onset®

HOBOlink®

Web Services

Developer's Guide

HOBO®

onsetcomp.com

Mailing Address:
P.O. Box 3450
Pocasset, MA 02559-3450

Phone: 1-800-LOGGERS (1-800-564-4377) or 508-759-9500
Fax: 508-759-9100
Hours of Operation: 8 AM to 5 PM ET, Monday through Friday

E-mail: loggerhelp@onsetcomp.com

Main Onset Web site: www.onsetcomp.com

If you purchased the products through an Onset Authorized Dealer, you can also refer to www.hobohelp.com for support information.

Doc #: 12789-C

Contents

NOTES

Section 1: Overview

HOBOlink® Web Services offer the ability for third party developers and partners to write their own programs to extract HOBO® U30 data from the HOBOlink database.

HOBOlink exposes a set of SOAP web services hosted on Onset's remote servers. These web services are accessed through a third party software program (the "caller").

HOBOlink Web Services deliver customized datasets to the caller in a known XML format (SensorML) with the interface to the data (WSDL file) published at a known URL and the XML schema published by Onset. This relies on HOBOlink's underlying infrastructure where data is stored as individual readings, such that custom datasets can be extracted and sent to the caller.

Specific capabilities of HOBOlink Web Services include:

- Data from the logger is stored as individual entries in a data warehouse type of database (SOS server), rather than only in binary .dtf files. This provides a tremendous amount of extensibility for allowing customized access to a user's data, both public and private.
- Data is streamed to the caller in SensorML, an industry standard XML format. Publishing the XML schema allows developers to use any number of data binding tools that allow manipulation of the XML document using simple programming objects.
- Datasets returned to the caller can span multiple devices, launches, and wraps, as well as being constrained to only a small subset of a deployment.
- Datasets are customizable by time, device S/N, sensor S/N and/or measurement type.
- Authentication is achieved via a token (provided by Onset) that is passed to HOBOlink as part of the web services call. HOBOlink manages access to the various web services using these tokens. Tokens will be provided free to customers who purchase HOBO U30 devices.

If you wish to discuss The HOBO Remote Monitoring System and Web Services in more detail, please contact an Onset Computer Application Specialist at (800) 564-4377 or sales@onsetcomp.com.

