

HOBOLink[®] Web Services V3 Developer's Guide

- Introduction 2**
- Required Technical Capabilities 2**
- OAuth Token 2**
 - Success 2**
 - Failure..... 3**
 - Command Line Requests 3**
- Get Data Endpoint 4**
 - Responses..... 4**
 - Success..... 4**
 - Failures 5**
 - Time Frame Querying..... 6**
 - Managed Data Tracking Querying 6**
 - Data Replay 7**

Introduction

This guide describes how external clients can access different REST endpoints. Onset exposes several web services that provide users with the ability to view loggers and retrieve data from loggers. The current version of the REST API expands upon previous versions (2.0 and 2.5) in that it uses OAuth to grant access. Below are details on how to generate tokens for the web service as well as how to access different web services and what they provide.

Required Technical Capabilities

Using Onset's web services require the following technical capabilities:

- Web development experience
- REST web services client development
- OAuth familiarity, particularly client credentials grants
- Access to a HOBOLink account and a Remote Monitoring Station and/or an MX series logger

OAuth Token

Onset supports the `client_credentials` grant types in the public API at this point:

- Grant: Client Credentials - defined in [rfc6749 section 4.4](#)
 - Required form fields: `grant_type=client_credentials`, `client_id`, `client_secret`

The endpoint to generate the tokens is POST `/auth/token`. Below is an example request, with the details passed in the body of the POST request:

Example Token Info Request

```
POST /ws/auth/token HTTP/1.1
Host: webservice.hobolink.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_id=Client+Name&client_secret=xxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxx
```

The `client_id` and `client_secret` should be replaced by the client-specific values provided by Onset.

The production API sandbox can be found at <https://webservice.hobolink.com/ws/auth/info/index.html#> and then navigating to Get Documentation → API → Token → `/token`.

Success

The response from this endpoint is defined based on [rfc6749 section 5.1](#) and is as follows:

Status Code: 200 OK

Content-Type: application/json

Additional Required Headers: Cache-Control: no-store, Pragma: no-cache

Body: JSON

- `access_token` - issued access token from OAuth server
- `token_type` - "bearer"
- `expires_in` - lifetime of token in seconds

Example Token Success

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store, no-transform
Pragma: no-cache
```

```
{
  "access_token": "44a11681c637fdb512764bb6b0965e44",
  "token_type": "bearer",
  "expires_in": 600,
}
```

Failure

Error response is defined based on [rfc6749 section 5.2](#) and is as follows:

Status Code: 400 Bad Request, 401 Unauthorized

Content-Type: application/json

Additional Required Headers: Cache-Control: no-store, Pragma: no-cache

Body: JSON

- error - ASCII error code limited to this list
 - invalid_request - missing/unsupported parameters (except grant type) or otherwise malformed (Status 400)
 - invalid_client - client authentication not included or authentication failed (Status 401)
 - invalid_grant - the grant parameters (resource owner credentials) are not valid or refresh token is not valid/expired/revoked (Status 400)
 - unsupported_grant_type - the grant type is not supported by the server (Status 400)
- error_description - Human-readable ASCII text used to provide additional understanding of error that occurred

Example Token Failure

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Cache-Control: no-store, no-transform
Pragma: no-cache
```

```
{
  "error_description": "The request is malformed, a required parameter is missing or a parameter
  has an invalid value.",
  "error": "invalid_request"
}
```

Command Line Requests

If desired, it is possible to run the same request via a CLI using commands like curl. The format of the request and response will be the same as above. An example of a curl request for a client_credentials token would look something like this:

And the response will look something like this:

```
curl -i -H "Content-type: application/x-www-form-urlencoded" -H "Accept: application/json" -d
"grant_type=client_credentials" -d "client_id=Client+Name" -d
"client_secret=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx" --data-urlencode -X POST
https://webservice.hobolink.com/ws/auth/token
```

In to order to control access to different endpoints and data, Onset uses an OAuth token privilege system. When a token is used to access an endpoint, the associated privileges are retrieved, and the request is passed to the endpoint only if the retrieved privileges contain the one required by the endpoint. If the privilege is not present, the user will receive a Forbidden status code. Privileges are assigned and managed at a user level. When a token is generated with client_credentials grant, then the default client user will be used to retrieve privileges. If access to a specific endpoint is required, please reach out to Onset.

