OPENDCS

REST API

Document Revision 8

July, 2023

This Document is part of the OpenDCS Software Suite for environmental data acquisition and processing. The project home is: <https://github.com/opendcs/opendcs>

See INTENT.md at the project home for information on licensing.

Table of Contents

1 Introduction 2

1.1 Document History 2

1.2 Glossary 2

2 REST Method Reference 2

2.1 Authentication and Authorization 2

2.2 Check a Token 2

2.3 Reference Lists 2

2.3.1 GET reflists 2

2.3.2 POST reflist 2

2.3.3 DELETE reflist 2

2.3.4 GET seasons 2

2.3.5 GET season 2

2.3.6 POST season 2

2.3.7 DELETE season 2

2.4 Network Lists 2

2.4.1 GET netlistrefs 2

2.4.2 GET netlist 2

2.4.3 POST netlist 2

2.4.4 DELETE netlist 2

2.5 DECODES Data Source Records 2

2.5.1 GET datasourcerefs 2

2.5.2 GET datasource 2

2.5.3 POST datasource 2

2.5.4 DELETE datasource 2

2.6 DECODES Site Records 2

2.6.1 GET siterefs 2

2.6.2 GET site 2

2.6.3 POST site 2

2.6.4 DELETE site 2

2.7 DECODES Platform Configurations 2

2.7.1 GET configrefs 2

2.7.2 GET config 2

2.7.3 POST config 2

2.7.4 DELETE config 2

2.8 DECODES Platform Records 2

2.8.1 GET platformrefs 2

2.8.2 GET platform 2

2.8.3 POST platform 2

2.8.4 DELETE platform 2

2.9 DECODES Presentation Group Records 2

2.9.1 GET presentationrefs 2

2.9.2 GET presentation 2

2.9.3 POST presentation 2

2.9.4 DELETE presentation 2

2.10 DECODES Routing Spec Records 2

2.10.1 GET routingrefs 2

2.10.2 GET routing 2

2.10.3 POST routing 2

2.10.4 DELETE routing 2

2.11 Schedule Entry Methods 2

2.11.1 GET schedulerefs 2

2.11.2 GET schedule 2

2.11.3 POST schedule 2

2.11.4 DELETE schedule 2

2.12 Loading Application Records 2

2.12.1 GET apprefs 2

2.12.2 GET app 2

2.12.3 POST app 2

2.12.4 DELETE app 2

2.13 Algorithm Methods 2

2.13.1 GET algorithmrefs 2

2.13.2 GET algorithm 2

2.13.3 POST algorithm 2

2.13.4 DELETE algorithm 2

2.14 Computation Methods 2

2.14.1 GET computationrefs 2

2.14.2 GET computation 2

2.14.3 POST computation 2

2.14.4 DELETE computation 2

2.14.5 POST resolvecomp 2

2.14.6 POST comptest 2

2.15 TSDB Properties Methods 2

2.15.1 GET tsdb\_properties 2

2.15.2 POST tsdb\_properties 2

2.16 Engineering Unit Methods 2

2.16.1 GET unitlist 2

2.16.2 POST eu 2

2.16.3 DELETE eu 2

2.16.4 GET euconvlist 2

2.16.5 POST euconv 2

2.16.6 DELETE euconv 2

2.17 Data Type Methods 2

2.17.1 GET datatypelist 2

2.18 Retrieving Property Specs 2

2.19 Retrieving DCP Messages 2

2.19.1 Retrieval a Single Message for Testing Decode 2

2.19.2 DCP Message Flag Bit Encoding 2

2.19.3 Multiple Message Retrieval 2

2.19.4 Retrieve LRGS Status 2

2.20 Test Decoding 2

3 Time Series Methods 2

3.1 GET Time Series References 2

3.2 GET Time Series Specification 2

3.3 GET Time Series Data 2

3.4 Interval Methods 2

3.4.1 GET intervals 2

3.4.2 POST interval 2

3.4.3 DELETE interval 2

3.5 Time Series Group Methods 2

3.5.1 GET tsgrouprefs 2

3.5.2 GET tsgroup 2

3.5.3 POST tsgroup 2

3.5.4 DELETE tsgroup 2

3.5.5 POST tsgroupeval 2

4 Network List Methods 2

4.1 POST Convert Netlist from “.nl” File 2

5 OpenDCS Process Monitor and Control (M&C) 2

5.1 App M&C 2

5.1.1 GET appstat 2

5.1.2 GET appevents 2

5.1.3 POST appstart 2

5.1.4 POST appstop 2

5.2 Routing Monitor and Control 2

5.2.1 GET routingstatus 2

5.2.2 GET routingexecstatus 2

5.2.3 GET dacqevents 2

5.2.4 GET platformstat 2

# Introduction

OpenDCS Rest API is web application that provides access to the OpenDCS database using JSON (Java Script Object Notation).

OpenDCS Rest API is intended to run as a stand-alone Java program. It uses embedded JETTY to implement the web services.

It was developed for the U. S. Army Corps of Engineers. Source and documentation may be found here:

<https://github.com/opendcs/odcsapi>

NOTE: THE PROJECT IS NOT YET UPLOADED TO GITHUB. THE “odcsapi.tgz” tarball contains the source image.

This contains a maven project which you can build with the commands:

cd ***[downloaded project image here]***

mvn clean install

This will create target/odcsapi-0.0.1-SNAPSHOT.tar.gz. It is recommended that you unpack this at the same place where you already have an OPENDCS installation running. It will create a subdirectory called “odcsapi-0.0.1-SNAPSHOT”. To run the JETTY application, use these commands, which you can place in a bash script if desired:

export DCSTOOL\_HOME=`pwd`

export DCSTOOL\_USERDIR=`pwd`

export JAVA\_OPTS="-DDCSTOOL\_HOME=$DCSTOOL\_HOME -DDCSTOOL\_USERDIR=$DCSTOOL\_USERDIR"

export JAVA\_ARGS="-p 8080 -c odcsapi"

cd odcsapi-0.0.1-SNAPSHOT

bin/start.sh

You may customize JAVA\_ARGS for your environment. Available arguments are:

* -p ***portnum*** The default listening port is 8080.
* -c ***context*** This sets the context for the webapp to use in URLs. The default is “odcsapi”.
* -P ***profile-file*** See below.
* -s Activates “secure mode.” In secure mode, tokens may only be passed in the HTTP header and *not* on the URL. Specifically, they must be passed in an Authentication Bearer header. In non-secure mode, tokens may be passed in either way.

As with most OPENDCS programs, this webapp will attempt to open $DCSTOOL\_USERDIR/decodes.properties to determine how to access the OpenDCS database and other settings. If you want to use a different file (for example, you may have multiple profiles accessing different databases) you can use the -P *profile-file* argument.

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).

## Document History

Revision 1, August 2022 – Based on a previous document that was for HydoJSON.

Revision 2, December 2022

* In all example URLs, “OHydroJson” has been replaced by “odcsapi.” This is the default application name used by the Jetty release.
* Time series methods have been completely reworked. The old HydroJSON methods are removed and replaced by the methods described in chapter 3.

Revision 3, March 2023

* Section 1.0 above, new “-s” command line argument to activate secure mode.
* Section 2.1 new text is added at the end of the section describing the new “GET credentials”
* New sections for reflist POST and DELETE methods.

Revision 4 May 2023

* New methods for GET/POST/DELETE seasons.
* New methods for GET/POST/DELETE intervals.
* New methods for retrieving DCP messages.

Revision 5 June 2023

* Added section 5.2, Routing Monitor and Control, along with its several subsections.

Revision 8 July 2023

* Added section on Time Series Group methods.

## Glossary

JSON Java Script Object Notation

WAR Web Archive – deployable web applications are typically distributed as WAR files.

## Installation and Configuration

There are two types of installations/configurations. One is Jetty and the other is a WAR file.

* Jetty
  + Change directory to the base directory of the project.
  + Run the following command:
    - mvn install
  + In the ‘target’ directory, there will be a .tar.gz file. This is the newly created jetty tar ball.
  + Move the tar ball to the desired location and extract it.
  + Then change directory to the ‘bin’ directory. There will be a ‘start.sh’ file.
  + Create a shell script, to run start.sh with some extra configurations.
    - Example shell script:
    - export DCSTOOL\_HOME=/home/opendcs/OPENDCS

export DCSTOOL\_USERDIR=/home/opendcs

export JAVA\_OPTS="-DDCSTOOL\_HOME=$DCSTOOL\_HOME -DDCSTOOL\_USERDIR=$DCSTOOL\_USERDIR"

export JAVA\_ARGS="-p 8081 -c odcsapi -cors /home/testuser/OPENDCS/opendcs\_web\_cors.cfg -s"

./start.sh

* + - The java args help configure the server
      * -cors
        + A cors file that helps configure the cors settings of the server. Below is an example of text in a cors file.

*Access-Control-Allow-Origin:\**

*Access-Control-Allow-Headers:X-Requested-With,Content-Type,Accept,Origin,authorization*

*Access-Control-Allow-Methods:GET,POST,HEAD,OPTIONS,DELETE*

* + - * -c
        + The context (relative url path) of the API.
      * -p
        + HTTP port number
      * -sp
        + HTTPS port number. Requires a key and a key password to work.
      * -key
        + Path to a key that can be accessed by the webserver.
      * -kp
        + Key Password (the password that was used to generate the key).
      * -P
        + Decodes Properties file path (by default it’s at $DCSTOOL\_HOME/decodes.properties.
      * -s
        + Secure mode. The authentication is done via the header, rather than as parameters passed through parameters.
* WAR file
  + Once mvn install has been run to create a jetty instance, a war file can then be created.
    - Run the following command after the jetty embedded server tar ball is created
      * ant -f build-war.xml war

The Root URL for the service is specified using the command line arguments when the JETTY server is started. If the defaults are used, you can access the service from the local machine with:

http://localhost:8080/odcsapi/***function-call***

For example, to get a list of platform references:

http://localhost:8080/odcsapi/platformrefs

## 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":"xyz"}

* 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), its 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 “xyz”

User xyz does not have OTSDB\_ADMIN or OTSDB\_MGR privilege – Not Authorized.

