AQUATIC INFORMATICS™

FASTER ANALYSIS.BETTER DECISIONS.

AQUARIUS Publish Web Service API

Sep.23.11

1 Introduction

This document describes software interfaces for retrieving time series and related data from AQUARIUS. The interface uses industry standard SOAP or REST web services whose outputs are simple CSV formatted strings.

2 Programming Environment:

The AQUARIUS Server software runs on Windows Server 2003 and Internet Information Server 6.0 or Windows Server 2008 and IIS 7.0. Web services are based on WCF under Microsoft .Net 3.5.

External tools can connect to the AQUARIUS Publish web service using either SOAP or REST protocols. If Microsoft Visual Studio is used then the webservice can be referenced directly, otherwise interfaces can be generated from the WSDL description or coded by hand.

Example code is shown in Appendix 1.

3 General Description of API

The API is based on a simple token-based authentication mechanism. Users acquire a token at the beginning of the session and use it for all subsequent access during the session. For the REST version of the interface, the token can be passed explicitly in the query string of embedded in the HTTP header. Examples of both are show in Appendix 1.

The API includes a set of methods for retrieving various lists of metadata from AQUARIUS (such as Approval codes, Quality codes, Parameter definitions, etc) as well as methods for retrieving actual data. General best practice would be to get the lists of metadata types and use those values in the data retrieval methods, but for relatively static systems it is also fine to ignore the metadata definitions and simply hard-code the known values into the retrieval methods.

AQUARIUS uses the concept of *Publish Views* to control access to time series data. The server administrator defines these views and specifies the data approval levels that each view is allowed to see. For example it might make sense to set up a “Public” view that hides data at an “unapproved” approval level, or a “Trusted” view that hides nothing.

4 Data Types and Technical Specs

- Unless otherwise specified, all string data is UTF8 Unicode and all byte arrays are UTF8 encoded strings.
- Any string that contains a comma must be wrapped in double quotes.
- Dates are represented as ISO 8601 formatted strings with fractional seconds and time zone. The format is:
  `yyyy-MM-ddTHH:mm:ss.fffzzz (eg 1997-07-16T19:20:30.451+01:00)`
- The SOAP interface is found at the following URL:
  `http://<yourserver>/AQUARIUS/Publish/AquariusPublishService.svc`
  And the WSDL is here:
  `http://<yourserver>/AQUARIUS/Publish/AquariusPublishService.svc?WSDL`
- The REST interface is at the following URL:
  `http://<yourserver>/AQUARIUS/Publish/AquariusPublishRestService.svc`

Method: GetAuthToken

This method is used to log in to the AQUARIUS Publish web service and obtain a token for accessing the API.

Interface

SOAP method: `string GetAuthToken (string user, string encodedPassword)`

REST method: `/GetAuthToken?user={user}&encPwd={encodedPassword}`

Input	
String user	AQUARIUS publish view access user
string encodedPassword	Hashed password (encoding is system-specific)

Output	
string authToken	The token is a SHA signature. The server caches this token for use during the session. Tokens expire after 30 minutes of no use.

Method: GetParameterList

This method returns the complete list of parameters defined in AQUARIUS. Common examples are HG (Gauge Height) and QR (Discharge) but there are often hundred of others.

Interface

SOAP method: `string GetParameterList()`

REST method: `/GetParameterList?token={string}`

Input

<none>

Output

CSV string, format:

```
Parameter,DisplayId,Name,Interpolation,Unit
HG,Stage,Gauge Height,1,m
QR,Flow,Discharge,1,m^3/s
```

Line 1: Header
Line 2: Data

Method: GetGradeList

This method returns the complete list of grades defined in AQUARIUS.

Interface

SOAP method: `string GetGradeList ()`

REST method: `/GetGradeList?token=s{tring}`

Input

<none>

Output

CSV string, format:

```
Code,TypeCode,Description,Color
-2,BAD,Bad data,#ff00ff
0,,Undefined Quality,
10,A,Partial Day,#34c203
15,B, Backwater,
...
```

Line 1: Header
Line 2: Data

Method: GetApprovalList

This method returns the complete list of approval levels defined in AQUARIUS.

Interface

SOAP method: `string GetApprovalList()`

REST method: `/GetApprovalList?token={string}`

Input

<none>

Output

CSV string, format:

```
Code,Description,Color
0,Unprocessed,#ffffff
1,Preliminary,#ff0000
2,Reviewed,#00ff00
3,Checked,#0000ff
4,Approved,#c0c0c0
...
```

Line 1: Header
Line 2: Data

Method: GetPublishViewList

This method returns the complete list of publishing views defined in AQUARIUS. Publishing views are used to control who sees what data based on approval levels.

