The AWDB web service provides access to data and metadata stored within the AWDB database via SOAP. To access this web service from a Java project, you will have to create the Java stubs from the WSDL and include the classes in your project. This document will cover how to generate these classes and what methods the web service provides.

**Generating the Java client stubs from the WSDL**

The utility to generate stubs from a WSDL, wsimport, 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 that first. This tool simply needs to be given the desired package to put the new stubs into and the WSDL address. From a command line, use the following command:

wsimport -p gov.usda.nrcs.wcc.awdbWebService -s . ‘<http://www.wcc.nrcs.usda.gov/awdbWebService/services?WSDL>’

This will generate the stubs for the AWDB web service and place them in the package gov.usda.nrcs.wcc.awdbWebService.

**Using the client stubs**

Once the client stubs are generated, they are imported and used just like any other Java packages.

To make calls to the web service, first you will need an instance of the AwdbWebService class that is pointed to the web service. This can be done once somewhere prior to using the 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 = new URL("http://www.wcc.nrcs.usda.gov/awdbWebService/services?wsdl");

AwdbWebService\_Service lookup = new AwdbWebService\_Service(wsURL, new

QName("http://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 (ex: m\_webService.getElements()).

**AWDB Web Service Method Listing**

[getElement 4](#_Toc256426559)

[getElements 4](#_Toc256426560)

[getHeightDepths 4](#_Toc256426561)

[getUnits 5](#_Toc256426562)

[getUnitName 5](#_Toc256426563)

[getStationElements 5](#_Toc256426564)

[getStations 6](#_Toc256426565)

[getStationMetadata 7](#_Toc256426566)

[getStationMetadataMultiple 7](#_Toc256426567)

[getReservoirMetadata 7](#_Toc256426568)

[getReservoirMetadataMultiple 8](#_Toc256426569)

[getData 9](#_Toc256426570)

[getInstantaneousData 10](#_Toc256426571)

[getAveragesData 11](#_Toc256426572)

[getDataAssuredFlags 12](#_Toc256426573)

[getPeakData 13](#_Toc256426574)

[getAveragesPeak 14](#_Toc256426575)

[getForecastPeriodAverages 15](#_Toc256426576)

[getForecastPoint 16](#_Toc256426577)

[getForecastPoints 16](#_Toc256426578)

[getForecast 17](#_Toc256426579)

[getForecasts 18](#_Toc256426580)

[getForecastsByPubDate 18](#_Toc256426581)

[getAllForecastsForStation 19](#_Toc256426582)

[getForecastValue 20](#_Toc256426583)

[getForecastConfigurations 20](#_Toc256426584)

[getForecastEquations 21](#_Toc256426585)

[getForecastEquationsMultiple 21](#_Toc256426586)

[getForecastPeriods 21](#_Toc256426587)

Each of the methods is described in more detail below along with an example call.

Parameters marked with a \* are required.

Several methods take a station triplet string or A List of triplets. These triplets identify a station and are composed of [station id]:[state code]:[network] ex: "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.

All of the signatures and examples below use Lists to hold collections of more than one item, such as station triplets or return values. The Java web service and the SOAP calls actually use Arrays to hold the collections, but when generating the web service stubs, the tool will automatically convert all these uses of Arrays to and from List objects for convenience. So for all practical purposes, when using the web service from Java, all input and return collections from the web service are Lists.

Each of the example calls given blow assumes there is already an instance of the AwdbWebService object created: m\_webService. Create this object using the code given on page 1.

# getElement

Element getElement(String elementCode)

This function gets the element with the given element code.

\* elementCode – The element code

**Example usage:**

Element element = m\_webService.getElement("WTEQ");

# getElements

List<Element> getElements()

This function gets all elements in AWDB.

**Example usage:**

List<Element> elements = m\_webService.getElements();

# getHeightDepths

List<HeightDepth> getHeightDepths()

This function gets all height/depths in AWDB.

**Example usage:**

List<HeightDepth> heights = m\_webService.getHeightDepths();

# getUnits

List<Unit> getUnits()

This function gets all units in AWDB.

**Example usage:**

List<Unit> units = m\_webService.getUnits();

# getUnitName

String getUnitName(String unitCode)

This function gets the plural name of a given unit code.

\* unitCode – The unit code

**Example usage:**

String unitName = m\_webService.getUnitName("in");

# getStationElements

List<StationElement> getStationElements(String stationTriplet, String beginDate, String endDate)

This function gets all station elements for the given station that were in service during the date range.