Starting in API Version 0.0.3, authentication credentials (username and password) may be passed as shown above in the POST body. They may also be passed in a GET call to the “credentials” method, (e.g. “http://localhost:8080/odcsapi/credentials”) containing an HTTP Authentication Basic header in the form “username:password”.

The returned data to the GET call will be empty. The returned HTTP header will contain an Authentication Bearer token containing the newly-generated token.

In subsequent calls to any methods that modify the database, the valid token must be provided. The preferred way to do this is in an HTTP Authentication Bearer header. If secure mode is *not* activated when the API is started, the token may also be passed as a “token” argument on the URL.

## Check a Token

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

http://localhost:8080/odcsapi/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}

## Reference Lists

Reference lists are used in OpenDCS to populate pulldown lists and extend the functionality of the software.

### GET reflists

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

Example:

http://localhost:8080/odcsapi/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.

The “name” argument may have multiple values. For example:

<http://localhost:8080/odcsapi/reflists?name=scripttype,dataorder>

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

The JSON returned is an array of reference lists. The data below contains the ScriptType and DataOrder reference lists:

{

"ScriptType": {

"reflistId": 3,

"enumName": "ScriptType",

"items": {

"standard": {

"value": "standard",

"description": "DECODES Format Statements and Unit Conversions",

"execClassName": "DecodesScript",

"editClassName": null,

"sortNumber": 3

},

"nos": {

"value": "nos",

"description": "Hard-coded NOS data parser",

"execClassName": "NOSMessageParser",

"editClassName": null,

"sortNumber": 2

},

"ndbc": {

"value": "ndbc",

"description": "National Data Buoy Center Context-Sensitive Parser",

"execClassName": "NDBCMessageParser",

"editClassName": null,

"sortNumber": 1

}

},

"defaultValue": null,

"description": null

},

"DataOrder": {

"reflistId": 9,

"enumName": "DataOrder",

"items": {

"a": {

"value": "a",

"description": "Ascending",

"execClassName": null,

"editClassName": null,

"sortNumber": null

},

"d": {

"value": "d",

"description": "Descending",

"execClassName": null,

"editClassName": null,

"sortNumber": null

}

},

"defaultValue": "d",

"description": null

}

}

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

### POST reflist

The ‘reflist’ POST method requires a valid token. It takes a single network list in JSON format. Note that the above GET method has a plural ‘reflists’ and returns an array of named reference lists. This POST method has singular ‘reflist’. The POST body should be a single reference list, not an array of lists.

For creating a new reference list, leave reflistId out of the passed data structure.

For overwriting an existing one, include the reflistId that was previously returned. The network list in the database is replaced with the one sent.

### DELETE reflist

The DELETE reflist method requires a valid token.

Required argument reflistid must be passed.

Take care in deleting reference lists. Several modules within OpenDCS require the existence of certain lists.

### GET seasons

Seasons are denoted by an abbreviation, a full name, start date/time, end date/time, and an optional time zone. Seasons are used in various places in OpenDCS, usually to specify some type of conditional processing. For example, a platform water-level sensor may be disabled during a winter period because the river is likely to be covered in ice.

* The abbreviation should be a single alpha-numeric word with no embedded spaces.
* The name may contain spaces.
* Start and End date/time are strings in the format MM/dd-HH:mm. The specify a date and time within a year.
* If time zone is omitted, local time on the server is assumed.

The GET seasons method is called with an URL like the following:

http://localhost:8080/odcsapi/seasons

A list of seasons defined on the server is returned in the format shown below:

[

{

"abbr": "autumn",

"name": "Autumn",

"start": "09/22-12:00",

"end": "12/21-12:00",

"tz": "EST5EDT"

},

{

"abbr": "winter",

"name": "Winter",

"start": "12/21-12:00",

"end": "03/20-12:00",

"tz": "EST5EDT"

},

{

"abbr": "all\_year",

"name": "All Year",

"start": "01/01-00:00",

"end": "12/31-23:59",

"tz": null

},

{

"abbr": "0211",

"name": "Feb through Nov",

"start": "02/01-00:00",

"end": "11/30-23:59",

"tz": "UTC"

}

]

### GET season

To return a single season data structure use an URL like the following:

http://localhost:8080/odcsapi/season?abbr=autumn

Instead of a list of seasons, the returned data is a single season data structure:

{

"abbr": "autumn",

"name": "Autumn",

"start": "09/22-12:00",

"end": "12/21-12:00",

"tz": "EST5EDT"

}

### POST season

The POST season operation requires a valid token. It takes a data structure like the one described above for GET season. It creates or overwrites a single season record.

An optional URL argument “fromabbr” may be used to modify the abbreviation for an existing season.

### DELETE season

The POST season operation requires a valid token. It also requires an argument “abbr” corresponding to the season abbreviation. For example, to DELETE the “autumn” season, use the following URL:

http://localhost:8080/odcsapi/season?abbr=autumn

## Network Lists

A Network List is simply a list of Platforms.

### GET netlistrefs

The GET netlistrefs method returns references to network lists. It is intended to populate a pick list of network lists and does not contain all of the list elements.

Authentication is not required for this method, but if a token argument is provided the

Examples:

http://localhost:8080/odcsapi/netlistrefs

http://localhost:8080/odcsapi/netlistrefs?tmtype=goes

With no arguments, a list of network lists in the database is returned. The format is as follows:

[

{

"lastModifyTime": "2020-08-22T14:36:55.705Z[UTC]",

"name": "BFD-BMD",

"netlistId": 1,

"numPlatforms": 3,

"siteNameTypePref": "nwshb5",

"transportMediumType": "goes"

},

{

"lastModifyTime": "2020-12-15T17:51:04.194Z[UTC]",

"name": "goes2",

"netlistId": 6,

"numPlatforms": 3,

"siteNameTypePref": "nwshb5",

"transportMediumType": "goes"

}

]

Note that the list contents (i.e. the references to the platforms in the list) is not included. Rather a count of platforms in the list is given.

The “netlistId” element is a unique key to be used for retrieving entire lists.

### GET netlist

The ‘netlists’ GET method will return a specific network list in its entirety.

Example:

http://localhost:8080/odcsapi/netlists?netlistid=1

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

The JSON structure of the returned data looks like this:

{

"items": {

"14159500": {

"description": "",

"platformName": "CGRO",

"transportId": "14159500"

},

"14372300": {

"description": "",

"platformName": "AGNO",

"transportId": "14372300"

}

},

"lastModifyTime": "2020-10-19T18:14:14.788Z[UTC]",

"name": "USGS-Sites",

"netlistId": 4,

"siteNameTypePref": "nwshb5",

"transportMediumType": "other"

}

Possible error codes:

* 406 (Not Acceptable): The required ‘netlistid’ parameter was missing in the URL.
* 404 (Not Found): The requested netlistid was not found in the database.

### POST netlist

The ‘netlist’ POST method requires a valid token. It takes a single network list in JSON format, as described for the GET method.

For creating a new network list, leave netlistId out of the passed data structure.

For overwriting an existing one, include the netlistId that was previously returned. The network list in the database is replaced with the one sent.

### DELETE netlist

The DELETE netlist method requires a valid token.

Required argument netlistid must be passed.

Error 405 will be returned if network list is used by one or more routing specs and cannot be deleted. The body of the error will be a message containing the name of the routing specs using the referenced netlist.

## DECODES Data Source Records

### GET datasourcerefs

Example:

http://localhost:8080/odcsapi/datasourcerefs

This method returns a JSON list of DECODES Data Source records suitable for displaying in a table or pick-list. The returned structure contains only the high-level descriptive information about each data source. The returned structure is as follows:

{

"CDADATA-As-MBHydro": {

"arguments": "host=cdadata.wcda.noaa.gov, password=\*\*\*\*, username=mbhydro",

"dataSourceId": 5,

"name": "CDADATA-As-MBHydro",

"type": "lrgs",

"usedBy": 0

},

"edl-tcp-poll": {

"arguments": "availablePorts=20, maxBacklogHours=48, pollNumTries=3, portType=tcpclient, saveSessionFile=$DCSTOOL\_USERDIR/edl-polled/$SITENAME-$DATE",

"dataSourceId": 9,

"name": "edl-tcp-poll",

"type": "polled",

"usedBy": 1

},

"testgroup": {

"dataSourceId": 10,

"name": "testgroup",

"type": "hotbackupgroup",

"usedBy": 0

}

}

The arguments (properties) are represented by a string with comma delimiter. Passwords within the string are replaced with four asterisks.

### GET datasource

The GET datasource method returns a single DECODES data source with all of its detail. The integer argument datasourceid is required.

