LIQUID ROBOTICS
A Boeing Company

Wave Glider Management System Data Service User Guide

WDS Documentation

Revision 07
Item Number: 11675

Wave Glider Management System Data Service User Guide

Trademarks

Liquid Robotics®, Wave Glider®, Wave Glider Management System, WGMS, and SHARC® are worldwide trademarks of Liquid Robotics. Other products and company names mentioned herein may be the trademarks of their respective owners.

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.

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 Liquid Robotics Terms and Conditions Articles 11.1 Limited Hardware Warranties, 11.2. Limited Service Warranties, and 11.3 Limitations. Nothing herein should be construed as constituting any additional warranties or representations. Liquid Robotics, Inc. shall not be liable for technical or editorial errors or omissions contained herein.

Contact

LIQUID ROBOTICS
A Boeing Company

460 Herndon Parkway

Herndon, VA 20170

Phone: +1 408-636-4200

Customer Support: +1 888-574-4574

Fax: +1 408-747-1923

Revision History

Document Part Number	Revision	Publication Date	Changes
11675	7	August 2025	• Updated the Matlab example with latest java libraries, Access Token changes, and correct parameter order • Updated Data Service URLs • Minor formatting fixes
	6	August 2025	• Added section for Access Tokens • Updated examples to reflect token update
	5	Jun 2022	• Removed Logging and Data Metering • Updated Documented Workarounds
	4	Apr 2022	• Updated References • Removed Legacy Data Portal known issues
	3	Sep 2021	• Added Streaming Features & Commands, Implementation Examples • Modified Overview to include Streaming
	2	May 2021	• Include references to sensor guides for metadata information • Formatted subheadings • Update Open Oceans reference to script library and PN 11884 • Capture default report to all gliders & orgs. • Specify all requests must specify the “vid” as the unique vehicle ID. • Include pointer and definitions of formatted data types • Update to include figures and captions. • Include date range limit • Include note for JSON formatting in Excel • Including list of known bugs and workarounds
	1	March 2021	Initial Release

References

Document Part Number	Revision
05910	WGMS User Guide
11884	WGMS Data Service (WDS) Examples
05573	Wave Glider® Integrated Sensor User Guide Turner C3 Submersible Fluorometer Sensor Software
05574	Wave Glider® Integrated Sensor User Guide Sea-Bird Electronics Glider Payload CTD Sensor Software
06258	Wave Glider® Integrated Sensor User Guide GPSWaves Sensor Software
08556	Wave Glider® Integrated Sensor User Guide Vemco VR2C Sensor Software
08640	Wave Glider® Integrated Sensor User Guide Teledyne RD Instruments Workhorse Monitor ADCP Sensor Software
09262	Wave Glider® Integrated Sensor User Guide RUDICS Software

Table of Contents

Table of Figures

Table of Tables

1 About this Document

The purpose of this document is to provide user guidance for interfacing with the Wave Glider Management System (WGMS) Data Service (WDS).

This document includes instructions and sample code for extracting data from WDS using the REST API, Python, and Matlab scripts.

2 Overview

The WDS allows users to access and export WGMS data with successful authentication via stateless programmatic requests. The primary mechanism utilizes Simple Object Access Protocol (SOAP) in order to access available data in WGMS Data Storage, including sensor data, raw data, weather data, and the Adaptable Modular Power System (AMPS) data. The secondary mechanism utilizes Remote Procedure Call (RPC) to initiate near-real-time streaming of data from a Wave Glider. Users with WGMS or Enterprise WGMS obtain access through controlled tokens and can access their specific organizations and respective Wave Glider Vehicles. Several reports can be requested and exported in JavaScript Object Notation (JSON) format by vehicle including sensor or data type and time range.

Commented [BH1]: We removed the RPC section from this document because it was never fully implemented in WDS. We should probably remove this sentence as well.

3 Features & Commands

All available commands in the data service utilize stateless, web-based requests, accessible via https://wds.wgms.com/WGMSData.asmx. The service description is available in the Web Service Definition Language (WSDL) format via https://wds.wgms.com/WGMSData.asmx?wsdl. The below are examples illustrating placeholders within SOAP requests. The placeholders shown need to be replaced with actual values.

Using applications and tools such as curl commands or 3rd party extensions like SOAP WSDL analyzers or SOAP clients may help to construct requests and use the WDS programmatically.

The following depicts an example through the use of *SoapUI*, a 3rd party testing application. The SOAP and WSDL analyzers will list the available operations and assist in building requests through defined parameters, such as token and result format.

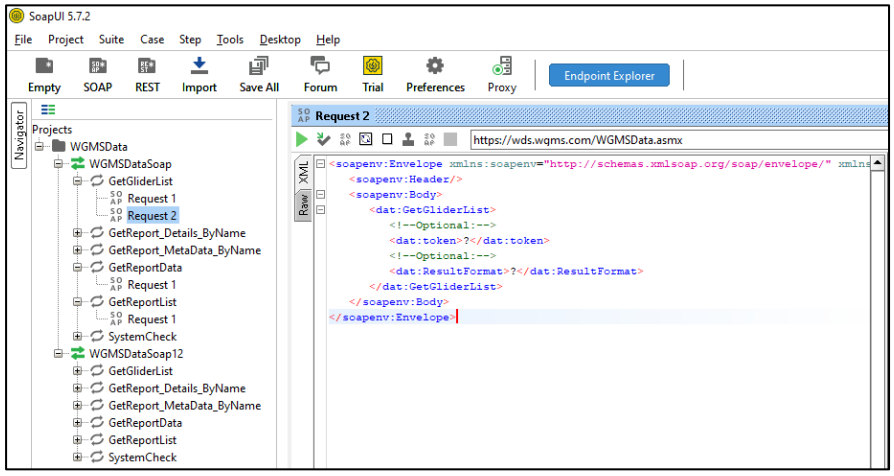

Figure 1 - Using SoapUI to access WDS

3.1 Authentication

Authentication through WDS is handled via WGMS Data Service Access Tokens. Each stateless request will consist of user's Access Token, Result Format, and any other potential string(s) as outlined further in this document. Each SOAP request must include user's Access Token to obtain access to available vehicle or sensor data. The Access Tokens are generated at the Org level, so users wanting to retrieve data from vehicles in multiple Orgs will need to use multiple Access Tokens.

