BigDecimal;
import java.text.SimpleDateFormat;
import java.util.*;

public class WSUseCases
{
    static final SimpleDateFormat dateFormat = new SimpleDateFormat(
        "yyyy-MM-dd");
    AwdbWebService m_webService = null;

/**
 * Constructor which initializes a web service instance.
 */

public WSUseCases()
{
    initWebService();
}

/**
 * Initialize the web service member variable m_webService.
 */

private void initWebService()
{
    try
    {
        URL wsURL = new URL
            "https://www.wcc.nrcs.usda.gov/awdbWebService/services?wsdl");

        AwdbWebService_Service lookup = new AwdbWebService_Service(wsURL,
            new QName("https://www.wcc.nrcs.usda.gov/ns/awdbWebService",
                "AwdbWebService"));
        m_webService = lookup.getAwdbWebServiceImplPort():
    }
```

```
    catch (Exception e)
    {
        System.out.println("Problem creating web service client: "
            + e.getMessage());


    }
}


 /**
    * Example main program entry point which:
    * 1 - instantiates this class and creates a web service object
    * 2 - gets a list of stations, loop through the stations and print out each
    *     station's metadata
    * 3 - uses Use Case 3 to get last 7 days data for each station
    *     and print out each day's date and values.
    * @param args
    */


    public static void main(String[] args){
        WSUseCases useCase = new WSUseCases();


        //Use Case 1: Get Inventory of Stations
        // NOTE:  For this tutorial, the selection criteria of
        // network, state and element  are hard-coded
        // as â€˜SNTLâ€™, â€™ORâ€™ and â€™WTEQâ€™ in the getData function.
        // NOTE: The getStationMetadata*  function
        // may return some sites that do not have data
        // the date range you are interested in.
        // You can remove those stations by checking their
        // begin and end dates before retrieving data.


        List<StationMetaData> metadata = useCase.getStations();


        String stationName, stationTriplet, beginDate, endDate, countyName,
        huc, shefID, fipsCountryCode, fipsCountyCode, fipsStateNumber;


        BigDecimal elevation, latitude, longitude, stationDataTimeZone;


        // Loop through list of stations and retrieve and
        // print out metadata for one station


        for (StationMetaData metadatum : metadata)
        {
            stationName = metadatum.getName();
            stationTriplet = metadatum.getStationTriplet();
            beginDate = metadatum.getBeginDate();
            endDate = metadatum.getEndDate();
            countyName = metadatum.getCountyName();
            elevation = metadatum.getElevation();
            huc = metadatum.getHuc();
            latitude = metadatum.getLatitude();
            longitude = metadatum.getLongitude();
            shefID = metadatum.getShefId();
            stationDataTimeZone = metadatum.getStationDataTimeZone();
            fipsCountryCode = metadatum.getFipsCountryCd();
            fipsCountyCode = metadatum.getFipsCountyCd();
            fipsStateNumber = metadatum.getFipsStateNumber();


        System.out.println("Station Name:       " + stationName + '\n' +
                           "Station Triplet:    " + stationTriplet + '\n' +
                           "Begin Date:         " + beginDate + '\n' +
                           "End Date:           " + endDate + '\n' +
                           "County Name:        " + countyName + '\n' +
                           "Elevation:          " + elevation + '\n' +
                           "HUC:                " + huc + '\n' +
                           "Latitude:           " + latitude + '\n' +
                           "Longitude:          " + longitude + '\n' +
                           "SHEF ID:            " + shefID + '\n' +
                           "Station Time Zone:  " + stationDataTimeZone + '\n' +
                           "Fips Country Code:  " + fipsCountryCode + '\n' +
                           "Fips County Code:   " + fipsCountyCode + '\n' +
                           "Fips State #:       " + fipsStateNumber);
```

```
            /*
             * Use Case 3: Get Last 7 Days Data for One Station
             * This will get DAILY data for each of the stations from
             * Use Case 1.
             * Note: To apply Use Case 2: Period of Record of Data for One Station,
             * see example method getPeriodOfRecord().
             */

            Data[] data = useCase.getLastSevenDaysData(stationTriplet);
            String beginDt = data[0].getBeginDate();
            String flags = data[0].getFlags().toString();
            BigDecimal[] stationValues = data[0].getValues().toArray(
                new BigDecimal[0]);
            // Get values for current station
            for (int i = 0; i <= stationValues.length; i++)
            {
                System.out.println("Date: Flags/Value: " + beginDt + ":  "
                    + flags + "/" + stationValues[i] + '\n');
            }
        }
    }
```

Use Case 1: Get Inventory of Stations

```
 /**
     * Use Case 1: Get Inventory of Stations
     * This example will get an inventory of all stations in Oregon
     * for SNOTEL stations that have Snow Water Equivalent
     * and return  list of stations
     */

    public List<StationMetaData> getStations(){
        List<String> stationIds = null;
        List<String> stateCds = null;
        List<String> networkCds = Arrays.asList("SNTL");
        List<String> hucs = null;
        List<String> countyNames = null;
        BigDecimal minLatitude = null;
        BigDecimal maxLatitude = null;
        BigDecimal minLongitude = null;
        BigDecimal maxLongitude = null;
        BigDecimal minElevation = null;
        BigDecimal maxElevation = null;
        List<String> elementCodes = Arrays.asList("WTEQ");
        List<Integer> ordinals = Arrays.asList(1);
        List<HeightDepth> heightDepths = null;

        /*
         * If (logicalAnd) is true, the getStations() call will return only
         * stations that match ALL of the parameters passed in, otherwise itâ€™ll
         * return stations that match ANY of the parameters passed in.
         */

        boolean logicalAnd = true;
        List<String> stationTriplets = m_webService.getStations(stationIds,
            stateCds, networkCds, hucs, countyNames, minLatitude,
            maxLatitude, minLongitude, maxLongitude, minElevation,
            maxElevation, elementCodes, ordinals, heightDepths, logicalAnd);
        List<StationMetaData> stations = m_webService
            .getStationMetadataMultiple(stationTriplets)
        return stations;
    }
```

Use Case 2: Get Period of Record for One Station

```
/**
     * Use Case 2: Get period of record for one station.
     * This will return period of Data that are SNOW WATER EQUIVALENT (element
     * code = WTEQ) for a given station and date range.
     * Note: Always use an ordinal of 1, and heightDepth of null
     * (height depth is only used for soil sensors)
     * @param p_stationTriplet The station to get data for, ex: "471:ID:SNTL"
```

```
     * @param p_beginDate The begin date - a String format "yyyy-MM-dd"
     * @param p_endDate The end date - a String format "yyyy-MM-dd"
     * @return An Array of Data Objects
     */

    public Data[] getPeriodOfRecord(String p_stationTriplet, String p_beginDate,     String
p_endDate){
        Data[] values = m_webService.getData(Arrays.asList(p_stationTriplet),
            "WTEQ", 1, null, Duration.DAILY, true, p_beginDate, p_endDate, true)
            .toArray(new Data[0]);
        return values;
    }
```

Use Case 3: Get Last Seven Days of Data

```
 /**
     * Use Case 3: Get last seven days of data.
     * This will return the last seven days of SNOW WATER EQUIVALENT (element
     * code = WTEQ data, relative to today, for a given station.
     * Note: Always use an ordinal of 1, and heightDepth of null
     * (height depth is only used for soil sensors)
     * @param p_stationTriplet The station to get data for, ex: "471:ID:SNTL"
     * @return An Array of Data Objects
     */

    public Data[] getLastSevenDaysData(String p_stationTriplet){

        String today = dateFormat.format(new Date());
        Calendar lastWeek = GregorianCalendar.getInstance();
        lastWeek.add(Calendar.DAY_OF_YEAR, -7);
        String sevenDaysAgo = dateFormat.format(lastWeek.getTime());

        Data[] values = m_webService.getData(Arrays.asList(p_stationTriplet),
            "WTEQ", 1, null, Duration.DAILY, true, sevenDaysAgo, today, ture)
            .toArray(new Data[0]);

