OPENDCS

HydroJSON Implementation

Document Revision 1

August, 2020

Table of Contents

1 Introduction

HydroJSON is a JSON (Java Script Object Notation) standard for interchanging hydro, meteorological, and other environmental data. It was developed by the U. S. Army Corps of Engineers. The original defining page can be found here:

```
https://github.com/gunnarleffler/hydroJSON
```

OpenDCS implements HydroJSON as a Web Application running under Apache Tomcat. This document will define the calls that are implemented by OpenDCS and the returned data structures in detail.

The backend database for data delivered can be either CWMS (Corps Water Management System) or OpenTSDB (Open Time Series Database – a module within OpenDCS).

1.1 Document History

Revision 1, August 2020 – Original document release.

1.2 Glossary

JSON	Java Script Object Notation
WAR	Web Archive – deployable web applications are typically distributed as WAR files.

2 Installation and Configuration

Configure for database
Build the WAR
Deploy to TOMCAT
Troubleshooting

3 Cove Software Demo Installation

4 REST Method Reference

The following methods are implemented for OpenTSDB HydroJSON. Following the table, subsections will provide details on each.

Keyword	Arguments	Section	Description
credentials	none	4.1	POST method to login to system and retrieve a new token.
check	token	4.2	GET method, passed a token value, returns complete token if valid.
reflists	name, token	4.3	GET method. Optional arguments are name of reference list desired, and a token. Returns reference list(s).
netlists	name, token	4.4	GET method. Optional arguments are name of network list desired, and a token. Returns network list(s).
netlist	token	4.5	POST method. Valid token required. POST data is a JSON Network List.
tscatalog	site, param, interval, active, token	4.6	Get method to return a catalog of Time Series Identifiers. All arguments are optional and can be used for filtering.
tsdata	site, param, interval, active, since, until, token	4.7	Get method to return Time Series Identifiers containing Time Series Data. All arguments optional and can be used for filtering.
platforms	tmtype, token	4.8	List platforms of a given transport medium type.
cnvtnl	token	4.9	Convert LRGS ".nl" file to JSON NetList

The Root URL for the service, unless changed during deployment, is OHydroJson. Thus to access the "platforms" method on a tomcat server running on localhost on port 8080, the complete URL would be:

```
http://localhost:8080/OHydroJson/platforms
```

4.1 Authentication and Authorization

Authentication and authorization is required for any method that modifies the database.

The 'credentials' POST method is used to obtain a new token. The POST data must be of the form:

```
{ "username" : "UserNameForLogin", "password" : "PasswordForLogin" }
```

The user name and password provided must be a valid login for the underlying database. Also, that user must be assigned either of the roles OTSDB_ADMIN or OTSDB_MGR. If successful, a token is returned with the following JSON structure:

```
{"lastUsed":1596807452610,"token":"9fe6390676c7dca9","username":"xxx"}
```

- The "username" argument will be the same username passed to the POST method.
- The "token" argument is a randomized hexadecimal string that must be passed to subsequent methods that require authentication.
- The "lastUsed" argument is a long integer number of milliseconds since the epoch (Jan 1 1970 at midnight UTC).

Tokens are valid for a finite period of time. Every time a token is used (i.e. passed to a method), it's lastUsed field is updated. If more than 3 hours goes by without a token being used, it is removed from the server and is no longer valid.

If the passed username or password is not valid, or if the user is not assigned the required role, an HTTP 401 error is returned with an explanatory message. It will typically be one of the following:

```
FATAL: password authentication failed for user "xxx"

User xxx does not have OTSDB_ADMIN or OTSDB_MGR privilege – Not Authorized.
```

4.2 Check a Token

The ‘check’ GET method can be called with a token argument. Example:

```
http://localhost:8080/OHydroJson/check?token=6d34fa0e3bb72fcd
```

If the token is valid, the token JSON object will be returned in the same format described above for authentication.