Get Data Endpoint

To request data for a particular logger, use the `/ws/data/file/{format}/user/{userId}` endpoint. This endpoint takes path and query parameters to indicate the user, logger, and range of data that is expected. When querying for data from this endpoint, data can be either returned for a specific time frame, or Onset can keep track of data previously returned and only new data will be returned on subsequent calls. Details on the different types of querying and what each parameter means can be found below:

```
GET /ws/data/file/{format}/user/{userId}?loggers={loggerList}&start_date_time={startTime}&end_date_time={endTime}&only_new_data={onlyNewData}&last_successful_query_time={lastSuccessfulQueryTime}
```

Parameter	Meaning
format	The format data should be returned in. Currently only JSON is supported.
userId	The numerical id of the user data should be grabbed for. This can be pulled from the HOBOLink URL: <code>www.hobolink.com/users/<userId></code>
loggerList	A comma-delimited list of logger serial numbers that data should be grabbed for. The list is limited to 10 different loggers at a time.
startTime	The date furthest in the past data should be grabbed for. Should be in the format <code>yyyy-MM-dd HH:mm:ss</code> . Example, if the date <code>2020-01-01 00:00:00</code> is used, only data from New Year's day 2020 and beyond will be returned.
endTime	This is only needed if time frame querying is desired, see below.
onlyNewData	This is only required if managed data tracking querying is desired, see below.
lastSuccessfulQueryTime	This is only required if using managed data tracking and it is desired to reset tracking to a previous call, see below.

The production API sandbox can be found at <https://webservice.hobolink.com/ws/data/info/index.html#> and then navigating to Get Documentation → API → File → `/file/{format}/user/{userId}`. The above parameters will be detailed there.

Responses

Success

On a successful query, the data will be returned as an `observation_list` in the specified format (only JSON currently supported). Each observation will indicate the logger, sensor, measurement type, data type, timestamp, value, and unit. Sample output can be seen below:

```
{
  "observation_list": [
    {
      "logger_sn": "99603325",
      "sensor_sn": "99508399-4",
      "timestamp": "2019-11-20 00:00:00Z",
      "data_type_id": "1",
      "si_value": 0.9995219339475939,
      "si_unit": "meters",
      "us_value": 3.2792714368359346,
      "us_unit": "feet",
      "scaled_value": 0,
      "scaled_unit": null,
      "sensor_key": 9658456,
      "sensor_measurement_type": "Water Level"
    },
    ...
    {
      "logger_sn": "99603325",
      "sensor_sn": "99508399-3",
      "timestamp": "2019-11-20 01:00:00Z",
      "data_type_id": "1",
      "si_value": 21.784423828125,

```

```

    "si_unit": "°C",
    "us_value": 71.211962890625,
    "us_unit": "°F",
    "scaled_value": 0,
    "scaled_unit": null,
    "sensor_key": 9658455,
    "sensor_measurement_type": "Water Temperature"
  }
],
"message": "OK: Found: 169 results.",
"max_results": false
}

```

In addition to the `observation_list` containing the list of data points, there is a `message` and `max_results` field. The `message` field will include a simple message indicating how many results were returned. The `max_results` field will be set to true if the maximum number of results was found, and false otherwise. Each query is capped at 100,000 data points, so even if more exist in a specified time range, only the first 100,000 will be returned. The `max_results` field provides an easy way to identify if there are more batches to query. When using the time frame querying (specified below), the time frame for the next batch will have to be specified manually. When using managed data tracking query (specified below), running the query again will retrieve the next batch.

Failures

In the case that data can not be returned from the query, an error will be returned in JSON format. Below is an example error:

```

{
  "error": "VAL-034",
  "message": "No user ID provided.",
  "error_description": "Invalid request."
}

```

In the above example, the fields correspond to the following:

error: A simple error code identifying which type of error was seen.

message: A message providing some context around the error.

error_description: A basic description of what type of error was seen.

The following is a list of all the possible error codes that can be returned:

Error	Description	Status Code
CFG-003	Error retrieving property.	400
DBM-004	Database error retrieving the user.	500
DBM-026	Database error retrieving data.	500
DBM-027	Database error saving data.	500
SYS-001	System is busy.	509
SYS-002	Too many requests.	509
VAL-001	Time period start time is null.	400
VAL-002	Time period start time is in the future.	400
VAL-004	Device serial number or user name is required.	400
VAL-006	Bad query date format.	400
VAL-033	Invalid format.	400
VAL-034	Invalid request.	400
WCE-001	WebCache connection not available.	500

Time Frame Querying

Time frame querying allows the client to query for data within a specified time frame. This requires the use of the `endTime` parameter. The `endTime` parameter takes on the same format of the `startTime` parameter (`yyyy-MM-dd HH:mm:ss`) and indicates the end of the range for which data should be retrieved. When the `endTime` parameter is used, the `lastSuccessfulQueryTime` must be omitted, and `onlyNewData` should be omitted or set to `false`. Data that has occurred between the start and end times, inclusive, will be returned. Note that all dates are in UTC.