To meet modern security standards, Access Tokens are set to expire after 90 days. Email reminders to generate a new token will be sent to the user's email address 2 weeks, 1 week, 2 days, and 1 day before, as well as on the day of the token expiration.

Commented [BH2]: Is this implemented, or to be implemented?

Generating WDS Access Tokens is done in the WGMS GUI.

1. Log in to WGMS and click on your account name.

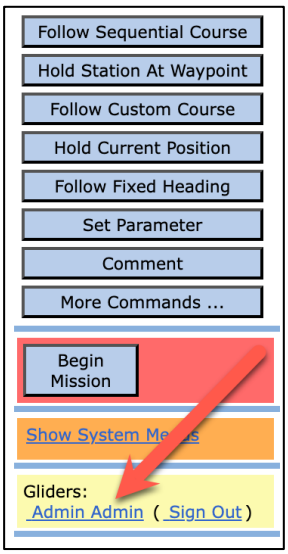

Figure 2: Admin Window Link

2. A new window will pop up – click on "Service Tokens" in the upper left.

Figure 3: Service Token Link

3. Click “Generate New Token” button.

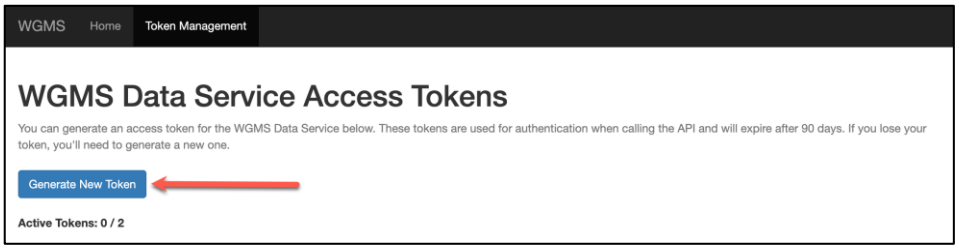

Figure 4: “Generate New Token” Button

4. If the user sees the message that “Session expired or not logged in. Please log in.” – they will need to refresh the browser tab, re-login to WGMS, and repeat steps 1-3 above.

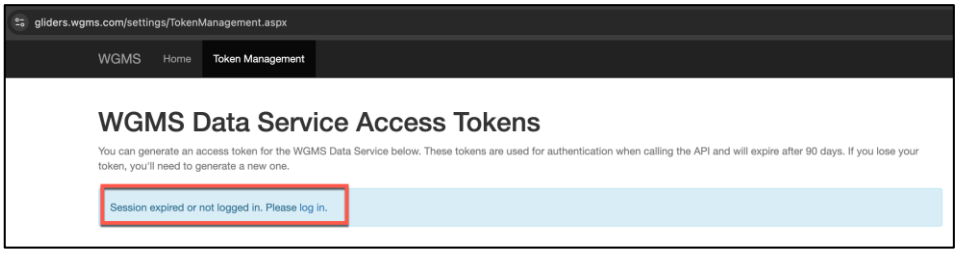

Figure 5: Session Expired Error

5. When the new token is successfully generated, that token is shown on the Active Tokens list. Use the Copy button to copy it to use for WDS API requests. Tokens are good for 90 days after creation. Users are allowed to have up to 2 active tokens per Org.

