Data Portal User Guide

Revision A
December 2016
Part Number 08125

LIQUID ROBOTICS®

Data Portal User Guide

Trademarks

Patents

The Wave Glider mechanism is patented under US Patent Number 7,371,136 and other patents pending. All aspects of the Wave Glider product have been developed at private expense by Liquid Robotics, Inc.

Warranty

The information contained herein is subject to change without notice. The only warranties for Liquid Robotics, Inc. products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as a warranty. LRI, Inc. is not liable for technical or editorial errors or omissions contained herein.

Revision History

Document Part Number	Revision	Publication Date	Changes
08125	A	December 2016	This is the initial release of this document.

Table of Contents

Overview

Liquid Robotics' Wave Gliders deliver their data to shore via Iridium, cellular, wireless, and other communications mechanisms. On the shore side, the data is stored, replicated, backed up, processed, and delivered to applications.

The Liquid Robotics Data Portal is a mechanism for development partners and customers to retrieve data in an automatable way for a variety of purposes.

The Data Portal constructs meaningful URLs that describe the data request and retrieve the data in JSON or CSV format.

Credentials are required, and authenticated against, for access to specific data.

Data can be specified by vehicle, sensor type, date/time range and format (JSON or CSV). If no end date range is specified the URL will continuously stream data to the recipient as it is created, allowing for a near-real-time connection to the data as it is produced at sea.

Chapter 1. Using the Data Portal

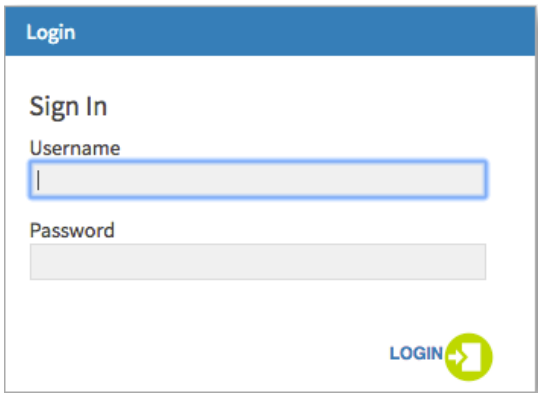
This chapter explains how you can:

- Access the Liquid Robotics Data Portal.
- Select vehicles, sensor data types, a data format, and a data time frame to generate a specific data retrieval URL.
- Download data.

Accessing the Data Portal User Interface

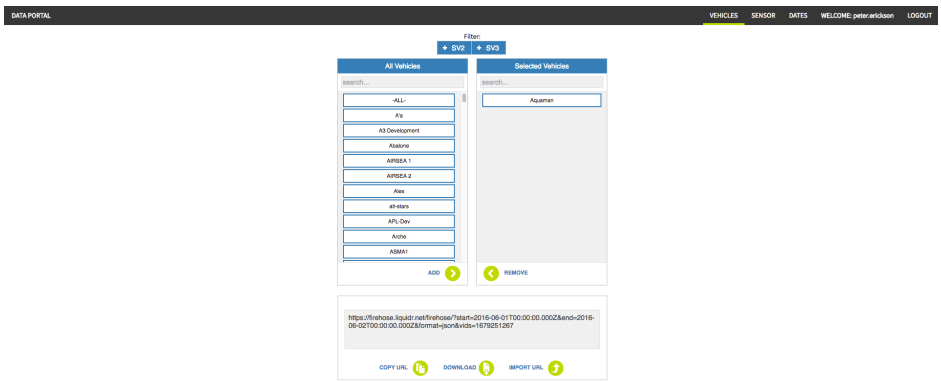
The Data Portal user interface can be accessed at: https://firehose.liquidr.net/DataPortal/login

Note that HTTPS is required. The server will not respond to HTTP requests. The Data Portal login page will prompt you for your username and password. Please use the credentials provided by Liquid Robotics customer support. These credentials are limited to the period of performance for the project.



Selecting Vehicles

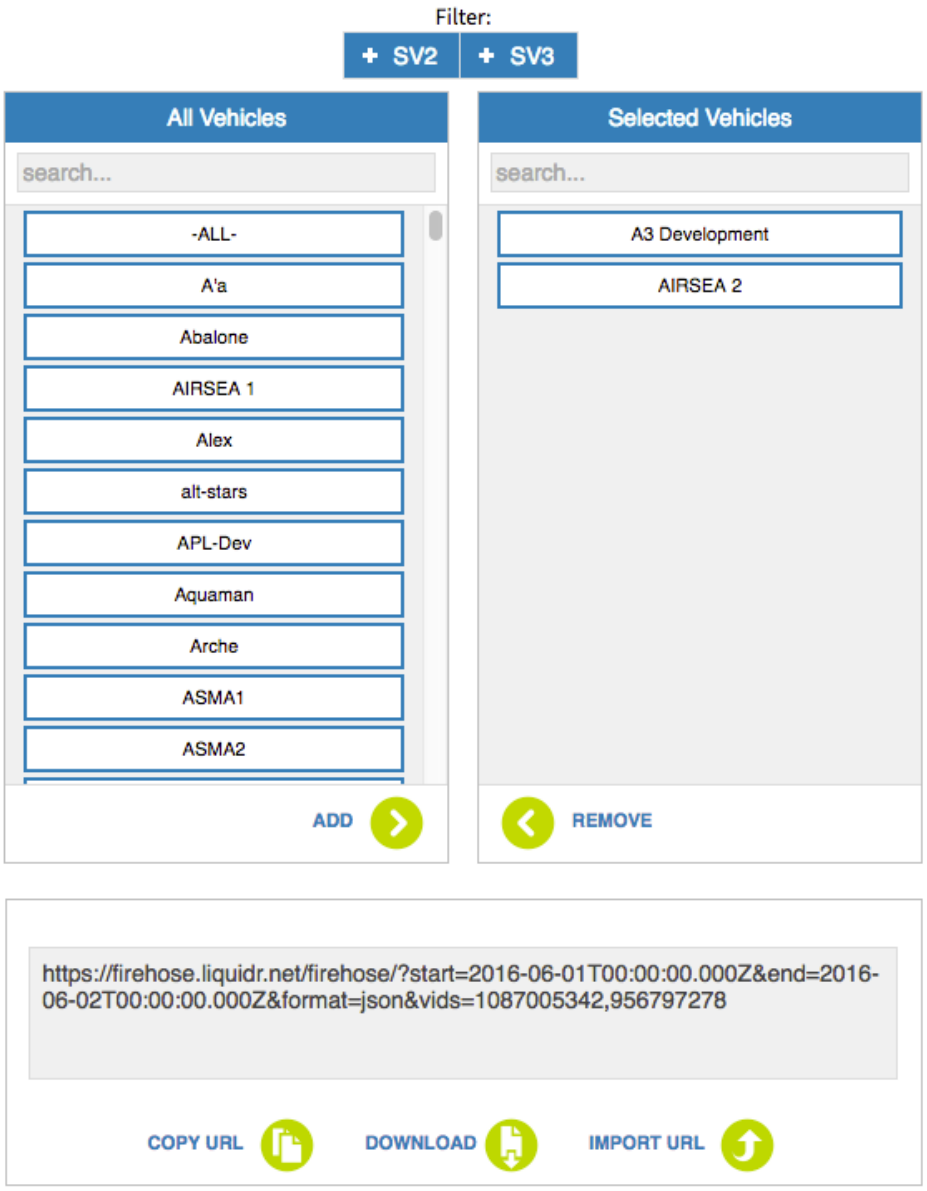
Upon successful login, the Data Portal **VEHICLES** screen will be displayed.