If the token is not valid, HTTP 410 is returned. The body of the response is JSON in the form:

```
{"errMessage":"Token '6d34fa0e3bb72fcd' does not exist.","status":410}
```

4.3 GET Reference List(s)

The ‘reflists’ GET method will return all reference lists or a specific reference list.

Example:

```
http://localhost:8080/OHydroJson/reflists?name=TransportMediumType
```

Authentication is not required for this method, but if a token argument is provided the lastUser timer in the token will be updated.

If no “name” argument is provided, then all reference lists in the database are returned.

The JSON structure of the returned data looks like this:


```
{
  "TransportMediumType": {
    "enumName": "TransportMediumType",
    "items": {
      "iridium": {
        "description": "Iridium IMEI",
        "value": "iridium"
      },
      "other": {
        "description": "Other",
        "value": "other"
      },
      "incoming-tcp": {
        "description": "Incoming TCP on a Listening Socket",
        "value": "incoming-tcp"
      },
      "goes": {
        "description": "GOES DCP",
        "value": "goes"
      },
      "goes-self-timed": {
        "description": "GOES DCP Self-Timed Message",
        "value": "goes-self-timed"
      },
      "polled-modem": {
        "description": "Polled via modem",
        "value": "polled-modem"
      },
      "polled-tcp": {
        "description": "Polled via TCP Socket",
        "value": "polled-tcp"
      },
      "data-logger": {
        "description": "Electronic Data Logger File",
        "value": "data-logger"
      }
    }
  }
}
```


The following reference lists are currently available:

- DataSourceType – Used in Database Editor to link a data source name to a Java class
- PortType – Used in the polling interface
- ScriptType – A list of DECODES Platform Configuration Script types (reserved for future use
- StatisticsCode – Valid statistics codes that can be used in a time series identifier
- SiteNameType – Known site name types, e.g. NWSHB5 (National Weather Service Handbook 5), USGS (Site Number), CODWR (Colorada Dept of Water Resources).
- DataConsumer – Links a data consumer type name to Java Code (e.g. File, Directory, Socket)
- Measures – A list of physical attributes measured by an engineering unit (e.g. force, temperature, mass)
- UnitConversionAlgorithm – A list of algorithms for unit conversion
- DataOrder – Ascending (oldest first) or Descending (newest first) time order
- GroupType – Used for annotation in time series groups
- LoggerType – Used in the polling interface
- LookupAlgorithm
- EquationScope
- UnitFamily – Metric, English, or Universal
- OutputFormat – Links a name to Java code for formatting output data (e.g. SHEF, CSV)
- DataTypeStandard – Known standards for specifying data type (e.g. SHEF-PE, CWMS, HDB)
- TransportMediumType – A Transport Medium uniquely identifies a platform. There are several types: GOES ID, Iridium IMEI, Polled Identifier, etc.
- Season – User can create any number of seasons that start/end at specified time of year
- EquipmentType
- ApplicationType – Computation Process, Comp Depends Daemon, DECODES Routing Scheduler, etc.
- RecordingMode

4.4 GET Network List(s)

The ‘netlists’ GET method will return all network lists or a specific network list.

Example:

```
http://localhost:8080/OHydroJson/netlists?name=test
```

Authentication is not required for this method, but if a token argument is provided the lastUser timer in the token will be updated.

If no “name” argument is provided, then all lists in the database are returned.

The JSON structure of the returned data looks like this:


```
{
  "test": {
    "items": {
      "CE344292": {
        "description": "Iowa River at Oakville, IA (USGS)",
        "platformName": "OKVI4",
        "transportId": "CE344292"
      },
      "CE628300": {
        "description": "IOWA RIVER NEAR MARENGO 1N",
        "platformName": "MROI4",
        "transportId": "CE628300"
      },
      "CE2DD632": {
        "description": "IOWA RIVER NEAR ROWAN 4NW",
        "platformName": "ROWI4",
        "transportId": "CE2DD632"
      },
      "CE3941F4": {
        "description": "Ball Mountain Dam, West River",
        "platformName": "BMD",
        "transportId": "CE3941F4"
      }
    },
    "name": "test",
    "siteNameTypePref": "nwshb5",
    "transportMediumType": "goes"
  }
}
```

4.5 POST Network List

The ‘netlist’ POST method requires a valid token. It takes a single network list in JSON format. Note that the GET netslists method returns a map of named lists – this method is given and returns a single list.