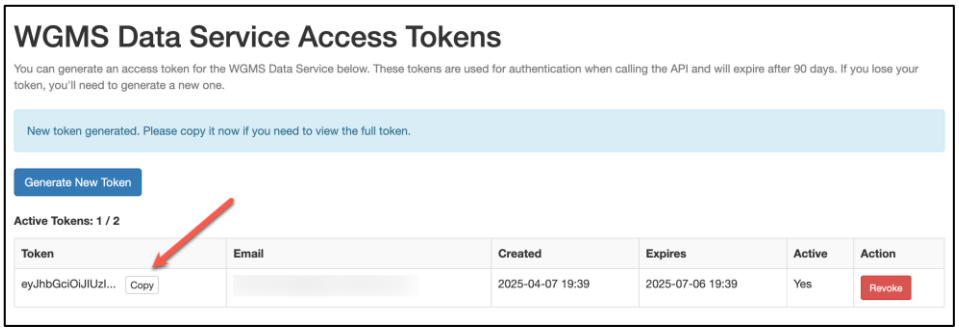

Figure 6:Token “Copy” Button

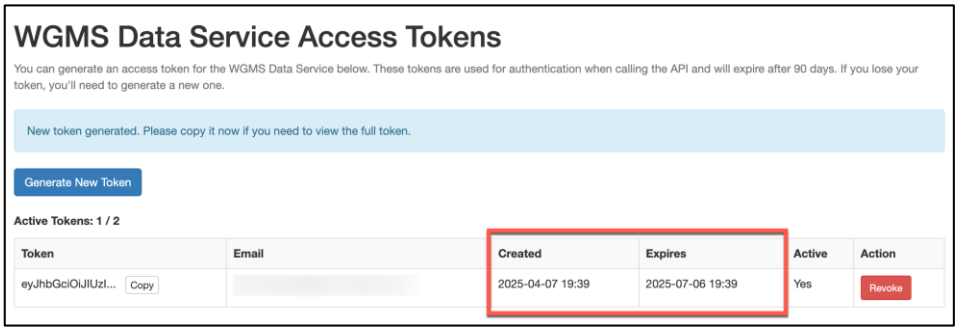

Figure 7: Token Expiration Date

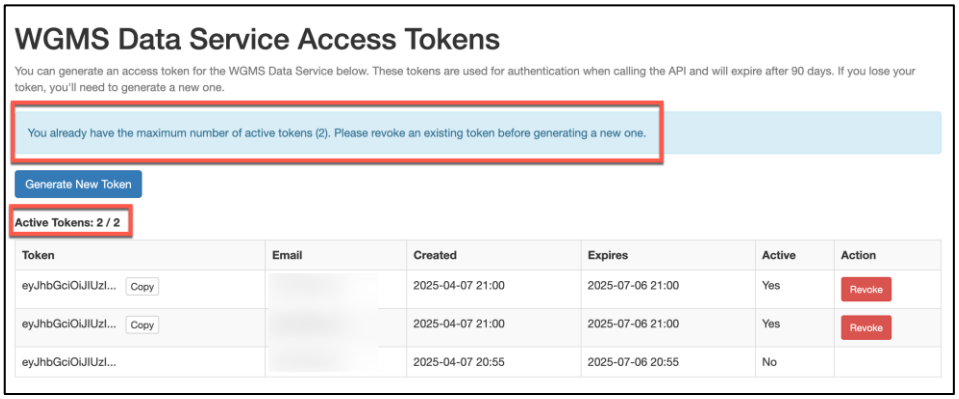

Figure 8: Max Tokens Error Message

6. Active tokens can be revoked before their 90 day expiration date. Revoked tokens cannot be reactivated; new token(s) must be generated to access the Data Service.

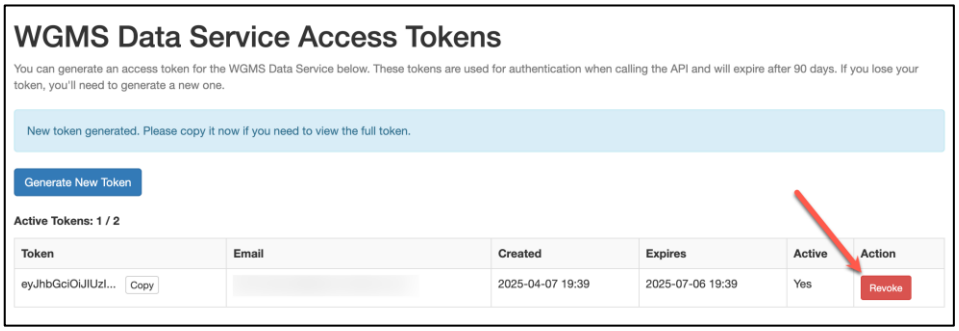

Figure 9: Token "Revoke" Button

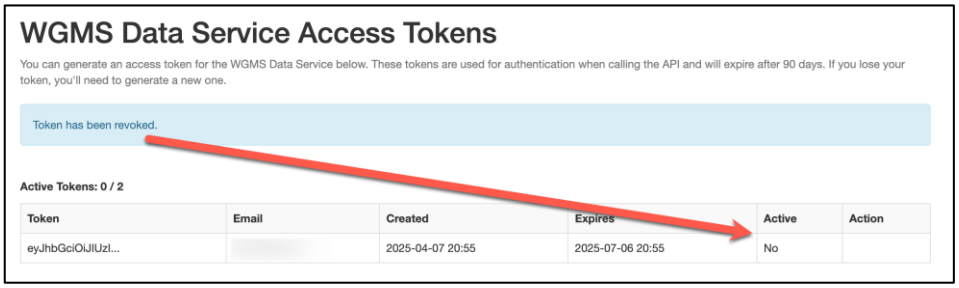

Figure 10: Token Inactive Indicator

3.2 Available Vehicles

WDS allows users to obtain the available Wave Glider Vehicles within their organization(s) in alignment with WGMS.

The following depicts the example request and expected response:

3.2.1 GetGliderList Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dat="https://dataservice.wgms.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <dat:GetGliderList>
         <!--Optional:-->
         <dat:token>string</dat:token>
         <!--Optional:-->
         <dat:ResultFormat>int</dat:ResultFormat>
      </dat:GetGliderList>
   </soapenv:Body>
</soapenv:Envelope>
```

3.2.2 GetGliderList Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <GetGliderListResponse xmlns="https://dataservice.wgms.com/">
         <GetGliderListResult>string</GetGliderListResult>
         <Error>false</Error>
      </GetGliderListResponse>
   </soap:Body>
</soap:Envelope>
```

The GetGliderList Response will follow the following metadata formatting:

```
        "vehicleName": "String",
        "vid": "Integer",
        "created": "String"
```

Note: The ResultFormat parameter should be entered in as a 1 by default.

3.3 Available Reports

WDS offers multiple reports to choose from. All vehicles in their respected orgs have a set of default reports, which include raw data, telemetry 6 reports, and AMPS data. All other data are specific to vehicle and organization, and may be found in reports customized for each vehicle and customer.

Note: Users should utilize the unique identifier "vid" for vehicle ID when requesting data. This ensures that consistent naming even when vehicles may have been moved or renamed.

Note: There is a known issue in Telemetry 6 Reports when utilizing the WDS. An analysis of Telemetry 6 when compared to WGMS Exports revealed the following:

- The WDS variables of ShoreSideCreated, LastLocationFix, TemperatureSub, LightState, OceanCurrent, and OceanCurrentDirection have no comparable fields in a WGMS Export.
- WDS `GliderDistance` has no equivalent in WGMS Export although *Distance Over Ground* is close.
- WDS `GliderSpeed` has no equivalent in WGMS Export although *Speed Over Ground* is close
- WDS `HeadingFloatDegrees` has no equivalent in WGMS Export although *Desired Heading* and *Sub Heading* are close.
- WDS `SurfaceTemperature` is called *Float Temp* in WGMS Export.
- WDS `GliderHeading` is called *Path Over Ground* in WGMS Export.
- WDS `DesiredBearingDegrees` is called *Desired Heading* in WGMS Export.

Data may be requested in date ranges within 90 days but we recommend limiting it to 30 days, as large data sets can affect performance, induce time outs, and inadvertently consume your allotted data usage (if your subscription is limited). To obtain data for longer periods, users must make multiple requests.