All SV2 and SV3 vehicles are selected by default as indicated by the **+ SV2** and **+ SV3** buttons with **-ALL-** vehicles in the **Selected Vehicles** column.

To select one or more specific vehicles for data retrieval, click on the vehicle name in the **All Vehicles** column and the button background color will change from white to green. In the following screen capture A3 Development and AIRSEA 2 have been selected.

By clicking the ADD button, the vehicles selected in the **All Vehicles** column are moved to the **Selected Vehicles** column. In the following screen capture A3 Development and AIRSEA 2 have been added to the **Selected Vehicles** column.

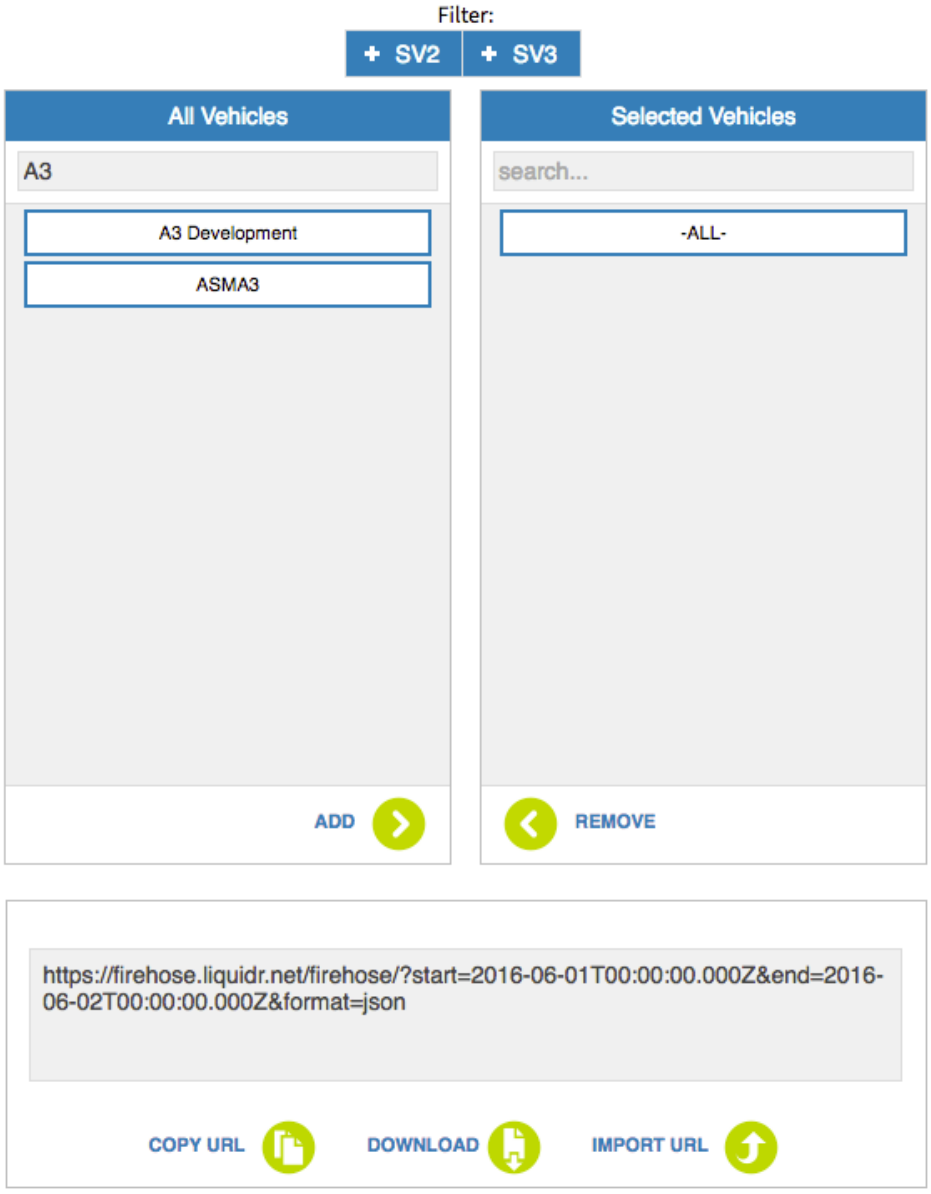


Note that the default -ALL- vehicles selection has been automatically moved to the **All Vehicles** column. You can also drag vehicles between the All Vehicles column and the Selected Vehicles column, rather than using the **ADD** and **REMOVE** buttons.

To limit the number of vehicles displayed in the **All Vehicles** column:

- You can filter by vehicle type (SV2 or SV3) by clicking on **+ SV2** or **+ SV3**. The + (plus) on the button will become a − (minus) and the background color will change from blue to white indicating that the vehicle type has been removed from the **All Vehicles** column. In the following screen capture SV2 vehicles have been filtered out of the **All Vehicles** column and all SV3 vehicles are selected in the **Selected Vehicles** column.

- You can also use the **search...** feature to find a specific vehicle. As you type in the vehicle name the list will become smaller until the vehicle you are searching for is the only vehicle displayed. The **search...** feature can be used to limit the number of vehicles displayed in either the **All Vehicles** column or the **Selected Vehicles** column. In the following screen capture A3 has been typed into the **search...** feature for the **All Vehicles** column:



NOTE: The **search...** feature is not case sensitive.

Once the vehicles you want to retrieve data from are listed in the **Selected Vehicles** column, open the sensor selection screen by clicking on **SENSOR** in the Data Portal toolbar:

Selecting Sensor Data Types and Format

The **SENSOR** screen opens with **-ALL-** sensor data selected in the + JSON (JSON) format.

By clicking on the **– CSV** the sensor data you select will be downloaded in the CSV format. You have the option to download sensor data in the JSON format or the CSV format, however you cannot download both formats using the same URL. You must generate a separate URL for each data format.

All sensor data types are available in both JSON and CSV format. You can download data from multiple (or ALL) sensors in JSON format using a single URL. With the CSV format, you must generate a separate URL for each sensor - you cannot download data from more than one sensor at a time in CSV format.