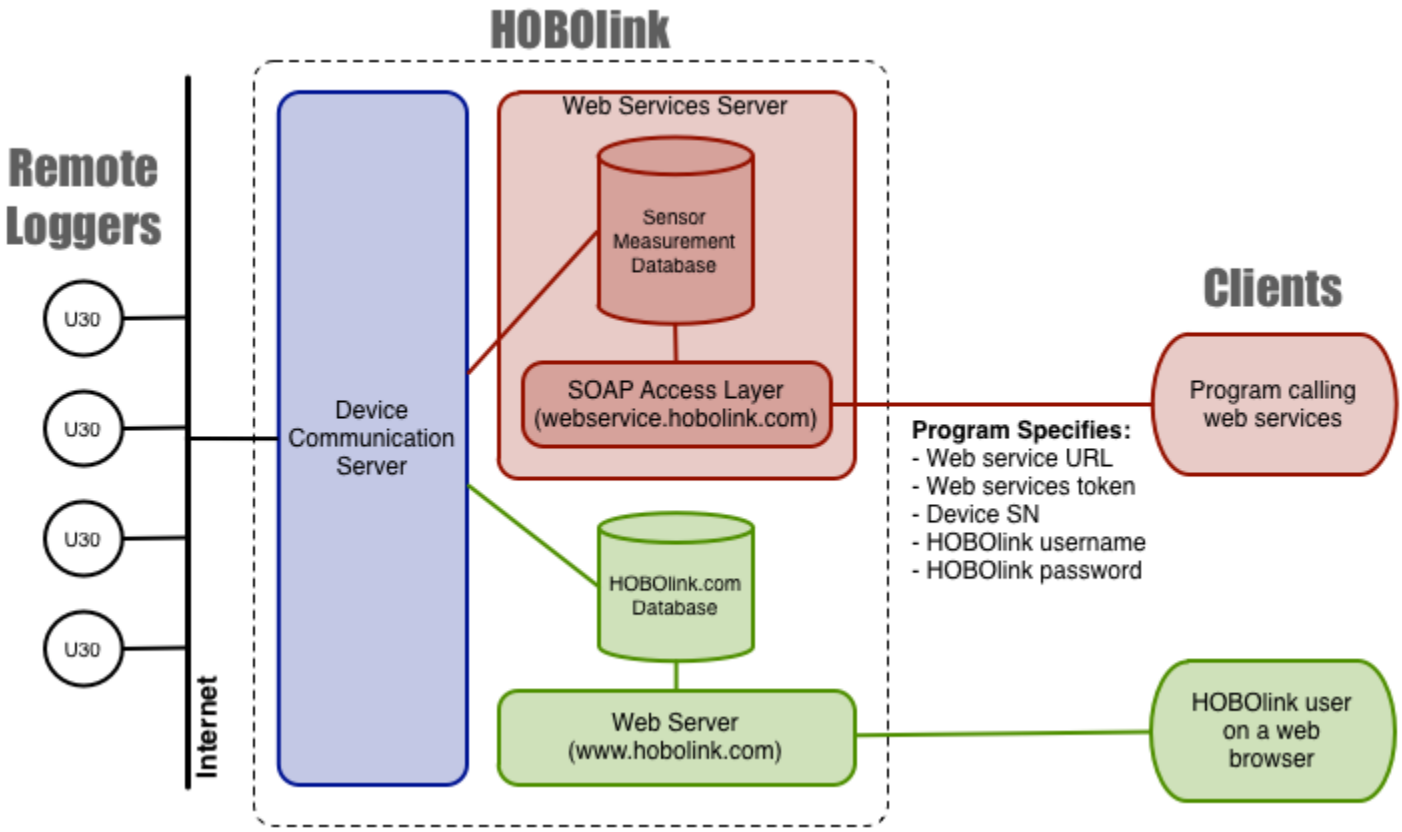
Requirements

- You should be a software engineer with a strong working knowledge of your programming language, its abilities, and its limitations.
- Ideally, you should have experience writing code that consumes Web Services.
- You should have a HOBOlink account and a U30 Remote Monitoring Station, or plans to purchase one in the near future.

Diagram

The following diagram shows the interaction between HOBOlink Web Services, clients, and devices, including:

- the difference between a user hitting HOBOlink and a program hitting the web services
- how the web services get data from the database, not directly from the devices
- the parameters the web service program needs to specify



Section 2: Tutorial

Step-by-Step Instructions

1. Get the Code

   The web service's methods are revealed through the WSDL file published by Onset at:

   https://webservice.hobolink.com/axis2/services/SensorObservationService?wsdl

   The WSDL is an XML representation of the web service's interface. The same WSDL is used for our test and production environments.

   NOTE: The WSDL may contain other web service offerings. You should not attempt to call these other services.

   Most programming languages and/or IDEs have tools available to help with this conversion.

   Run the program appropriate to your environment that converts the WSDL to usable code. This will vary depending on your programming language.

Tips	
Java	• We recommend using axis2 to generate Java code from the WSDL: Java - http://ws.apache.org/axis2/ In particular, check out the documentation that explains the command line tool and available plug-ins: http://ws.apache.org/axis2/1_4_1/reference.html http://ws.apache.org/axis2/tools/index.html • Let your IDE do the work for you. If there is a plug-in for your IDE listed on the axis2 tools page or in the documentation for your IDE, install & run it. We have successfully used both NetBeans and IntelliJ to create test clients for our web services. • Local java classes will be created that represent the web service interface.
C#	• See: C# - http://msdn.microsoft.com/en-us/library/7h3ystb6.aspx
PHP4	• See : PHP4 - http://www.nusphere.com/php_script/nusoap.htm
PHP5	• See : PHP5 - http://www.urdalen.no/wsdl2php/
Ruby	• See : Ruby - http://dev.ctor.org/soap4r

2. Contact Onset for Required Information

   Before you run client code, contact Onset Sales or Technical Support (1-800-LOGGERS) to obtain the necessary authentication and validation information. You will be provided with:

   - A device serial number and HOBOlink username/password, for use in testing
   - One generic validation token for use in testing
   - One individualized token you can use in your production environment with your purchased devices.