\* stationTriplet – The triplet that identifies a station

beginDate – The date of the earliest outService to retrieve

endDate - The date of the latest inService date to retrieve

**Example usage:**

List<StationElement> elements = m\_webService.getStationElements("10311000:NV:USGS", null, null);

# getStations

List<String> getStations(List<String> stationIds, List<String> stateCodes, List<String> networkCodes,

List<String> hucs, List<String> countyNames, BigDecimal minLatitude, BigDecimal maxLatitude, BigDecimal minLongitude, BigDecimal maxLongitude, BigDecimal minElevation,

BigDecimal maxElevation, List<String> elementCodes, List<int> ordinals,

List<HeightDepth> heightDepths, boolean logicalAnd)

This function gets A List of station triplets identifying all stations that match the given criteria. Null parameters will not be used as search filter criteria.

stationIds– A List of wildcard patterns of station ids and/or names

stateCodes – A List of wildcard patterns of FIPS postal state codes

networkCodes – A List of wildcard patterns of network codes

hucs – A List of wildcard patterns of hucs

countyNames – A List of wildcard patterns of county names

minLatitude – The minimum latitude

maxLatitude – The maximum latitude

minLongitude – The minimum longitude

maxLongitude – The maximum longitude

minElevation – The minimum elevation

maxElevation – The maximum elevation

elementCodes – A List of element codes

ordinals – A List of ordinals

heightDepths – A List of height/depths

\* p\_logicalAnd – If true, returned stations must match all criteria. If false, returned stations will match any criteria.

**Example usage:**

List<String> triplets = m\_webService.getStations(Arrays.asList(new String[] {"302", "ARBUCKLE\*"}),

Arrays.asList(new String[] {"OR"}), Arrays.asList(new String[] {"SNTL"}), null, null, null, null, null, null, null, null, null, null, null, true);

# getStationMetadata

StationMetadata getStationMetadata(String stationTriplet)

This function gets metadata for the given station.

\* stationTriplet – The triplet that identifies the station

**Example usage:**

StationMetaData station = m\_webService.getStationMetadata("945:OR:SNTL");

# getStationMetadataMultiple

List<StationMetadata> getStationMetadataMultiple(List<String> stationTriplets)

This function gets metadata for the given set of stations.

\* stationTriplets – A List of triplets that identifies the stations

**Example usage:**

String[] triplets = {"945:OR:SNTL", "0645:NV:COOP"};

List<StationMetaData> stations = m\_webService.getStationMetadataMultiple(Arrays.asList(triplets));

# getReservoirMetadata

ReservoirMetadata getReservoirMetadata(String stationTriplet)

This function gets reservoir metadata for one station.

\* stationTriplet – The triplet that identifies the station

**Example usage:**

ReservoirMetadata metadata = m\_webService.getReservoirMetadata("308:AZ:SNTL");

# getReservoirMetadataMultiple

List<ReservoirMetadata> getReservoirMetadataMultiple(List<String> stationTriplets)

This function gets reservoir metadata for multiple stations.

\* stationTriplets – A List of triplets that identifies the stations

**Example usage:**

List<ReservoirMetadata> metadata = m\_webService.getReservoirMetadataMultiple(

Arrays.asList(new String[] { "308:AZ:SNTL", "323:ID:SNTL","302:OR:SNTL"}));

# getData

List<Data> getData(List<String> p\_stationTriplets, String p\_elementCode, int p\_ordinal,

HeightDepth p\_heightDepth, Duration p\_duration, boolean p\_getFlags,

String p\_beginDate, String p\_endDate)

This function gets data for one or more stations for the given input and date range.

\* p\_stationTriplets – A List of triplets that identifies the stations

\* p\_elementCode – The element code

\* p\_ordinal – The ordinal

P\_heightDepth – The height/depth

\* p\_duration – The duration can be one of: Duration.DAILY, Duration.SEMIMONTHLY, or

Duration.MONTHLY

\* p\_getFlags – If true, will include A List of flags in the result

\* p\_beginDate – The first date of data to retrieve

\* p\_endDate – The last date of data to retrieve

**Example usage:**

List<Data> allData = m\_webService.getData(Arrays.asList("302:OR:SNTL"), "PREC", 1, null,

Duration.DAILY, true, "1979-01-01 00:00", "1980-01-01 00:00");

# getInstantaneousData

List<InstantaneousData> getInstantaneousData(List<String> p\_stationTriplets, String p\_elementCd,

int p\_ordinal, HeightDepth p\_heightDepth, String p\_beginDate, String p\_endDate, InstantaneousDataFilter p\_filter, UnitSystem p\_unitSystem)

This function gets instantaneous data and flags from SNOTEL for one or more stations for the given input and date range.

A common usage is to get the 23:59 value for a single day. To do this, get the first\_of\_day with a begin and end date of yyyy-MM-dd 23:59. An example is below.