The network list in the database is replaced with the one sent.

4.6 GET Time Series Catalog

The tscatalog method can be used to return the entire, or subset of, the time series catalog that is known to the server.

Examples:

```
http://localhost:8080/OHydroJson/tscatalog?site=BFD,CHCR&param=Elev

http://localhost:8080/OHydroJson/tscatalog?param=Stage&interval=1Hour,4Hours
```

Without arguments, it will return the entire catalog of time series. The following option arguments may be included:

- site: comma-separated list of site names. Only TSIDs from specified site(s) will be returned.
- param: comma-separated list of parameter values
- interval: comma-separated list of interval values
- active: boolean value (e.g. yes, no, true, false, T, F, Y, N).
- tsids: comma-separated list of complete time series IDs. If present, this overrides the site, param, interval, and active filters.
- token: Not required, but if included, the token lastUsed time will be updated.

The arguments may be combined for an increasingly fine filter.

The return JSON value contains time series identifiers according to the original HydroJSON specification. The format is as follows:


```
{
  "BFD": {
    "HUC": "",
    "active_flag": "T",
    "coordinates": {
      "datum": "WGS84",
      "latitude": 42.4278,
      "longitude": -72.0261
    },
    "elevation": {
      "accuracy": 0,
      "datum": "NGVD29",
      "method": "",
      "value": 234.7
    },
    "location_id": "BFD",
    "location_type": "",
    "name": "BFD",
    "responsibility": "",
    "time_format": "%Y-%m-%dT%H:%M:%S%z",
    "timeseries": {
      "BFD.Precip.Inst.15Minutes.0.DCP-raw": {
        "active_flag": "T",
        "count": 0,
        "duration": "0",
        "interval": "15Minutes",
        "notes": "BFD.Precip.Inst.15Minutes.0.DCP-raw",
```

```
        "param": "Precip",
        "sigfig": 3,
        "units": "in"
      },
      "BFD.Elev-Pool.Inst.15Minutes.0.DCP-raw": {
        "active_flag": "T",
        "count": 0,
        "duration": "0",
        "interval": "15Minutes",
        "notes": "BFD.Elev-Pool.Inst.15Minutes.0.DCP-raw",
        "param": "Elev-Pool",
        "sigfig": 3,
        "units": "ft"
      }
    },
    "timezone": "America/New_York",
    "tz_offset": 0
  }
}
```

4.7 GET Time Series Data

The tsdata method extends the tscatalog method. The same filtering arguments described for tscatalog may be used here. In addition, you may specify a time range for data retrieval by using the since and until arguments.

Example:

```
http://localhost:8080/OHydroJson/tsdata?site=BFD,CHCR&active=true&since=201
1/021/22:00&until=2011/021/23:30
```

The since and until arguments may have any of the following formats:

`now-1day`	The word "now" minus an increment times a unit. Examples: now-1day, now-5hours, now-1week, etc.
`now`	The current time that the web service call was made.
`YYYY/DDD/HH:MM:SS`	A complete Julian Year, Day-of-Year, and Time
`YYYY/DDD/HH:MM`	Seconds omitted means zero.
`DDD/HH:MM:SS`	Assume current year
`DDD/HH:MM`	
`HH:MM:SS`	Assume current day
`HH:MM`	

If 'since' is omitted, then the earliest values in storage are included. If 'until' is omitted, then the latest values are included.

Data is returned in a format extended from tscatalog described above. After the "units" field will be a set of values. For each value, the date/time, value, and flags are provided.