        return values;
     }

/*  End class WSUseCases */

}
```

Frequently-Asked Questions

Answers to questions from our users.

Q Why am I getting an HTTP 502 (Bad Gateway) error when trying to make calls to the AWDB Web Service from a .NET application?

A By default, .NET applications include the “HTTP” header “Expect: 100-continue”. This header causes the AWDB Web Service to return the HTTP 502 (Bad Gateway) error. This can be resolved by telling the .NET application to **not** include this HTTP header when making requests to the web service. Adding the following line will prevent the header from being included in requests to the AWDB Web Service. Ensure that this is set before any calls are made to the web service.

System.Net.ServicePointManager.Expect100Continue = false;

Q How do I determine the current version number of the AWDB Web Service?

A To determine the currently-deployed version number and build date of the AWDB Web Service, go to **https://wcc.sc.egov.usda.gov/awdbWebService/version.html**. The url will return a path containing the version number and the build date

Q I've written some calls to the AWDB Web Service. Is there a way to test these?

A Yes. Use the **AWDB Web Service Test Tool** to run any methods of the Web Service, set the SOAP request, SOAP response, and see the actual data returned. This is a powerful tool that lets users look at data or test the Web Service to ensure it returns what is wanted before implementing the calls in code.

Q I want to be sure I'm using the latest version of AWDB Web Service WSDL file. Where is it located?

A The latest version of the AWDB Web Service WSDL (Web Service Description Language) file can be found at: **https://wcc.sc.egov.usda.gov/awdbWebService/services?WSDL**

Q How do I get hourly data over a date range?

A The 'getHourlyData' method has parameters for 'beginDate' and 'endDate', as well as 'beginHour' and 'endHour'. The 'beginDate' and 'endDate' can start and end on a specific hour. Therefore, if you want to collect hourly data from 22:00 yesterday to 4:00 today, you can set the beginDate to 'YYYY-mm-01 22:00' to 'YYYY-mm-02 04:00' and leave the 'beginHour' and 'endHour'parameters blank (or null)

Q How do I get hourly data for a specific hour over a date range?

A The 'getHourlyData' method has parameters for 'beginDate' and 'endDate', as well as 'beginHour' and 'endHour'. The purpose of the'beginHour' and 'endHour' parameters are to return a subset of data for each day requested. Therefore, if you wanted to collect data for a specific hour or range of hours for every day between your begin and end dates, you would specify the beginHour and endHour parameters. For example, if you wanted the 00:00 hour data for each day from July 1 through July 4, you would set the beginDate to '2015-07-01', the endDate to '2015-07-04', the beginHour to '00:00', and the endHour to '00:00

Reference Guide

The National Water and Climate Center (NWCC) maintains a large database of soil, water, and climate data known as AWDB (air and water database). This database includes data from NWCC's extensive automated data collection system of snowpack and related climate data called SNOTEL (for SNOw TELemetry), its Soil Climate Analysis Network (SCAN) stations, as well as data from snow courses, streamflow stations, reservoirs, climate indices, National Weather Service COOP stations, 30-year normals, water supply forecasts, and more. All of this data can be accessed publicly through the AWDB Web Service.

What is a Web Service?

A **Web Service** is a way to communicate between two devices over the Internet. A user (client) of a Web Service invokes remote procedure calls on a Web Service that provides some data or other services. The user of a Web Service is typically another application. The application can be written in any language and on any platform.

A Web Service provider will provide a **WSDL** (Web Services Description Language) file that fully describes the Web Service. The WSDL file is an XML document that has all of the information needed to access and use a Web Service.

The WSDL file describes all of the calls that can be made on the Web Service, the parameters and return values for each call, and the address of the service. Most programming languages provide a way to translate a WSDL file into code that can be used to make calls to the Web Service.

AWDB Web Service Overview

The AWDB Web Service is a **SOAP** (Simple Object Access Protocol) Web Service. The AWDB Web Service provides methods for accessing metadata (about elements, units, stations, etc.), data, 30-year normals, and water supply forecasts.

This web page gives an overview of all of the methods of the AWDB Web Service, a description of the parameters of each method, a description of what is returned by each method, a detailed description of all the **classes** used by the web service, and the full list of possible **network codes** and **element codes** that can be used.

Web Service URL

The URL for the WSDL for the AWDB Web Service is: https://wcc.sc.egov.usda.gov/awdbWebService/services?WSDL

Conventions

The following conventions apply to the AWDB Web Service:

- Method parameters marked with an asterisk (*) are required.
- Several methods take a station triplet string or a list of triplets. These triplets identify a station and are composed of a three part string that contains the station id, the state code, and the network code of the station all delimited by a colon (':'), as follows:

  [station id]:[state code]:[network code]

  Example: 302:OR:SNTL

- Methods that request results for multiple stations will return an array holding the results in the same order the stations were given.
- Dates must be in the format YYYY-MM-DD. For example January 4, 2013 would be formatted as 2013-01-04.

Web Service Methods Overview

Station and Station Metadata Methods

Method	Description
getReservoirMetadata	Retrieves metadata about a reservoir station. These metadata are the capacity, usable capacity, and the elevation (in feet) of the reservoir when it is at capacity. (*Note: The AWDB database currently only has reservoir capacity populated.*)
getReservoirMetadataMultiple	Retrieves metadata for several reservoir stations in a single call. The information returned is the same as the **getReservoirMetadata** method.
getStations	Gets a list of stations that match one or more search criteria. Stations can be searched by station id, name, state, network, HUC, latitude, longitude, elevation, county name, element, and more. The method returns the three-part station identifiers (referred to as "station triplets") that are used to make calls to other methods to retrieve data or metadata for the stations.
getStationElements	Gets a list of all elements that a station has (or had) data, Normals, or forecasts for between a specified begin and end date.
getStationMetadata	Retrieves metadata for a station given the three-part station triplet identifier. This method returns a StationMetadata object that contains several pieces of metadata about the given station (such as name, latitude, longitude, elevation, county, and HUC).
getStationMetadataMultiple	Retrieves station metadata for multiple stations in a single call. This returns the same information as getStationMetadata for each station requested.

Data Methods

Method	Description
getData	Gets DAILY, MONTHLY, or SEMI_MONTHLY data for one more stations, for a single element, for a range of dates.
getDataInsertedOrUpdatedSince	Gets data (any duration except HOURLY) for one or more stations, for a single element, for a range of dates, *but only returns data that has been inserted or modified since some date that the user specifies.*
getHourlyData	Gets data for one or more stations, for a single element, for a range of dates. This will return data starting on the first value that actually has data for each station and will return a value for everything in between until the last non-null value between the requested dates and times.
getInstantaneousData	Gets instantaneous SNOTEL or SCAN data for one or more stations for a single element for a range of dates. Most SNOTEL and SCAN stations report data at least every few hours (and some every hour). Use this method to get SNOTEL or SCAN data that has a finer granularity than just daily.
getInstantaneousDataInsertedOrUpdatedSince	Gets instantaneous SNOTEL or SCAN data for one or more stations for a single element for a range of dates, *but only returns the value that have been insertedor updated since some date that the user specifies.* Most SNOTEL and SCAN stations report data at least every few hours (and some every hour). Use this method to get SNOTEL or SCAN data that has a finer granularity than just daily and where you only want to get the values that have been inserted since some date that the user passes in.
getPeakData	Gets the annual peak data value for each water year requested for one or more stations, for a single element. The peak data value is the value with the highest value in each water year. If there are multiple values that equal the highest value, the last one is returned.

30 Year Central Tendencies Methods

Method	Description
getCentralTendencyData	Gets the 30-year central tendency (*average* or *median*) value(s) for the current normals period, for one or more stations, for a single element, for a given duration, and for some range of dates. The method takes a **Duration** in order to retrieve DAILY, MONTHLY, SEMIMONTHLY, or ANNUAL central tendencies. (Note: For daily data, there is always a tendency value for February 29. If asking for data for January 1 to December 31, 366 values will be returned.)
getCentralTendencyPeakData	Gets the peak 30-year central tendency (*average* or *median*) value(s) for the current normals period, for one or more stations, for a single element, for a given duration, and for some range of dates. The method takes **Duration** in order to retrieve DAILY, MONTHLY, SEMIMONTHLY, or ANNUAL central tendencies. The peak central tendency value is the highest value (if there are multiple values that equal the highest value, the last one is returned).
getForecastPeriodCentralTendency	Gets the 30-year forecast period central tendency (*average* or *median*) value(s) for the current normals period, for one or more stations, for a given element, and for one or more forecast periods.

Metadata Methods

Method	Description
getElement	Retrieves metadata about a single element given an element code. The method returns an Element object which contains the element code, the name of the element, and the stored unit code (the units that the data were stored in and will be returned in unless otherwise specified).
getElements	Gets a list of all possible elements in the database. Each item in the returned list will be an Element object and will contain the same information described in the **getElement** method.
getHeightDepths	Gets a list of all possible height/depths that are defined in AWDB (returned as a list of HeightDepth objects). Each **HeightDepth** object consists of a value (positive for a height, and negative for a depth) along with the unit code of the value (usually “in” for inches).
getUnitName	Retrieves the (plural) name of a unit given a unit code. Calling this method with “in” for example returns “inches”.
getUnits	Retrieves a list of all of units defined in the AWDB database. Each item in the returned list will be a Unit object that contains the unit code and the plural name of the unit.