Example:

http://localhost:8080/odcsapi/datasource?datasourceid=10

Returned structure:

{

"dataSourceId": 10,

"groupMembers": [

{

"dataSourceId": 4,

"dataSourceName": "Cove-LRGS"

},

{

"dataSourceId": 5,

"dataSourceName": "CDADATA-As-MBHydro"

},

{

"dataSourceId": 7,

"dataSourceName": "USGS-Web"

}

],

"name": "testgroup",

"props": {

"whatevs": "something 567",

"abc": "def"

},

"type": "hotbackupgroup",

"usedBy": 0

}

### POST datasource

The ‘datasource’ POST method requires a valid token. It takes a single datasource in JSON format, as described for the GET method.

For creating a new network list, leave datasourceId out of the passed data structure.

For overwriting an existing one, include the datasourceId that was previously returned by GET. The network list in the database is replaced with the one sent.

### DELETE datasource

The DELETE datasource method requires a valid token.

Required argument datasource id must be passed.

Error 405 will be returned if datasource list is used by one or more routing specs and cannot be deleted. The body of the error will be a message containing the name of the routing specs using the referenced netlist.

## DECODES Site Records

### GET siterefs

Example:

http://localhost:8080/odcsapi/siterefs

This method returns a JSON list of DECODES Site records suitable for displaying in a table or pick-list. The returned structure contains only the numeric ID (unique), description, and an array of site names in the following format:

[

{

"description": "Iowa River at Oakville, IA (USGS)",

"siteId": 1,

"sitenames": {

"CWMS": "OKVI4",

"nwshb5": "OKVI4",

"usgs": "05465700",

"local": "OKVI4"

}

},

{

"description": "IOWA RIVER NEAR ROWAN 4NW",

"siteId": 2,

"sitenames": {

"CWMS": "ROWI4",

"USGS": "05449500"

}

},

{

"description": "Site used for Regression Testing",

"siteId": 3,

"sitenames": {

"CWMS": "TESTSITE1"

}

}

]

The numeric ID may be used in subsequent calls to the “site” method.

### GET site

Example:

http://localhost:8080/odcsapi/site?siteid=3

This method returns a JSON representation of a single, complete DECODES Site record. The structure is as follows:

{

"country": "USA",

"description": "Barre Falls Dam, Ware River",

"elevUnits": "M",

"elevation": 234.7,

"latitude": "42.4278",

"longitude": "-72.0261",

"nearestCity": "Barre Falls Dam",

"properties": {

"some": "thing",

"something": "else"

},

"region": "",

"siteId": 7,

"sitenames": {

"CWMS": "BFD",

"NWSHB5": "BFD"

},

"state": "MA",

"timezone": "America/New\_York"

}

### POST site

The POST site method requires a valid token. It takes a single DECODES site record in JSON format, as described above for GET site.

For creating a new site, leave siteId out of the passed data structure.

For overwriting an existing one, include the siteId that was previously returned. The site list in the database is replaced with the one sent.

### DELETE site

The DELETE site method requires a valid token.

Required argument siteid must be passed.

Error 405 will be returned if site is used by one or more platforms and cannot be deleted.

## DECODES Platform Configurations

### GET configrefs

Example:

http://localhost:8080/odcsapi/configrefs

This method returns a JSON list of DECODES Config records suitable for displaying in a table or pick-list. The returned structure contains an array in the following format:

[

{

"configId": 1,

"description": "WSC SHEF - 2 sensors - HG, VB",

"name": "Shef-WSC-Hydro-RCOYCHER",

"numPlatforms": 1

},

{

"configId": 2,

"description": "WSC SHEF - 2 sensors - QR=QF, HG",

"name": "Shef-WSC-Hydro-RBRDDDVH",

"numPlatforms": 1

},

{

"configId": 3,

"description": "WSC SHEF - 2 sensors - HG, VB",

"name": "Shef-WSC-Hydro-RBLOCLEE",

"numPlatforms": 1

},

{

"configId": 4,

"description": "AE SHEF - 4 sensors - PC, TA, SW, YB=VB",

"name": "Shef-AE-Met-SESK",

"numPlatforms": 1

},

{

"configId": 5,

"description": "WSC SHEF - 2 sensors - HG, VB",

"name": "Shef-WSC-Hydro-RBULLRES",

"numPlatforms": 1

},

{

"configId": 6,

"description": "WSC SHEF - 2 sensors - HG, VB",

"name": "Shef-WSC-Hydro-RREDBIN",

"numPlatforms": 1

}

]

The numeric ID may be used in subsequent calls to the “config” method.

### GET config

Example:

http://localhost:8080/odcsapi/config?configid=12

This method returns a JSON representation of a single, complete DECODES Config record. Example:

{

"configId": 12,

"configSensors": [

{

"dataTypes": {

"NL-SHEF": "WL",

"SHEF-PE": "HG",

"W-SHEF": "WL"

},

"properties": {},

"recordingInterval": 3600,

"recordingMode": "F",

"sensorName": "WL",

"sensorNumber": 1,

"timeOfFirstSample": 0

},

{

"dataTypes": {

"SHEF-PE": "VB"

},

"properties": {},

"recordingInterval": 3600,

"recordingMode": "F",

"sensorName": "VB",

"sensorNumber": 4,

"timeOfFirstSample": 0

}

],

"description": "WSC SHEF - 2 sensors - HG, VB",

"name": "Shef-WSC-Hydro-RVERMMAR",

"numPlatforms": 0,

"scripts": [

{

"dataOrder": "D",

"formatStatements": [

{

"format": "s(50,':',DONE),x,F(F,A,10d' ')",

"label": "getlabel",

"sequenceNum": 0

},

{

"format": "s(12,'#',getlabel),x,f(mint,a,3d' +-',1),32(w,c(N,skiphg),F(S,A,12d' +-:',1)), >GETLABEL",

"label": "hg",

"sequenceNum": 1

},

{

"format": "s(12,'.',getlabel),-2x,32(w,c(N,getlabel),F(S,A,12d' +-:',4)), >GETLABEL",

"label": "vb",

"sequenceNum": 2

},

{

"format": "s(12,'.',enddata),-2x,32(w,c(N,enddata),F(S,A,12d' +-:',4)), >GETLABEL",

"label": "x-nointerval",

"sequenceNum": 3

},

{

"format": ">vb",

"label": "battery",

"sequenceNum": 4

},

{

"format": "-3x,>getlabel",

"label": "enddata",

"sequenceNum": 5

},

{

"format": "/",

"label": "error",

"sequenceNum": 6

},

{

"format": "/,>getlabel",

"label": "done",

"sequenceNum": 7

},

{

"format": "-2x,5(S(50,':',done),S(3,'HG',enddata))>getlabel",

"label": "skiphg",

"sequenceNum": 8

},

{

"format": "s(12,'#',getlabel),x,f(mint,a,3d' +-',6),32(w,c(N,enddata),F(S,A,12d' +-:',6)), >GETLABEL",

"label": "pr",

"sequenceNum": 9

},

{

"format": ">pr",

"label": "pc",

"sequenceNum": 10

},

{

"format": ">hg",

"label": "hk",

"sequenceNum": 11

},

{

"format": "s(12,'#',getlabel),x,f(mint,a,3d' +-',7),32(w,c(N,enddata),F(S,A,12d' +-:',7)), >GETLABEL",

"label": "tx",

"sequenceNum": 12

},

{

"format": "s(12,'#',getlabel),x,f(mint,a,3d' +-',8),32(w,c(N,enddata),F(S,A,12d' +-:',8)), >GETLABEL",

"label": "tn",

"sequenceNum": 13

},

{

"format": "s(12,'.',enddata),-2x,32(w,c(N,enddata),F(S,A,12d' +-:',4)), >GETLABEL",

"label": "vb-x",

"sequenceNum": 14

},

{

"format": "s(12,'#',getlabel),x,f(mint,a,3d' +-',9),32(w,c(N,skiphg),F(S,A,12d' +-:',9)), >GETLABEL",

"label": "hh",

"sequenceNum": 15

}

],

"name": "ST",

"scriptSensors": [

{

"sensorNumber": 1,

"unitConverter": {

"a": 0,

"algorithm": "none",

"b": 0,

"c": 0,

"d": 0,

"e": 0,

"f": 0,

"fromAbbr": "raw",

"toAbbr": "M"

}

},

{

"sensorNumber": 4,

"unitConverter": {

"a": 0,

"algorithm": "none",

"b": 0,

"c": 0,

"d": 0,

"e": 0,

"f": 0,

"fromAbbr": "raw",

"toAbbr": "V"

}

}

]

}

]

}

### POST config

The POST config method requires a valid token. It takes a single DECODES Platform Configuration record in JSON format, as described above for GET config.

For creating a new config, leave configId out of the passed data structure.

For overwriting an existing one, include the configId that was previously returned. The configuration in the database is replaced with the one sent.

### DELETE config

The DELETE config method requires a valid token.

Required argument configid must be passed.

Error 405 will be returned if the referenced configuration is used by one or more platforms and cannot be deleted.

## DECODES Platform Records

### GET platformrefs

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

* tmtype – the transport medium type desired. If not provided, all types are returned. The method will return any platform that has a transport medium with the given type. The returned data structure will still contain all of the transport media in the platform, but the list of platforms will be filtered to only include platforms with a TM of the given type:
  + NOTE: medium type “goes” will match either goes-self-timed or goes-random.
* token: Not required, but if included, the token lastUsed time will be updated.

Data Structure TBD but will include

* name – an index into the hashed set. Combination of site name and designator
* agency – The agency that owns and/or maintains this platform
* configId – Numeric surrogate key to the configuration record
* description
* platformId - Numeric surrogate key to the platform record
* siteId - Numeric surrogate key to the site record
* transportMedia – a list of tmtype/tm id pairs.