For the query:

```
http://localhost:8080/OHydroJson/tsdata?tsid=BFD.Precip.Inst.15Minutes.0.DCP-
raw,BFD.Elev-Pool.Inst.15Minutes.0.DCP-raw&since=2011/22/00:00&until=2011/22/01:00
```

Here is the response:

```
{
  "BFD": {
    "HUC": "",
    "active_flag": "T",
    "coordinates": {
      "datum": "WGS84",
      "latitude": 42.4278,
      "longitude": -72.0261
    },
    "elevation": {
      "accuracy": 0,
      "datum": "NGVD29",
      "method": "",
      "value": 234.7
    },
    "location_id": "BFD",
    "location_type": "",
    "name": "BFD",
    "responsibility": "",
    "time_format": "%Y-%m-%dT%H:%M:%S%z",
    "timeseries": {
```

```
"BFD.Precip.Inst.15Minutes.0.DCP-raw": {
  "active_flag": "T",
  "count": 5,
  "duration": "0",
  "end_timestamp": "2011-01-22T01:00:00",
  "interval": "15Minutes",
  "max_value": 15.51,
  "min_value": 15.51,
  "notes": "BFD.Precip.Inst.15Minutes.0.DCP-raw",
  "param": "Precip",
  "sigfig": 3,
  "start_timestamp": "2011-01-22T00:00:00",
  "units": "in",
  "values": [
    [
      "2011-01-22T00:00:00",
      15.51,
      0
    ],
    [
      "2011-01-22T00:15:00",
      15.51,
      0
    ],
    [
      "2011-01-22T00:30:00",
      15.51,
      0
    ],
    [
      "2011-01-22T00:45:00",
      15.51,
      0
    ],
    [
      "2011-01-22T01:00:00",
      15.51,
      0
    ]
  ]
},
"BFD.Elev-Pool.Inst.15Minutes.0.DCP-raw": {
  "active_flag": "T",
  "count": 5,
  "duration": "0",
  "end_timestamp": "2011-01-22T01:00:00",
  "interval": "15Minutes",
  "max_value": 776.94,
  "min_value": 776.94,
  "notes": "BFD.Elev-Pool.Inst.15Minutes.0.DCP-raw",
  "param": "Elev-Pool",
  "sigfig": 3,
  "start_timestamp": "2011-01-22T00:00:00",
  "units": "ft",
  "values": [
    [
      "2011-01-22T00:00:00",
```

```
            776.94,
            0
          ],
          [
            "2011-01-22T00:15:00",
            776.94,
            0
          ],
          [
            "2011-01-22T00:30:00",
            776.94,
            0
          ],
          [
            "2011-01-22T00:45:00",
            776.94,
            0
          ],
          [
            "2011-01-22T01:00:00",
            776.94,
            0
          ]
        ]
      }
    },
    "timezone": "America/New_York",
    "tz_offset": 0
  }
}
```

4.8 GET Platform List

The GET platforms method returns a list of with a given Transport Medium type. It accepts the following arguments:

- tmtype – the transport medium type desired. If not provided, the default is GOES. The method will return any platform that has a transport medium with the given type.
- token: Not required, but if included, the token lastUsed time will be updated.

Data Structure TBD but will include

- Platform name
- Agency
- Transport ID
- Config Name
- Description

The returned JSON data is structured as follows:


```
{
  "CE344292": {
    "agency": "MVR",
    "config": "OKVI4",
    "description": "Iowa River at Oakville, IA (USGS)",
    "name": "unknownSite",
    "transportId": "CE344292"
  },
  "CE628300": {
    "agency": "MVR",
    "config": "MROI4",
    "description": "Iowa River @ Marengo,IA",
    "name": "unknownSite",
    "transportId": "CE628300"
  }
}
```

4.9 POST Convert Netlist from ".nl" File

This POST method is sent the contents of an LRGS ".nl" file as plain text and returns a JSON data structure representation of the file.


```
{"items":{
    "12345678":{"description":"Some Big Long Description",
    "platformName":"SomeName","transportId":
    "12345678"},"87654321":{"description":"Description Two",
    "platformName":"NameNumber2","transportId":"87654321"}
    }
}
```