The output is a list of publish views where each view has a code number, a list of visible approval levels, and a view name.

Interface

SOAP method: `string GetPublishViewList()`

REST method: `/GetPublishViewList?token={string}`

Input

<none>

Output

CSV string, format:

```
Code,Approvals,PublishView
0,4,Public
1,0 3 4,Partner
2,0 2 3 4,InternalAccess
...
```

Line 1: Header
Line 2: Data

Method: GetDataSetsList

This method is used to get a list of all the time series that are in the AQUARIUS database for a given location.

The output is a list of datasets, where each line has a dataset name, the parameter it is measuring, the unit of measurement, and a set of simple statistics such as max, min, mean, total number of samples, etc.

There is an option to return a list of only those time series that have been modified since a given time.

Interface

SOAP method:
```
string GetDataSetsList(string locationId, string changesSinceTime)
```

REST method:
```
/GetDataSetsList?token={string}&locId={string}&changesSince={datetime}
```

Input	
string locationId	AQUARIUS location identifier. (eg 02GA010).). If not specified then the time series for all locations are returned
string changesSinceTime	By specifying this time you limit the returned time series to those that have been modified since this time. If not specified then all time series are returned. This should only be used in conjunction with locationId (i.e. if you specify changesSinceTime, you must also specify locationId).

Output	
CSV string, format:	
`LocationId,DataId,Parameter,Unit,Min,Max,Mean,StartTime,StartVal`	Line 1: Header

```
ue,EndTime,EndValue,TotalGaps,TotalSamples,LastModified
02GA010,HG.Working@02GA010,HG,m,2.131,7.304,3.455,1996-01-
01T00:15:00:00.000-06:00,6.212,2010-05-05T20:00:01.000-
06:00,6.707,60,153771,2010-07-01T00:00:01.000-06:00
02GA010,QR.Working@02GA010,QR,m^3/s,0,422.525,17.152,1996-01-
01T00:15:00:00.000-06:00,4.321,2010-05-05T20:00:01.000-
06:00,5.775,60,153771,2010-07-01T00:00:01.000-06:00
...
```

Line 2: Data

Line 3

Method: GetTimeSeriesData

This method is used to retrieve the contents of a given time series. It can:

- retrieve the entire signal, or it can look at a particular time range.
- retrieve only data that has changed since a specified time.
- "roll back the clock" to look at the signal as it existed at some time in the past.

It is important to note that the data returned by this method is governed by the publishView that is specified, which in turn depends on the approval levels of data in the time series. This means that data can come into "view" at any time (based on approval changes) but can also go out of view.

The method returns two sets of data:

1- The time ranges that are relevant (one line per time range)
2- The data for each time range (many lines, one per data point)

If changesSinceTime is specified then the method will return data that has changed (or come into view) since that time. Note that the returned data may include some points that were returned in a previous poll.

If the caller is maintaining a copy of the time series then the returned data must be processed as follows for each returned time range:

1- First, the overall time range should be deleted
2- Then the returned data should be copied in

This may result in data being deleted as well as added. In the extreme case, a given time range may not have any corresponding points, in which case it is a full delete.

If changesSinceTime is not specified then the time series will be returned in a single time range (according to the specified publishView).

In the case where no data is available, the number of ranges will be 0 (ie the file will have only 2 lines).

Interface

SOAP method:

```
string GetTimeSeriesData(string dataId, string publishView,
                         string queryFromTime, string queryToTime,
                         string changesSinceTime, string asAtTime)
```

REST method:

```
/GetTimeSeriesData?token={string}&dataId={string}&view={string}
                  &queryFrom={datetime}&queryTo={datetime}
                  &changesSince={datetime}&asAtTime={datetime}
```

Input	
String dataId	AQUARIUS data identifier. This should be taken from the DataId parameter as returned from the call to GetDataSetsList. e.g. HG.Working@02GA010 QR.Working@02GA010 TA.Field Visits@02GA010
string publishView	See method GetPublishViewList. If the current user has a default publishing view assigned then the method will filter based on user's default publishing view. If the current user does not have a default publishing view specified a publishing view can be specified via this parameter. If the user has a publishing view specified and the publishView parameter is not null or empty GetTimeSeriesData will return an error message instead of data.
string queryFromTime	The start time of the range to be returned. For example if you are interested in only 2009 then this might be 2009-01-01T00:00:00.000-05:00. If not specified then the start of signal is used.
string queryToTime	The end time of the range to be returned. For example if you are interested in only 2009 then this might be 2009-12-31T23:59:59.999-05:00. If not specified then the end of signal is used.
string changesSinceTime	By specifying this time you limit the returned data to only changes that have occurred since this time. The data is returned in multiple time ranges according to what data has changed. If not specified then all data is returned and there will be only one time range.
string asAtTime	By specifying this time you can roll back the clock to retrieve the signal as it was at this time. If not specified then the current signal is used.