A listing of data reports available to each customer may be obtained through the following request and expected response:

3.3.1 GetReportList Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dat="https://dataservice.wgms.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <dat:GetReportList>
         <!--Optional:-->
         <dat:token>Your access token</dat:token>
         <!--Optional:-->
         <dat:ResultFormat>Format selector integer</dat:ResultFormat>
      </dat:GetReportList>
   </soapenv:Body>
</soapenv:Envelope>
```

3.3.2 GetReportList Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <GetReportListResponse xmlns="https://dataservice.wgms.com/">
         <GetReportListResult>string</GetReportListResult>
```

```
        <Error>false</Error>
      </GetReportListResponse>
   </soap:Body>
</soap:Envelope>
```

An example export from the training org returns the following report list:

```
"Raw Data Report","Telemetry 6 Report","Amps Power Summary Report","Amps
Output Power Status Report","Amps Solar Input Port Report","Amps Battery
Port Report","Amps Bridge Power Port Report","Test Amps Power Summary
Report","ADCP Sensor Bathymetry","ADCP Sensor C2 Samples","ADCP Sensor C2T
Samples","ADCP Sensor C4 Samples","ADCP Sensor C4T Samples","ADCP Sensor
Header Samples","Datawell MOSE Records","GPS Waves Sensor","Aquatracka
Samples","Fluorometer Samples (C3)","Seabird CTD Records","VEMCO Vr2C
Status","VEMCO Vr2c Tags"
```

3.4 JSON Exports

Obtaining report data is executed by the following request and expected response. The Data Service serves responses in JSON Format.

3.4.1 Exporting to Microsoft Excel

Depending on the tool sets used, users may export the data or copy into a text file for JSON and utilize Microsoft Excel for data parsing and analysis.

Using older versions of Microsoft Excel (2013), you can connect to a JSON file via selecting **Data**, navigating to **Power Query**, **From Other Sources** & Selecting **Blank Query**. Using the Advanced Editor, update the query string to point to your file & path:

```
let
    Source =
Json.Document(File.Contents("C:\Users\Name\Desktop\JSONTest.json")),
    #"Converted to Table" = Record.ToTable(Source)
in
    #"Converted to Table"
```

In newer versions of Excel, Select **Data**, **Get Data**, **From File**, **From JSON**, and choose **Import Data**. Navigate to the JSON file, and select **Open**.

3.4.2 Understanding Timestamps

When using the data service or WGMS timestamps, multiple times are reported. The a*scensionTime* is reported by Iridium and is mostly useful for diagnostics and troubleshooting. The *gliderTimestamp* maps to when the data is created onboard the vehicle. The *shoreSideCreated* time represents when the sensor measurements are instantiated into the shore side database. The *shoresideCreated* and *gliderTimeStamp* may be used in conjunction to determine end-to-end latency.

3.4.3 GetReportData Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dat="https://dataservice.wgms.com/">
   <soapenv:Header/>
   <soapenv:Body>
```

```
      <dat:GetReportData>
         <!--Optional:-->
         <dat:token>Your access token</dat:token>
         <!--Optional:-->
         <dat:ReportName>string</dat:ReportName>
         <!--Optional:-->
         <dat:SerialNo>Glider serial number</dat:SerialNo>
         <dat:StartDate>dateTime</dat:StartDate>
         <dat:EndDate>dateTime</dat:EndDate>
         <dat:ResultFormat>Report format selector integer</dat:ResultFormat>
      </dat:GetReportData>
   </soapenv:Body>
</soapenv:Envelope>
```

3.4.4 GetReportData Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <GetReportDataResponse xmlns="https://dataservice.wgms.com/">
      <GetReportDataResult>string</GetReportDataResult>
      <Error>boolean</Error>
    </GetReportDataResponse>
  </soap12:Body>
</soap12:Envelope>
```

An example response of the *Raw Data Report* returns the following metadata:

```
  "metaData": {
      "vehicleName": "String",
      "vid": "Integer",
      "payloadLength": "Integer",
      "payload": "Byte[]",
      "gliderTimeStamp": "DateTime",
      "ascensionTime (Unix)": "Integer",
      "ascensionTime": "DateTime",
      "shoresideCreated": "DateTime",
      "structureType": "Integer",
      "source": "String",
      "satelliteLatitude": "Decimal",
      "satelliteLongitude": "Decimal",
      "satelliteCEPRadius": "Decimal"
    }
```

3.4.5 Sample GetReportData Request

The following is an example XML request. It should be used with the GetReportData WebMethod for the WGMS Data Service.

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
```

```
  <soap12:Body>
    <GetReportData xmlns="https://dataservice.wgms.com/">
      <token>Your access token</token>
      <ReportName>[reportname: e.g "Raw Data Report"]</ReportName>
      <SerialNo>[vid]</SerialNo>
      <StartDate>2020-11-01T00:00:00Z</StartDate>
      <EndDate>2020-11-30T00:00:00Z</EndDate>
      <ResultFormat>1</ResultFormat>
    </GetReportData>
  </soap12:Body>
</soap12:Envelope>
```

Users may write out the XML file to disk manually or send it via a curl command using

```
curl -d @[Request file] -H "Content-type: application/soap+xml" [url] >
[outfile.xml]
```

3.4.6 Example Curl Command