The returned JSON data is structured as follows:

{

"BFD": {

"agency": "CWMS",

"configId": 5,

"description": "Barre Falls Dam, Ware River, MA",

"name": "BFD",

"platformId": 4,

"siteId": 7,

"transportMedia": {

"goes-random": "CE3234CC",

"goes-self-timed": "CE3234CC"

}

},

"BMD-tailwater": {

"agency": "CWMS",

"configId": 7,

"description": "Ball Mountain TW",

"designator": "tailwater",

"name": "BMD-tailwater",

"platformId": 6,

"siteId": 8,

"transportMedia": {

"goes-random": "CE3941F4",

"goes-self-timed": "CE3941F4"

}

},

"BMD": {

"agency": "CWMS",

"configId": 6,

"description": "Ball Mountain Dam, West River,VT",

"name": "BMD",

"platformId": 5,

"siteId": 8,

"transportMedia": {

"goes-random": "CE31D030",

"goes": "CE31D030",

"goes-self-timed": "CE31D030"

}

}

}

### GET platform

Example:

http://localhost:8080/odcsapi/platform?platformid=5

This method returns a JSON representation of a single, complete DECODES Platform record. The following structure is returned:

{

"agency": "CWMS",

"configId": 6,

"description": "Ball Mountain Dam, West River,VT",

"lastModified": "2022-01-21T14:18:21.176Z[UTC]",

"platformId": 5,

"platformSensors": [

{

"max": 120,

"min": -40,

"sensorNum": 1,

"sensorProps": {}

},

{

"sensorNum": 1,

"sensorProps": {}

},

{

"sensorNum": 1,

"sensorProps": {}

},

{

"sensorNum": 1,

"sensorProps": {}

},

{

"actualSiteId": 5,

"sensorNum": 1,

"sensorProps": {}

}

],

"production": false,

"properties": {},

"siteId": 8,

"transportMedia": [

{

"assignedTime": 2095,

"baud": 0,

"channelNum": 161,

"dataBits": 0,

"doLogin": false,

"mediumId": "CE31D030",

"mediumType": "goes-self-timed",

"parity": "U",

"scriptName": "ST",

"stopBits": 0,

"timeAdjustment": 0,

"timezone": "UTC",

"transportInterval": 3600,

"transportWindow": 5

},

{

"baud": 0,

"channelNum": 129,

"dataBits": 0,

"doLogin": false,

"mediumId": "CE31D030",

"mediumType": "goes-random",

"parity": "U",

"scriptName": "RD",

"stopBits": 0,

"timeAdjustment": 0,

"timezone": "UTC",

"transportInterval": 900

},

{

"baud": 0,

"channelNum": 235,

"dataBits": 0,

"doLogin": false,

"mediumId": "CE31D030",

"mediumType": "goes",

"parity": "U",

"scriptName": "Network\_Tower",

"stopBits": 0,

"timeAdjustment": 0,

"timezone": "UTC",

"transportInterval": 900

}

]

}

### POST platform

The POST platform method requires a valid token. It takes a single DECODES Platform record in JSON format, as described above for GET.

For creating a new platform, leave platformId out of the passed data structure.

For overwriting an existing one, include the platformId that was previously returned. The platform in the database is replaced with the one sent.

### DELETE platform

The DELETE platform method requires a valid token.

Required argument platformid must be passed.

## DECODES Presentation Group Records

### GET presentationrefs

Returns a list of references to presentation groups suitable for displaying a list. The returned structure is:

[

{

"groupId": 1,

"lastModified": "2022-01-14T14:43:47.297Z[UTC]",

"name": "CWMS-English",

"production": false

},

{

"groupId": 2,

"lastModified": "2020-05-11T20:23:20.103Z[UTC]",

"name": "CWMS-Metric",

"production": false

},

{

"groupId": 3,

"lastModified": "2020-05-11T20:23:20.118Z[UTC]",

"name": "SHEF-English",

"production": false

},

{

"groupId": 4,

"lastModified": "2020-05-11T20:33:10.275Z[UTC]",

"name": "regtest",

"production": false

},

{

"groupId": 5,

"lastModified": "2022-03-04T18:17:58.885Z[UTC]",

"name": "NRCS",

"production": true

},

{

"groupId": 6,

"inheritsFrom": "CWMS-English",

"lastModified": "2021-03-05T14:06:18.435Z[UTC]",

"name": "hydrodcs",

"production": false

}

]

### GET presentation

Example:

http://localhost:8080/odcsapi/presentation?groupid=4

This method returns a JSON representation of a single, complete DECODES Presentation Group record. The following structure is returned. Note that the optional min and max elements are not always present.

{

"elements": [

{

"dataTypeCode": "Elev",

"dataTypeStd": "CWMS",

"fractionalDigits": 2,

"units": "ft"

},

{

"dataTypeCode": "Elev-Pool",

"dataTypeStd": "CWMS",

"fractionalDigits": 2,

"units": "ft"

},

{

"dataTypeCode": "FLOW-HOLDOUT",

"dataTypeStd": "CWMS",

"fractionalDigits": 2,

"units": "cfs"

},

{

"dataTypeCode": "FLOW-INFLOW",

"dataTypeStd": "CWMS",

"fractionalDigits": 2,

"units": "cfs"

},

{

"dataTypeCode": "Temp",

"dataTypeStd": "CWMS",

"fractionalDigits": 2,

"max": 212,

"min": 0,

"units": "degF"

},

{

"dataTypeCode": "Flow",

"dataTypeStd": "CWMS",

"fractionalDigits": 2,

"units": "cfs"

}

],

"groupId": 4,

"lastModified": "2022-03-04T19:00:21.752Z[UTC]",

"name": "regtest",

"production": false

}

### POST presentation

The POST presentation method requires a valid token. It takes a single DECODES Presentation Group in JSON format, as described above for GET.

For creating a new record, leave groupId out of the passed data structure.

For overwriting an existing one, include the groupId that was previously returned. The presentation group in the database is replaced with the one sent.

### DELETE presentation

The DELETE platform method requires a valid token.

Required argument groupid must be passed in the URL.

## DECODES Routing Spec Records

### GET routingrefs

Example: [http://localhost:8080/odcsapi/routingrefs](http://localhost:8080/OHydroJson/routingrefs)

The returned structure is:

[

{

"dataSourceName": "OKVI4",

"destination": "pipe()",

"lastModified": "2020-05-11T20:24:53.052Z[UTC]",

"name": "OKVI4-input",

"routingId": 1

},

{

"dataSourceName": "OKVI4",

"destination": "opentsdb()",

"lastModified": "2022-01-14T14:50:50.982Z[UTC]",

"name": "OKVI4-ingest",

"routingId": 2

},

{

"dataSourceName": "MROI4-ROWI4",

"destination": "opentsdb()",

"lastModified": "2020-05-11T20:33:10.274Z[UTC]",

"name": "rs-MROI4-ROWI4",

"routingId": 3

},

{

"dataSourceName": "BFD-BMD",

"destination": "pipe()",

"lastModified": "2022-01-14T14:44:23.26Z[UTC]",

"name": "BFD-BMD-Decode-Test",

"routingId": 4

}

]

### GET routing

Example:

http://localhost:8080/odcsapi/routing?routingid=20

This method returns a JSON representation of a single routing spec. The returned structure is:

{

"applyTimeTo": "Both",

"ascendingTime": true,

"dataSourceId": 4,

"dataSourceName": "Cove-LRGS",

"destinationArg": "some-directory-path",

"destinationType": "directory",

"enableEquations": true,

"goesChannels": [

123

],

"goesRandom": true,

"goesSelfTimed": true,

"goesSpacecraftCheck": true,

"goesSpacecraftSelection": "East",

"iridium": true,

"lastModified": "2022-03-22T17:44:15.941Z[UTC]",

"name": "test",

"netlistNames": [

"goes1"

],

"networkDCP": true,

"outputFormat": "emit-ascii",

"outputTZ": "EST5EDT",

"parityCheck": true,

"paritySelection": "Good",

"platformIds": [

"12345678",

"004085C8"

],

"platformNames": [

"MROI4"

],

"presGroupName": "CWMS-English",

"production": true,

"properties": {},

"qualityNotifications": true,

"routingId": 20,

"settlingTimeDelay": true,

"since": "2022/065 00:00:00"

}

### POST routing

The POST routing method requires a valid token. It takes a single DECODES Routing Spec in JSON format, as described above for GET.

For creating a new record, leave routingId out of the passed data structure.

For overwriting an existing one, include the routingId that was previously returned. The routing spec in the database is replaced with the one sent.

### DELETE routing

The DELETE routing method requires a valid token.

Required argument routingid must be passed in the URL.

## Schedule Entry Methods

### GET schedulerefs

Example: [http://localhost:8080/odcsapi/schedulerefs](http://localhost:8080/OHydroJson/schedulerefs)

The returned structure is:

[

{

"appName": "RoutingScheduler",

"enabled": false,

"lastModified": "2020-12-15T17:52:13.934Z[UTC]",

"name": "goes1",

"routingSpecName": "goes1",

"schedEntryId": 9

},

{

"appName": "RoutingScheduler",

"enabled": false,

"lastModified": "2020-12-15T17:53:06.043Z[UTC]",

"name": "goes2",

"routingSpecName": "goes2",

"schedEntryId": 10

},

{

"enabled": false,

"lastModified": "2022-03-23T13:54:09.188Z[UTC]",

"name": "no\_app\_assigned",

"routingSpecName": "polltest",

"schedEntryId": 17

},

{

"appName": "RoutingScheduler",

"enabled": false,

"lastModified": "2022-01-14T14:45:01.336Z[UTC]",

"name": "junk",

"routingSpecName": "polltest",

"schedEntryId": 14

}

]

Note in the third entry above that appName may be omitted. In the database, a schedule entry may not (yet) be assigned to an application.