Forecast Metadata and Forecast Value Methods

Method	Description
getForecast	Gets a water supply forecast value for a given forecast point, element, forecast period, and publication date.
getForecastPeriods	Gets a list of all forecast periods that are defined in the database.
getForecastPoint	Gets the forecast point metadata for a given station. The forecast point metadata includes the name of the forecast point (which may be different from the name of the actual station), the exceedence probabilities that will be calculated for forecasts for this point, and the name of the forecaster responsible for producing forecasts for this point.
getForecastPoints	Finds one or more forecast points using search criteria such as station ids, state codes, network codes, forecast point names, HUCs, or name of the responsible forecaster.
getForecasts	Gets all the forecasts for a given station, element, and forecast period (for all publication dates).
getForecastsByPubDate	Gets a list of forecasts for a given station, element, forecast period, and whose publication dates are between a given begin and end date.
getForecastValue	Gets a single forecast (with no additional metadata) for a given station, element, forecast period, exceedence probability, and publication date.

Station & Station Metadata Methods - Details

getReservoirMetadata

This method retrieves metadata about a reservoir station that indicates the capacity, usable capacity, and the elevation (in feet) of the reservoir when it is at capacity. (*Note: The AWDB database currently only has the reservoir capacity populated*.)

Returns: A **ReservoirMetadata** object that contains the reservoir metadata for the requested station.

Parameters:

Parameter	Type	Description
*stationTriplet	String	The three-part station identifier of the reservoir station for which to get the metadata.

Example usage:

stationTriplet=”14080400:OR:BOR”

This will return a reservoir capacity of 153,000 acre-feet for this station (PRINEVILLE) in Oregon.

getReservoirMetadataMultiple

Retrieves metadata for several reservoir stations in a single call. The reservoir metadata for each station indicate the capacity, usable capacity, and the elevation (in feet) of the reservoir when it is at capacity. (*Note: The AWDB database currently only has the reservoir capacity populated.*)

Returns: A list of **ReservoirMetadata**objects that contain the reservoir metadata for each station requested in the order of the requested stations.

Parameters:

Parameter	Type	Description
*stationTriplets	List<String>	A list of the three-part station identifiers of the reservoir stations for which to get the metadata.

Example usage:

stationTriplets[0]= ”14080400:OR:BOR” // Reservoir station PRINEVILLE

stationTriplets[1]= “14361900:OR:BOR” // Reservoir station APPLEGATE

This will return a reservoir capacity of 153,000 acre-feet for this station (PRINEVILLE) in Oregon and 75,200 acre-feet for APPLEGATE.

getStations

Gets a list of stations that match one or more search criteria. Stations can be searched by station ids, names, states, networks, HUC, latitude, longitude, elevation, county, element, and more. The method returns the three-part station identifiers (“station triplets”) that are used to make calls to other methods to retrieve data or metadata for the stations.

Returns: A list of strings that are the stationTriplets for the stations which match the search criteria passed in.

Parameters: All are optional unless marked as required with an ‘*’

Parameter	Type	Description
stationIds	List<String>	A list of station ids or station names (can contain wildcards ‘*’ or ‘?’).
stateCds	List<String>	A list of two-character FIPS state codes of the states to search stations in.
networkCds	List<String>	A list of **network codes** of the stations for which to search.
hucs	List<String>	A list of Hydrologic Unit Codes (HUCs) of the stations for which to search. This includes an implied wildcard (*) at the end of each HUC. For example, if a user searches for a station with the HUC "17010101", it will get all stations whose HUC begins with "17010101".
countyNames	List<String>	A list of county names that the stations are in. The county names can contain the ‘*’ and ‘?’ wildcards.
minLatitude	BigDecimal	The minimum latitude (in decimal degrees) of the stations to find (inclusive)
maxLatitude	BigDecimal	The maximum latitude (in decimal degrees) of the stations to find (inclusive)
minLongitude	BigDecimal	The minimum longitude (in decimal degrees) of the stations to find (inclusive)
maxLongitude	BigDecimal	The maximum longitude (in decimal degrees) of the stations to find (inclusive)
minElevation	BigDecimal	The minimum elevation (in feet) of the stations to find (inclusive).
maxElevation	BigDecimal	The maximum elevation (in feet) of the stations to find (inclusive).
elementCd	List<String>	A list of **element codes** of the elements for which the stations have some data. The element codes can contain wildcards.
ordinals	List<Integer>	A list of the ordinals of the elements the stations must have. The ordinal is a number that starts at 1 and is used to distinguish multiple sensors of the same element. Elements with a lower ordinal usually have better data (more likely to be quality controlled).
heightDepths	List<HeightDepth>	A list of the height/depths that the elements of the stations must have.
*logicalAnd	boolean	If true, the method will return stations that match ALL of the parameters passed in. If false, the method will return stations that match ANY of the parameters passed in.

Example usage:

Case1: Find all SNOTEL stations in Oregon (SNOTEL only, not SCAN). SNOTEL station ids are numbers between 300 and 1999 and SCAN stations are 2000+ (and also station 15).

stationIds=”???”, “1???”

states=”OR”

networkCds=”SNTL”

logicalAnd=true

getStationElements

Gets a list of all elements that a station has (or had) data or normals for a specified begin and end date.

Returns: A list of **StationElement** objects that contain the elements that are available for the station and dates passed in.

Parameters:

Parameter	Type	Description
*stationTriplet	String	The three-part station identifier of the station for which to get the station elements.
beginDate	String	The date the element must have been put out of service after (in format yyyy-MM-dd HH:mm or MM/dd/yyyyHH:mm). Use null if the begin date doesn't matter.
endDate	String	The date the element must have been put into service before (in format yyyy-MM-dd HH:mm or MM/dd/yyyyHH:mm). Use null if the end date doesn't matter.

Example usage:

stationTriplet="302:OR:SNTL"

beginDate=null

endDate=null

This will return all elements that have been defined for station 302 - ANEROID LAKE #2.

getStationMetadata

Retrieves metadata for a station given the three-part station triplet identifier. This method returns a **StationMetadata** object that contains several pieces of metadata about the given station (such as name, latitude, longitude, elevation, county, and HUC).

Returns: A **StationMetadata** object that contains the station metadata for the station requested.

Parameters:

Parameter	Type	Description
*stationTriplet	String	The three-part station identifier of the station for which to get the metadata.

Example usage:

stationTriplet="302:OR:SNTL"

This will return a **StationMetadata** object that contains all of the metadata for station 302 - ANEROID LAKE #2.

getStationMetadataMultiple

Retrieves station metadata for multiple stations in a single call. This method returns a **StationMetadata** object that contains several pieces of metadata about each station passed in (such as name, latitude, longitude, elevation, county, and HUC).

Returns: A list of **StationMetadata** objects that contain the station metadata for each station requested. The metadata will be returned in the same order as the stations requested.

Parameters:

Parameter	Type	Description

*stationTriplets	List<String>	A list of the three-part station identifiers of the stations for which to get the metadata.

Example usage:

stationTriplets="302:OR:SNTL", "0645:NV:COOP"

This will return the station metadata for SNOTEL Station 302 - ANEROID LAKE #2 and National Weather Service COOP Station 0645 in Nevada.

Data Methods - Details

getData

Gets data for one or more stations, for a single element, for a range of dates. This will return data starting on the first value that actually has data for each station and will return a value for everything in between until the last non-null value between the requested dates.

Returns: An array of **Data** objects that contain the data values for each station requested. The data will be returned in the same order as the stations requested.

Parameters:

Parameter	Type	Description
*stationTriplets	List<String>	A list of the three-part station identifiers of the stations for which to get data.
*elementCd	String	The element code of the data to retrieve.
*ordinal	int	The ordinal for element of the data to retrieve. This will almost always be 1. The ordinal distinguishes multiple sensors of the same element, of which ordinal 1 is the primary sensor, 2 is the secondary sensor (if there are multiple sensors), and so on.
heightDepth	**HeightDepth**	The height/depth for the element of the data to retrieve, or null if the element doesn't have a height/depth. For most elements, this should be null. This is usually only non-null for soil-related elements.
*duration	**Duration**	The duration of the data to retrieve.
*getFlags	boolean	If true, data flags will be retrieved and returned, otherwise the flags won't be returned.
*beginDate	String	The earliest date to get data for (in format yyyy-MM-dd or MM/dd/yyyy)
*endDate	String	The latest date to get data for (in format yyyy-MM-dd or MM/dd/yyyy)
*alwaysReturnDailyFeb29	boolean	If 'true' or 'null' is passed in for this parameter, requests for daily data will always return a slot for February 29 (regardless of whether the year is a leap year or not). For leap years, the actual February 29 value will be returned; for non-leap years, null will be returned. If 'false' is passed in, requests for daily data will only return a slot for February 29 or leap years.

Example usage:

stationTriplets="302:OR:SNTL" // SNOTEL Station ANEROID LAKE #2 in Oregon

elementCd="PREC" // Accumulated Precipitation

ordinal=1

heightDepth=null

duration=Duration.DAILY

getFlags=true

beginDate="2010-01-01"

endDate="2010-01-31"

alwaysReturnDailyFeb29=false

Calling the **getData** method with these parameters will return daily accumulated precipitation for SNOTEL station 302 - ANEROID LAKE #2 for 01/01/2010 to 01/31/2010 and will include the data flags. There will be no placeholder "slot" for February 29.

getDataInsertedOrUpdatedSince

Gets data (any duration but HOURLY) for one or more stations, for a single element, for a range of dates, *but only returns data that has been inserted or modified since some date that the user specifies.*

Returns: An array of **Data** objects that contain the data values for each station requested. The data will be returned in the same order as the stations requested.

Parameters:

Parameter	Type	Description
*stationTriplets	List<String>	A list of the three-part station identifiers of the stations for which to get data.
*elementCd	String	The **element code** of the data to retrieve.
*ordinal	int	The ordinal for element of the data to retrieve. This will almost always be 1. The ordinal distinguishes multiple sensors of the same element, of which ordinal 1 is the primary sensor, 2 is the secondary sensor (if there are multiple sensors), and so on.