```
curl-d @request.xml -H "Content-type: application/soap+xml"
http://localhost:59128/WGMSData.asmx?op=GetReportData> result-1mo.json
```

3.4.7 Data Formatting

Note: Refer to *PN 04438* for details on message format details.

Sensor Sample Record (Unparsed)

All sensors locate each sample in space and time using the following format. The software managing the sensor provides the timestamp and the lat/lon. The SMC (and Payload Interface Boards (PIBs) - these are SV2 Sensor Port Payloads adapted for SV3) provide callable routines that report these values in a form that can be directly inserted into sensor sample records. Message types 0x90 and 0x91 provide for repeat counts for multiple samples within a message.

Table 1: Payload Sensor Data Format

Field Name	Bytes Used	Byte #	Format
Latitude	4	0-3	Signed 32-bit integer
Longitude	4	4-7	Signed 32-bit integer
Unix Timestamp	4	8-11	Signed 32-bit integer
Sensor Data	[variable]	12-	

Telemetry Type 148 (Payload 0x94)

This message format is intended to report generic minimally-parsed sensor telemetry and make the maximum space available for the payload. The payload type ID uniquely identifies the type of sensor. The Board ID uniquely identifies the computer on the Wave Glider that is reporting the data. The Task ID identifies a sub-component on the computer that is reporting the data. The combination of Board ID and Task ID identify the reporting instance.

This message format is a truncated version of type 0x91 message format, and is used to pass data through the shoreside infrastructure to clients. Sensor data passed using type 0x94 is not parsed by shoreside systems.

Table 2: Payload 0x94 Message Format

Field Name	Bytes Used	Byte #	Format
PayloadType	4	0-3	See Telemetry type 0x80
Board Id	1	4	Unsigned 8-bit integer
Task Id	1	5	Unsigned 8-bit integer
Sensor Data payload	[variable]	6-	Unsigned 4-bit integer

Standard Telemetry Data 0x06 is formatted per the following:

Table 3: Telemetry 0x06 Standard Telemetry Data Format

Field Name	Byte Index	Size	Bit Index
Sub Leak Alarm	0	1	0
Float Leak Alarm	0	1	1
Sub to Float Comms Alarm	0	1	2
Payload Error Condition Alarm	0	1	3
Float Battery Low Alarm	0	1	4
Umbilical Fault Alarm	0	1	5
Sub Pressure Threshold Alarm	0	1	6
GPS Not Functioning	0	1	7
Internal Vehicle Comm Error	1	1	0
Sub Reboot Alarm	1	1	1
Float to Sub Comms Error	1	1	2
Over Current Error	1	1	3
Float Temperature Threshold Exceeded	1	1	4
Float Pressure Threshold Exceeded	1	1	5
Sub Temperature Threshold Exceeded	1	1	6
Float Reboot Alarm	1	1	7
Temp_Surface	2	2	
Latitude	4	4	
Longitude	8	4	
Last_Fix_Time_Hours	12	1	
Last_Fix_Time_Minutes	13	1	
Last_Fix_Time_Seconds	14	1	
Last_Fix_Date_Month	15	1	
Last_Fix_Date_Day	16	1	
Last_Fix_Date_Year	17	1	
File Read Transaction Timeout Error	18	1	0
File Write Fallback Image Activated	18	1	1
File Write Program Image Burned	18	1	2
File Write Fallback Image Bad or Not Found	18	1	3
File Write Selected Image Bad or Not Found	18	1	4
File Write Tentative Image Retry	18	1	5
Boot Config Corrupt	18	1	6
IPIB Not Responding	18	1	7

Field Name	Byte Index	Size	Bit Index
Temperature_Sub	19	2	
Distance_Over_Ground (m)	21	2	
Rudder Count	23	4	
Heading_Sub (degrees)	27	2	
Iridium_Signal_Strength	29	1	
Target_WayPoint	30	1	
Desired_Bearing (Degrees)	31	2	
Heading_Float (Degrees)	33	2	
Pressure_Sensor_Float	35	1	
Pressure_Sensor_Sub	36	1	
Navigation_Mode	37	1	See Nav Modes in *PN 04438*
Light_State	38	1	0
IR_State	38	1	1
Xbee_State	38	1	2
Payload 1 Power	38	1	3
Payload 2 Power	38	1	4
Sub Payload Power	38	1	5
Bubble Com Error	38	1	6
Battery Com Error	38	1	7
Reset_Flags	39	1	
Total_Power (10mWh)	40	2	
Iridium_Read_Errors	42	1	
Iridium_Write_Errors	43	1	
Iridium_Session_Errors	44	1	
Water Speed	45	2	
Water Direction	47	2	
Water Speed Std Dev	49	1	
Water Direction Std Dev	50	1	
Humidity	51	1	
Payload 3 Status	52	1	0
Payload 4 Status	52	1	1
PEP Status	52	1	2
IPIB Status	52	1	3
Modem Power	52	1	4
Increased Mailbox Checks	52	1	5
Reserved (SV3 – Thruster on/off)	52	1	6
Reserved (SV3 – Auto Evading on/off)	52	1	7

3.5 Metadata

When requesting report metadata, WDS will return a Metadata Report for the requested report. Available metadata includes but is not limited to tags, descriptions, vehicle identifiers or parameters, units, data field names, and data field descriptions.

Some metadata is specific to a sensor. Refer to the specific sensor user guides in the References

~~References~~ section for details on metadata.

3.5.1 GetReport_MetaData_ByName Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dat="https://dataservice.wgms.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <dat:GetReport_MetaData_ByName>
         <!--Optional:-->
         <dat:token>string</dat:token>
         <!--Optional:-->
         <dat:ReportName>string</dat:ReportName>
         <dat:ResultFormat>int</dat:ResultFormat>
      </dat:GetReport_MetaData_ByName>
   </soapenv:Body>
</soapenv:Envelope>
```

3.5.2 GetReport_MetaData_ByName Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <GetReport_MetaData_ByNameResponse xmlns="https://dataservice.wgms.com/">
    <GetReport_MetaData_ByNameResult>string</GetReport_MetaData_ByNameResult>
      <Error>boolean</Error>
    </GetReport_MetaData_ByNameResponse>
  </soap12:Body>
</soap12:Envelope>
```

3.6 System Check & Diagnostics

The WDS offers diagnostics to assist with data access and problem troubleshooting.

3.6.1 SystemCheck Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dat="https://dataservice.wgms.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <dat:SystemCheck>
         <!--Optional:-->
         <dat:token>string dat:token>
      </dat:SystemCheck>
   </soapenv:Body>
</soapenv:Envelope>
```