\* p\_stationTriplets – A List of triplets that identifies the stations

\* p\_elementCode – The element code

\* p\_ordinal – The ordinal

p\_heightDepth – The height/depth

\* p\_beginDate – The first date of data to retrieve

\* p\_endDate – The last date of data to retrieve

\* p\_filter – The filter can be one of: InstantaneousDataFilter.ALL,

InstantaneousDataFilter.FIRST\_OF\_DAY, or InstantaneousDataFilter.MIDNIGHT\_ONLY

P\_unitSystem – The units the data will be returned in and can be either UnitSystem.ENGLISH or

UnitSystem.LAST\_COLLECTED.

**Example usage:**

String[] stationTriplets = new String[3];

stationTriplets[0] = "476:NV:SNTL";

stationTriplets[1] = "302:OR:SNTL";

stationTriplets[2] = "311:MT:SNTL";

HeightDepth heightDepth = new HeightDepth();

heightDepth.setValue(new BigDecimal(-9));

heightDepth.setUnitCd("mm");

List<InstantaneousData> result = m\_webService.getInstantaneousData(Arrays.asList(stationTriplets),

"WTEQ", 1, heightDepth, "2004-12-30 00:00", "2005-01-30 00:00", InstantaneousDataFilter.FIRST\_OF\_DAY, UnitSystem.LAST\_COLLECTED);

**Example request for Sept 30th 23:59 PREC value:**

List<InstantaneousData> last = m\_webService.getInstantaneousData(Arrays.asList("302:OR:SNTL"),

"PREC", 1, null, "2004-09-30 23:59", "2004-09-30 23:59", InstantaneousDataFilter.FIRST\_OF\_DAY, UnitSystem.ENGLISH);

# getAveragesData

List<AveragesData> getAveragesData(List<String> p\_stationTriplets, String p\_elementCode,

HeightDepth p\_heightDepth, Duration p\_duration, boolean p\_getFlags, int p\_beginMonth,

int p\_beginDay, int p\_endMonth, int p\_endDay)

This function gets averages data for one or more stations for the given input and date range.

The start and end days and months will be ignored if inappropriate for the given duration.

\* p\_stationTriplets – A List of triplets that identifies the stations

\* p\_elementCode – The element code

p\_heightDepth – The height /depth

\* p\_duration – The duration can be one of: Duration.DAILY, Duration.SEMIMONTHLY,

Duration.MONTHLY, or Duration.ANNUAL

\* p\_getFlags – If true, will include A List of flags in the result

\* p\_beginMonth – The first month of data to retrieve

\* p\_beginDay – The first day of data to retrieve

\* p\_endMonth – The last month of data to retrieve

\* p\_endDay – The last day of data to retrieve

**Example usage:**

List<AveragesData> allAvgs = m\_webService.getAveragesData(Arrays.asList("471:ID:SNTL"),

"WTEQ", null, Duration.DAILY, true, 2, 27, 3, 2);

# getDataAssuredFlags

List<StationDataAssuredFlags> getStationDataAssuredFlags(List<String> stationTriplets,

String elementCode, int ordinal, HeightDepth heightDepth, String durationCode,

int beginWaterYear, int endWaterYear)

This function gets the station data assured flags for the given stations and inputs.

\* stationTriplets – A List of triplets that identifies the stations

\* elementCode – The element code

\* ordinal – The ordinal

heightDepth – The height/depth

\* durationCode – The duration code

\* beginWaterYear– The first water year to retrieve

\* endWaterYear – The last water year to retrieve

**Example usage:**

List<StationDataAssuredFlags> stationFlags =

m\_webService.getStationDataAssuredFlags(Arrays.asList("302:OR:SNTL"),

"WTEQ", 1, null, "E", 2000, 2006);

# getPeakData

List<PeakData> getPeakData(List<String> p\_stationTriplets, String p\_elementCode,

int p\_ordinal, HeightDepth p\_heightDepth, Duration p\_duration, boolean p\_getFlags,

int p\_beginYear, int p\_endYear)

This function gets the annual peak values for the given stations, input, and water year range.

\* p\_stationTriplets – A List of triplets that identifies the stations

\* p\_elementCode – The element code

\* p\_ordinal – The ordinal

p\_heightDepth – The height/depth

\* p\_duration – The duration can only be Duration.DAILY

\* p\_getFlags – If true, will include the flag in the result

\* p\_beginYear – The starting water year

\* p\_endYear – The ending water year

**Example usage:**

List<PeakData> allData = m\_webService.getPeakData(Arrays.asList("302:OR:SNTL"), "WTEQ", 1, null,

Duration.DAILY, true, 2000, 2003);

# getAveragesPeak

List<AveragesPeakData> getAveragesPeak(List<String> p\_stationTriplets, String p\_elementCode,

HeightDepth p\_heightDepth, Duration p\_duration, boolean p\_getFlags)

This function gets the peak averages for the given stations and input.