Example Query Format: `http://webservice.hobolink.com/ws/data/file/JSON/user/99999?loggers=99999999&start_date_time=2019-11-20+00%3A00%3A00&end_date_time=2019-11-20+01%3A00%3A00`

Expected Return: Data points from logger 99999999 configured by user 99999 at or after 2019-11-20 00:00:00, and before or at 2019-11-20 01:00:00.

Data returned may contain gaps depending upon system processing time or motes in recovery mode.

Managed Data Tracking Querying

In cases where consistent querying is performed, clients may not want to keep track of what data they have received, or want to add logic to update the URL. To help alleviate this, Onset provides an option to use a managed data tracking query format. In order to use this type of querying, the `only_new_data` parameter must be present, and must be set to `'true'`. The `startTime` has a similar function in this type of query, where no data prior to the `startTime` will be returned. However, unlike the Time Frame Querying, the data being returned doesn't start at the `startTime`, but rather at the last timestamp returned in a managed query. For example, if we use the following query:

`http://webservice.hobolink.com/ws/data/file/JSON/user/99999?loggers=99999999&start_date_time=2019-11-20+00%3A00%3A00&only_new_data=true`

Then the first time the query is executed, then data starting at 2019-11-20 00:00:00 will be looked up. The number of results returned is limited to 100,000 per query, so the first 100,000 results at or after 2019-11-20 00:00:00 will be returned. The second time the *same* query is executed, the next batch of 100,000 data points will be returned, and so on. Assuming that there is only one sensor with data to be returned, and the sensor was logging at a 30 sec interval, the below table shows the time ranges that would be returned for each query. Assumption is current time is February 1st, 2020 at midnight. Note that all dates are in UTC.

Query Number	Time Range Returned	Notes
1	2019-11-20 00:00:00 - 2019-12-24 17:19:30	First query, so begins at <code>startTime</code> and continues for 100,000 points
2	2019-12-24 17:20:00 - 2020-01-28 10:40:00	Next batch of 100,000 points
3	2020-01-28 10:40:00 - 2020-01-31 23:59:30	Last batch catches up to current time, returning the remaining 10,240 data points

The same query can continue to be used as long as needed, grabbing only the data that has been generated since the last query. Each time the query returns a successful result code, the time of the latest data point is saved, keeping the time range moving forward. The downside to using this query format is that a query can't be retried if some data point are lost or corrupted. Time frame querying format can always be used to grab a specific selection of data, but is only really useful if the time frame can be specified. If data needs to be re-queried in a more programmatic way, the managed data query format can take an additional parameter to back track the data being returned; the `last_successful_query_time` parameter.

Data returned will not contain gaps and will only be delivered as full, contiguous data sets are available. This means that data processing delays or data recovery mode as exists on HOBONet wireless sensors may delay data availability until all data has been obtained.

Data Replay

The `last_successful_query_time` parameter can only be used for managed data tracking queries, so `only_new_data` must be set to true. The parameter takes a date in the same format as `start_date_time` (yyyy-MM-dd HH:mm:ss), again in UTC, and is used to back track the current state of the managed data queries. Each time a managed data query is completed successfully, the time of the query and the current latest data point are saved. When used, the most recent successful query that occurred before or at the specified `last_successful_query_time` will be identified, and the latest data point returned will be used to determine the next batch. Using the example above, let's say that the 3 queries were completed at the following times:

Query Number	Time of Query
1	2019-12-30 00:00:00
2	2020-01-29 00:00:00
3	2020-02-01 00:00:00

We then go back and realize the data from queries 2 and 3 got corrupted in transit back to the caller. In order to avoid replaying all the data or figuring out exactly which `start_date_time` we need to specify, we just use the `last_successful_query_time`. Our "replay" query would then look like this:

```
http://webservice.hobolink.com/ws/data/file/JSON/user/99999?loggers=99999999&start_date_time=2019-11-20+00%3A00%3A00&only_new_data=true&last_successful_query_time=2019-12-30+00%3A00%3A00
```

This will re-do query number 2. If that is successful and the data is correctly retrieved, then we can execute the original query (without the `last_successful_query_time`) again, and it will pick up with query number 3 and so forth. If we want to replay all queries, we can simply set the `last_successful_query_time` to date far in the past, and it will act as if none of the queries have been run before.