heightDepth	**HeightDepth**	The height/depth for the element of the data to retrieve, or null if the element doesn't have a height/depth. For most elements, this should be null. This is usually only non-null for soil-related elements.
*duration	**Duration**	The duration of the data to retrieve.
*getFlags	boolean	If true, data flags will be retrieved and returned, otherwise the flags won't be returned.
*beginDate	String	The earliest date to get data for (in format yyyy-MM-dd or MM/dd/yyyy)
*endDate	String	The latest date to get data for (in format yyyy-MM-dd or MM/dd/yyyy)
*insertOrUpdateBeginDate	String	The earliest insert or update date to get data for (in format yyyy-MM-dd or MM/dd/yyyy. The only data values that will be returned will be values that have been inserted or updated on or after this date.
*alwaysReturnDailyFeb29	boolean	If 'true' or 'null' is passed in for this parameter, requests for daily data will always return a slot for February 29 (regardless of whether the year is a leap year or not). For leap years, the actual February 29 value will be returned; for non-leap years, null will be returned. If 'false' is passed in, requests for daily data will only return a slot for February 29 or leap years.

Example usage:

stationTriplets="302:OR:SNTL" // SNOTEL Station ANEROID LAKE #2 in Oregon

elementCd="PREC" // Accumulated Precipitation

ordinal=1

heightDepth=null

duration=Duration.DAILY

getFlags=true

beginDate="2010-01-01"

endDate="2010-01-31"

insertOrUpdateBeginDate="2010-02-01"

alwaysReturnDailyFeb29=false

Calling the **getData** method with these parameters will return daily accumulated precipitation for SNOTEL station 302 - ANEROID LAKE #2 for 01/01/2010 to 01/31/2010 and will include the data flags. There will be no placeholder "slot" for February 29.

getHourlyData

Gets data for one or more stations, for a single element, for a range of dates. This will return data starting on the first value that actually has data for each station and will return a value for everything in between until the last non-null value between the requested dates and times.

Returns: An array of **HourlyData** objects that contain the data values for each station requested. The data will be returned in the same order as the stations requested.

Parameters:

Parameter	Type	Description
*stationTriplets	List<String>	A list of the three-part station identifiers of the stations for which to get data.
*elementCd	String	The **element code** of the data to retrieve.
*ordinal	int	The ordinal for element of the data to retrieve. This will almost always be 1. The ordinal distinguishes multiple sensors of the same element, of which ordinal 1 is the primary sensor, 2 is the secondary sensor (if there are multiple sensors), and so on.
heightDepth	**HeightDepth**	The height/depth for the element of the data to retrieve, or null if the element doesn't have a height/depth. For most elements, this should be null. This is usually only non-null for soil-related elements.
*beginDate	String	The earliest date to get data for (in format yyyy-MM-dd or MM/dd/yyyy)
*endDate	String	The latest date to get data for (in format yyyy-MM-dd or MM/dd/yyyy)
beginHour	int	The earliest hour to get data for (in format HH:mm:ss). Note: 00:00 assumed if nothing specified for mm:ss.
endHour	int	The latest hour to get data for (in format HH:mm:ss). Note: 00:00 assumed if nothing specified for mm:ss.

Example usage 1: Get hourly data for a specific time range over a date range

stationTriplets="302:OR:SNTL" // SNOTEL Station ANEROID LAKE #2 in Oregon

elementCd="PREC" // Accumulated Precipitation

ordinal=1

heightDepth=null

beginDate=2010-01-01

endDate=2010-01-31

beginHour=2

endHour=16

Example usage 2: Get hourly data for a specific hour over a date range

stationTriplets="302:OR:SNTL" // SNOTEL Station ANEROID LAKE #2 in Oregon

elementCd=”PREC” // Accumulated Precipitation

ordinal=1

heightDepth=null

beginDate=2010-01-01

endDate=2010-01-31

beginHour=00:00

endHour=00:00

Example usage 3: Get hourly data over a date range

stationTriplets=“302:OR:SNTL” // SNOTEL Station ANEROID LAKE #2 in Oregon

elementCd=”PREC” // Accumulated Precipitation

ordinal=1

heightDepth=null

beginDate=2010-01-01 22:00

endDate=2010-01-02 04:00

beginHour=2

endHour=16

Calling the **getData** method with these parameters will return hourly accumulated precipitation for SNOTEL station 302 - ANEROID LAKE #2 from 2 a.m. 01/01/2010 to 4 p.m. 01/31/2010, contiguous.

getInstantaneousData

Gets instantaneous SNOTEL or SCAN data for one or more stations for a single element for a range of dates. Most SNOTEL and SCAN stations report data at least every few hours (and some every hour), so use this to get SNOTEL or SCAN data that has a finer granularity than just daily.

A common usage of this method is to get the annual accumulated precipitation value. This value is stored in the database with a timestamp of yyyy-09-30 23:59 for each year. To do this, get the filter=InstantaneousDataFilter.FIRST_OF_DAY value with a begin date and end date of yyyy-09-30 23:59 for elementCd=’PREC’.

Returns: A list of **InstantaneousData** objects that contain the data values for each station requested. The data will be returned in the same order as the stations requested.

Parameters:

Parameter	Type	Description
*stationTriplets	List<String>	A list of the three-part station identifiers of the stations for which to get data.
*elementCd	String	The **element code** of the data to retrieve.
*ordinal	int	The ordinal for the element of the data to retrieve. This will almost always be 1. The ordinal distinguishes multiple sensors of the same element, of which ordinal 1 is the primary sensor, 2 is the secondary sensor (if there are multiple sensors), and so on.
heightDepth	**HeightDepth**	The height/depth for the element of the data to retrieve, or null if the element doesn't have a height/depth. For most elements, this should be null. This is usually only non-null for soil-related elements.
*beginDate	String	The earliest date to get data for (in format yyyy-MM-dd HH:mm or MM/dd/yyyy HH:mm)
*endDate	String	The latest date to get data for (in format yyyy-MM-dd HH:mm or MM/dd/yyyy HH:mm)
*filter	**InstantaneousDataFilter**	This determines which subset of data to retrieve. This can be ALL to get all values, FIRST_OF_DAY to get only the first value of each day, or MIDNIGHT_ONLY to get only the midnight value of each day requested
*unitSystem	**UnitSystem**	This indicates what units in which to return the data. Possible options are ENGLISH for English units, or LAST_COLLECTED to return the data in the units in which the sensor for this element most recently collected data.

Example usage for hourly SNOTEL data:

Using these parameters will retrieve hourly Snow Water Equivalent data for January 1, 2010 in English units (inches) for SNOTEL station ANEROID LAKE #2 (302).

stationTriplets="302:OR:SNTL"

elementCd="WTEQ"

ordinal=1

heightDepth=null

beginDate="2010-01-01"

endDate="2010-01-01 23:59"

filter=InstantaneousDataFilter.ALL

unitSystem=UnitSystem.ENGLISH

Example usage for annual precipitation value:

Using these parameters will retrieve the annual accumulated precipitation value for 2010 in English units (inches) for SNOTEL station ANEROID LAKE #2 (302).

stationTriplets="302:OR:SNTL"

elementCd="PREC"

ordinal=1

heightDepth=null

beginDate="2010-09-30 23:59"

endDate="2010-09-30 23:59"

filter=InstantaneousDataFilter.FIRST_OF_DAY

unitSystem=UnitSystem.ENGLISH

getInstantaneousDataInsertedOrUpdatedSince

Gets instantaneous SNOTEL or SCAN data for one or more stations for a single element for a range of dates, but only returns the values that have been inserted or updated since some date that the user specifies. Most SNOTEL and SCAN stations report data at least every few hours (and some every hour), so use this to get SNOTEL or SCAN data that has a finer granularity than just daily and where the user only wants to get the values that have been inserted or updated since some date the user passes in.

Returns: A list of **InstantaneousData** objects that contain the data values for each station requested. The data will be returned in the same order as the stations requested.

Parameters:

Parameter	Type	Description
*stationTriplets	List<String>	A list of the three-part station identifiers of the stations for which to get data.
*elementCd	String	The **element code** of the data to retrieve.
*ordinal	int	The ordinal for the element of the data to retrieve. This will almost always be 1. The ordinal distinguishes multiple sensors of the same element, of which ordinal 1 is the primary sensor, 2 is the secondary sensor (if there are multiple sensors), and so on.
heightDepth	**HeightDepth**	The height/depth for the element of the data to retrieve, or null if the element doesn't have a height/depth. For most elements, this should be null. This is usually only non-null for soil-related elements.
*beginDate	String	The earliest date to get data for (in format yyyy-MM-dd HH:mm or MM/dd/yyyy HH:mm)
*endDate	String	The latest date to get data for (in format yyyy-MM-dd HH:mm or MM/dd/yyyy HH:mm)
*insertOrUpdateBeginDate	String	The earliest insert or update date to get data for (in format yyyy-MM-dd HH:mm or MM/dd/yyyy HH:mm. The only values that will be returned will be values that have been inserted or updated on or after this date.
*filter	**InstantaneousDataFilter**	This determines which subset of data to retrieve. This can be ALL to get all values, FIRST_OF_DAY to get only the first value of each day, or MIDNIGHT_ONLY to get only the midnight value of each day requested
*unitSystem	**UnitSystem**	This indicates what units in which to return the data. Possible options are ENGLISH for English units, or LAST_COLLECTED to return the data in the units in which the sensor for this element most recently collected data.

Example usage for hourly SNOTEL data:

Using these parameters will retrieve hourly Snow Water Equivalent data for January 1, 2010 in English units (inches) for SNOTEL station ANEROID LAKE #2 (302).

stationTriplets="302:OR:SNTL"

elementCd="WTEQ"

ordinal=1

heightDepth=null

beginDate="2010-01-01"

endDate=”2010-01-01 23:59”

insertOrUpdateBeginDate=”2010-01-02”

filter=InstantaneousDataFilter.ALL

unitSystem=UnitSystem.ENGLISH

Example usage for annual precipitation value:

Using these parameters will retrieve the annual accumulated precipitation value for 2010 in English units (inches) for SNOTEL station ANEROID LAKE #2 (302).

stationTriplets="302:OR:SNTL"

elementCd=”PREC”

ordinal=1

heightDepth=null

beginDate=”2010-09-30 23:59”

endDate=”2010-09-30 23:59”

insertOrUpdateBeginDate=”2010-10-01”

filter=InstantaneousDataFilter.FIRST_OF_DAY

unitSystem=UnitSystem.ENGLISH

getPeakData

Gets the annual peak data value for each water year requested, for one or more stations, for a single element. The peak data value is the highest value in each water year (if there are multiple values that equal the highest value, the last one is returned).

Returns: A list of **PeakData** objects that contain the data values for each station requested. The data will be returned in the same order as the stations requested.

Parameters:

Parameter	Type	Description
*stationTriplets	List<String>	A list of the three-part station identifiers of the stations for which to get the peak values.
*elementCd	String	The **element code** of the data to retrieve.