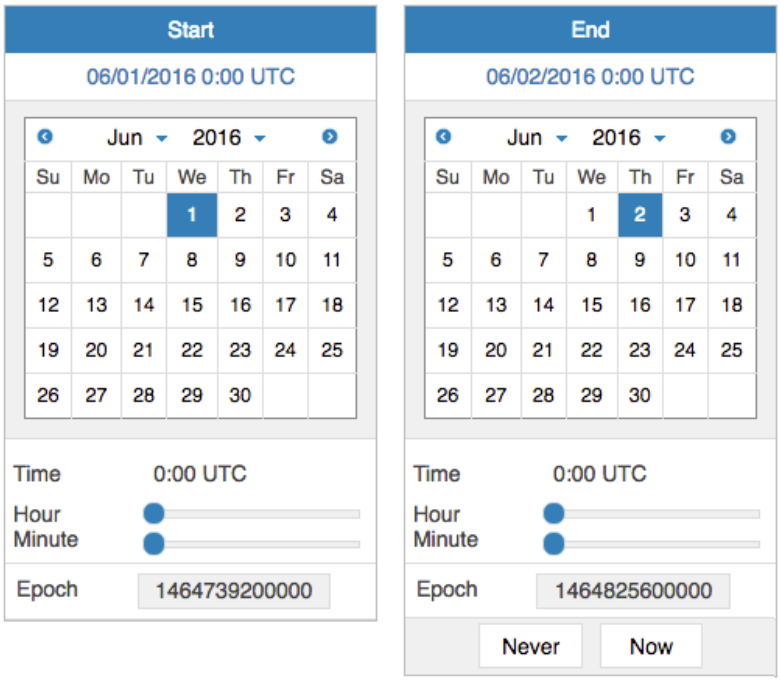
In the same way that you select vehicles in the **VEHICLES** screen, you can use the **ADD** and **REMOVE** buttons to move sensor data types between the **All Sensors** column and the **Selected Sensors** column.

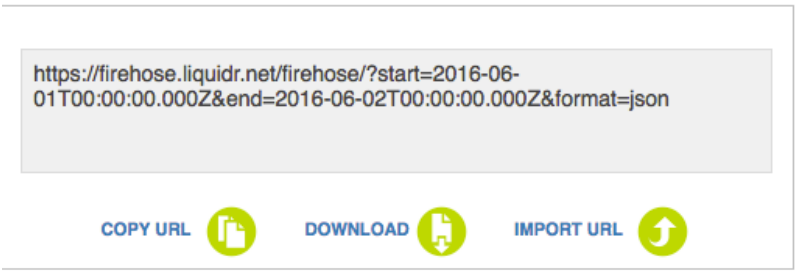
The **search...** feature in the **SENSOR** screen works the same way that it does in the **VEHICLES** screen. You can use the **search...** feature to locate specific sensor data types in the **All Sensors** column, or in the **Selected Sensors** column. As you type in the name of the sensor data type the list will become smaller until the sensor data type you are searching for is the only one displayed. The **search...** feature is not case sensitive.

Once the data format has been selected (JSON or CSV) and the sensor data types you want to retrieve are listed in the **Selected Sensors** column, click on **DATES** in the Data Portal toolbar to select the time frame for the data you want to retrieve:

Setting a Time Frame

The **DATES** screen opens with a time frame that includes the entire previous day. All time settings are in UTC.

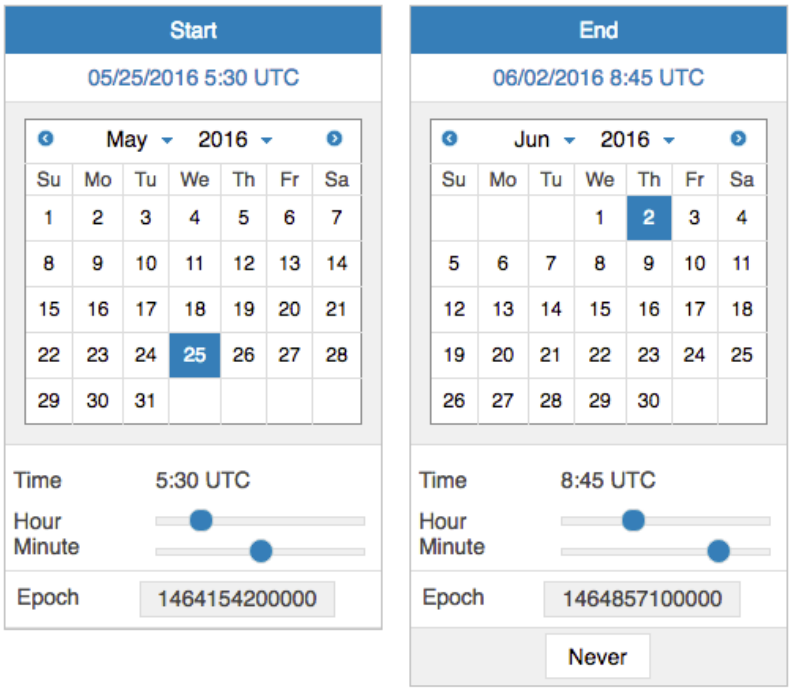


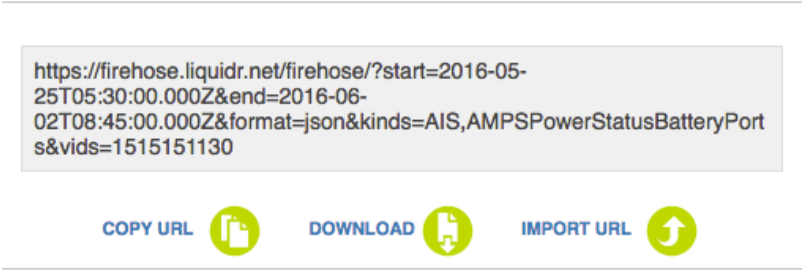


To set a time frame for the sensor data you selected:

- Use the **Start** calendar to set the date, then click and drag the hour and minute sliders to set the beginning of the time frame.
- Use the **End** calendar to set the date, then click and drag the hour and minute sliders to set the end of the time frame.

For example, in the following screen capture the time frame has been set to begin on May 25, 2016 at 5:30 UTC and end on June 2, 2016 at 8:45 UTC.





Liquid Robotics recommends selecting relatively short periods of time (a few days or less) initially, as large ranges (weeks to months) can take several minutes to several hours to generate a URL. For example, 6 months of AIS data from one Wave Glider vehicle takes approximately 30 minutes to download.

Note

By selecting **Now**, the end time will update to the current date, time, and minute.

By selecting **Never**, no end time is specified, the URL will not terminate and will instead continue to stream data as it becomes available.

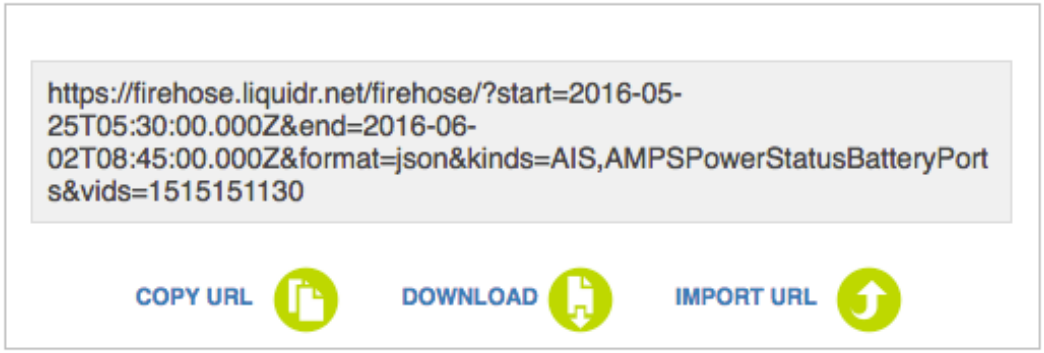
When setting a time frame for sensor data retrieval, you can set an end time in the future. For example, at noon you could set a start time in the past and an end time at 3PM - the Data Portal will retrieve data until 3 PM. You can also set a time frame for data retrieval that is entirely in the future - starting at a future time and ending at a later future time.