\* p\_stationsTriplets – A List of triplets that identifies the stations

\* p\_elementCode – The element code

p\_heightDepth – The height/depth

\* p\_duration – The duration can be one of: Duration.DAILY, Duration.SEMIMONTHLY,

Duration.MONTHLY, or Duration.ANNUAL

\* p\_getFlags – If true, will include the flags in the results

**Example usage:**

List<AveragesPeakData> peakAvgs =

m\_webService.getAveragesPeak(Arrays.asList("09498500:AZ:USGS"), "SRVO", null,

Duration.MONTHLY, true);

# getForecastPeriodAverages

List<ForecastPeriodAverage> getForecastPeriodAverages(List<String> p\_stationTriplets,

String p\_elementCode, List<String> p\_periods)

This function gets the averages for given stations and forecast period.

\* p\_stationsTriplets – A List of triplets that identifies the stations

\* p\_elementCode – The element code

\* p\_period s– A List of forecast periods (ex. “APR-JUL”)

**Example usage:**

String[] periods = {"JUN-JUL", "APR-JUL", "MAY-JUL"};

List<ForecastPeriodAverage> averages = m\_webService.getForecastPeriodAverages(

Arrays.asList("09498500:AZ:USGS"), "SRVO", Arrays.asList(periods));

# getForecastPoint

ForecastPoint getForecastPoint(String stationTriplet)

This function gets the forecast point for the given station.

\* stationTriplet – The triplet that identifies the station

**Example usage:**

ForecastPoint fcstPt = m\_webService.getForecastPoint("10311000:NV:USGS");

# getForecastPoints

List<ForecastPoint> getForecastPoints(List<String> stationIds, List<String> stateCodes,

List<String> networkCodes, List<String> forecastPointNames, List<String> hucs,

List<String> forecasters, boolean logicalAnd)

This function gets the forecast points that match the given criteria.

stationIds – A List of wildcard patterns of station ids

stateCodes – A List of wildcard patterns of state codes

networkCodes – A List of wildcard patterns of network codes

forecastPointNames – A List of wildcard patterns of forecast point names

hucs – A List of wildcard patterns of hucs

forecasters – A List of forecaster usernames

\* p\_logicalAnd – If true, returned forecast points must match all criteria. If false, returned points will match any criteria.

**Example usage:**

List<ForecastPoint> fcstPts = m\_webService.getForecastPoints(Arrays.asList(new String[] {"10311000"}),

Arrays.asList(new String[] {"NV"}), Arrays.asList(new String[] { "USGS" }), null, null, null, true);

# getForecast

Forecast getForecast(String stationTriplet, String elementCode, String forecastPeriod,

String publicationDate)

This function gets the forecasts for the given station, element, forecast period, and publication date.

\* stationTriplet – The triplet that identifies the station

\* elementCode – The element code

\* forecastPeriod – The forecast period (ex. “APR-JUL”)

\* publicationDate– The publication date

**Example usage:**

Forecast fcst = m\_webService.getForecast("10311200:NV:USGS", "SRVO", "MAR-JUL",

"2002-01-01 00:00");

# getForecasts

List<Forecast> getForecasts(String stationTriplet, String elementCode, String forecastPeriod)

This function gets the forecasts for the given station, element, and forecast period.

\* stationTriplet – The triplet that identifies the station

\* elementCode – The element code

\* forecastPeriod – The forecast period (ex. “APR-JUL”)

**Example usage:**

List<Forecast> fcsts = m\_webService.getForecasts("10311200:NV:USGS", "SRVO", "MAR-JUL");

# getForecastsByPubDate

List<Forecast> getForecastsByPubDate(String stationTriplet, String elementCode, String forecastPeriod,

String beginPublicationDate, String endPublicationDate)

This function gets the forecasts for the given station, element, and forecast period whose publication dates are between the begin and end dates.

\* stationTriplet – The triplet that identifies the station

\* elementCode – The element code

\* forecastPeriod – The forecast period (ex. “APR-JUL”)

\* beginPublicationDate– The publication date

\* endPublicationDate– The publication date

**Example usage:**

List<Forecast> fcsts = m\_webService.getForecastsByPubDate("10311200:NV:USGS", "SRVO",

"MAR-JUL", "1997-01-01 00:00:00", "1999-03-01 00:00:00");

# getAllForecastsForStation

List<ForecastFull> getAllForecastsForStation (String stationTriplet, String beginPublicationDate,

String endPublicationDate)

This function gets all forecasts for the given station whose publication date is within the specified range.