*ordinal		The ordinal for the element of the data to retrieve. This will almost always be 1. The ordinal distinguishes multiple sensors of the same element, of which ordinal 1 is the primary sensor, 2 is the secondary sensor (if there are multiple sensors), and so on.
heightDepth	**HeightDepth**	The height/depth for the element of the data to retrieve, or null if the element doesn’t have a height/depth. For most elements, this should be null. This is usually only non-null for soil-related elements.
*duration	**Duration**	The duration of the data to retrieve (currently, this method only supports the DAILY duration).
*getFlags	boolean	If true, data flags will be retrieved and returned, otherwise the flags won’t be returned.
*beginYear	int	The first water year for which to get the peak data value.
*endYear	int	The last water year for which to get the peak data value.

Example usage:
The following parameters will retrieve the highest snow-water equivalent (WTEQ) value for each water year from 2000 to 2003 for SNOTEL station ANEROID LAKE #2 (302). There will be four values returned in total - one for each water year.

stationTriplets=”302:OR:SNTL”

elementCd=”WTEQ”

ordinal=1

heightDepth=null

duration=DAILY

getFlags=true

beginYear=2000

endYear=2003

30 Year Central Tendency Methods - Details

The three methods take a **CentralTendencyType,** an enumeration with three constants: AVERAGE, MEDIAN and NORMAL. Use the AVERAGE or MEDIAN central tendency type to retrieve the type of values you seek. To retrieve the default central tendency (*average* or *median*) values for a given station and element, use NORMAL. The methods are smart enough to know whether the default central tendency type is average or median, and will return the appropriate values. For example, if a SNOTEL or Snow Course station's element is Snow Water Equivalent, you will receive its default central tendency type as median values; and if a SNOTEL station's element is Precipitation, you will receive its default central tendency type as average values.

getCentralTendencyData

Gets the 30-year central tendency (*average* or *median*) value(s) for the current normals period, for one or more stations, for a single element, for a given duration, and for some range of dates. The method takes a **Duration** in order to retrieve DAILY, MONTHLY, SEMIMONTHLY, or ANNUAL central tendencies. (Note: For daily data, there is always a tendency value for February 29. If asking for data for January 1 to December 31, 366 values will be returned.)

Returns: A list of **Central Tendency Data** objects that contain the 30-year central tendency (*average* or *median*) values for each station requested. The requested central tendency values will be returned in the same order as the stations requested.

Parameters:

Parameter	Type	Description
*stationTriplets	List<String>	A list of the three-part station identifiers of the stations for which to get central tendencies.
*elementCd	String	The **element code** of the central tendencies to retrieve.
*heightDepth	**HeightDepth**	The height/depth for the element of the central tendencies to retrieve or null if the element doesn't have a height/depth. For most elements, this should be null. This is usually only non-null for soil-related elements.
*duration	**Duration**	The duration of the central tendencies to retrieve. This can be DAILY, MONTHLY, SEMIMONTHLY, or ANNUAL.
*getFlags	boolean	If true, flags associated with the central tendency will be retrieved and returned, otherwise the flags won't be returned.
*beginMonth	int	The first month of central tendencies to retrieve (1-12).
*beginDay	int	The first day of the first month for which to retrieve central tendencies (1-31).
*endMonth	int	The last month of central tendencies to retrieve (1-12).
*endDay	int	The last day of the last month for which to retrieve central tendencies (1-31).
*centralTendencyType	**CentralTendencyType**	The type of central tendency values to retrieve. This can be AVERAGE, MEDIAN or NORMAL; where NORMAL will be the default type (Average or Median) for the given station and element.

Example usage:
The following parameters can be used to get daily 30-year medians for SNOTEL station EMIGRANT SUMMIT (471) in Idaho for Snow Water Equivalent for days Feb 27-Mar 2. Setting the **CentralTendencyType** to "NORMAL" will return median values here, as the default central tendency for this station and element is median.

stationTriplets="471:ID:SNTL"

elementCd="WTEQ"

heightDepth=null

duration=Duration.DAILY

getFlags=true

beginMonth=2

beginDay=27

endMonth=3

endMonth=2

centralTendencyType="NORMAL"

getCentralTendencyPeakData

Gets the peak 30-year central tendency (*average* or *median*) value(s) for the current normals period, for one or more stations, for a single element, for a given duration, and for some range of dates. The method takes a **Duration** in order to retrieve DAILY, MONTHLY, SEMIMONTHLY, or ANNUAL central tendencies. The peak central tendency value is the highest value (if there are multiple values that equal the highest value, the last one is returned).

Returns: A list of **CentralTendencyPeakData** objects that contain the 30-year central tendency (*average* or *median*) peak values for each station requested. The requested central tendency (average or median) peak data values will be returned in the same order of the stations requested.

Parameters:

Parameter	Type	Description
*stationTriplets	List<String>	A list of the three-part station identifiers of the stations for which to get peak central tendencies.
*elementCd	String	The **element code** of the peak central tendencies to retrieve.
heightDepth	**HeightDepth**	The height/depth for the element of the peak central tendencies to retrieve, or null if the element doesn't have a height/depth. For most elements, this should be null. This is usually only non-null for soil-related elements.
*duration	**Duration**	The duration of the peak central tendencies to retrieve. This can be DAILY, MONTHLY, SEMIMONTHLY, or ANNUAL.
*getFlags	boolean	If true, flags associated with the peak central tendency will be retrieved and returned, otherwise the flags won't be returned.
*centralTendencyType	**CentralTendencyType**	The type of central tendency values to retrieve. This can be AVERAGE, MEDIAN or NORMAL, where NORMAL will be the default type (Average or Median) for the given station and element.

Example usage:
The following parameters can be used to request the peak daily 30-year median for snow-water equivalent (WTEQ) for SNOTEL station ANEROID LAKE #2 (302).

stationTriplets="302:OR:SNTL"

elementCd="WTEQ"

heightDepth=null

duration=Duration.DAILY

getFlags=false

centralTendencyType="MEDIAN"

getForecastPeriodCentralTendency

Gets the 30-year forecast period central tendency (*average* or *median*) value(s) for the current normals period, for one or more stations, for a given element, and for one or more forecast periods.

Returns: A list of ForecastPeriodCentralTendency objects that contain the 30-year forecast period central tendencies for each station and period requested. The requested central tendency (average or median) values will be returned in the same order as the stations requested.

Parameters:

Parameter	Type	Description
*stationTriplets	List<String>	A list of the three-part station identifiers of the stations for which to get the forecast period averages.
*elementCd	String	The **element code** of the forecast period averages to retrieve.
*periods	List<String>	A list of the forecast periods for which to get period averages. This will be a list of Strings that are the names of the periods (such as JAN-MAR, or APR-JUL).
*centralTendencyType	**CentralTendencyType**	The type of central tendency values to retrieve. This can be AVERAGE, MEDIAN or NORMAL, where NORMAL will be the default type (Average or Median) for the given station and element.

Example usage:
The following set of parameters can be used to request forecast period averages for USGS station BLUE LAKE INFLOW (14162200) in Oregon for the forecast periods April-May, May-July, and June-July.

stationTriplets=”14162200:OR:USGS”

elementCd=”SRVO”

periods[0]="APR-MAY"

periods[1]="MAY-JUL"

periods[2]="JUN-JUL"

centralTendencyType=”AVERAGE”

Metadata Methods - Details

getElement

Retrieves metadata about a single element given an element code. The method returns an **element** object which contains the element code, the name of the element, and the stored unit code (the units that the data was stored in and will be returned in unless otherwise specified).

Returns: An element object that contains the metadata about the element code passed in.

Parameters:

Parameter	Type	Description
*elementCd	String	The **element code** about which to get additional metadata

Example Usage:

elementCd=’WTEQ’

getElements

Retrieves a list of all elements in the AWDB database. Each item in the returned list will be an Element object and will contain the same information described in the **getElement** method. The elements will be ordered alphabetically by element code.

Returns: A list of Element objects (one for each element defined in the AWDB database).

Parameters: None

getHeightDepths

Gets a list of all possible height/depths that are defined in AWDB (returned as a list of **HeightDepth** objects). Each **HeightDepth** object consists of a value (positive for a height, and negative for a depth) along with the unit code of the value (usually "in" for inches).

Returns: A list of **HeightDepth** objects that contain the information about all the height/depths defined in the AWDB database.

Parameters: None

getUnitName

Retrieves the (plural) name of a unit given a unit code. Calling this method with "in" for example returns "inches".

Returns: The (plural) name of the unit for the unit code passed in.

Parameters:

Parameter	Type	Description
*unitCode	String	The unit code of the unit for which to get the name.

Example usage:

unitCode="in" returns "inches"

getUnits

Retrieves a list of all of the units defined in the AWDB database. Each item in the returned list will be a **unit** object that contains the unit code and the name (plural name) of the unit.

Returns: A list of **unit** objects - one for each unit defined in the database.

Parameters: None

Forecast Methods - Details

getForecast

Gets a water supply forecast value for a given forecast point, element, forecast period, and publication date.

Returns: A **Forecast** object that contains the forecast values (for all probabilities) for the requested station, element, forecast period, and publication date.

Parameters:

Parameter	Type	Description
*stationTriplets	String	The three-part station identifier of the station for which to get the forecast.
*elementCd	String	The **element code** of the forecast to retrieve.
*forecastPeriod	String	The forecast period for which to get the forecast. This will be a String that is the name of the period (such as JAN-MAR, or APR-JUL) for which to get the forecast.
*publicationDate	String	The publication date of the forecast to retrieve. This must be a String in the format yyyy-MM-dd or MM/dd/yyyy.

Example usage:
Using the following parameters will retrieve the forecast values for APR-JUL that were published in April 2011 for the forecast point “Applegate River Nr Copper” (14362000) in Oregon.

stationTriplet=”14362000:OR:USGS”

elementCd=”SRVO”

forecastPeriod=”APR-JUL”

publicationDate=”2011-04-01”

getForecastPeriods

Gets a list of all forecast periods defined in the AWDB database.