You cannot stream CSV data, only the JSON data format can be streamed.

At this point you have selected the vehicles, the sensor data types, the data format, and the time frame necessary to generate a functional data retrieval URL.

Downloading Data

After selecting the vehicles, sensor data types, data format, and time frame, a functional data retrieval URL is displayed in the bottom panel of the DATES screen:



Note how the URL is constructed, as it can be modified.

Start and end times are represented as ISO 8601 time (https://www.w3.org/TR/NOTE-datetime).

`format` specifies the output formatting (JSON or CSV).

`kinds`is a list of the sensor types that you selected on the **`sensors`**tab.

`vids` is a list of vehicle identification numbers.

You can add **`&doc`** to include the description of each field.

When a URL is properly modified, the selections in the user interface (vehicles, sensors, data time frame) will be updated according to the information in the URL.

Click **DOWNLOAD** and a JSON or CSV file containing the data you selected will be downloaded to your system. Note, you will be prompted for your login credentials before the data is downloaded.

You have the option to click **COPY URL** to copy the URL in the Data Portal user interface into your clipboard so you can paste it into a browser, or use it for some other data retrieval technique. Using the URL to access data will require authentication.

Note

URLs for CSV data cannot be copied.

To import a URL into the Data Portal, paste the URL into the URL box in the Data Portal user interface, then click **IMPORT URL**. You will receive a verification message, if the import is successful. When a URL is successfully imported, the selections in the user interface (vehicles, sensors, data time frame) will be updated according to the information in the URL.

Note

URLs for CSV data cannot be imported.

Chapter 2. Message Server

The Message Server allows you to query Liquid Robotics for messages sent from Wave Gliders. By modifying the URL provided by the Data Portal, you can set parameters that define the query. In response to a query, the Data Portal returns the messages as JSON text (relaxed or strict), or as comma separated values (CSV). There are lots of options [https://firehose.liquidr.net/firehose/welcome.jsp#Structure] for the kinds of queries you can build.

Getting Started

This basic query gives you the most recent messages: Get me the most recent messages [https://firehose.liquidr.net/firehose/] (the client connection remains open to receive messages as they arrive).

See Example Queries for examples of a wide range of message retrieval queries.

Example Queries

Retrieve These Messages	Example Queries
All incoming messages,starting from a given time in the past: (the start query parameter)	messages, starting from 10 minutes ago [https://firehose.liquidr.net/firehose/?start=-10m] messages starting two hours ago, strict JSON format [https://firehose.liquidr.net/firehose/?start=-2h&format=json] messages starting from yesterday [https://firehose.liquidr.net/firehose/?start=-1d] messages starting from September 1st 2015 at 3pm [https://firehose.liquidr.net/firehose/?start=2015-09-01T15.00] (potentially large dataset !)
All messages within a given time interval (the start and end query parameter)	messages between and hour and two hours ago, strict JSON format [?start=-2h&end=-1h&format=json] messages sent between 1pm and 5pm on September 2nd 2015 [https://firehose.liquidr.net/firehose/?start=2015-09-02T13.00&end=2015-09-02T17.00]
Messages of a particular type or types (the kinds query parameter)	Weather messages that came since yesterday, CSV [https://firehose.liquidr.net/firehose/?start=-1d&end=-1m&kinds=Weather&format=csv] Wave Glider and CTD, GenericAlarm and AIS messages in the last hour [https://firehose.liquidr.net/firehose/?start=-1h&kinds=Waveglider,CTD,GenericAlarm,AIS]
Outputs in various formats	Wave Glider messages from the last 2 hours, as loose JSON [https://firehose.liquidr.net/firehose/?start=-2h&end=-1m&kinds=Waveglider] Wave Glider messages from the last 2 hours, as strict JSON [https://firehose.liquidr.net/firehose/?start=-2h&end=-1m&kinds=Waveglider&format=json]

	Wave Glider messages from the last 2 hours, as CSV [https://firehose.liquidr.net/firehose/?start=-2h&end=-1m&kinds=Waveglider&format=csv]
Messages generated by a particular vehicle or vehicles, (the vids parameter)	Latest messages from vehicle: 771545959 and vehicle: 1531231814 [https://firehose.liquidr.net/firehose/?start=-30m&vids=771545959,1531231814] messages from vehicle: 771545959 of type GenericAlarm starting 12 hours ago [https://firehose.liquidr.net/firehose/?start=-12h&vids=771545959&kinds=GenericAlarm]
Messages coming from a particular geographical location (bounded box) (the lat-bounds and long-bounds parameters)	Latest messages from vehicles in Sunnyvale [https://firehose.liquidr.net/firehose/?start=-5m&lat-bounds=37.419795,37.379634&long-bounds=-122.055886,-121.973489] Weather messages in last day from Hawaii, strict JSON format [https://firehose.liquidr.net/firehose/?start=-1d&lat-bounds=20.281280,18.337336&long-bounds=-157.244166,-154.096584&kinds=Weather&format=json]

Structure of the Query

<URL to message server>/firehose/?<param1>&<param2=value>...<paramN=value>

Query Parameter	Description	Default value, if absent
start	start time for data (dates are ISO 8601 [https://en.wikipedia.org/wiki/ISO_8601]),	10 mins ago.
end	end time for data (dates are ISO 8601 [https://en.wikipedia.org/wiki/ISO_8601]),	never
kinds	comma separated list of message types you want in the data	all message types
vids	comma separated list of Unique IDs of the vehicles you want in the data	all vehicles
lat-bounds	comma separated min and max of the latitude from which you want data	all latitudes
long-bounds	comma separated min and max of the longitude from which you want data	all longitudes
format	the output format of the data, value either csvjson (for relaxed json, less noisy), or json (for strict json, works with all parsers), or csv for comma separated values.	csvjson
gzip (no value)	if present, the output will be sent compressed	uncompressed output
doc (no value)	if present, the documentation accompanying the messages will be sent, message type "documentation"	no documentation

Chapter 3. JSON Data Structure

JSON (JavaScript Object Notation www.json.org [http://www.json.org/]) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of JavaScript [http://javascript.crockford.com/], Standard ECMA-262 3rd Edition - December 1999 [http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf].

JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON standard used: Java JSON implementation in Java EE 7.