Output

CSV string, format:

```
Time,Parameter,LocationId,DataId,NumRanges,NumPoints,
PublishView,FlagType????,QueryFrom,QueryTo,ChangesSince,AsAtTime
2010-07-01T00:00:01.000-
06:00,HG,02GA010,HG.Telemetry@02GA010,2,35,Public,,,,,
RangeNumber,StartTime,EndTime,NumPoints
1,2010-02-01T00:00:01.000-06:00,2010-02-06T23:59:59.000-06:00,20
2,2010-06-01T00:00:01.000-06:00,2010-06-06T23:59:59.000-06:00,15
RangeNumber,Time,Value,Flag???,Quality,Interpolation,Approval
1,2010-02-01T00:00:01.000-06:00,10.1,,10,1,3
1,2010-02-01T00:15:01.000-06:00,10.2,,10,1,3
1,2010-02-01T00:30:01.000-06:00,10.3,,10,1,3
1,2010-02-01T00:45:01.000-06:00,10.4,,10,1,3
1,2010-02-01T01:00:01.000-06:00,10.5,,10,1,3
...
2,2010-06-01T00:00:01.000-06:00,10.1,,10,1,3
2,2010-06-01T00:15:01.000-06:00,10.2,,10,1,3
2,2010-06-01T00:30:01.000-06:00,10.3,,10,1,3
2,2010-06-01T00:45:01.000-06:00,10.4,,10,1,3
2,2010-06-01T01:00:01.000-06:00,10.5,,10,1,3
...
```

Line 1: Meta header
Line 2: Meta data

Table1: Data Ranges

Table 2: Data

Method: GetRatingTable

This method is used to get the rating curve(s) for a given location. It returns a set of interpolated curves in the form of regularly sampled rating tables, each with its own period of applicability.

The output is composed of two logical tables. The first table lists the rating curves and their periods of applicability. The second table gives the points for each curve.

Interface

SOAP method:

```
string GetRatingTable(string locationId, string inputParameter,
                      string outputParameter)
```

REST method:

```
/GetRatingTable?token={string}&locId={string}&input={string}&output={string}
```

Input	
string locationId	AQUARIUS location identifier. (eg 02GA010)
string inputParameter	Independent parameter (eg HG)
string outputParameter	Dependent parameter (eg QR)

Output

CSV string, format:

```
LocationId,InputParameter,OutputParameter,NumTables
02GA010,HG,QR,2
TableNumber,StartDate,EndDate,NumPoints
1,1990-06-01T00:00:01.000-06:00,2001-06-01T00:00:01.000-06:00,56
2,2001-06-01T00:00:01.000-06:00,2002-06-01T00:00:01.000-06:00,87
TableNumber,Input,Output
1,12.1,20.3
1,12.2,20.4
1,12.2,20.4
1,12.2,20.4
1,12.2,20.4
...
2,12.2,20.4
2,12.2,20.4
2,12.2,20.4
2,12.2,20.4
...
```

Line 1: Meta header
Line 2: Meta data
Table 1: Rating table info

Table 2: Rating points

Method: GetTemplateList

This method returns the complete list of report templates defined in AQUARIUS. Report templates are used to generate reports for a given data set.

The output is a list of report templates where each template has an id, a label and a type.

Interface

SOAP method:
```
string GetTemplateList()
```

REST method:
```
/GetTemplateList?token={string}
```

Input

<none>

Output

CSV string, format:

```
Id,Label,Type (0=Template; 1=Description)                    Line 1: Meta header
798fa260-ab19-435c-9997-850373d8da71,Daily Mean by Year,0    Line 2: Meta data
```

Method: GetReportData

This method generates and retrieves a report for a given data set using the provided report template. The report templates are configured using the AQUARIUS Report Tool User Interface.

The output is the report as generated by the AQUARIUS reporting engine. The contents of the report are specific according to the provided report template and data set.

Interface

SOAP method:

```
string GetReportData(string dataId, string reportId, int reportType,
                     string outputPath, int reportType)
```

REST method:

```
/GetReportData?token={string}&dataId={string}&reportId={string}&reportType={int}&
               outputPath={string}&reportName={string}
```

Input	
string dataId	AQUARIUS data identifier, e.g. HG.Working@02GA010
string reportId	See method GetTemplateList. The report template is set up in the AQUARIUS Reporting Tool.
Int reportType	See method GetTemplateList: 0 = Report Template 1 = Report Description
string outputPath	Full path to the location on disk where the report will be saved. e.g. C:\ProgramData\Aquatic Informatics\AQUARIUS\Reports
string reportName	Report name (file extension will be appended to the report name once the report is generated according to how the specified report template is set up). e.g. DailyMeansReport_01012011.