Returns: A list of **ForecastPeriod** objects that represents all forecast periods defined in the AWDB database.

Parameters: None

getForecastPoint

Gets the forecast point metadata for a given station. The forecast point metadata includes the name of the forecast point (which may be different from the name of the actual station), theÂ exceedenceprobabilities that will be calculated for forecasts for this point, and the name of the forecaster responsible for producing forecasts for this point.

Returns: A **ForecastPoint** object that contains the forecast point metadata for a given station.

Parameters:

Parameter	Type	Description
*stationTriplet	String	The three-part station identifier of the station for which to get the forecast point metadata.

Example usage:

Using the following parameter will retrieve the forecast point metadata for USGS station “Applegate River Nr Copper” (14362000) in Oregon.

stationTriplet=”14362000:OR:USGS”

getForecastPoints

Finds one or more forecast points using search criteria such as station ids, state codes, network codes, forecast point names, HUCs, or name of responsible forecaster. Any of the parameters that take a list can be null if you don't want to search by that criteria. If any of the parameters that take a list have more than one item, the search will return forecast points that match any of those items.

Returns: A list of **ForecastPoint** objects for all the forecast points that match the search criteria used.

Parameters:

Parameter	Type	Description
stationIds	List<String>	A list of the station ids of the forecast points for which to search. The station ids can contain the wildcard characters '*' or '?'
stateCds	List<String>	A list of the two-character FIPS state postal codes of the states of the forecast points to find.
networkCds	List<String>	A list of the **of the forecast points for which to search.**
forecastPointNames	List<String>	A list of the names of the forecast points for which to search. The names can contain the '*' or '?' wildcard characters.
hucs	List<String>	A list of the Hydrologic Unit Codes (HUC) of the forecast points for which to search. The HUCs can contain the wildcard characters '*' or '?'. Always include the '*' wildcard at the end of the HUC (otherwise you'll have to match the exact number of digits of the HUC for the forecast point as defined in the AWDB database - which is currently 8-digits, but will change to 12-digits in the future).
forecasters	List<String>	A list of user names of forecasters that are responsible for producing the forecasts of the forecast points for which to search. User names are usually the first-name initial followed by the last name of the forecaster (all lowercase).
*logicalAnd	boolean	If true, the forecast points that are returned must match all of the search criteria passed in. If false, the forecast points that are returned must match at least one of the search criteria passed in.

Example usage:
The following parameters can be used to find all forecast points in Oregon or Washington that are USGS stations and where the forecast point name begins with "CL".

stateCds="OR", "WA"

networkCds="USGS"

forecastPointNames="CL*"

logicalAnd=true

getForecasts

Gets all the forecasts for a given station, element, and forecast period (for all publication dates).

Returns: A list of **Forecast** objects for all the forecasts requested.

Parameters:

Parameter	Type	Description
*stationTriplet	String	The three-part station identifier of the station for which to get the forecasts.
*elementCd	String	The **element code** of the forecasts to retrieve.
*forecastPeriod	String	The forecast period to get the forecasts for. This will be a string that is the name of the period (such as JAN-MAR or APR-JUL) for which to get the forecast.

Example usage:
The following parameters will retrieve all forecasts for APR-JUL for the forecast point “Applegate River Nr Copper” (14362000) in Oregon, for all publication dates.

stationTriplet=”14362000:OR:USGS”

elementCd=”SRVO”

forecastPeriod=”APR-JUL”

getForecastsByPubDate

Gets a list of forecasts for a given station, element, forecast period, and whose publication dates are between a given begin and end date.

Returns: A list of **Forecast** objects for all the forecasts requested.

Parameters:

Parameter	Type	Description
*stationTriplet	String	The three-part station identifier of the station for which to get the forecasts.
*elementCd	String	The **element code** of the forecasts to retrieve.
*forecastPeriod	String	The forecast period to get the forecasts for. This will be a string that is the name of the period (such as JAN-MAR or APR-JUL) for which to get the forecast.
*beginPublicationDate	String	The earliest publication date of the forecasts to retrieve. This must be a string in the format yyyy-MM-dd or MM/dd/yyyy.
*endPublicationDate	String	The latest publication date of the forecasts to retrieve. This must be a string in the format yyyy-MM-dd or MM/dd/yyyy.

Example usage:
Using the following parameters will retrieve the forecasts for APR-JUL that were published between April 2011 and April 2012 for the forecast point “Applegate River Nr Copper” (14362000) in Oregon.

stationTriplet=”14362000:OR:USGS”

elementCd=”SRVO”

forecastPeriod=”APR-JUL”

beginPublicationDate=”2011-04-01”

endPublicationDate=”2012-04-01”

getForecastValue

Gets a single forecast value (with no additional metadata) for a given station, element, forecast period, exceedence probability, and publication date.

Returns: A BigDecimal that represents a single forecast value (in thousands of acre-feet) for the requested forecast point, period, probability and publication date.

Parameters:

Parameter	Type	Description
*stationTriplet	String	The three-part station identifier of the station for which to get the forecast.
*elementCd	String	The **element code** of the forecast to retrieve.
*forecastPeriod	String	The forecast period for which to get the forecast. This will be a String that is the name of the period (such as JAN-MAR or APR-JUL) for which to get the forecast.
*probability	int	The exceedence probability of the forecast to retrieve (a number between 0 and 100). This is usually one of 5, 10, 30, 50, 70, 90, or 95.
*publicationYear	int	The year of the publication date of the requested forecast.
*publicationMonth	int	The month of the publication date of the requested forecast (1-12).
*publicationDay	int	The day of month of the publication date of the requested forecast (1-31).

Example usage:
Using the following parameters below retrieve the forecast value (in thousands of acre-feet) for APR-JUL for the forecast point “Applegate River Nr Copper” (14362000) in Oregon that was published April 1, 2011.

stationTriplet=”14362000:OR:USGS”

elementCd=”SRVO”

forecastPeriod=”APR-JUL”

publicationYear=2011

publicationMonth=4

publicationDay=1

CentralTendencyData

Attribute	Type	Description
beginDay	int	The day of month of the first central tendency value in the values list (1-31).
beginMonth	int	The month of the first central tendency value in the values list (1-12).
dataSetFlag	String	The flag assigned to the set of 30-year central tendencies returned.
duration	**Duration**	The duration of the 30-year central tendencies in the values list.
endDay	int	The day of month of the last central tendency value in the values list (1-31).
endMonth	int	The month of the last central tendency value in the values list (1-12).
flags	List<String>	The flags associated with each of the 30-year central tendency values. There will be one entry for each value in the list.
values	List<BigDecimal>	The 30-year central tendency values.
centralTendencyType	**CentralTendencyType**	An enumeration with three constants: AVERAGE, MEDIAN and NORMAL. The AVERAGE and MEDIAN are the calculated value types. NORMAL is the default central tendency (*average* or *median*) for a specific station and element.

CentralTendencyPeakData

Attribute	Type	Description
duration	**Duration**	The duration of the peak 30-year central tendency value.
flag	String	The flag associated with the peak 30-year central tendency value.
peakDay	int	The day of month of the peak 30-year central tendency value (1-31).
peakMonth	int	The month of the peak 30-year central tendency values (1-12).
value	BigDecimal	The peak 30-year central tendency value.
centralTendencyType	**CentralTendencyType**	An enumeration with three constants: AVERAGE, MEDIAN and NORMAL. The AVERAGE and MEDIAN are the calculated value types. NORMAL is the default central tendency (*average* or *median*) for a specific station and element.

CentralTendencyType (Enum)

Value	Description
AVERAGE	Used to request average values.
MEDIAN	Used to request median values.
NORMAL	Used to request the default central tendency (average or median) values for a given station and element, whichever is their default central tendency type.

Data

Attribute	Type	Description
beginDate	String	The date of the first data value returned (in format yyyy-MM-dd).
duration	**Duration**	The duration of the data values.
endDate	String	The date of the last data value returned (in format yyyy-MM-dd).
flags	List<String>	The flags for each of the data values (if requested).
values	List<BigDecimal>	The actual data values.

DataSource (Enum)

Value	Description
OBSERVED	Indicates that data was measured/collected directly by the station.
DERIVED	Indicates that data was derived from another data source (usually another duration of the same element)
INTERPRETED	Indicates that the data was interpreted from another data source.

Duration (Enum)

Value	Description
ANNUAL	Used to request water year precipitation data and normals.
DAILY	Used to request daily data and normals.
INSTANTANEOUS	*Deprecated.* The getStationElements method may return this for a SNOTEL/SCAN station that has instantaneous data.
MONTHLY	Used to request monthly data and normals.
SEMIMONTHLY	Used to request semi-monthly data and normals.
SEASONAL	Used to request water supply forecast values or seasonal streamflow normals.

Element

Attribute	Type	Description
elementCd	String	The **element code** of the element.
name	String	The name of the element.
storedUnitCd	String	The units that the data for this element are stored in. This will be the units of the data returned by the web service unless otherwise specified.

Forecast

Attribute	Type	Description
calculationDate	String	The date and time that the forecast values were calculated. In the formatyyyy-MM-dd HH:mm:ss.
elementCd	String	The **element code** of the forecast values.
exceedenceProbabilities	List<Integer>	The exceedence probabilities that were calculated (these will usually be [5, 30, 50, 70, 95] or [10, 30, 50, 70, 90].
exceedenceValues	List<BigDecimal>	The calculated forecast values for each of the exceedence probabilities for this forecast.
forecastPeriod	String	The forecast period for this forecast.
periodAverage	BigDecimal	The 30-year seasonal period average for the forecast period that this forecast.
publicationDate	String	The publication date of this forecast.
stationTriplet	String	The three-part station identifier that identifies the station for this forecast.
unitCd	String	The unit code that indicates the units of the calculated forecast values and the period average.

ForecastPeriod

Attribute	Type	Description
description	String	A description of the forecast period (if any).
forecastPeriod	String	The forecast period as a string. This will be something like APR-JUL (for April to July).

ForecastPeriodCentralTendency

Attribute	Type	Description
flag	String	The flag associated with the forecast period average value.