\* stationTriplet – The triplet that identifies the station

\* p\_beginPublicationDate – The earliest publication date

\* p\_endPublicationDate – The last publication date

**Example usage:**

List<ForecastFull> fcsts = m\_webService.getAllForecastsForStation("10311200:NV:USGS",

"2000-01-01", "2000-12-31");

# getForecastValue

BigDecimal getForecastValue(String stationTriplet, String elementCode, String forecastPeriod,

int probability, int publicationYear, int publicationMonth, int publicationDay)

This function gets the forecast value for the given station, forecast period, probability, and publication date.

\* stationTriplet – The triplet that identifies the station

\* elementCode – The element code

\* forecastPeriod – The forecast period (ex. “APR-JUL”)

\* probability – The probability (between 0 and 100)

\* publicationYear– The publication year

\* publicationMonth– The publication month

\* publicationDay– The publication day

**Example usage:**

BigDecimal fcstValue = m\_webService.getForecastValue("10311200:NV:USGS", "SRVO",

"MAR-JUL", 50, 1997, 1, 1);

# getForecastConfigurations

List<Configuration> getForecastConfigurations (String forecaster)

This function gets all configurations made by the given forecaster.

\* p\_forecaster – The forecaster username

**Example usage:**

List<Configuration> configurations = m\_webService.getForecastConfigurations("user");

# getForecastEquations

List<ForecastEquation> getForecastEquations(String stationTriplet)

This function gets all forecast equations for the given station.

\* stationTriplet – The triplet that identifies the station

**Example usage:**

List<ForecastEquation> equations = m\_webService.getForecastEquations("10321000:NV:USGS");

# getForecastEquationsMultiple

List<ForecastEquation> getForecastEquationsMultiple(List<String> stationTriplets)

This function gets all forecast equations for the given stations.

\* p\_stationsTriplets – A List of triplets that identifies the stations

**Example usage:**

List<ForecastEquation> equationss = m\_webService.getForecastEquationsMultiple(

Arrays.asList(new String[] {"302:OR:SNTL", "10321000:NV:USGS"}));

# getForecastPeriods

List<ForecastPeriod> getForecastPeriods()

This function gets all possible forecast periods.

**Example usage:**

List<ForecastPeriod> forecastPeriods = m\_webService.getForecastPeriods();

**AWDB Web Service Class Listing**

The web service makes use of a number of container classes for both inputs and returning data to the user. Each of these classes and their publicly accessible fields are given below. All of these classes have a public getter and a setter for each field with the standard Java bean naming conventions (ex: getName() and setName() for the String field “name” or isDaily() for the boolean field “daily”).

## AveragesData

int beginDay

int beginMonth

String dataSetFlag

Duration duration

int endDay

int endMonth

List<String> flags

List<BigDecimal> values

## AveragesPeakData

Duration duration

String flag

int peakDay

int peakMonth

BigDecimal value

## Configuration

String analysisType

int calibrationEndYear

int calibrationStartYear

String comment

int componentsToRetain

String creationDate

boolean dailyForecast

BigDecimal dailyForecastThreshold

boolean dailyForecastThresholdRelative

boolean dataForcedSeriallyComplete

int degreesOfFreedom

String forecastCondition

String forecastType

String forecaster

BigDecimal functionArgument

String functionCd

String helperDataSource

String helperElementCd

int helperEndDay

int helperEndMonth

String helperEndYearFlag

int helperMixedPastEndDay

int helperMixedPastEndMonth

String helperMixedPastEndYearFlag

int helperMixedPastStartDay

int helperMixedPastStartMonth

String helperMixedPastStartYearFlag

boolean helperPredictorUsed

int helperStartDay

int helperStartMonth

String helperStartYearFlag

String helperStationTriplet

boolean helperUsedInOverlappingYears

boolean helperVarianceMatchedToObserved

boolean jackKnifingUsed

int key

BigDecimal minimumR2ForPredictor

int minimumYearsForPredictor

int mixedPastEndDay

int mixedPastEndMonth

String mixedPastEndYearFlag

int mixedPastStartDay

int mixedPastStartMonth

String mixedPastStartYearFlag

boolean mixedPastUsed

String name

int numYearsUsed

boolean optimalWeightingUsed

BigDecimal overrideAverage

BigDecimal overrideStderr

BigDecimal pastFlowRealtime

List<PredictorGroup> predictorGroups

List<Predictor> predictors

BigDecimal preprocessingConstant

BigDecimal preprocessingMultiplier

String publicationCd

BigDecimal tTestLimit

String targetDataSource

String targetElementCd

int targetEndDay

int targetEndMonth

String targetEndYearFlag

int targetStartDay

int targetStartMonth

String targetStartYearFlag

String targetStationName

String targetStationTriplet

boolean transformErrorsOnly

## Data

String beginDate

Duration duration

String endDate

List<String> flags

List<BigDecimal> values

## DataSource

An enum that can be used as:

DataSource.OBSERVED

DataSource.DERIVED

DataSource.INTERPRETED

## Duration

An enum that can be used as:

Duration.DAILY

Duration.MONTHLY

Duration.SEMIMONTHLY

Duration.INSTANTANEOUS

Duration.ANNUAL

Duration.SEASONAL

## Element

String elementCd

String name

String storedUnitCd

## Forecast

String calculationDate

String elementCd

List<Integer> exceedenceProbabilities

List<BigDecimal> exceedenceValues

String forecastPeriod

BigDecimal periodAverage

String publicationDate

String stationTriplet

String unitCd

## ForecastEquation

String comment

String elementCd

List<ForecastEquationTerm> equationTerms

String footnotes

String forecastCondition

String forecastPeriod

String forecastType

String function

BigDecimal functionArgument

BigDecimal interceptConstant

int key

BigDecimal maxOfRecord

BigDecimal minOfRecord

List<BigDecimal> monthlyCorrelationCoefficients

List<BigDecimal> monthlyStderrValues

String name

List<MonthAndDay> publicationMonthsAndDays

boolean published

String stationTriplet

boolean transformErrorsOnly

String unitCd

## ForecastEquationTerm

BigDecimal coefficient

String dataPeriodDays

String dataPeriodMonth

String dataYearFlag

StationElement stationElement

String unitCd

boolean upstreamForecast

## ForecastFull

String comment

Configuration configuration

String coordinatedForecastIssueDate

List<BigDecimal> coordinatedForecastProbabilities

List<BigDecimal> coordinatedForecastValues

String elementCd

ForecastEquation equation

boolean forecastFinal

String forecastPeriodCd

String forecastPeriodName

String forecaster

String hydrologistForecastIssueDate

List<BigDecimal> hydrologistForecastProbabilities

List<BigDecimal> hydrologistForecastValues

String originalForecastIssueDate

List<BigDecimal> originalForecastProbabilities

List<BigDecimal> originalForecastValues

List<String> otherAgencyCds

List<BigDecimal> otherAgencyForecastValues

BigDecimal periodAverage

String publicationDate

String stationTriplet

String unitCd

## ForecastPeriod

String description

String forecastPeriod

## ForecastPeriodAverage

String flag

String forecastPeriod

String stationTriplet

BigDecimal value

## ForecastPoint

List<Integer> exceedenceProbabilities

String name

String responsibleForecaster

String stationTriplet

## HeightDepth

String unitCd

BigDecimal value

A note when using HeightDepth objects, both unitCd and value must be set. Use -9 mm if you want

an unknown/not applicable height depth value.

## InstantaneousData

String beginDate

String endDate

int stationId

String unitCd

List<InstantaneousDataValue> values

## InstantaneousDataFilter

An enum that can be used as:

InstantaneousDataFilter.ALL

InstantaneousDataFilter.FIRST\_OF\_DAY

InstantaneousDataFilter.MIDNIGHT\_ONLY

## InstantaneousDataValue

String flag

String time

BigDecimal value

## MonthAndDay

int day

int month

## PeakData

int beginYear

Duration duration

int endYear

List<String> flags

List<Integer> peakDays

List<Integer> peakMonths

List<BigDecimal> values

## Predictor

int endDay

int endMonth

String endYearFlag

int groupNumber

int predictorNumber

boolean predictorUsed

int startDay

int startMonth

String startYearFlag

String stationTriplet

## PredictorGroup

String customStationListName

String dataSourceCd

String elementCd

boolean globalMonthChangeAllowed

String groupName

Integer groupNumber

boolean negativeWeightAllowed

String networks

String states

boolean upstreamForecast

## ReservoirMetadata

BigDecimal elevationAtCapacity

BigDecimal reservoirCapacity

String stationTriplet

BigDecimal usableCapacity

## StationDataAssuredFlags

List<String> assuredFlags

String stationTriplet

## StationElement

String beginDate

Integer dataPrecision

DataSource dataSource

Duration duration

String elementCd

String endDate

HeightDepth heightDepth

int ordinal

String originalUnitCd

String stationTriplet

String storedUnitCd

## StationMetaData

String actonId

String beginDate

String countyName

BigDecimal elevation

String endDate

String fipsCountryCd

String fipsCountyCd

String fipsStateNumber

String huc

BigDecimal latitude

BigDecimal longitude

String name

String shefId

BigDecimal stationDataTimeZone

BigDecimal stationTimeZone

String stationTriplet

## Unit

String name

String unitCd

## UnitSystem

An enum that can be used as:

UnitSystem.ENGLISH

UnitSystem.LAST\_COLLECTED

# Appendix A: Network Codes

You need three pieces of information to uniquely identify a station. These are the station id, state code, and network code. The state code is just the 2-character FIPS alphabetic code. For example, the state code for Oregon is OR. The network codes that are currently being used by the AWDB database are the following:

|  |  |
| --- | --- |
| **Network Code** | **Description** |
| COOP | NULL |
| WBAN | NULL |
| METAR | NULL |
| SHEF | NULL |
| SNTL | NRCS SNOTEL DATA COLLECTION SYSTEM |
| SNOW | NRCS SNOW COURSE SITES |
| USGS | USGS MONTHLY DATA |
| BOR | MONTHLY RESERVOIR DATA |
| SCAN | NRCS SOIL CLIMATE ANALYSIS DATA COLLECTION SYSTEM |
| MPRC | Manual Precipitation Sites |
| MSNT | Manual Snotel non-telemetered, non-real time sites |
| OFCST | SITES THAT ARE USED ONLY IN THE FORECAST EQUATIONS |
| CLMIND | NETWORK "CREATED" BY NRCS WCC TO INCORPORATE ALL CLIMDATE INDEX TYPE STATIONS |
| WELL | NETWORK "CREATED" BY NRCS WCC TO INCORPORATE ALL WELL DATA |
| NA | NOT APPLICABLE |

# Appendix B: Element Codes

When you request data from the service, you need to specify the element code for the data that you are interested in. The table below lists the element codes available and the units that the data is stored in.