3. Write Client Code

   Referencing the web service interface generated in Step 1, write client code that calls the available methods. Your code needs to specify the web service URL, HOBOlink username, password, token, device serial number(s), and sensor serial number(s). Other details such as time period and other filters are also available through the interface code. It is up to you whether you want to use a time period or other data filters when calling the web service.

   The top-most method of the Sensor Observation Service is *GetObservationFull* which takes a request object as its argument. You will populate the request with the details listed above, before passing it to *GetObservationFull.* The details of this and all the objects contained in this web service are defined in the Sensor Observation Service section in Section 3.

   In your test code, use the test values provided by Onset along with the following endpoint URL:

   https://webservice.hobolink.com:89/axis2/services/SensorObservationService

   The test account should help you to get your basic framework up & running. You will not be able to run your code until you finish Step 4: Configure Client for SSL.

   Once your tests are working properly, modify your client code to request data from your device serial number and sensors, using your own HOBOlink account information and your individualized validation token. Make sure to also remove the port (:89) from your web service URL, which will allow you to call into our production environment.

4. Configure Client for SSL

   Getting your client code to connect to an SSL URL will vary depending on your code. Generally speaking, you will need to add the certificate from the web service server to a trusted certificate file, and then reference that from within your code.

5. Write SensorML Parsing Code

   Data will be returned to the caller in SensorML format. If necessary, Onset can provide sample Java data binding code to marshal SensorML into Java objects.

Guidelines

- Bad or missing sensors will record –888.88 as their reading. You should screen for this value.
- The maximum number of the measurements that will be returned in a single call to the getObservationRequest service is 20,000.
- There is a small chance that duplicate entries can exist for the same sensor at the same time. You should ensure that your code can handle duplicate measurements.
- Observations are returned to the caller in UTC, not the local time of the device.
- The web services do not use the special XML markup `CDATA` to encode the SensorML in the SOAP response. Some programming languages may need to do additional processing to unescape this response. In our testing, the Java and C# languages needed no additional processing.

Section 3: Reference

Sensor Observation Service

- The WSDL can be found at https://webservice.hobolink.com/axis2/services/SensorObservationService?wsdl
- The Sensor Observation Service endpoint is located at https://webservice.hobolink.com/axis2/services/SensorObservationService
- GetObservationFull(SosGetObservationRequestFull getObservationFull)
- Returns: String of Sensor Observation Service/O&M xml

Parameters

Hierarchy

The following illustration shows the relationship between parameters.

SosGetObservationRequestFull

- user
- serial numbers
- **TemporalFilter**
  - operator
  - **TimePeriod**
    - start
    - end
    - startIndet
    - endIndet
  - preset
- **sensors**
  - **SimpleSensor**
    - sensor serial number
    - measurement type
- **SpatialFilter**
- **ComparisonFilter**
  - operator
  - **SimpleSensor**
    - sensor serial number
    - measurement type
  - value
- **Authentication**
  - token
  - user
  - password

Parameter Information

Name	Description
SosGetObservationRequestFull	The object defining the request itself. Required: Y - Type: Object
• user	HOBOlink account user. If this parameter is populated, data will be returned for all devices that have been registered by that user. The serial_numbers parameter does not need to be populated. Either a list of serial number(s) or a user is required. Required: N - Type: String
• serial numbers	Serial numbers of the U30 devices. If this parameter is populated, data will be returned for the specified devices. The user parameter does not need to be populated. Either a list of serial number(s) or a user is required. Required: N - Type: Array of Objects
• **TemporalFilter**	Programming object that represents a time based filter of the data set. Required: N - Type: Object
o operator	The type of time based filter. Supported values are "Before", "After" and "During". Required: Y Type: String
o **TimePeriod**	Time Period for which the data is desired. Required: Y - Type: Object
▪ start	Start time for the data to be returned. Required for the After and During operators. Required: N - Type: Object
▪ end	End time for the data to be returned. Required for the Before and During operators. Required: N - Type: Object
▪ startIndet	Can be used in lieu of the start parameter to represent an indeterminate time. Supported value is "Now". Required: N - Type: String
▪ endIndet	Can be used in lieu of the end parameter to represent an indeterminate time. Supported value is "Now". Required: N - Type: String
o preset	Use to preset the time rather than using a TimePeriod object. Valid arguments are: "Past_24", "Past_1", "Since_Midnight" Required: N - Type: String
• **sensors**	The list of sensors to be returned in the data set. Required: N - Type: Array of Objects
o **SimpleSensor**	Programming object that represents a sensor. Required: N - Type: Object