forecastPeriod	String	The forecast period as a string. This will be something like APR-JUL (for April to July).
stationTriplet	String	The three-part station identifier that identifies the station for which this forecast period average is.
value	BigDecimal	The forecast period average value.
centralTendencyType	**CentralTendencyType**	An enumeration with three constants: AVERAGE, MEDIAN and NORMAL. The AVERAGE and MEDIAN are the calculated value types. NORMAL is the default central tendency (*average* or *median*) for a specific station and element.

Forecast Point

Attribute	Type	Description
exceedenceProbabilities	List<Integer>	A list of the forecast exceedence probabilities that are to be calculated when forecasts for this forecast point are calculated. Each of these will be a number between 5 and 95. Usually a group of five probabilities that are [5, 30, 50, 70, 95] or [10, 30, 50, 70, 90].
name	String	The name of the forecast point.
responsibleForecaster	String	An abbreviated name of the NWCC forecaster responsible for producing the forecasts for this forecast point.
stationTriplet	String	The three-part station identifier that identifies the station to which this forecast point corresponds.

HeightDepth

Attribute	Type	Description
unitCd	String	The unit code of the height/depth value. This is usually “in” for inches or “mm” for millimeters.
value	String	The height or depth value. Depths will be negative; heights will be positive.

Example: If you want a soil moisture element at a depth of two inches, the unitCd would be “in” and the value would be -2.

Note: When using HeightDepth objects, both unitCd and value must be set. Use *null* in place of a HeightDepth object if an element doesn’t have a height or depth (the HeigthDepth is typically only used for soil-related elements).

HourlyData

Attribute	Type	Description
beginDate	String	The date of the first data value returned (in format yyyy-MM-dd).
endDate	String	The date of the last data value returned (in format yyyy-MM-dd).
stationTriplet	String	The three-part station identifier that identifies the station for which the hourly data values are for.
values	List<**HourlyDataValue**>	A list of **HourlyDataValue** objects.

HourlyDataValue

Attribute	Type	Description
value	BigDecimal	The actual data value.
flag	String	The flag associated with the data value.
dateTime	String	The date and time of the data value (in format yyyy-MM-dd HH:mm).

InstantaneousData

Attribute	Type	Description
beginDate	String	The date of the first instantaneous data value returned (in format yyyy-MM-dd).
endDate	String	The date of the last instantaneous data value returned (in format yyyy-MM-dd).
stationId	int	The SNOTEL/SCAN station whose data this object represents.
unitCd	String	The unit code of the units of the data values contained in this object.
values	List<InstantaneousDataValue>	A list of the instantaneous data values that were requested.

InstantaneousDataFilter (Enum)

Value	Description
ALL	Used to request all instantaneous data values (no filtering).
FIRST_OF_DAY	Used to request only the first value of each day.
MIDNIGHT_ONLY	Used to request only the midnight value of each day.

InstantaneousDataValue

Attribute	Type	Description
flag	String	The flag associated with the data value. This will be 'V' for valid, 'S' for suspect, or 'E' for edited.
time	String	The date and time of the data value (in format yyyy-MM-dd HH:mm).
value	BigDecimal	The instantaneous data value.

PeakData

Attribute	Type	Description
beginYear	int	The water year of the first peak data value in the list of values and flags.
duration	Duration	The duration of the peak data values.
endYear	int	The water year of the last peak data value in the list of values and flags.
flags	List<String>	The data flags associated with the peak data values.
peakDays	List<Integer>	The days of months of each of the peak data values.
peakMonths	List<Integer>	The month of each of the peak data values.
values	List<BigDecimal>	The peak data values for each water year requested.

ReservoirMetadata

Attribute	Type	Description
elevationAtCapacity	BigDecimal	The elevation (in feet) of the reservoir when the reservoir is filled to capacity.
reservoirCapacity	BigDecimal	The capacity of the reservoir (in acre-feet).
stationTriplet	String	The three-part station identifier that identifies the reservoir station.
usableCapacity	BigDecimal	The usable capacity of the reservoir (in acre-feet).

StationElement

Attribute	Type	Description
beginDate	String	The date that the station started collecting data for this element (in yyyy-MM-dd HH:mm:ss format).

dataPrecision	Integer	The precision for which the data for this element is accurate (it is also the precision that the data/normals will be returned in).
dataSource	**DataSource**	The data source of the element.
duration	**Duration**	The duration of the element.
elementCd	String	The **element code** of the element.
endDate	String	The date that the station stopped collecting data for this element (in yyyy-MM-dd HH:mm:ss format). If the station is still collecting data for this element, this will be set to 2100-01-01 00:00:00.
heightDepth	HeightDepth	Indicates the height or depth of the sensor collecting data for the element. This is usually only used for soil sensors and will be *null* for everything else.
ordinal	int	The ordinal of the station element. The ordinal is a number that indicates whether the element is from the primary sensor, the secondary sensor, and so on (if there are multiple sensors collecting data for the same element). This will typically be 1.
originalUnitCd	String	The unit code of the units that the data for this element were collected in.
stationTriplet	String	The three-part station identifier that identifies the station to which the station element belongs.
storedUnitCd	String	The unit code of the units that the data for this element were stored in (this is, units that the web service will return data in unless otherwise specified)

StationMetadata

Attribute	Type	Description
actonId	String	The “acton” id of a station. This is only used for SNOTEL stations and has also been known as the “CDBS” id. (For example, for station ‘302’, the acton id is ‘17D02S’).
beginDate	String	The date (yyyy-MM-dd HH:mm:ss) that the station was installed.
countyName		The name of the county in which the station is located.
elevation	BigDecimal	The elevation (in feet) of the station.
endDate	String	The date (yyyy-MM-dd HH:mm:ss) that the station was discontinued. If the station is still active, the end date will be set to 2100-01-01 00:00:00.
fipsCountyCd	String	The FIPS county code of the county (and state) in which the station is located.
fipsCountryCd	String	The FIPS country code of the country in which the station is located.
fipsStateNumber	String	The FIPS state number of the state in which the station is located.
huc	String	The 12-digit HUC (based on the Watershed Boundary Dataset) in which the station is located.
hud	String	The 8-digit HUC (based on the Hydrologic Unit Dataset) in which the station is located.
latitude	BigDecimal	The station latitude.
longitude	BigDecimal	The station longitude.
name	String	The station name.
shefId	String	The id of the station that is used when sending data to the National Weather Service via the SHEF system.
stationDataTimeZone	BigDecimal	The time zone in which the station collects data.
stationTimeZone	BigDecimal	The time zone of the actual location of the station (*Note: This is currently set to the same value as thestationDataTimeZone*).
stationTriplet	String	The three-part identifier of the station, in the format stationId:stateCode:networkCode.

Unit

Attribute	Type	Description
name	String	The plural name of the unit.

unitCd	String	The unit code of the unit.

UnitSystem (Enum)

Value	Description
ENGLISH	Used to request instantaneous data in English units (e.g., precipitation in inches, temperature in Fahrenheit, etc)
LAST_COLLECTED	Used to request data in the units that the data was most recently collected in (e.g., if a station was collecting observed temperature in Fahrenheit, but later switched to Celsius, then if the data is requested in LAST_COLLECTED units, all of the data will be returned in Celsius units).

Network Codes

Three pieces of information are needed to identify a station uniquely:

1. **Station id.**
2. **State code.** The state code is the two-character FIPS alphabetic code. For example, the state code for Oregon is OR.
3. **Network code.** The network code usually indicates the source of the data, but is primarily used to avoid clashes in station ids, by grouping ids that come from certain sources together.

AWDB Network Codes

The network codes used by the AWDB database are in the following table.

Network Code	Description
BOR	Any Bureau of Reclamation reservoir stations plus other non-BOR reservoir stations
CLMIND	Used to store climate indices (such as Southern Oscillation Index or Trans-Nino Index)
COOP	National Weather Service COOP stations
MPRC	Manual precipitation sites
MSNT	Manual SNOTEL non-telemetered, non-real-time sites
SNOW	NRCS Snow Course Sites
SNTL	NWCC SNOTEL and SCAN stations
USGS	Any USGS station, but also other non-USGS streamflow stations

Element Codes

When you request data from the AWDB Web Service, you need to specify the element code for the data that you are interested in. The following table lists the element codes available and the units in which the data are stored.

Commonly-Used Element Codes

Name	Element Code	Unit
AIR TEMPERATURE AVERAGE	TAVG	Fahrenheit
AIR TEMPERATURE MAXIMUM	TMAX	Fahrenheit
AIR TEMPERATURE MINIMUM	TMIN	Fahrenheit
AIR TEMPERATURE OBSERVED	TOBS	Fahrenheit

DIVERSION FLOW VOLUME OBSERVED	DIV	acre-feet
DIVERSION DISCHARGE OBSERVED MEAN	DIVD	cfs
DISCHARGE MANUAL/EXTERNAL ADJUSTED MEAN	SRDOX	cfs
PRECIPITATION ACCUMULATION	PREC	inches
PRECIPITATION INCREMENT	PRCP	inches
PRECIPITATION INCREMENT â€“ SNOW-ADJUSTED	PRCPSA	inches
RESERVOIR STORAGE VOLUME	RESC	acre-feet
RIVER DISCHARGE OBSERVED MEAN	SRDOO	cfs
SNOW DEPTH	SNWD	inches
SNOW WATER EQUIVALENT	WTEQ	inches
STREAM VOLUME, ADJUSTED	SRVO	acre-feet
STREAM VOLUME, ADJUSTED EXTERNALLY	SRVOX	acre-feet
STREAM VOLUME, OBSERVED	SRVO	acre-feet
TELECONNECTION INDEX (also known as OSCILLATION INDEX)	OI	N/A

Other Element Codes

Name	Element Code	Unit
BAROMETRIC PRESSURE	PRES	inches of Mercury
BATTERY-ETI PRECIP GAGE	ETIB	volt
CONDUCTIVITY	COND	umho
DEW POINT TEMPERATURE	DPTP	Fahrenheit
DIAGNOSTICS	DIAG	unitless
DISSOLVED OXYGEN	DISO	mgram/l
DISSOLVED OXYGEN - PERCENT SATURATION	DISP	pct
ENERGY GAIN OR LOSS FROM GROUND	HFTV	watt/m2
EVAPORATION	EVAP	inches
FUEL MOISTURE	FUEL	pct
FUEL TEMPERATURE INTERNAL	FMTMP	Fahrenheit
GROUND SURFACE INTERFACE TEMPERATURE AVERAGE	TGSV	Fahrenheit
GROUND SURFACE INTERFACE TEMPERATURE MAXIMUM	TGSX	Fahrenheit
GROUND SURFACE INTERFACE TEMPERATURE MINIMUM	TGSN	Fahrenheit
GROUND SURFACE INTERFACE TEMPERATURE OBSERVED	TGSI	Fahrenheit
JULIAN DATE	JDAY	julian_day
NET SOLAR RADIATION AVERAGE	NTRDV	watt/m2
NET SOLAR RADIATION MAXIMUM	NTRDX	watt/m2
NET SOLAR RADIATION MINIMUM	NTRDN	watt/m2
NET SOLAR RADIATION OBSERVED	NTRDC	watt/m2
PH	H2OPH	unitless
PHOTOSYNTHETICALLY ACTIVE RADIATION (PAR) AVERAGE	PARV	micromole/m2/s
PULSE LINE MONITOR-ETI GUAGE	ETIL	volt
REAL DIELECTRIC CONSTANT	RDC	unitless
RELATIVE HUMIDITY	RHUM	pct
RELATIVE HUMIDITY AVERAGE	RHUMV	pct
RELATIVE HUMIDITY ENCLOSURE	RHENC	pct
RELATIVE HUMIDITY MAXIMUM	RHUMX	pct
RELATIVE HUMIDITY MINIMUM	RHUMN	pct

RESERVOIR STAGE	REST	feet
SALINITY	SAL	gram/l
SNOW DEPTH AVERAGE	SNWDV	inches