3.6.2 SystemCheck Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
```

```
Content-Length: length
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <SystemCheckResponse xmlns="https://dataservice.wgms.com/">
      <SystemCheckResult>string</SystemCheckResult>
    </SystemCheckResponse>
  </soap12:Body>
</soap12:Envelope>
```

3.7 Known Issues

Liquid Robotics adds issues and bugs to the product backlog, and provides fixes based on priority and available workaround. The following lists the known issues that are deferred until future release.

1. ADCP C2T Data Bin Count = 30 but only 25 bins are served. Please contact Liquid Robotics support if this is an issue for you.
2. Sensor Report Types NOT included in field definitions for AMPS Power Summary Report
3. Large data pulls from queries may result in timeouts on your application

4 Example Use Cases

4.1 Connecting MATLAB to the WGMS Data Service

The purpose of this section is to provide detailed instructions on how to interface with the WGMS Data Service using Matlab. The following information was adapted from MathWorks Help Center, in particular their description of the matlab.wsdl.createWSDLClient function. This function creates an interface to SOAP-based web services. The webpage can be accessed with this link.

4.1.1 Install required libraries

1. Download and install OpenJDK:
    a. From your internet Browser go to: https://adoptium.net/.
    b. Select latest version and download the file. (Note: At the time this document was written, the version used was Temurin 21 LTS).
    c. Install the downloaded file (its filename will start with OpenJDK…), and select default parameters during the install process (unless you have specific JDK needs)
    d. Record the path of the installation, it will be needed later in the process.
2. Download ApacheCXF:
    a. Download ApacheCXF using this link. (Note: There are newer versions of ApacheCXF, but Matlab WSDL requires version 3.4.X, and we've tested through 3.4.8 only).
    b. Unzip the compressed folder
    c. Record the path, it will be needed later in the process.

4.1.2 Set up Matlab environment

Instructions in this section are executed from within the Matlab command window.

1. Assign a path to the OpenJDK software:

    Ex. If the path to the JDK software is C:\Program Files\Eclipse Adoptium\jdk-21.0.8.9-hotspot\, assign a variable, jdk, in Matlab to that path.

    ```
    >>jdk = 'C:\Program Files\Eclipse Adoptium\jdk-21.0.8.9-hotspot'
    ```

2. Assign a path to the Apache CXF Software:

    Ex. If the path to Apache CXF software is C:\Program Files\apache-cxf-3.2.14 assign a variable, cxf, in Matlab to that path.

    ```
    >>cxf = 'C:\Program Files\apache-cxf-3.4.8'
    ```

3. Execute the following command:

    ```
    >>matlab.wsdl.setWSDLToolPath('JDK',jdk,'CXF',cxf)
    ```

4. Execute the following command:

    ```
    >>matlab.wsdl.createWSDLClient('https://wds.wgms.com/wgmsdata.asmx?wsdl
    ')
    ```

5. Verify WGMSData.m and WGMSData1.m were created in the working directory.

    (Note: WGMSData.m is for use with SOAP 1.2 and WGMSData1.m is for use with SOAP 1.1.)

6. Add the directory where the classes were created to the Java path by executing the following command:

```
>>javaaddpath('.\+wsdl\wds.jar')
>>javaaddpath('.\+wsdl\wgmsdata.jar')
```

The methods within WGMSData.m and WGMSData1.m can now be used to extract data from the WGMS Data Service. The methods including the inputs and outputs are described in the *.m files and are shown below. The output of GetReportList, GetGliderList, and GetReportData are JSON formatted. To convert that output to a Matlab structure, use the jsondecode.m function.

***Note**: DO NOT edit the WGMSData.m and WGMSData1.m files*

4.1.3 Retrieve data from WDS

1. All functions require the WGMSData Object as an input variable. To create the WGMSData Object:
   a. Execute the following command:

```
>> obj = WGMSData
```

   b. Verify the Endpoint and WSDLFile:

```
Endpoint: `{https://wds.wgms.com/}WGMSData'
WSDLFile: 'https://wds.wgms.com/wgmsdata.asmx?wsdl'
```

2. Set up Org and Token variables
   a. Execute the following commands:

```
>> Org = "Your Org Name"
>> Token = "Your Token String"
```

3. **SystemCheck**: This function is not necessary for retrieving data but can be useful for troubleshooting. To run the system check:
   a. Execute the following command:

```
>> SystemCheckResult = SystemCheck(obj, Token)
```

   b. Verify the SystemCheckResult is Nominal:

```
SystemCheckResult =

    'WGMS Data Service System is Nominal
```

4. **GetReportList**: This function provides the report names for all the data available in the WGMS Org. *(Note: The report names shown in GetReportListResult are not necessarily available for <u>all</u> vehicles)*
   a. Execute the following command:

```
>>[GetReportListResult,Error] =
GetReportList(obj,Token,Org,ResultFormat)
```

      i. Where obj = The WGMSData object (step 9)
      ii. Token = String representing your JWT security token
      iii. Org = String representing the WGMS Org you are requesting data from

iv. ResultFormat = 1

b. Verify the output in GetReportListResult and Error = 0. The output in GetReportListResult is used as the input variable “ReportName” in calls to the GetReportData function (step 14).

5. **GetGliderList**: This function provides a list of vehicle names and unique IDs from a WGMS Org.
   a. Execute the following command in the Matlab Command Window:

```
>> [GetGliderListResult,Error] =
GetGliderList(obj,Token,Org,ResultFormat)
```

      i. Where obj = The WGMSData object (step 9)
      ii. Token = String representing your JWT security token
      iii. Org = String representing the WGMS Org you are requesting data from
      iv. ResultFormat = 1

   b. Verify the Outputs, GetGliderList Result and Error = 0 in the Matlab Command Window. The ID output is used as the input variable “SerialNo” in calls to the GetReportData function (step 14).

6. **GetReportData**: This function retrieves data from the WGMS Data Service for the selected vehicle and the selected report name.
   a. Execute the following command in the Matlab Command Window

```
>> [GetReportDataResult,Error] =
GetReportData(obj,Token,Org,ReportName,SerialNo,StartDate,EndDate
,ResultFormat)
```

      i. Where obj = The WGMSData object (step 9)
      ii. Token = String representing your JWT security token
      iii. Org = String representing the WGMS Org you are requesting data from
      iv. ReportName = String representing the ReportName (Step 12b)
      v. SerialNo = String representing the vehicle unique ID (Step 13b)
      vi. Start Date = The beginning time bound in the format dd-mmm-yyyy HH:MM:SS
      vii. End Date = The end time bound in the format dd-mmm-yyyy HH:MM:SS
      viii. ResultFormat = 1

      NOTE: The format of StartDate and EndDate is explained in Table 1 of the help page for the datestr.m function.

   b. Verify the outputs in GetReportDataResult and Error = 0.

7. To convert the data from JSON to a Matlab structure, run the following command:

```
>>Data = jsondecode(GetReportDataResult)
```

4.2 Python Script to Interface with the WGMS Data Service

4.2.1 Setup

If a customer has python and pip installed on their system this Python script is meant to be an easy to use interface for customers to access Data Service.

After downloading the script the user will need to install the 'zeep' module, which provides the SOAP protocol library:

```
pip3 install zeep
```

The user needs to modify certain constants such as org, user, and token in the source code.