### GET schedule

Example:

http://localhost:8080/odcsapi/schedule?scheduleid=21

This method returns a JSON representation of a single schedule entry. The returned structure is:

{

"appId": 14,

"appName": "RoutingScheduler",

"enabled": false,

"lastModified": "2022-03-28T18:02:13.659Z[UTC]",

"name": "something-else",

"routingSpecId": 9,

"routingSpecName": "goes1",

"schedEntryId": 21,

"timeZone": "America/New\_York"

}

### POST schedule

The POST schedule method requires a valid token. It takes a single DECODES Schedule Entry in JSON format, as described above for GET.

For creating a new record, leave schedEntryId out of the passed data structure.

For overwriting an existing one, include the schedEntryId that was previously returned. The routing spec in the database is replaced with the one sent.

### DELETE schedule

The DELETE schedule method requires a valid token.

Required argument scheduleid must be passed in the URL.

## Loading Application Records

### GET apprefs

Example: [http://localhost:8080/odcsapi/apprefs](http://localhost:8080/OHydroJson/apprefs)

The returned structure is:

[

{

"appId": 1,

"appName": "decodes",

"comment": "Manual DECODES Routing Specs",

"lastModified": "2020-05-11T20:23:21Z[UTC]"

},

{

"appId": 3,

"appName": "compedit",

"comment": "Computation Editing Application",

"lastModified": "2020-05-11T20:23:21Z[UTC]"

},

{

"appId": 4,

"appName": "compproc",

"appType": "computationprocess",

"comment": "Main Computation Process",

"lastModified": "2022-03-30T20:49:44Z[UTC]"

},

{

"appId": 8,

"appName": "utility",

"appType": "utility",

"comment": "Utility Program -- used by several apps",

"lastModified": "2022-01-14T15:28:40Z[UTC]"

},

{

"appId": 10,

"appName": "dcpmon",

"comment": "DCP Monitor Web Application",

"lastModified": "2020-05-11T20:23:21Z[UTC]"

},

{

"appId": 14,

"appName": "RoutingScheduler",

"appType": "routingscheduler",

"comment": "Schedules and Executes DECODES Routing Specs",

"lastModified": "2020-05-11T20:23:21Z[UTC]"

},

{

"appId": 15,

"appName": "compdepends",

"appType": "compdepends",

"comment": "Daemon to determine computation dependencies",

"lastModified": "2020-05-11T20:23:21Z[UTC]"

}

]

### GET app

Example:

[http://localhost:8080/odcsapi/app?appid=4](http://localhost:8080/OHydroJson/app?appid=4)

Return data format:

{

"appId": 4,

"appName": "compproc",

"appType": "computationprocess",

"comment": "Main Computation Process",

"lastModified": "2022-03-30T20:49:44Z[UTC]",

"manualEditingApp": false,

"properties": {

"fromName": "idk"

}

}

Note: appType may be omitted if it is not defined in the database.

### POST app

The POST app method requires a valid token. It takes a single DECODES Loading Application in JSON format, as described above for GET.

For creating a new record, leave appId out of the passed data structure.

For overwriting an existing one, include the appId that was previously returned. The app in the database is replaced with the one sent.

### DELETE app

The DELETE schedule method requires a valid token.

Required argument appid must be passed in the URL.

This operation will fail if the loading application is currently being used by any computations or schedule entries, or if it is currently running and has an active CP\_COMP\_PROC\_LOCK record.

## Algorithm Methods

### GET algorithmrefs

Example: [http://localhost:8080/odcsapi/algorithmrefs](http://localhost:8080/OHydroJson/algorithmrefs)

The returned structure is:

[

{

"algorithmId": 1,

"algorithmName": "AlarmScreening",

"description": "Perform value, rate-of-change, stuck-sensor, and missing-val",

"execClass": "decodes.tsdb.alarm.AlarmScreeningAlgorithm",

"numCompsUsing": 2

},

{

"algorithmId": 2,

"algorithmName": "Bridge Clearance",

"description": "Computes bridge clearance by subtracting waterlevel from con",

"execClass": "decodes.tsdb.algo.BridgeClearance",

"numCompsUsing": 0

},

{

"algorithmId": 3,

"algorithmName": "CentralRunningAverageAlgorithm",

"description": "CentralRunningAverageAlgorithm averages single 'input' param",

"execClass": "decodes.tsdb.algo.CentralRunningAverageAlgorithm",

"numCompsUsing": 0

},

{

"algorithmId": 8,

"algorithmName": "ExpressionParserAlgorithm",

"description": "Allow up to 5 inputs labeled in1",

"execClass": "decodes.tsdb.algo.ExpressionParserAlgorithm",

"numCompsUsing": 0

},

…

]

### GET algorithm

Example:

[http://localhost:8080/odcsapi/algorithm?algorithmid=4](http://localhost:8080/OHydroJson/algorithm?algorithmid=4)

Return data format:

{

"algoScripts": [],

"algorithmId": 4,

"description": "Given two inputs, output the best one: If only one is present at the time-slice, output it. If one is outside the specified upper or lower limit (see properties) output the other. If both are acceptable, output the first one. Useful in situations where you have redundant sensors.",

"execClass": "decodes.tsdb.algo.ChooseOne",

"name": "ChooseOne",

"numCompsUsing": 1,

"parms": [

{

"parmType": "i",

"roleName": "input1"

},

{

"parmType": "i",

"roleName": "input2"

},

{

"parmType": "o",

"roleName": "output"

}

],

"props": {

"input1\_MISSING": "IGNORE",

"chooseHigher": "true",

"upperLimit": "999999999999.9",

"lowerLimit": "-999999999999.9",

"input2\_MISSING": "IGNORE"

}

}

### POST algorithm

The POST algorithm method requires a valid token. It takes a single OpenDCS Algorithm Record in JSON format, as described above for GET.

For creating a new record, leave algorithmId out of the passed data structure.

For overwriting an existing one, include the algorithmId that was previously returned. The algorithm in the database is replaced with the one sent.

### DELETE algorithm

The DELETE algorithm method requires a valid token.

Required argument algorithmid must be passed in the URL.

This operation will fail if the algorithm is currently being used by any computation records.

## Computation Methods

### GET computationrefs

Example: [http://localhost:8080/odcsapi/computationrefs](http://localhost:8080/OHydroJson/algorithmrefs)

The returned structure is:

[

{

"algorithmId": 24,

"algorithmName": "AverageAlgorithm",

"computationId": 1,

"description": "",

"enabled": false,

"name": "Daily Ave [ ... )",

"processId": 5,

"processName": "compproc\_regtest"

},

{

"algorithmId": 24,

"algorithmName": "AverageAlgorithm",

"computationId": 2,

"description": "",

"enabled": false,

"name": "Daily Ave ( ... ]",

"processId": 5,

"processName": "compproc\_regtest"

},

{

"algorithmId": 24,

"algorithmName": "AverageAlgorithm",

"computationId": 3,

"description": "",

"enabled": false,

"name": "Daily Ave [ ... ]",

"processId": 5,

"processName": "compproc\_regtest"

},

…

]

### GET computation

Example:

[http://localhost:8080/odcsapi/computation?computationid=4](http://localhost:8080/OHydroJson/algorithm?algorithmid=4)

Return data format:

{

"algorithmId": 24,

"algorithmName": "AverageAlgorithm",

"appId": 5,

"comment": "",

"computationId": 4,

"effectiveEndType": "No Limit",

"effectiveStartType": "No Limit",

"enabled": false,

"groupId": -1,

"lastModified": "2022-05-17T17:17:04.693Z[UTC]",

"name": "Daily Ave ( ... )",

"parmList": [

{

"algoParmType": "i",

"algoRoleName": "input",

"dataType": "Stage",

"dataTypeId": 48,

"deltaT": 0,

"duration": "0",

"interval": "15Minutes",

"paramType": "Inst",

"siteId": 1,

"siteName": "OKVI4",

"tsKey": 1,

"version": "raw"

},

{

"algoParmType": "o",

"algoRoleName": "average",

"dataType": "Stage",

"dataTypeId": 48,

"deltaT": 0,

"duration": "1Day",

"interval": "1Day",

"paramType": "Ave",

"siteId": 1,

"siteName": "OKVI4",

"tsKey": 5,

"version": "OO"

}

],

"props": {

"minSamplesNeeded": "1",

"aggUpperBoundClosed": "false",

"aggregateTimeZone": "UTC",

"average\_tsname": "HG-Ave-Open-Open",

"aggLowerBoundClosed": "false"

}

}

### POST computation

The POST computation method requires a valid token. It takes a single OpenDCS Computation Record in JSON format, as described above for GET.

For creating a new record, leave computationId out of the passed data structure.

For overwriting an existing one, include the computationId that was previously returned. The computation in the database is replaced with the one sent.

### DELETE computation

The DELETE computation method requires a valid token.

Required argument computationid must be passed in the URL.

### POST resolvecomp

Example URL for POST:

http://localhost:8080/odcsapi/resolvecomp?token=6b994be905e1fddf

This method requires a valid session token.

The POST data should be a computation in the format described above for GET computation.