Java EE 7 supports:

www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf [http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf]

General - Wave Glider Metadata

The JSON file begins with a Wave Glider metadata summary, for example:

```
[{
     "kind": "wgmeta",
     "vid": "1718503568",
     "name": "SV3-001",
     "avatar": "http://data.liquidr.com/icon/SV3-001.png",
     "auth": "lriops | pitcairn",
     "ip": "10.20.32.85",
     "model": "SV3",
     "imei": "300434060700180",
     "RV": "1.0.2",
     "build": "Regulus-2015-05-14T17:58:23.469Z",
     "buildnumber": "12",
     "systembuild": "true"
}],
```

The metadata description is as follows:

- `kind` - The type of data.
- `vid` - The vehicle identification number.
- `name` - The name of the vehicle.
- `avatar` - A URL to an image that can be used as an icon to represent the vehicle.
- `auth` - A list of permissions for who has authority to view the data.
- `ip` - The IP address of the vehicle Web page.
- `model` - The vehicle model name.

- `imei` - The unique identifier for the Iridium modem in the vehicle.
- `RV` - The revision number for the onboard firmware.
- `build` - The build identifier for the onboard firmware.
- `buildnumber` - The build number for the onboard firmware.
- `systembuild` - A boolean for whether this is a system or a standalone module. True = system.

Appendix A. Java Code to Ensure a Secure Connection

Use the following code to always connect to HTTPS to secure user authentication credentials:

```
/*
 * THE SOFTWARE PRODUCT IS PROVIDED “AS IS” WITHOUT REPRESENTATIONS OR WARRANTIES OF ANY KIND. TO THE
 * FULLEST EXTENT PERMISSIBLE UNDER APPLICABLE LAW, LRI SPECIFICALLY DISCLAIMS ALL WARRANTIES, CONDITIONS,
 * AND REPRESENTATIONS REGARDING THE SOFTWARE PRODUCT, INCLUDING WITHOUT LIMITATION ANY WARRANTY OF
 * FITNESS FOR ANY PARTICULAR PURPOSE, MERCHANTABILITY, TITLE, QUIET ENJOYMENT, OR NON-INFRINGEMENT.
 *
 * Copyright 2016 Liquid Robotics
 */
package logintest;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.Charset;
import java.util.Base64;

/*
 * Always connect to https to secure user authentication credentials.
 */
public class LoginTest {
    /**
     * @param args the command line arguments
     *
     */
    public static void main(String[] args) {
        final String defaultURL = "https://mds.liquidr.net/firehose/?start=now&kinds=Waveglider"; // Stream from current date/time
        final String defaultUser = "YourUser";
        final String defaultPassword = "YourUserPassword";
        String url;
        String user;
        String password;

        if (args.length != 0 && args.length != 3) {
            System.err.println("Usage: LoginTest [URL User Password]");
            System.exit(1);
        }
        if (args.length == 3) {
            url = args[0];
            user = args[1];
            password = args[2];
        } else {
            url = defaultURL;
            user = defaultUser;
            password = defaultPassword;
        }
        try {
            byte c[] = new byte[1];
            int readCount;

            // Attempt to connect
            InputStream is = openAuthenticatedConnection(url, user, password);

            // Connection established, start reading stream
            while ((readCount=is.read(c)) >= 0) {
                if (readCount > 0) {
                    System.out.print(new String(c, Charset.forName("ISO-8859-1"))); System.out.flush();
                } else {
                    // Should the read return 0 bytes read
                    System.out.println("sleeping...");
                    Thread.sleep(1000);
                }
            }
            System.err.println("EOF on input stream");
        } catch(Exception e) {
            System.err.println("Login exception: " + e);
        }
        System.exit(0);
    }
    public static InputStream openAuthenticatedConnection(String urlStr, String username, String password) throws IOException {
       URL url = new URL(urlStr);
       System.err.println("BasicAuthClient: Attempt to authenticate on " + url + " for user " + username);

       String authString = username + ":" + password;
       byte[] authEncBytes = Base64.getEncoder().encode(authString.getBytes());
       String authStringEnc = new String(authEncBytes);
```

```
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setUseCaches(false);
        connection.setReadTimeout(0);
        //connection.setConnectTimeout(1000*60*15);
        connection.setRequestProperty("Authorization", "Basic " + authStringEnc);
        if (connection.getResponseCode() == HttpURLConnection.HTTP_OK) {
            System.err.println("User: " + username + " authenticated");
            return connection.getInputStream();
        } else {
            throw new IOException("Connection or authentication exception: Http Status code (" + connection.getResponseCode() + ")");
        }
    }
}
```

Note

This code can be downloaded from Liquid Robotics Open Oceans Portal. Go to <URL> and log in with credentials provided by Liquid Robotic. Look for <file name> in the User Documentation > Data Portal section.

Appendix B. Data Descriptions

AIS

Name	Units/Type	Description
Latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
Longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
Time	Milliseconds	The time at which this measurement was made
AccessionTime	Milliseconds	The time at which this measurement was added to the shore side data repository.
Altitude		
COG	Degrees True	Course over ground
Heading	Degrees True	Heading
MMSI		MMSI
NavigationStatus		Navigation Status
PacketSource		Packet source
ROT		Rate of turn (ROT_AIS)
SOG	Knots	Speed over ground. The value 102.2 is used to denote all speeds 102.2 kts and higher
Undefined		

Example:


```
{"kind":"AIS","vid":"316477414","latitude":"28.654543333333333","longitude":"-88.47274",
"time":"1454803200000","accessionTime":"1454803329417","COG":"203.0","heading":"165.0","MMSI":"538004433",
"navigationStatus":"3","packetSource":"18","ROT":"-1","SOG":"0.0","undefined":"1970-01-01T00:00Z"}],
```


MDABroadBand (Level 1 BB Report)

Name	Units/Type	Description
latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
time	milliseconds	The time at which this measurement was made
accessionTime	milliseconds	The time at which this measurement was added to the shore side data repository.
altitude		
bands	Array of band reports	The band reports in the packet.
bearingTolerance		+/-10 (Debug messages only.) Precision tolerance of bearing in degree in integer.
boardId	Integer	ID of the hardware board that transmits the packet.
dataFormat	Integer	Data format code of the message.
dataMode	0 or 1	Data mode of the message. 0 for regular mode, 1 for debug.

filterLevel	0 or 1 or 2	(Debug messages only.) The level of the filters applied to the data. 0 for raw 1 for NB Bearing Filter 2 for CrossedBands Filter 3 for AIS Filter.
numberOfBands	Integer	The total number of bands included in the packet.
sequenceNumber	Integer	Sequential number for tracking purpose.
taskId	Integer	ID of the software task that transmits the packet.
undefined		
vehicleName	String	Name of the vehicle sensing the packet.
versionId	Number	Version of the message format.