SNOW DEPTH MAXIMUM	SNWDX	inches
SNOW DEPTH MINIMUM	SNWDN	inches
SNOW FALL	SNOW	inches
SNOW WATER EQUIVALENT AVERAGE	WTEQV	inches
SNOW WATER EQUIVALENT MAXIMUM	WTEQX	inches
SNOW WATER EQUIVALENT min	WTEQN	inches
SOIL MOISTURE BARS AVERAGE	SMV	inches of Mercury
SOIL MOISTURE BARS MAXIMUM	SMX	inches of Mercury
SOIL MOISTURE BARS MINIMUM	SMN	inches of Mercury
SOIL MOISTURE BARS OBSERVED	SMO	inches of Mercury
SOIL MOISTURE PERCENT	SMS	pct
SOIL TEMPERATURE AVERAGE	STV	Fahrenheit
SOIL TEMPERATURE MAXIMUM	STX	Fahrenheit
SOIL TEMPERATURE MINIMUM	STN	Fahrenheit
SOIL TEMPERATURE OBSERVED	STO	Fahrenheit
SOLAR RADIATION	SRAD	watt/m2
SOLAR RADIATION AVERAGE	SRADV	watt/m2
SOLAR RADIATION MAXIMUM	SRADX	watt/m2
SOLAR RADIATION MINIMUM	SRADN	watt/m2
SOLAR RADIATION TOTAL	SRADT	watt/m2
SOLAR RADIATION/LANGLEY	LRAD	langley
SOLAR RADIATION/LANGLEY MAXIMUM	LRADX	langley
SOLAR RADIATION/LANGLEY TOTAL	LRADT	langley
STREAM STAGE (GAUGE HEIGHT) AVERAGE	SRMV	feet
STREAM STAGE (GAUGE HEIGHT) MAXIMUM	SRMX	feet
STREAM STAGE (GAUGE HEIGHT) MINIMUM	SRMN	feet
STREAM STAGE (GAUGE HEIGHT) OBSERVED	SRMO	feet
STRM FLOW AVERAGE	STRV	cfs
TURBIDITY	TURB	ntu
USABLE LAKE STORAGE VOLUME	RESA	acre-feet
VAPOR PRESSURE - PARTIAL	PVPV	kPa
VAPOR PRESSURE - SATURATED	SVPV	kPa
WATER LEVEL AVERAGE	WLEVV	inches
WATER LEVEL MAXIMUM	WLEVX	inches
WATER LEVEL MINIMUM	WLEVN	inches
WATER LEVEL OBSERVED	WLEV	inches
WATER TEMPERATURE	WTEMP	Fahrenheit
WATER TEMPERATURE AVERAGE	WTAVG	Fahrenheit
WATER TEMPERATURE MAXIMUM	WTMAX	Fahrenheit
WATER TEMPERATURE MINIMUM	WTMIN	Fahrenheit
WELL DEPTH	WELL	feet
WIND DIRECTION AVERAGE	WDIRV	degree
WIND DIRECTION OBSERVED	WDIR	degree
WIND MOVEMENT AVERAGE	WDMVV	mile

WIND MOVEMENT MAXIMUM	WDMVX	mile
WIND MOVEMENT MINIMUM	WDMVN	mile
WIND MOVEMENT OBSERVED	WDMV	mile
WIND MOVEMENT TOTAL	WDMVT	mile
WIND SPEED AVERAGE	WSPDV	mph
WIND SPEED MAXIMUM	WSPDX	mph
WIND SPEED MINIMUM	WSPDN	mph
WIND SPEED OBSERVED	WSPD	mph

Unsupported Methods

These methods are used by NWCC applications, and are not supported for public use.

getAllForecastsForStation

getForecastConfigurations

getForecastEquations

getForecastEquationsMultiple

getStationDataAssuredFlags

Announcements and Release Notes

Contains important information associated with updates and changes to the AWDB Web Service. It also contains Release Notes describing recent enhancements and defect fixes.

- **October 1, 2021:** The following changes apply to the getCentralTendency method:
  - The getFlags parameter will be replaced with getNumberObservationsUsed
  - The dataSetFlag will be replaced with dataset_NumberObservationsUsed
  - The getForecastPeriodAverages and getForecastPeriodCentralTendency methods will now have the getNumberObservationsUsedParameter
- **January 25, 2017:** USDA moves all websites to HTTP Secure. The National Water and Climate Center (NWCC) is part of the United States Department of Agriculture (USDA) Natural Resources Conservation Service (NRCS). Over the last several weeks, all USDA websites (including the NWCC website) have moved from Hyper Text Transfer Protocol (HTTP) to Hyper Text Transfer Protocol Secure (HTTPS) for internet data transfer. The move to HTTPS ensures that all communications between the user's browser and USDA websites are encrypted and secure. Web browsers such as Internet Explorer, Firefox, and Chrome now display a padlock icon in the address bar to visually indicate an HTTPS connection.

If users are experiencing problems with bookmarks, saved URLs, web service calls, or other functions, it may be related to the change from HTTP to HTTPS. If there's a reference or bookmark to **http://www.wcc.nrcs.usda.gov**, try changing it to **https://www.wcc.nrcs.usda.gov**.

- **Attention Users Connecting to the AWDB Web Service from a .NET Application:** If you are attempting to connect to the AWDB Web Service from a .NET client, you may receive an HTTP 502 (Bad Gateway) error. This is because, by default, .NET applications include the "HTTP" header "Expect: 100-continue". This header causes the AWDB Web Service to return the HTTP 502 (Bad Gateway) error. This can be resolved by telling the .NET application to **not** include this HTTP header when making requests to the web service. Adding the following line will prevent the header from being included in requests to the AWDB Web Service. **Note:** Ensure that this is set **before** any calls are made to the web service.

System.Net.ServicePointManager.Expect100Continue = false;

- On Thursday, **March 31, 2016**, the Air-Water Database (AWDB) Web Service moved from the Portland, OR, data center to a new server at the National Information Technical Center in Kansas City, MO. Most users of the AWDB Web Service are being automatically redirected from the Portland server to the Kansas City server and **no action is necessary**.

However, some users whose systems do not support redirected urls may not see this change. These users will need to change the WSDL (Web Service Definition Language) url from the old location to the new WSDL url location. For more information on changing the url to the AWDB Web Service, click here.

- **January 14, 2015** - The AWDB Web Service has been updated to version 2.18.0. This release contains the addition of two attributes to the Data object that is returned by the getData method. The first attribute is the stationTriplet. This identifies the station that each Data object which is in the array returned by the getData method is for. The second attribute is an array of collectionDates that will contain the date of collection for each value which is returned when a user asks for SEMIMONTHLY data.

If you use the Visual Basic API, any Excel spreadsheet or other application that uses the API will continue to work, and you will not have to make any changes to your code. If you want to use the stationTriplet or the collectionDates returned by the getData method, you can get the updated Visual Basic API here. The zip file AwdbWebServiceVBClasses-1.9.10.zip contains the latest version of the API that supports retrieval of the stationTriplet and the collectionDates from the getData method.

If your application does not call the getData method of the AWDB Web Service you also are not affected by the changes and you do not have to do anything to your application(s).

If you have generated "stubs" from the WSDL for the AWDB Web Service to have your application communicate with the AWDB Web Service, you will need to re-generate the "stubs" and rebuild your application. To prevent any downtime for your application, you can rebuild your application in advance of January 14 by generating new "stubs" using the new WSDL file.

If you have any questions about these changes, please contact **ServiceNow**.

Release Notes

Date	Version	Description

Sept. 13, 2016	2.25.0	Added new methods: **getDataInsertedOrUpdatedSince** and **getInstantaneousDataInsertedOrUpdatedSince**.
Jan. 15, 2015	2.18.0	1. Removed the dependence on the SNOTEL Web Service for the getInstantaneousData call (this may result in a slight performance improvement also). 2. Modified the getStationMetadata method so that it doesn't issue an error when some of the metadata values are null (such as county or timezone). 3. Modified the getData method to return the stationTriplet in the return object for each station included in the results. 4. Modified the getData method to return collectionDates when the duration is 'SEMIMONTHLY' if collection dates exist for the data being requested.
Oct. 23, 2013	2.13.0	Fixed a defect in the getStations() method where using wildcards in the stationId parameter was not working correctly. Also, the getStationMetadata() and getStationMetadataMultiple() methods now return the precise latitude and longitude for all stations.
Aug. 7, 2013	2.12.1	Fixed a defect in the getInstantaneousData method where it was not returning an empty slot when a station did not have data for the requested element. This was causing data to be misinterpreted by applications when data for multiple stations was being requested. If someone had requested data for two stations and the first station didn't have any data for the requested element, nothing was being returned for the first station, so applications would assume that the data returned was for the first station requested and there was no data for the second station. Fixed a problem in the getHourlyData method where it would always retrieve ordinal 1 data regardless of the ordinal passed in. Also modified getInstantaneousData so that it will work to retrieve ordinal 2 or greater data even if there is only an ordinal 1 station element that exists.
July 24, 2013	2.11.0	Modified the getData and getInstanteousData methods so that network codes are not hard-coded and instead are table-driven. This will allow changing the network codes of stations (such as splitting SNTL into SNTL and SCAN) in the database. Also added a timestamp on the status page.
May 9, 2013	2.10.0	The getData method now has an additional parameter called 'alwaysReturnDailyFeb29'. If 'true' is passed in for this parameter (or null), requests for daily data will always return a slot for February 29 (regardless of whether the year is a leap year or not. For leap years, the actual February 29 value will be returned; for non-leap years, null will be returned). If 'false' is passed in, requests for daily data will only return a slot for February 29 for leap years. The methods that return central tendencies (and peak central tendencies) now return values rounded to the precision of the sensor that collected the data. The value that is returned for snow depth data is no longer affected by the snow water equivalent data co-located at a station (previously if the snow water equivalent value was zero, the snow depth value would be forced to zero).
Apr. 5, 2013	2.9.1	Defect fix on getHourlyData method which was producing return of erroneous results.
Mar. 7, 2013	2.8.0	Modified the getStations call so that the "hucs" parameter has an implied wildcard at the end. Modified the getStationMetadata call so that the "huc" attribute now returns the 12-digit HUC (hydrologic unit code) of the station. There is a new attribute that is now returned called "hud" which is the 8-digit HUC (based on the Hydrologic Unit Dataset).
Feb. 11, 2013	2.7.0	Defect fix on the status page which was causing the status page to fail in production.

Jan. 14, 2013	2.6.0	The data retrieval code was modified so that if a SWE value is between 0 and -1, it will be returned as 0. Deployed Web Service Test Tool to https://www.wcc.nrcs.usda.gov/awdbWebService
Dec. 17, 2013	2.5.0	Modified the getStationMetadata call so that the “huc” attribute now returns the 12-digit HUC of the station. There is a new attribute that is now returned also called “hud” which is the 8-digit HUC (that is based on the Hydrologic Unit Dataset).
Nov. 26, 2013	2.4.1	Modified the code so that derived PRCP normals are rounded to the precision of the sensor before subtraction.
Oct. 15, 2013	2.4.0	Fixed a defect in getCentralTendencyData where it was not setting the centralTendencyType correctly in the return object.
Oct. 1, 2012	2.3.0	Added the following 30-Year Central Tendency methods: getCentralTendencyData getCentralTendencyPeakData getForecastPeriodCentralTendency

Terms of Use

Terms of Use for individuals or organizations planning to use the National Resources Conservation Service, National Water and Climate Center AWDB Web Service.

THIS SOFTWARE IS PROVIDED BY THE HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.