The returned data will be a JSON array all time-series that can act as triggering inputs for the specified computation. For group-type computations, this may be many. The format of the returned data is as follows:

[

{ "uniqueString": "ROWI4.Stage.Inst.15Minutes.0.raw",

"key": 8,

"description": "ROWI4 - Stage (created by DECODES)",

"storageUnits": "ft"

},

{ "uniqueString": "MROI4.Stage.Inst.30Minutes.0.raw",

"key": 11,

"description": "MROI4 - Stage (created by DECODES)",

"storageUnits": "ft"

}

]

### POST comptest

Example URL for POST:

http://localhost:8080/odcsapi/comptest?token=6b994be905e1fddf

This method requires a valid session token.

The POST data is a special request structure that contains the computation to be tested, the time series to be used for testing, and the following three variables:

* “since” – date/time for the start of data
* “until” – date/time for the end of the data
* “traceOutput” – a Boolean value indicating whether you want trace-log data in the returned results. Note: if a large time window is selected, the trace results can be quite voluminous.

A sample comptest request data structure is shown below:

{

"computation":

{

"computationId": 5,

"name": "MROI4-ROWI4-Stage-Copy",

"comment": "",

"appId": 5,

"applicationName": null,

"lastModified": 1654538252952,

"enabled": false,

"effectiveStartType": "No Limit",

"effectiveStartDate": null,

"effectiveStartInterval": null,

"effectiveEndType": "No Limit",

"effectiveEndDate": null,

"effectiveEndInterval": null,

"algorithmId": 26,

"algorithmName": "CopyAlgorithm",

"parmList": [

{

"algoParmType": "i",

"algoRoleName": "input",

"tsKey": null,

"dataTypeId": null,

"dataType": "",

"interval": "",

"deltaT": 0,

"deltaTUnits": null,

"unitsAbbr": null,

"siteId": null,

"siteName": "",

"tableSelector": null,

"modelId": null,

"paramType": "",

"duration": "",

"version": "",

"ifMissing": null

},

{

"algoParmType": "o",

"algoRoleName": "output",

"tsKey": null,

"dataTypeId": null,

"dataType": "",

"interval": "",

"deltaT": 0,

"deltaTUnits": null,

"unitsAbbr": null,

"siteId": null,

"siteName": "",

"tableSelector": null,

"modelId": null,

"paramType": "",

"duration": "",

"version": "rev",

"ifMissing": null

}

],

"props": {},

"groupId": 1,

"groupName": "MROI4-ROWI4-HG"

},

"tsid": {

"uniqueString": "ROWI4.Stage.Inst.15Minutes.0.raw",

"key": 8,

"description": "ROWI4 - Stage (created by DECODES)",

"storageUnits": "ft"

},

"since": "2000-12-19T20:15:00.000Z[UTC]",

"until": "2022-12-19T20:15:00.000Z[UTC]",

"traceOutput": true

}

The returned “comptest results” data structure contains all the input data fetched to run the computation, all of the output data produced by the computation, and, if specified, the trace log output. It looks like this:

{

"compParmData": [

{

"algoRoleName": "input",

"parmData": {

"tsid": {

"uniqueString": "ROWI4.Stage.Inst.15Minutes.0.raw",

"key": 8,

"description": "Stage @ ROWI4",

"storageUnits": "ft"

},

"values": [

{

"sampleTime": "2010-12-19T20:15:00.000Z[UTC]",

"value": 4.46,

"flags": 1

},

{

"sampleTime": "2010-12-19T20:30:00.000Z[UTC]",

"value": 4.47,

"flags": 1

},

***Etc. … all the data samples for this parameter***

]

}

},

{

"algoRoleName": "output",

"parmData": {

"tsid": {

"uniqueString": "ROWI4.Stage.Inst.15Minutes.0.rev",

"key": 68,

"description": null,

"storageUnits": "ft"

},

"values": [

{

"sampleTime": "2010-12-19T20:15:00.000Z[UTC]",

"value": 4.46,

"flags": 268435460

},

{

"sampleTime": "2010-12-19T20:30:00.000Z[UTC]",

"value": 4.47,

"flags": 268435460

},

***Etc. … all the data samples for this parameter***

]

}

}

],

"logMessages": [

{

"timeStamp": "2022-08-31T18:36:06.229Z[UTC]",

"priority": "DBG1 ",

"text": "DBG1 08/31/22 18:36:06 ComputationDaoWrapper Setting manual connection for algorithmDAO"

},

{

"timeStamp": "2022-08-31T18:36:06.229Z[UTC]",

"priority": "DBG3 ",

"text": "DBG3 08/31/22 18:36:06 OpenTimeSeriesDAO Query1 'SELECT TS\_ID, SITE\_ID, DATATYPE\_ID, STATISTICS\_CODE, INTERVAL\_ID, DURATION\_ID, TS\_VERSION, ACTIVE\_FLAG, STORAGE\_UNITS, STORAGE\_TABLE, STORAGE\_TYPE, MODIFY\_TIME, DESCRIPTION, UTC\_OFFSET, ALLOW\_DST\_OFFSET\_VARIATION, OFFSET\_ERROR\_ACTION from TS\_SPEC'"

},

***Etc. … all the trace log messages generated during the computation run***

]

}

## TSDB Properties Methods

### GET tsdb\_properties

Example: [http://localhost:8080/odcsapi/tsdb\_properties?token=1241234](http://localhost:8080/OHydroJson/tsdb_properties?token=1241234)

* The token argument is optional. If supplied it will reset the timer on the token.

The tsdb\_properties table in the database will be returned in a structure as follows:

{

"offsetErrorAction": "ROUND",

"storagePresentationGroup": "hydrodcs",

"api.datasource": "Cove-LRGS",

"allowDstOffsetVariation": "true"

}

### POST tsdb\_properties

The POST tsdb\_properties method requires a valid token. It takes one ore more properties in a structure as defined above for the GET method.

* Any property with the same name as one supplied will be overwritten by the passed value.
* If there is no property in the database with a matching name, a new property is added.
* To delete a property from the database, pass an empty string as the value.

## Engineering Unit Methods

### GET unitlist

Example: [http://localhost:8080/odcsapi/unitlist](http://localhost:8080/OHydroJson/unitlist)

* The token argument is optional. If supplied it will reset the timer on the token.

An array of data structures representing all known Engineering Units will be returnd as shown below.

[

{

"abbr": "$",

"family": "univ",

"measures": "Currency",

"name": "Dollars"

},

{

"abbr": "%",

"family": "univ",

"measures": "ratio",

"name": "percent"

},

{

"abbr": "1000 m2",

"family": "Metric",

"measures": "Area",

"name": "Thousands of square meters"

},

{

"abbr": "1000 m3",

"family": "Metric",

"measures": "Volume",

"name": "Thousands of cubic meters"

},

{

"abbr": "C",

"family": "Metric",

"measures": "Temperature",

"name": "Centigrade"

},

…

]

### POST eu

Create a new, or update an existing Engineering Unit

Example URL for POST:

http://localhost:8080/odcsapi/eu?token=6b994be905e1fddf

This method requires a valid session token.

The optional URL argument “fromabbr” can be used to modify the abbreviation of an existing EU.

The POST data should contain a single engineering unit as described above for unitlist. For example, to create a new unit with abbreviation “blob”, the data could be:

{

"abbr": "blob",

"family": "Metric",

"measures": "stuff",

"name": "A Blob of Stuff"

}

### DELETE eu

Delete an existing Engineering Unit record.

Example URL for DELETE:

http://localhost:8080/odcsapi/eu?token=6b994be905e1fddf&abbr=blob

### GET euconvlist

Returns a list of Engineering Unit Conversions defined in the database.

Example: <http://localhost:8080/odcsapi/euconvlist>

* The token argument is optional. If supplied it will reset the timer on the token.

An array of data structures representing all known conversions will be returned as shown below.

[

{

"ucId": 3689,

"fromAbbr": "m^3/s",

"toAbbr": "cms",

"algorithm": "none",

"a": 0,

"b": 0,

"c": 0,

"d": 0,

"e": 0,

"f": 0

},

{

"ucId": 3690,

"fromAbbr": "ft",

"toAbbr": "in",

"algorithm": "linear",

"a": 12,

"b": 0,

"c": 0,

"d": 0,

"e": 0,

"f": 0

},

. . .

]

### POST euconv

Create a new, or update an existing Engineering Unit Conversion

Example URL for POST:

http://localhost:8080/odcsapi/euconv?token=6b994be905e1fddf

This method requires a valid session token.

The POST data should contain a single engineering unit conversion as described above for euconvlist. For example, to create a new conversion that declares “cms” to be a synonym of “blob”, the data could be:

{

"fromAbbr": "cms",

"toAbbr": "blob",

"algorithm": "none",

"a": 0,

"b": 0,

"c": 0,

"d": 0,

"e": 0,

"f": 0

}

Note that the “none” algorithm means that no conversion is required and the coefficients A-F are ignored. It essentially means that the two units are synonyms.

Also note that we left off the “ucId” member since we were creating a new conversion. To update an existing one, include “ucId”.

The returned data structure will be the same as the data passed, except that if this is a new conversion the ucId member will be added.

### DELETE euconv

Delete an existing Engineering Unit conversion record.

Example URL for DELETE:

http://localhost:8080/odcsapi/euconv?token=6b994be905e1fddf&euconvid=1459

This deletes the EU Conversion record with ID 1459.