Example:


```
[{"kind":"MDABroadband","vid":"751882248","latitude":"20.015583333333332","longitude":"-156.066635",
"time":"1454770894925","accessionTime":"1454770894926","bands":
[{"ambientNoise":"66","bandNumber":"2","bearing":"146.25","bearingRate":"0.0","contactConfidence":"0",
"contactId":"194","contactStatus":"0","detectionStatistics":"0.0","latitude":"20.015423333333334",
"longitude":"-156.06661333333332","mmsi":"0","time":"1454770772"},{"ambientNoise":"62","bandNumber":"3",
"bearing":"116.71875","bearingRate":"0.0","contactConfidence":"0","contactId":"22","contactStatus":"0",
"detectionStatistics":"0.0","latitude":"20.015423333333334","longitude":"-156.06661333333332","mmsi":"0",
"time":"1454770772"},{"ambientNoise":"55","bandNumber":"5","bearing":"248.90625","bearingRate":"0.0",
"contactConfidence":"0","contactId":"252","contactStatus":"0","detectionStatistics":"0.0",
"latitude":"20.015423333333334","longitude":"-156.06661333333332","mmsi":"0","time":"1454770772"},
{"ambientNoise":"55","bandNumber":"6","bearing":"257.34375","bearingRate":"0.0","contactConfidence":"0",
"contactId":"36","contactStatus":"0","detectionStatistics":"0.0","latitude":"20.015423333333334",
"longitude":"-156.06661333333332","mmsi":"0","time":"1454770772"}],"bearingTolerance":"0","boardId":"0",
"dataFormat":"1","dataMode":"0","filterLevel":"0","numberOfBands":"4","sequenceNumber":"3464","taskId":"0",
"undefined":"1970-01-01T00:00Z","vehicleName":"SV3-063","versionId":"1"}],
```


MDA Health Status

Name	Units/Type	Description
latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
time	milliseconds	The time at which this measurement was made
accessionTime	milliseconds	The time at which this measurement was added to the shore side data repository.
alarmCode	Integer	Code of the specific error.
altitude		
boardId	Integer	ID of the hardware board that transmits the packet.
commandsReceived	Integer	Number of commands received by the sensor.
dataFormat	Integer	Data format code of the message.
freeMemorySpace	Integer	The amount of free disk space available in MB.
functionStatus		Flags indicating the operational state of various functional blocks. 1 = enabled 0 = disabled 0 Signal Processing; 1 Data Logging Requested Setting;

		2 FIFO Logging Mode (Delete oldest files when disk is below a specified free space); 3 Data Logging Actual Setting; 4 Trip Wire Detection; 5 Noise Excision Processing
level1UpdateRate	Integer	The data rate of the comprehensive reports.
level2UpdateRate	Integer	The data rate of the level 2 contact reports.
mACAddress	String	MAC address.
sensorSWVersion	String	Software version of the sensor.
sequenceErrorCountCompass	Integer	Number of times a compass data flow sequence error has occurred.
sequenceErrorCountHydrophone	Integer	Number of times a hydrophone data flow sequence error has occurred.
sequenceErrorCountStatus	Integer	Number of times a status sequence error has occurred.
sequenceNumber	Integer	Sequential number for tracking purpose.
sPRebootCount	Integer	Number of times the SP has rebooted.
sPRestartCount	Integer	Number of times the SP restarted without the OS rebooting.
sPShutdownCount	Integer	Number of times the SP has been commanded to shutdown.
statusFlag	String	The status flag.
supplyVoltage	Volts	Voltage supplied to sensor.
taskId	Integer	ID of the software task that transmits the packet.
timestamp		
undefined		
uptime	Integer	Minutes since sensor OS boots.
vehicleName	String	Name of the vehcile sensing the packet.
versionId	Number	Version of the message format.

Example:


```
[{"kind":"MDAHealthStatus","vid":"1718503568","latitude":"-19.418645","longitude":"-126.92908",
"time":"1454718327163","accessionTime":"1454718327164","alarmCode":"1","boardId":"0",
"commandsReceived":"16723","dataFormat":"5","freeMemorySpace":"1806","functionStatus":"0x35",
"level1UpdateRate":"120","level2UpdateRate":"60","mACAddress":"0:15:C9:29:1F:85","sensorSWVersion":"0x20D",
"sequenceErrorCountCompass":"1","sequenceErrorCountHydrophone":"1","sequenceErrorCountStatus":"1",
"sequenceNumber":"1","sPRebootCount":"37","sPRestartCount":"64","sPShutdownCount":"21",
"statusFlag":"8800:F0:1F80:0:0:0:3C:40:0:20:0:0","supplyVoltage":"15023","taskId":"0",
"timestamp":"1454718318","undefined":"1970-01-01T00:00Z","uptime":"65535","vehicleName":"SV3-001",
"versionId":"1"}],
```


MDA Narrow Band (L2 - NB Report)

Name	Units/Type	Description
latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude

longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
time	milliseconds	The time at which this measurement was made
accessionTime	milliseconds	The time at which this measurement was added to the shore side data repository.
altitude		
bearingTolerance	+/-10	(Debug messages only.) Precision tolerance of bearing in degree in integer.
boardId	Integer	ID of the hardware board that transmits the packet.
contacts	Array of contact reports	The contact reports in the packet: Bearing, bearing rate, contact confidence, contact ID, contact length, contact status, latitude, longitude, mmsi, nbDetStat, nbFc, nbFcExt, numbNbSignals, time.
dataFormat	Integer	Data format code of the message.
dataMode	0 or 1	Data mode of the message. 0 for regular mode; 1 for debug.
filterLevel	0 or 1 or 2	(Debug messages only.) The level of the filters applied to the data. 0 for raw. 1 for NB Bearing Filter. 2 for CrossedBands Filter. 3 for AIS Filter.
numberOfContacts	Integer	The total number of contact reports included in the packet.
sequenceNumber	Integer	Sequential number for tracking purpose.
taskId	Integer	ID of the software task that transmits the packet.
undefined		
vehicleName	String	Name of the vehicle sensing the packet.
versionId	Number	Version of the message format.

Example:


```
[{"kind":"MDANarrowband","vid":"1145967850","latitude":"19.99539","longitude":"-156.06822166666666",
"time":"1454630538772","accessionTime":"1454969203049","bearingTolerance":"0","boardId":"0","contacts":
[{"bearing":"324.84375","bearingRate":"0.0","contactConfidence":"0","contactId":"5315","contactLength":"0",
"contactStatus":"0","latitude":"19.995686666666668","longitude":"-156.06832","mmsi":"0","nbDetStat":"6.75",
"nbFc":"3846.0059","nbFcExt":"9.725491","numbNbSignals":"10","time":"1454630537"},{"bearing":"241.875",
"bearingRate":"0.0","contactConfidence":"0","contactId":"10872","contactLength":"0","contactStatus":"0",
"latitude":"19.995686666666668","longitude":"-156.06832","mmsi":"0","nbDetStat":"3.75","nbFc":"78.95628",
"nbFcExt":"0.4392157","numbNbSignals":"10","time":"1454630537"}],"dataFormat":"2","dataMode":"0",
"filterLevel":"0","numberOfContacts":"2","sequenceNumber":"21630","taskId":"0",
"undefined":"1970-01-01T00:00Z","vehicleName":"SV3-002","versionId":"1"}],
```


MDA Tripwire Event (TW Report)