```
# Constants
wsdl = "https://wds.wgms.com/WGMSData.asmx?wsdl"  # New Data Service WSDL
token = "your_access_token" #WDS Access Token from WGMS
reportName = "Telemetry ó Report"  # Data to be sent back
resultFormat = 1  # Default Option
```

Figure 11 - Python scripting example, modified constraints

The script requires the user to add inputs to determine the commands that are to be sent to Data Service.

```
python3 DataService.py --help
```

```
theja@theja-VirtualBox:~/dev/DataService$ python3 DataService.py --help
usage: DataService.py [-h] [--getGliderList] [--getReportList] [--getReportData] [--getPeriodicData] [--vehicles [VEHICLES [VEHICLES ...]]] [--startDate STARTDATE] [--endDate ENDDATE]
                      [--interval INTERVAL] [--time TIME]

Python script to utilize the WGMS Data Service.

optional arguments:
  -h, --help            show this help message and exit
  --getGliderList       Command to get list of gliders in org.
  --getReportList       Command to get list of reports.
  --getReportData       Command to get raw data from a specified vehicle in a date range. Requires user to specify startDate and endDate.
  --getPeriodicData     Command to get data over a specified period of time at specified intervals. Requires user to specify interval and time
  --vehicles [VEHICLES [VEHICLES ...]]
                        The vehicle serial number. For multiple vehicle spaces needed between numbers.Example --vehicles 123 456 789
  --startDate STARTDATE
                        Start date and time of the requested data. Format: YYYY-mm-DDTHH:MM:SSZ
  --endDate ENDDATE     End date and time of the requested data. Format: YYYY-mm-DDTHH:MM:SSZ
  --interval INTERVAL   Length of time interval in minutes.
  --time TIME           Length of time you want the python script to run in minutes.
```

Figure 12 - Python script example, Help

4.2.2 GetGliderList

To get a list of all the gliders in an Org the user would send the following command:

```
python3 DataService.py --getGliderList
```

```
theja@theja-VirtualBox:~/dev/DataService$ python3 DataService.py --getGliderList
{
  "metaData": {
    "vehicleName": "String",
    "unique ID": "Integer",
    "created": "String"
  },
  "recordData": [
    {
      "vehicleName": "A'a",
      "unique ID": 860481287,
      "created": "04/26/2012"
    },
    {
      "vehicleName": "Alex",
      "unique ID": 787336913,
      "created": "10/07/2011"
    },
    {
      "vehicleName": "Aquaman {dry docked}",
      "unique ID": 123456789,
      "created": "06/28/2013"
    },
    {
      "vehicleName": "Aukai (Moved to PCA Org 180328)",
      "unique ID": 282,
      "created": "08/26/2009"
    },
    {
      "vehicleName": "Benjamin (PCX_2)",
```

Figure 13 - Python example, getGliderList

4.2.3 GetReportList

To get a list of all the reports that can be requested from Data Service the user would send the following command:

```
python3 DataService.py --getReportList
```

Example:

Figure 14 - Python example, getReportList

4.2.4 GetReportData

To get data from a specified list of vehicles over a time period the user would send the following command. Note that requesting data from multiple specified vehicles is done in script, making separate GetReportData calls for each vehicle specified.

```
python3 DataService.py --getReportData --vehicle [list of vehicle serial
numbers] --startDate [start date and time in YYYY-mm-DDTHH:MM:SSZ] --endDate
[end date and time in YYYY-mm-DDTHH:MM:SSZ]
```

Get data from 1 vehicle:

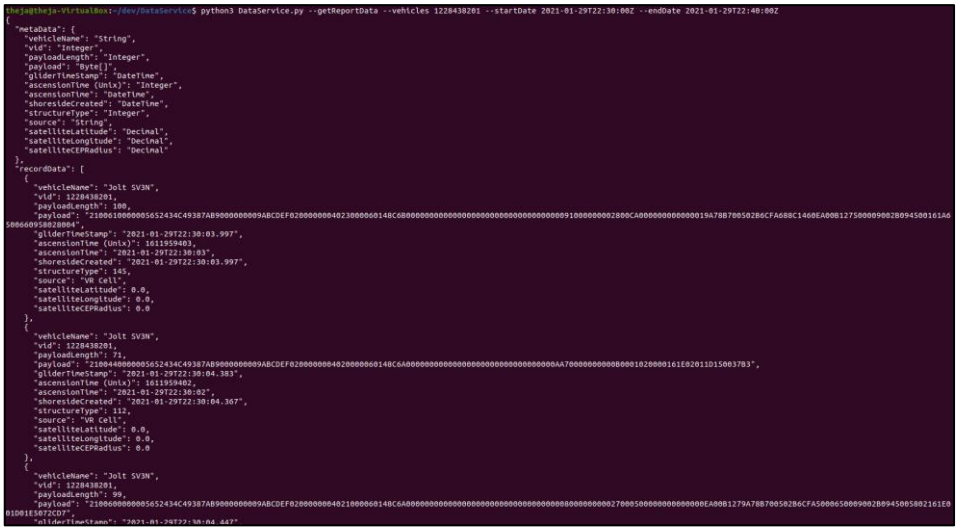

Figure 15 - Python example, getReportData from One Vehicle

Get data from multiple vehicles:

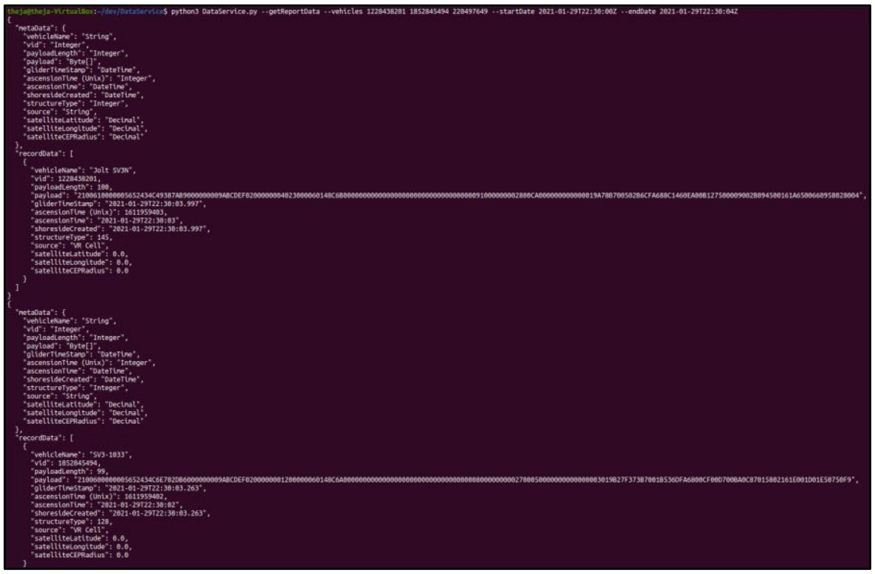

Figure 16 - Python example, getReportData from Multiple Vehicles

To get data from all the vehicles in the org simply omit the vehicles option.