## Data Type Methods

### GET datatypelist

Examples:

[http://localhost:8080/odcsapi/datatypelist](http://localhost:8080/OHydroJson/datatypelist)

[http://localhost:8080/odcsapi/datatypelist?standard=cwms](http://localhost:8080/OHydroJson/datatypelist?standard=cwms)

The returned data structure is an array of JSON data type objects:

[

{

"code": "72114",

"displayName": "CWMS:72114",

"id": 367,

"standard": "CWMS"

},

{

"code": "Address",

"displayName": "CWMS:Address",

"id": 368,

"standard": "CWMS"

},

{

"code": "Code-Channel",

"displayName": "CWMS:Code-Channel",

"id": 382,

"standard": "CWMS"

},

{

"code": "Code-DCPAddress",

"displayName": "CWMS:Code-DCPAddress",

"id": 372,

"standard": "CWMS"

},

{

"code": "Depth-Snow",

"id": 72,

"standard": "CWMS"

},

…

]

If the optional ‘standard’ argument is supplied, then only data types with the matching standard are returned. Otherwise all data types in the database are returned sorted by (standard, code).

## Retrieving Property Specs

GET propspecs?class=***FullyQualifiedClassName***

Many of the Java classes within OpenDCS maintain a set of properties that can alter the object’s behavior. This method allows the caller to get a list of acceptable properties for a given class.

For example, here are the properties accepted by the class decodes.db.Platform

GET propspecs?class=decodes.db.Platform

[

{

"description": "(default=0) Set to 1, 2, 3 for increasing levels of debug information when this platform is decoded.",

"dynamic": false,

"name": "debugLevel",

"type": "i"

},

{

"description": "Set to have this platform ignored during a specified season.",

"dynamic": false,

"name": "ignoreSeason",

"type": "e:Season"

},

{

"description": "Set to have this platform only processed during a specified season.",

"dynamic": false,

"name": "processSeason",

"type": "e:Season"

},

{

"description": "(default=3) For polled stations, this determines the order in which they will be polled (1 = highest priority = polled first)",

"dynamic": false,

"name": "pollPriority",

"type": "i"

}

]

An array of PropertySpec objects is returned in JSON format:

* “name” is the property name
* “description” is a description displayed to the user as a tooltip or in a dialog to set the property.
* “type” gives the code information about acceptable values for this property.
* “dynamic” is not used and should be ignored.

The “type” variable can be used by the code to offer a pull-down list or to limit the range of acceptable values. “type” will be one of the following:

|  |  |
| --- | --- |
| i | Integer – Value can only be a whole number. |
| n | Number – may be integer or floating point |
| b | Boolean – Value can be “true” or “false” |
| f | Filename |
| d | Directory Name |
| s | Free-form string |
| t | Time Zone Abbreviation – See the Java TimeZone doc for a list of acceptable abbreviations. |
| e:***RefListName*** | A pulldown list of values from a DECODES Reference List should be offered to the user (with a blank space at the top meaning ‘no selection’). See section 4.3 above for info on how to retrieve reference lists. |
| h | A host name or IP address |
| l | (That’s lower case L) indicates a long string that should be displayed in a multi-line text area with an optional scroll bar. But be aware that most of the property tables in the database limit a property value to 240 chars. |

The following class names currently support property specs. More may be added in the future:

* decodes.comp.AreaRatingCompResolver
* decodes.comp.RdbRatingCompResolver
* decodes.comp.StationExcludeCompResolver
* decodes.comp.TabRatingCompResolver
* decodes.consumer.DirectoryConsumer
* decodes.consumer.FileAppendConsumer
* decodes.consumer.FileConsumer
* decodes.consumer.PipeConsumer
* decodes.consumer.StringBufferConsumer
* decodes.consumer.TcpClientConsumer
* decodes.consumer.AlbertaLoaderFormatter
* covesw.azul.consumer.CsvFormatter
* decodes.consumer.EmitAsciiFormatter
* decodes.consumer.EmitOracleFormatter
* decodes.consumer.HeaderFormatter
* decodes.consumer.HtmlFormatter
* decodes.consumer.HumanReadableFormatter
* decodes.consumer.HydroJSONFormatter
* decodes.consumer.KHydstraFormatter
* decodes.consumer.KistersFormatter
* decodes.consumer.NullFormatter
* decodes.consumer.RawFormatter
* decodes.consumer.ShefFormatter
* decodes.consumer.ShefitFormatter
* decodes.consumer.TransmitMonitorFormatter
* decodes.consumer.TsImportFormatter
* decodes.cwms.CwmsConsumer
* decodes.datasource.DirectoryDataSource
* decodes.datasource.FtpDataSource
* decodes.datasource.HotBackupGroup
* decodes.datasource.LrgsDataSource
* decodes.datasource.RoundRobinGroup
* decodes.datasource.ScpDataSource
* decodes.datasource.SocketStreamDataSource
* decodes.datasource.UsgsWebDataSource
* decodes.datasource.WebAbstractDataSource
* decodes.datasource.WebDataSource
* decodes.datasource.WebDirectoryDataSource
* decodes.polling.PollingDataSource
* decodes.db.Platform
* decodes.db.PlatformSensor
* decodes.db.ConfigSensor
* decodes.routing.RoutingScheduler
* decodes.tsdb.CompAppInfo
* decodes.tsdb.algo.HdbRating
* decodes.tsdb.algo.RunningAverageAlgorithm
* decodes.tsdb.algo.AverageAlgorithm
* decodes.tsdb.algo.BigAdder
* decodes.tsdb.algo.Multiplication
* decodes.tsdb.algo.SumOverTimeAlgorithm
* decodes.tsdb.algo.ExpressionParserAlgorithm
* decodes.tsdb.algo.FillForward
* decodes.tsdb.algo.HdbReservoirMassBalance
* decodes.tsdb.algo.AddToPrevious
* decodes.tsdb.algo.Stat
* decodes.tsdb.algo.HdbEvaporation
* decodes.tsdb.algo.EstimatedInflow
* decodes.tsdb.algo.GroupAdder
* decodes.tsdb.algo.ReservoirFull
* decodes.tsdb.algo.ScalerAdder
* decodes.tsdb.algo.HdbACAPSRating
* decodes.tsdb.algo.ChooseOne
* decodes.tsdb.algo.BridgeClearance
* decodes.tsdb.algo.UsgsEquation
* decodes.tsdb.algo.IncrementalPrecip
* decodes.tsdb.algo.FlowResIn
* decodes.tsdb.algo.CopyAlgorithm
* decodes.tsdb.algo.DisAggregate
* decodes.tsdb.algo.WeightedWaterTemperature
* decodes.tsdb.algo.PythonAlgorithm
* decodes.tsdb.algo.Resample
* decodes.tsdb.algo.ShowAlgoProps
* decodes.tsdb.algo.PeriodToDate
* decodes.tsdb.algo.CopyNoOverwrite
* decodes.tsdb.algo.SubSample
* decodes.tsdb.algo.CentralRunningAverageAlgorithm
* decodes.tsdb.algo.TabRating
* decodes.tsdb.algo.RdbRating
* decodes.tsdb.algo.Division
* decodes.tsdb.algo.VirtualGage
* decodes.tsdb.alarm.AlarmScreeningAlgorithm
* decodes.hdb.algo.InflowAdvancedAlg
* decodes.hdb.algo.EquationSolverAlg
* decodes.hdb.algo.EstGLDAInflow
* decodes.hdb.algo.BeginofPeriodAlg
* decodes.hdb.algo.DynamicAggregatesAlg
* decodes.hdb.algo.HdbShiftRating
* decodes.hdb.algo.HdbLookupTimeShiftRating
* decodes.hdb.algo.SideInflowAlg
* decodes.hdb.algo.VolumeToFlowAlg
* decodes.hdb.algo.NVRNUnreg
* decodes.hdb.algo.PowerToEnergyAlg
* decodes.hdb.algo.GLDAUnreg
* decodes.hdb.algo.CallProcAlg
* decodes.hdb.algo.EndofPeriodAlg
* decodes.hdb.algo.GLDAEvap
* decodes.hdb.algo.BMDCUnreg
* decodes.hdb.algo.CRRCUnreg
* decodes.hdb.algo.ParshallFlume
* decodes.hdb.algo.FlowToVolumeAlg
* decodes.hdb.algo.SimpleDisaggAlg
* decodes.hdb.algo.EOPInterpAlg
* decodes.hdb.algo.TimeWeightedAverageAlg
* decodes.hdb.algo.InflowBasicAlg
* decodes.hdb.algo.MPRCUnreg
* decodes.hdb.algo.GlenDeltaBSMBAlg
* decodes.hdb.algo.FLGUUnreg
* decodes.cwms.rating.CwmsRatingSingleIndep
* decodes.cwms.rating.CwmsRatingMultIndep
* decodes.cwms.validation.ScreeningAlgorithm
* decodes.util.DecodesSettings
* lrgs.lrgsmain.LrgsConfig
* opendcs.opentsdb.OpenTsdbSettings

## Retrieving DCP Messages

### Retrieval a Single Message for Testing Decode

Example URLs for GET:

http://localhost:8080/odcsapi/message?tmid=435107EA&token=xxxxxx

http://localhost:8080/odcsapi/message?tmid=300434063963670&token=xxxx

This method requires a valid session token. It may be passed on the URL through the token argument or in the Authentication header.

The tmid argument is required. It must match a transport medium ID in a DCP Message. For GOES messages this is the 8-hex-digit GOES DCP Address. For Iridium messages this is the 15 digit IMEI. For Polled-DCP messages this is the name assigned to the logger.