Name	Units/Type	Description
latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
time	milliseconds	The time at which this measurement was made
accessionTime	milliseconds	The time at which this measurement was added to the shore side data repository.
altitude		
boardId	Integer	ID of the hardware board that transmits the packet.

contactID	Integer	Contact ID which causes the trip event.
contactLength	Seconds	(Debug messages only.) Duration of the contact that causes the trip wire event.
dataFormat	Integer	Data format code of the message.
dataMode	0 or 1	Data mode of the message. 0 for regular mode. 1 for debug.
frames	Array of Frames	(Debug messages only.) The frames in the message.
numberFrames	Integer	(Debug messages only.) Number of frames that contain both BB and NB.
reportType	0 (initial trip). or 1 (continual trip).	(Debug messages only.) Type of the trip wire report.
sequenceNumber	Integer	Sequential number for tracking purpose.
taskId	Integer	ID of the software task that transmits the packet.
timeSecsFirstFrame		
timestamp		
timestampLobCrossing		
triplineCrossed	Float between 0.0 and 360.0	Bearing from the wave glider to the target which crosses the trip line.
triplineDirection	0 (clockwise) or 1 (counterclockwise)	Direction that the target is going when the line is tripped.
undefined		
vehicleName	String	Name of the vehcile sensing the packet.
versionId	Number	Version of the message format.

Example:


```
[{"kind":"MDATripwireEvent","vid":"751882248","latitude":"20.026976666666666",
"longitude":"-156.06921666666668","time":"1454638902232","accessionTime":"1454969203049","boardId":"0",
"contactID":"1005","contactLength":"0","dataFormat":"3","dataMode":"0","numberFrames":"0","reportType":"0",
"sequenceNumber":"0","taskId":"0","timeSecsFirstFrame":"0","timestamp":"1454638893",
"timestampLobCrossing":"946705553","triplineCrossed":"45.0","triplineDirection":"0",
"undefined":"1970-01-01T00:00Z","vehicleName":"SV3-063","versionId":"1"}],
```


MDA Tripwire Settings

Name	Units/Type	Description
latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
time	milliseconds	The time at which this measurement was made
accessionTime	milliseconds	The time at which this measurement was added to the shore side data repository.
altitude		
boardId	Integer	ID of the hardware board that transmits the packet.
dataFormat	Integer	Data format code of the message.
numberOfLines	Integer	Total number of lines included in this message.

sequenceNumber	Integer	Sequential number for tracking purpose.
taskId	Integer	ID of the software task that transmits the packet.
timestamp		
triplines	Array of triplines	The triplines included in the message.
undefined		
vehicleName	String	Name of the vehcile sensing the packet.
versionId	Number	Version of the message format.

Example:


```
[{"kind":"MDATripwireSettings","vid":"1145967850","latitude":"19.988956666666667",
"longitude":"-156.06646666666666","time":"1454705628428","accessionTime":"1454705628429","boardId":"0",
"dataFormat":"4","numberOfLines":"4","sequenceNumber":"0","taskId":"0","timestamp":"1454705627",
"triplines":[{"bearing":"45.0","direction":"3","lineNumber":"1"},{"bearing":"45.0","direction":"3",
"lineNumber":"1"},{"bearing":"45.0","direction":"3","lineNumber":"1"},{"bearing":"45.0","direction":"3",
"lineNumber":"1"}],"undefined":"1970-01-01T00:00Z","vehicleName":"SV3-002","versionId":"1"}],
```


Wave Glider

Name	Units/Type	Description
latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
time	milliseconds	The time at which this measurement was made
accessionTime	milliseconds	The time at which this measurement was added to the shore side data repository.
altitude		
currentBearing	degrees	Current bearing
currentSpeed	knots	Current speed
distanceOverGround	m	The distance travelled over ground since the last RobotTelemetryReport. This is computed by sampling the GPS coordinates much more frequently than from one report to another. Comparing this to the distance between consecutive reports gives a rough measure of how 'wiggly' the course of the Wave Glider is.
floatBatteryLowAlarm		Float battery low alarm
floatLeakAlarm		Float leak alarm
floatPressureThresholdExceededAlarm		Float pressure threshold exceeded alarm
floatRebootAlarm		Float reboot alarm
floatTempThresholdExceededAlarm		Float temperature threshold exceeded alarm
floatToSubCommsAlarm		Float to sub comms alarm
gpsNotFunctioningAlarm		GPS not functioning alarm
headingDesired	°	
headingSkew	Δ°	The difference between the true direction travelled over the ground and the direction of thrust (headingSub). Think of it as a measure of how hard the Wave Glider is fighting the current.

headingSub	°	The direction of thrust (usually the heading of the submerged part of the Wave Glider)
internalVehicleCommAlarm		Internal vehicle comm alarm
overCurrentAlarm		Over current alarm
payloadErrorConditionAlarm		Payload error condition alarm
speedToGoal	knots	Speed-to-goal
subLeakAlarm		Sub leak alarm
subPressureThresholdAlarm		Sub pressure threshold alarm
subRebootAlarm		Sub reboot alarm
subTempThresholdExceededAlarm		Float sub temperature threshold exceeded alarm
subToFloatCommsAlarm		Sub to float comms alarm
targetWaypoint		
tempSub	˚C	Water temperature measured in the submerged glider unit. It is usually at a depth of 7 meters. This sensor measures integer number of degrees.
totalPower	milliwatt-hours	Total power available in the bstteries
umbilicalFaultAlarm		Umbilical fault alarm
undefined		
waterSpeed	knots	Water speed

Example:


```
{"kind":"Waveglider","vid":"751882248","latitude":"20.01571","longitude":"-156.06844666666666",
"time":"1454630452000","accessionTime":"1454969203049","currentBearing":"0.0","currentSpeed":"0.0",
"distanceOverGround":"33","floatBatteryLowAlarm":"false","floatLeakAlarm":"false",
"floatPressureThresholdExceededAlarm":"false","floatRebootAlarm":"false",
"floatTempThresholdExceededAlarm":"false","floatToSubCommsAlarm":"false","gpsNotFunctioningAlarm":"false",
"headingDesired":"197","headingSkew":"0.0","headingSub":"197","internalVehicleCommAlarm":"false",
"overCurrentAlarm":"false","payloadErrorConditionAlarm":"false","speedToGoal":"0.0",
"subLeakAlarm":"false","subPressureThresholdAlarm":"false","subRebootAlarm":"false",
"subTempThresholdExceededAlarm":"false","subToFloatCommsAlarm":"false","targetWaypoint":"68",
"tempSub":"26","totalPower":"65535","umbilicalFaultAlarm":"false","undefined":"1970-01-01T00:00Z",
"waterSpeed":"0.468"}],
```


Waves 1

Name	Units/Type	Description
latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
time	milliseconds	The time at which this measurement was made
accessionTime	milliseconds	The time at which this measurement was added to the shore side data repository.
altitude		
Dirp	degrees	Direction of significant waves
Hs	Meters	Significant wave height

Tav	Seconds	Average period of all waves
Tp	Seconds	Period between significant waves
undefined		

Example:


```
[{"kind":"Waves","vid":"1765014915","latitude":"-23.613086666666668","longitude":"-42.98546666666667",
"time":"1454292000000","accessionTime":"1454975125384","Dirp":"71.72143264995904","Hs":"2.0161909686196724",
"Tav":"5.086206896551724","Tp":"5.565217391304348","undefined":"1970-01-01T00:00Z"}],
```


Waves 2

There are currently two sensors that can generate a wave motion report: the Datawell MOSE sensor (http://www.datawell.nl) and the Liquid Robotics GPS Waves sensor. You can distinguish between the two using the sensorType field.