Output

(Report output in format as defined by report template or report description in Report Tool)

SOAP Client Example (C#)

This example shows sample code for using the SOAP interface. If you are using Visual Studio you can create the client proxy class by using the "Add Service Reference" feature. Otherwise you will need to generate a proxy class from the WSDL (not shown).

```
//Add web references "http://aiapp1/AQUARIUS/Publish/AquariusPublishService.svc"
AquariusPublishServiceClient client = new AquariusPublishServiceClient(
     "BasicHttpBinding_IAquariusPublishService",
     @"http://aiapp1/AQUARIUS/Publish/AquariusPublishService.svc");

//get token and set it into the SOAP header
string authToken = client.GetAuthToken("dummy_user", "dummy_pw");
OperationContextScope context = new OperationContextScope(client.InnerChannel);
OperationContext current = OperationContext.Current;
MessageHeader runtimeHeader = MessageHeader.CreateHeader(
     "AQAuthToken", "", authToken, false);
OperationContext.Current.OutgoingMessageHeaders.Add(runtimeHeader);

//basic methods
string paramList = client.GetParameterList();
string qualityList = client.GetGradeList();
string approvalList = client.GetApprovalList();
string ratingTable = client.GetRatingTable("02GA010", "HG", "QR");
string dataList = client.GetDatasetsList("02GA010");

//Time series changes in the last 30 hours
string changeSince = (DateTime.Now-TimeSpan.FromHours(30)).ToString(
                      "yyyy-MM-ddTHH:mm:ss.fffzzz");
byte[] datas = client.GetTimeSeriesData("HG.Telemetry@02GA010","Public",
                                        null, null, changeSince, null);
string csv = System.Text.Encoding.UTF8.GetString(datas);
```

REST (HTTP) Client Example (C#)

This example shows a mechanism for using the REST interface. To simplify things it includes a custom HttpGet function that deals with making http requests, setting the authToken, and marshalling the response data.

```
string address =
     "http://aiapp1/AQUARIUS/Publish/AquariusPublishRestService.svc";

//get auth token
string authToken = HttpGet(address + "/GetAuthToken" +
                           "?user=dummy.supe&encPwd=dummy1", null);

//basic methods
string paramList = HttpGet(address + "/GetParameterList", authToken);
string qualityList = HttpGet(address + "/GetGradeList", authToken);
string approvalList = HttpGet(address + "/GetApprovalList", authToken);
string ratingTable = HttpGet(address + "/GetRatingTable" +
                             "?locId=02GA010" +
                             "&input=HG" +
                             "&output=QR",
                             authToken);
string dataList = HttpGet(address + "/GetDatasetsForLocation" +
                          "?locId=02GA010",
                          authToken);

//Time series changes in the last 30 hours
string changeSince = (DateTime.Now - TimeSpan.FromHours(30)).ToString(
                      "yyyy-MM-ddTHH:mm:ss.fffzzz");
string csv = HttpGet(address + "/GetTimeSeriesData" +
                     "?dataId=HG.Telemetry@02GA010" +
                     "&view=Public" +
                     "&changesSince=" + changeSince,
                     authToken);
```

```
//-------------------------------------------------------
// Here is the custom HttpGet function, in C#
//-------------------------------------------------------
public static string HttpGet(string url, string authToken)
{
    return HttpGet(url, authToken, System.Text.Encoding.UTF8);
}
```

```
public static string HttpGet(string url, string authToken,
                             System.Text.Encoding encoding)
{
    System.Net.HttpWebRequest request = null;
    System.Net.WebResponse response = null;
    System.IO.Stream responses = null;
    System.IO.StreamReader sr = null;
    try
    {
        request = (System.Net.HttpWebRequest)System.Net.WebRequest.Create(url);
        request.Method = "GET";
        if (!string.IsNullOrEmpty(authToken))
        {
            request.Headers.Add("AQAuthToken", authToken);
        }
        response = request.GetResponse();
        responses = response.GetResponseStream();
        sr = new System.IO.StreamReader(responses, encoding);
        string ret = sr.ReadToEnd();
        return ret;
    }
    catch (Exception e)
    {
        throw e;
    }
    finally
    {
        if (sr != null) { try { sr.Close(); } catch { } }
        if (responses != null) { try { responses.Close(); } catch { } }
    }
}
```