```
python3 DataService.py --getReportData --startDate [start date and time in
YYYY-mm-DDTHH:MM:SSZ] --endDate [end date and time in YYYY-mm-DDTHH:MM:SSZ]
```

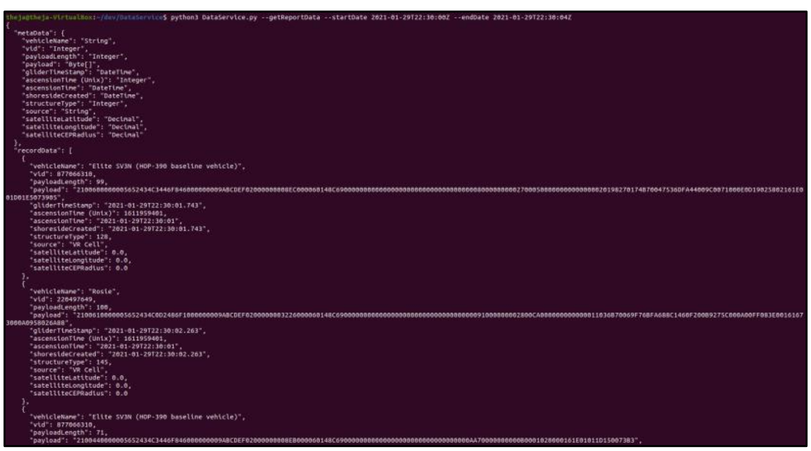

Figure 17 - Python example, getReportData for All Vehicles in Org

4.2.5 GetPeriodicData

This script sets up a polling interval to retrieve data from specified vehicles at a set interval over a period of time.

```
python3 DataService.py --getPeriodicData --vehicle [list of vehicle serial numbers] --interval [time in minutes to get data] --time [time in minutes the script will run]
```

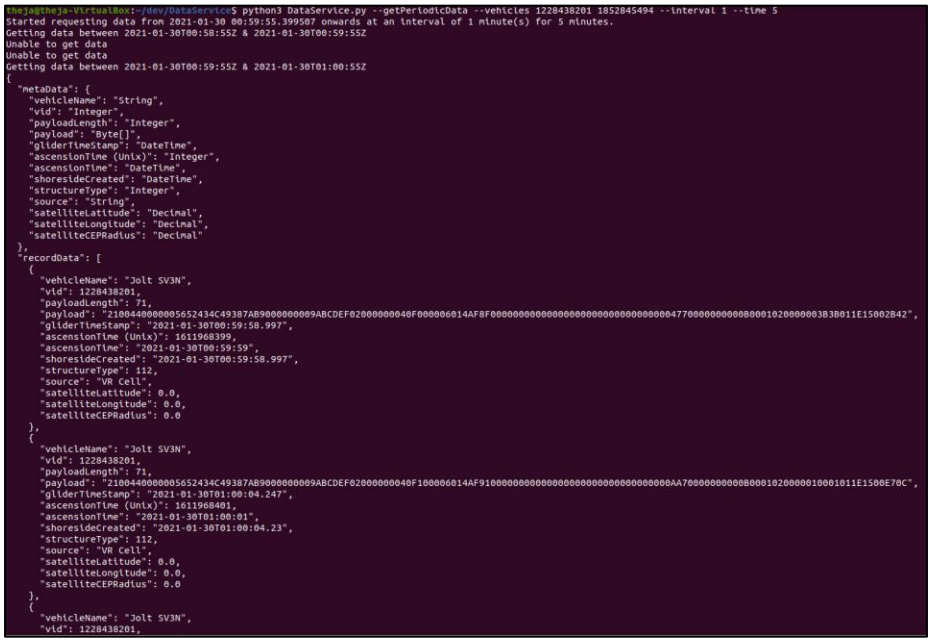

Figure 18 - Python example, getReportData using periodic polling for multi-vehicle

To get data from all the vehicles in the org simply omit the vehicles option:

```
python3 DataService.py --getPeriodicData --interval [time in minutes to get data] --time [time in minutes the script will run]
```

4.3 Script Reference Library

Examples will become available via application notes or training modules within the Open Oceans Portal.

Python & Java script tools are available under PN 11884.