Name	Units	Description
Latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
Longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
Time	Milliseconds	The time at which this measurement was made
AccessionTime	Milliseconds	The time at which this measurement was added to the shores ide data repository.
Altitude		
Dirp	Degrees	Direction of significant waves / Dominant Direction
Fs	Hz	Sampling frequency. (GPS Only)
Hs	meters	Significant wave height
SamleGaps		Number of missing samples. (GPS only)
Samples		N umber of samples used (GPS only)
sensorType		Name signifying which Wave Sensor made the report. Either the DataWell Mose sensor or the GPS Sensor.
Tav	Seconds	Average period of all waves (Mose only)
Tp	Seconds	Period between significant waves / Peak Period

Example:


```
{"kind":"Waves","vid":"102740746","latitude":"37.41096666666667","longitude":"-122.00477",
"time":"1457496001000","accessionTime":"1458847797392","Dirp":"9999.0","Fs":"4.000557777926567",
"Hs":"1.004491240113532","SampleGaps":"0","Samples":"7201","sensorType":"GpsSensor","Tav":"-1.0",
"Tp":"4.3383985924179","undefined":"1970-01-01T00:00Z"}],
```


Weather & Wave Glider Meta Data

Name	Units/Type	Description
Latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
Longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue

Time	Milliseconds	The time at which this measurement was made
AccessionTime	Milliseconds	The time at which this measurement was added to the shores ide data repository.
Altitude		
AvgWindDirection		
AvgWindSpeed	Knots	Average Wind Direction
MaxWindSpeed	Knots	Average Wind Speed
Pressure	Millibars	Air Pressure
StdDevWindDir		Standard Deviation of Wind Direction
StdDevWindSpeed		Standard Deviation of Wind Speed
Temperature	Degrees C	Air Temperature
Undefined		

Weather Example:


```
{"kind":"Weather","vid":"751882248","latitude":"20.014701666666667","longitude":"-156.06870166666667",
"time":"1454630400000","accessionTime":"1454969203049","avgWindDirection":"259.3","avgWindSpeed":"6.8",
"maxWindSpeed":"10.1","pressure":"1014.7","stdDevWindDir":"7.5","stdDevWindSpeed":"1.4","temperature":"23.6",
"undefined":"1970-01-01T00:00Z"}],
```


Wave Glider Meta Data Example:


```
[[{"kind":"wgmeta","vid":"751882248","name":"SV3-063","avatar":"http://data.liquidr.com/icon/SV3-063.png",
"auth":"lriops | islandr","ip":"10.20.32.94","model":"SV3","imei":"300434060620310","RV":"1.0.3",
"build":"Regulus-2015-11-16_19-45-29","buildnumber":"33","systembuild":"true"},
```


ADCP-C2

Name	Units	Description
Latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
Longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
Time	Milliseconds	The time at which this measurement was made
AccessionTime	Milliseconds	The time at which this measurement was added to the shore side data repository.
Altitude		
DepthDirectionSpeed	m/s & degrees	Bin depth, direction and speed
Undefined		

Example:


```
kind:ADCP-C2,vid:1765014915,latitude:-25.461026666666665,longitude:-42.851881666666664,time:1446340316000,
accessionTime:1455747989298,depthDirectionSpeed:6.2200,25.7100,0.2397,10.2200,51.8056,0.1730,14.2200,95.4923,
0.1567,18.2200,93.8460,0.1193,22.2200,101.9207,0.0920,26.2200,117.0835,0.0988,30.2200,120.4111,0.1067,34.2200,
113.1314,0.1120,38.2200,118.0092,0.1065,42.2200,116.5650,0.1073,46.2200,123.6901,0.1046,50.2200,119.6237,
0.1173,54.2200,121.7014,0.1199,58.2200,129.1605,0.1251,62.2200,126.3844,0.1416,66.2200,129.0046,0.1557,
70.2200,129.5830,0.1648,74.2200,122.6405,0.1817,78.2200,122.2889,0.1928,82.2200,123.5412,0.2136,86.2200,
```

127.4898,0.2218,90.2200,145.1915,0.2558,94.2200,149.3255,0.3058,98.2200,157.5992,0.3018,102.2200,119.0889,
0.3250,106.2200,171.8328,0.2182,110.2200,176.4237,0.6412,undefined:1970-01-01T00:00Z,

C3

Name	Units	Description
Latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
Longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
Time	Milliseconds	The time at which this measurement was made
AccessionTime	Milliseconds	The time at which this measurement was added to the shores ide data repository.
Altitude		
C1		
C2		
C3		
Pressure		
Temperature	Degrees C	
undefined		
SensorIdentifier	Board_task	Identifies the particular sensor this data came from. Only useful on craft with multiple CTDs

Example:

{"kind":"C3","vid":"1167408762","latitude":"36.72759833333333","longitude":"-122.07875666666666",
"time":"1454803503000","accessionTime":"1458852403701","c1":"10.0600004196167","c2":"65.94999694824219",
"c3":"249.75999450683594","pressure":"0.0","temperature":"13.95","undefined":"1970-01-01T00:00Z",
"sensorIdentifier":"8215"}],

CTD - Conductivity/Temperature/Depth Sensor

Name	Units	Description
Latitude	Degrees North	http://en.wikipedia.org/wiki/Latitude
Longitude	Degrees East	http://en.wikipedia.org/wiki/Longitue
Time	Milliseconds	The time at which this measurement was made
AccessionTime	Milliseconds	The time at which this measurement was added to the shores ide data repository.
Altitude		
Conductivity	Siemens/Meter	
DissolvedOxygen	ml/l	Amount of oxygen dissolved in the water
OxygenHz	hertz	
OxygenSolubility	ml/l	Volume of oxygen gas at standard temperature and pressure conditions (STP) absorbed from humidity-saturated air at a total pressure of one atmosphere, per unit volume of the liquid at the temperature of measurement (ml/l)

Pressure	Decibars	
Salinity	psu	Practical Salinity (psu or ~ g/kg)
Temperature	Degrees C	
Undefined		
sensorIdentifier	Board_task	Identifies the particular sensor this data came from. Only useful on craft with multiple CTDs

Example:


```
kind:CTD,vid:1765014915,latitude:-25.472498333333334,longitude:-42.856028333333335,time:1446336116000,
accessionTime:1455747989298,conductivity:5.3323097,oxygenHz:3906.6,oxygenSolubility:4.8643723,
pressure:4.4399996,salinity:36.955437,temperature:22.8198,undefined:1970-01-01T00:00Z,sensorIdentifier:257,
```