▪ sensor serial number	Serial number of the sensor. Required: Y - Type: String
▪ measurement type	Measurement type of the sensor. Required: Y - Type: String
• **SpatialFilter**	
• **ComparisonFilter**	Programming object that represents a comparison based filter of the data set. Required: N - Type: Array of Objects
o operator	The type of comparison based filter. Supported values are “PropertyIsEqualTo”, “PropertyIsNotEqualTo”, “PropertyIsLessThan”, “PropertyIsGreaterThan”, “PropertyIsLessThanOrEqualTo” or “PropertyIsGreaterThanOrEqualTo”. Required: Y - Type: String
o **SimpleSensor**	Programming object that represents a sensor. Required: Y - Type: Object
▪ sensor serial number	Serial number of the sensor. Required: Y - Type: String
▪ measurement type	Measurement type of the sensor. Required: Y - Type: String
o value	The value to be filtered upon. Required: Y - Type: String
o value upper	Required: Y - Type: String
• **Authentication**	Required: Y - Type: Object
o token	Authenticates the caller as a valid CDS Full consumer. Contact Onset Computer for a free token. SSL must be used to ensure encryption of this information. Required: Y - Type: String
o user	HOBOlink account under which the specified serial number is registered. Required: N - Type: String
o password	Password for the specified HOBOlink account. SSL must be used to ensure encryption of this information. Required: N - Type: String

Error Codes

Configuration

- CFG-001 – Error loading logging properties file.
- CFG-002 – Error loading the webservice properties file.

Database

- DBM-001 – Database error validating token.
- DBM-002 – Database error retrieving communication plans by VAR.
- DBM-003 – Database error retrieving HOBOlink status.
- DBM-004 – Database error retrieving the user.
- DBM-005 – No active deployments for user.
- DBM-006 – Database error retrieving active deployments for user.
- DBM-007 – Database error retrieving the device by serial number.
- DBM-008 – Database error retrieving the communications plan.
- DBM-009 – Database error saving the device.
- DBM-010 – Database error retrieving the deployment by device.
- DBM-011 – No valid communication plans for the respective VAR.
- DBM-012 – No subscribers found.
- DBM-013 – Database error retrieving list of subscribers.
- DBM-014 – Database error retrieving subscriber.
- DBM-015 – Database error retrieving credit type.
- DBM-016 – Database error saving subscriber.
- DBM-017 – Database error retrieving subscriber's webservice.
- DBM-018 – Database error retrieving webservice.
- DBM-019 – Database error saving subscriber's webservice.
- DBM-020 – No text files for deployment.
- DBM-021 – Database error retrieving latest text file for deployment.

Authentication

- ATH-001 – Invalid token.
- ATH-002 – Invalid user.
- ATH-003 – Encryption algorithm error during authentication.
- ATH-004 – Invalid password.
- ATH-005 – Invalid device serial number.
- ATH-006 – Invalid communications plan.
- ATH-007 – Invalid status.
- ATH-008 – Device not owned by respective user.

Email

- EML-001 – Error sending plan renewal confirmation email.

I/O

- IOE-001 – Could not read text file.
- IOE-002 – Error communicating with Sensor Observation Service.
- IOE-003 – The Sensor Observation Service reported an error.