|  |  |  |
| --- | --- | --- |
| **Element Code** | **Description** | **Unit** |
| WTEQ | SNOW WATER EQUIVALENT | in |
| PREC | PRECIPITATION ACCUMULATION | in |
| TAVG | AIR TEMPERATURE AVERAGE | degF |
| TMAX | AIR TEMPERATURE MAXIMUM | degF |
| TMIN | AIR TEMPERATURE MINIMUM | degF |
| SRAD | SOLAR RADIATION | watt/m2 |
| PRCP | PRECIPITATION INCREMENT | in |
| EVAP | EVAPORATION | in |
| SNWD | SNOW DEPTH | in |
| WDMV | WIND MOVEMENT OBSERVED | mile |
| SNOW | SNOW FALL | in |
| MNPN | MINIMUM | degF |
| MXPN | MAXIMUM | degF |
| STN | SOIL TEMPERATURE MINIMUM | degF |
| STX | SOIL TEMPERATURE MAXIMUM | degF |
| STO | SOIL TEMPERATURE OBSERVED | degF |
| SRVO | STREAM VOLUME, ADJUSTED | ac\_ft |
| RESC | RESERVOIR STORAGE VOLUME | ac\_ft |
| TOBS | AIR TEMPERATURE OBSERVED | degF |
| OI | TELECONNECTION INDEX | unitless |
| JDAY | JULIAN DATE | julian\_day |
| REST | RESERVOIR STAGE | ft |
| STRM | STRM FLOW | cfs |
| WTEQX | SNOW WATER EQUIVALENT MAXIMUM | in |
| SRVOO | STREAM VOLUME, OBSERVED | ac\_ft |
| WELL | WELL DEPTH | ft |
| WDMVN | WIND MOVEMENT MINIMUM | mile |
| HFTV | ENERGY GAIN OR LOSS FROM GROUND | watt/m2 |
| VOLT | GENERIC VOLTAGE | volt |
| PRES | BAROMETRIC PRESSURE | inch\_Hg |
| BATT | BATTERY | mvolt |
| BATX | BATTERY MAXIMUM | mvolt |
| BATN | BATTERY MINIMUM | mvolt |
| ETIB | BATTERY-ETI PRECIP GUAGE | volt |
| COND | CONDUCTIVITY | umho |
| DPTP | DEW POINT TEMPERATURE | degF |
| DIAG | DIAGNOSTICS | unitless |
| DISO | DISSOLVED OXYGEN | mgram/l |
| DISP | DISSOLVED OXYGEN - PERCENT SATURATION | pct |
| FUEL | FUEL MOISTURE | pct |
| FMTMP | FUEL TEMPERATURE INTERNAL | degF |
| TGSV | GROUND SURFACE INTERFACE TEMPERATURE AVERAGE | degF |
| TGSX | GROUND SURFACE INTERFACE TEMPERATURE MAXIMUM | degF |
| TGSN | GROUND SURFACE INTERFACE TEMPERATURE MINIMUM | degF |
| TGSI | GROUND SURFACE INTERFACE TEMPERATURE OBSERVED | degF |
| NTRDV | NET SOLAR RADIATION AVERAGE | watt/m2 |
| NTRDX | NET SOLAR RADIATION MAXIMUM | watt/m2 |
| NTRDN | NET SOLAR RADIATION MINIMUM | watt/m2 |
| NTRDC | NET SOLAR RADIATION OBSERVED | watt/m2 |
| H2OPH | PH | unitless |
| ETIL | PULSE LINE MONITOR-ETI GUAGE | volt |
| RDC | REAL DIELECTRIC CONSTANT | unitless |
| RHUM | RELATIVE HUMIDITY | pct |
| RHUMV | RELATIVE HUMIDITY AVERAGE | pct |
| RHENC | RELATIVE HUMIDITY ENCLOSURE | pct |
| RHUMX | RELATIVE HUMIDITY MAXIMUM | pct |
| RHUMN | RELATIVE HUMIDITY MINIMUM | pct |
| SAL | SALINITY | gram/l |
| SNWDV | SNOW DEPTH AVERAGE | in |
| SNWDX | SNOW DEPTH MAXIMUM | in |
| SNWDN | SNOW DEPTH MINIMUM | in |
| WTEQV | SNOW WATER EQUIVALENT avg | in |
| WTEQN | SNOW WATER EQUIVALENT min | in |
| SMV | SOIL MOISTURE BARS AVERAGE | bar |
| SMX | SOIL MOISTURE BARS MAXIMUM | bar |
| SMN | SOIL MOISTURE BARS MINIMUM | bar |
| SMO | SOIL MOISTURE BARS OBSERVED | bar |
| SMS | SOIL MOISTURE PERCENT | pct |
| STV | SOIL TEMPERATURE AVERAGE | degF |
| SRADV | SOLAR RADIATION AVERAGE | watt/m2 |
| SRADX | SOLAR RADIATION MAXIMUM | watt/m2 |
| SRADN | SOLAR RADIATION MINIMUM | watt/m2 |
| SRADT | SOLAR RADIATION TOTAL | watt/m2 |
| LRAD | SOLAR RADIATION/LANGLEY | langley |
| LRADX | SOLAR RADIATION/LANGLEY MAXIMUM | langley |
| LRADT | SOLAR RADIATION/LANGLEY TOTAL | langley |
| SRMV | STREAM STAGE (GAUGE HEIGHT) AVERAGE | ft |
| SRMX | STREAM STAGE (GAUGE HEIGHT) MAXIMUM | ft |
| SRMN | STREAM STAGE (GAUGE HEIGHT) MINIMUM | ft |
| SRMO | STREAM STAGE (GAUGE HEIGHT) OBSERVED | ft |
| TURB | TURBIDITY | ntu |
| WLEVV | WATER LEVEL AVERAGE | in |
| WLEVX | WATER LEVEL MAXIMUM | in |
| WLEVN | WATER LEVEL MINIMUM | in |
| WLEV | WATER LEVEL OBSERVED | in |
| WTEMP | WATER TEMPERATURE | degF |
| WDIRV | WIND DIRECTION AVERAGE | degree |
| WDIR | WIND DIRECTION OBSERVED | degree |
| WDMVV | WIND MOVEMENT AVERAGE | mile |
| WDMVX | WIND MOVEMENT MAXIMUM | mile |
| WDMVT | WIND MOVEMENT TOTAL | mile |
| WSPDV | WIND SPEED AVERAGE | mph |
| WSPDX | WIND SPEED MAXIMUM | mph |
| WSPDN | WIND SPEED MINIMUM | mph |
| WSPD | WIND SPEED OBSERVED | mph |
| RVST | RIVER STAGE LEVEL | ft |
| WTAVG | WATER TEMPERATURE AVERAGE | degF |
| WTMAX | WATER TEMPERATURE MAXIMUM | degF |
| WTMIN | WATER TEMPERATURE MINIMUM | degF |
| STRV | STRM FLOW AVERAGE | cfs |
| STRN | STRM FLOW MINIMUM | cfs |
| STRX | STRM FLOW MAXIMUM | cfs |
| SWSI | SWSI ELEMENT | unknown |
| BATV | BATTERY AVERAGE | mvolt |
| SVPV | VAPOR PRESSURE - SATURATED | kPa |
| PVPV | VAPOR PRESSURE - PARTIAL | kPa |
| PARV | PHOTOSYNTHETICALLY ACTIVE RADIATION (PAR) HOURLY AVG | micromole/m2/s |
| PART | PHOTOSYNTHETICALLY ACTIVE RADIATION (PAR) TOTAL | millimole/m2/s |
| PARDY | PHOTOSYNTHETICALLY ACTIVE RADIATION (PAR) DAILY TOTAL | millimoles/m2/s |
| HOLD | PLACE HOLDER | unitless |