Validation

- VAL-001 – Time period start time is null.
- VAL-002 – Time period start time is in the future.
- VAL-003 – No data found in specified time range.
- VAL-004 – Device serial number or user name is required.
- VAL-005 – Unsupported preset time period.
- VAL-006 – Unsupported indeterminate value for time period.
- VAL-007 – A start time is required for time period with the “during” operator.
- VAL-008 – An end time is required for time period with the “during” operator.
- VAL-009 – A start time is required for time period with the “after” operator.
- VAL-010 – An end time is required for time period with the “before” operator.
- VAL-011 – Invalid time period operator.
- VAL-012 – An operator is required with a time period.
- VAL-013 – Sensor information is required with “greater than” filter operator.
- VAL-014 – The value is required with “greater than” filter operator.
- VAL-015 – Sensor information is required with “less than” filter operator.
- VAL-016 – The value is required with the “less than” filter operator.
- VAL-017 – Sensor information is required with “greater than or equal to” filter operator.
- VAL-018 – The value is required with the “greater than or equal to” filter operator.
- VAL-019 – Sensor information is required with “less than or equal to” filter operator.
- VAL-020 – The value is required with the “less than or equal to” filter operator.
- VAL-021 – Sensor information is required with “equal to” filter operator.
- VAL-022 – The value is required with the “equal to” filter operator.
- VAL-023 – Sensor information is required with “not equal to” filter operator.
- VAL-024 – The value is required with the “not equal to” filter operator.
- VAL-025 – Invalid filter operator.
- VAL-026 – An operator is required with a filter.

Consuming Web Services

Java - http://ws.apache.org/axis2/

C# - http://msdn.microsoft.com/en-us/library/7h3ystb6.aspx

PHP4 - http://www.nusphere.com/php_script/nusoap.htm

PHP5 - http://www.urdalen.no/wsdl2php/

Ruby - http://dev.ctor.org/soap4r

SensorML

http://www.opengeospatial.org/standards/sensorml

XML Data Binding

Onset has successfully generated data binding code with the Java toolset XMLBeans (http://xmlbeans.apache.org/). The SensorML schema was too complex for basic Java tools like Castor or JAXB. To date, we have not investigated any XML data binding tools for other programming languages.

Section 4 : Sample Code

Java

```
. . . . . .

// This line is required for SSL to work.  The cert file needs to contain
the certificate at https://webservice.hobolink.com

System.setProperty("javax.net.ssl.trustStore", CERT_FILE);

// Use the locator to setup the web service

SensorObservationServiceLocator locator = new
SensorObservationServiceLocator();

// The service is defined by its endpoint:
// dev:
//
https://webservice.hobolink.com:89/axis2/services/SensorObservationService
// stable:
//
https://webservice.hobolink.com/axis2/services/SensorObservationService

SensorObservationServicePortType service =

      locator.getSensorObservationServiceHttpSoap11Endpoint(new
      URL(WS_URL));

// Set up the request with serial numbers, authentication object,
// time period, and other filters if you like

SosGetObservationRequestFull request = new SosGetObservationRequestFull();

Authentication auth = new Authentication(PASSWORD, TOKEN, USERNAME);

request.setAuth(auth);

String[] serialNumbers = {DEVICE_SERIAL_NUMBER};

request.setSerialNumbers(serialNumbers);

Calendar start= Calendar.getInstance();

start.set(2008, Calendar.OCTOBER, 1);

TimePeriod time = new TimePeriod();

time.setStart(start);

TemporalFilter timePeriod = new TemporalFilter("after", "", time);

request.setTimePeriod(timePeriod);

String response = service.getObservationFull(request);

. . . . . .
```

C#, PHP4, PHP5, Ruby - Not available at this time

Section 5: Glossary

SOAP	Simple Object Access Protocol. A protocol for exchanging XML-based messages over computer networks.
WSDL	Web Services Description Language. Describes the web service interface. Typically used to develop web service clients.
Stub	Also known as skeleton code. A method of generating class files based off a web service definition file (WSDL). Most programming languages have SOAP tools designed to generate stub code from a WSDL file (e.g. wsdl2java).
Sensor Observation Service	Provides an API for managing deployed sensors and retrieving sensor data and specifically “observation” data.
Endpoint	A web service endpoint is a (referenceable) entity, processor, or resource to which web service messages can be addressed.
Marshal	The act of converting an XML document to and from a programming object.
Bind	The process of representing the information in an XML document as an object in computer memory.

Onset Computer Corporation
470 MacArthur Blvd.
Bourne, MA 02532