The return data will be a JSON representation of raw DCP message. The most recent message will be returned:

{

"flags": 71765,

"platformId": "CE31D030",

"sequenceNum": 25693,

"localRecvTime": "2023-05-09T12:34:54.428Z[UTC]",

"carrierStart": "2023-05-09T12:34:55.284Z[UTC]",

"carrierStop": "2023-05-09T12:34:56.752Z[UTC]",

"baud": 300,

"goodPhasePct": 100,

"freqOffset": 0.5,

"signalStrength": 44.8,

"phaseNoise": 1.97,

"xmitTime": null,

"momsn": null,

"mtmsn": null,

"cdrReference": null,

"sessionStatus": null,

"base64": "Q0UzMUQwMzAyMzEyOTEyMzQ1NUc0NSswTk4xNjFFTjIwMDAyN2JCMURBTXRBTXRBTXRBTXM6WUIgMTMuNTkgIA=="

}

The structure contains various header fields:

* flags: A binary encoded integer value. See the following section for bit definitions.
* platformId: This is the unique transport medium identifier, DCP address for GOES, or IMEI for Iridium
* sequenceNum: typically assigned by the server
* localRecvTime: The time this message was acquired by the server
* carrierStart & carrierStop: for GOES messages, this will indicate the time when the platform activated its carrier signal, effectively occupying the channel.
* baud, goodPhasePct, freqOffset, signalStrength, phaseNoise: These are GOES quality measurements.
* xmitTime: typically only set for network DCPs which are polled by the server.
* momsn, mtmsn: Iridium sequence numbers for Mobile-Originated and Mobile-Terminated messages
* cdrReference, sessionStatus: Iridium quality measurements
* base64: This is the BASE64-encoded raw DCP message as it was transmitted. It may include a GOES or Iridium header. BASE64 encoding is used to allow special characters within the message and to preserve the original white-space.

The API code will use a DECODES Data Source to retrieve the message from an LRGS. Set a TSDB property named “api.datasource” to the name of the Data Source you want the API to use. See section 2.15 for information on getting and setting TSDB properties.

### DCP Message Flag Bit Encoding

DCP Message Flag Bits are defined within the OpenDCS class lrgs.common.DcpMsgFlag. Here is the relevant section of the source code with the comments preserved:

public static final int *MSG\_PRESENT* = 0x0001;

/\*\* This msg should be ignored. \*/

public static final int *MSG\_DELETED* = 0x0002;

public static final int *SRC\_MASK* = 0x001C;

public static final int *SRC\_DOMSAT* = 0x0000;

public static final int *SRC\_NETDCP* = 0x0004;

public static final int *SRC\_DRGS* = 0x0008;

public static final int *SRC\_NOAAPORT* = 0x000C;

public static final int *SRC\_LRIT* = 0x0010;

public static final int *SRC\_DDS* = 0x0014;

public static final int *SRC\_IRIDIUM* = 0x0018;

public static final int *SRC\_OTHER* = 0x001C;

public static final int *UNDEFINED\_1* = 0x0020;

/\*\* Message does NOT have \_DOMSAT\_ sequence num \*/

public static final int *MSG\_NO\_SEQNUM* = 0x0040;

public static final int *DUP\_MSG* = 0x0080;

/\*\* Means Carrier times were estimated on DRGS link. \*/

public static final int *CARRIER\_TIME\_EST* = 0x0100;

/\*\* Message flagged as binary \*/

public static final int *BINARY\_MSG* = 0x0200;

/\*\* Mask for baud-rate bits \*/

public static final int *BAUD\_MASK* = 0x0C00;

/\*\* Message baud rate is unknown \*/

public static final int *BAUD\_UNKNOWN* = 0x0000;

/\*\* Message baud rate is 100 \*/

public static final int *BAUD\_100* = 0x0400;

/\*\* Message baud rate is 300 \*/

public static final int *BAUD\_300* = 0x0800;

/\*\* Message baud rate is 1200 \*/

public static final int *BAUD\_1200* = 0x0C00;

/\*\* Unused bits \*/

public static final int *HAS\_CARRIER\_TIMES* = 0x1000;

// /\*\* Coopted for extended msg types (see below). \*/

// public static final int MSG\_TYPE\_EXT = 0x2000;

/\*\* DRGS Address was Corrected from original. \*/

public static final int *ADDR\_CORRECTED* = 0x4000;

/\*\* Bit indicating GOES (default for legacy) or Iridium Msg. \*/

public static final int *MSG\_TYPE\_MASK* = 0x0003A000;

// non-specific any GOES:

public static final int *MSG\_TYPE\_GOES* = 0x00000000;

public static final int *MSG\_TYPE\_IRIDIUM* = 0x00008000;

public static final int *MSG\_TYPE\_NETDCP* = 0x00002000;

public static final int *MSG\_TYPE\_OTHER* = 0x0000A000;

// For flag rev 3, new types are added above 16 bits.

public static final int *MSG\_TYPE\_GOES\_ST* = 0x00010000;

public static final int *MSG\_TYPE\_GOES\_RD* = 0x00020000;

// Goes international

public static final int *MSG\_TYPE\_GOES\_INT* = 0x00030000;

// Indicates a binary message that has detected errors

public static final int *HAS\_BINARY\_ERRORS* = 0x00040000;

// Platform Type is either CS1 or CS2

public static final int *PLATFORM\_TYPE\_MASK* = 0x00080000;

public static final int *PLATFORM\_TYPE\_CS1* = 0x00000000;

public static final int *PLATFORM\_TYPE\_CS2* = 0x00080000;

// No EOT was detected on this message

public static final int *NO\_EOT* = 0x00100000;

public static final int *ARM\_UNCORRECTABLE\_ADDR* = 0x00200000;

public static final int *ARM\_ADDR\_NOT\_IN\_PDT* = 0x00400000;

public static final int *ARM\_PDT\_INCOMPLETE* = 0x00800000;

// Message overlaps its PDT window

public static final int *ARM\_TIMING\_ERROR* = 0x01000000;

// Message completely outside its timing window

public static final int *ARM\_UNEXPECTED\_MSG* = 0x02000000;

// GOES message received on wrong channel

public static final int *ARM\_WRONG\_CHANNEL* = 0x04000000;

### Multiple Message Retrieval

The API provides methods for retrieving a set of messages based on user-defined search criteria. The dynamics of the retrieval work like this:

1. Client does a “POST searchcrit” operation. This defines time ranges, message types, platform IDs, network lists, etc., that collectively determine what messages are to be retrieved.
   1. The server saves the search criteria within the client’s session data.
2. Client does a “GET messages” operation. This returns a block of messages along with a flag indicating whether more messages are to follow.
3. Client repeats step 2 until an empty message block is returned or until the “moreToFollow” flag is false.

Both methods require a valid session token. It may be passed on the URL through the token argument or in the Authentication header.

**POST searchcrit**

Example URL for POST searchcrit:

http://localhost:8080/odcsapi/searchcrit?token=6b994be905e1fddf

The POST data should contain a search criteria structure as shown below:

{

"since": "now - 1 hour",

"until": "now",

"settlingTimeDelay": false,

"platformIds": [],

"platformNames": ["BMD-NRE","BFD","OKVI4","BMD-tailwater"],

"netlistNames": [],

"goesChannels": [],

"goesSelfTimed": false,

"goesRandom": false,

"networkDCP": false,

"iridium": false,

"qualityNotifications": false,

"goesSpacecraftCheck": false,

"goesSpacecraftSelection": "East",

"parityCheck": false,

"paritySelection": "Good"

}

**GET searchcrit**

A method for retrieving the previously-posted searchcrit is provided for testing. Note that the session will discard a search criteria when either a new one is posted, or when a message retrieval is complete.

**GET messages**

The “GET messages” operation will return a set of messages as shown below:

{

"messages": [

{

"flags": 71765,

"platformId": "CE31D030",

"sequenceNum": 68909,

"localRecvTime": "2023-05-09T14:34:57.174Z[UTC]",

"carrierStart": "2023-05-09T14:34:55.284Z[UTC]",

"carrierStop": "2023-05-09T14:34:56.756Z[UTC]",

"baud": 300,

"goodPhasePct": 100,

"freqOffset": 2.7,

"signalStrength": 44.8,

"phaseNoise": 1.93,

"xmitTime": null,

"momsn": null,

"mtmsn": null,

"cdrReference": null,

"sessionStatus": null,

"base64": "Q0UzMUQwMzAyMzEyOTE0MzQ1NUc0NSswTk4xNjFFTjEwMDAyN2JCMURBTXNBTXNBTXNBTXM6WUIgMTMuNTkgIA=="

},

{

"flags": 137301,

"platformId": "CE31D030",

"sequenceNum": 59093,

"localRecvTime": "2023-05-09T14:08:22.779Z[UTC]",

"carrierStart": "2023-05-09T14:08:21.111Z[UTC]",

"carrierStop": "2023-05-09T14:08:23.966Z[UTC]",

"baud": 300,

"goodPhasePct": 100,

"freqOffset": -0.9,

"signalStrength": 46,

"phaseNoise": 1.88,

"xmitTime": null,

"momsn": null,

"mtmsn": null,

"cdrReference": null,

"sessionStatus": null,

"base64": "Q0UzMUQwMzAyMzEyOTE0MDgyMUc0Ni0wTk4xMjlFTjIwMDA3OWJaR1VgT2BAQEBTZkBSeEBTeUBTeEBTb0BTX0BUQkBTZkBTfEBURGBAQEBBR0BBR0BBR0BBR0BBR0BBR0BBR0BBR0BBR0BBR2dlYUZUTCA="

},

{

"flags": 596053,

"platformId": "CE3234CC",

"sequenceNum": 69792,

"localRecvTime": "2023-05-09T14:37:04.918Z[UTC]",

"carrierStart": "2023-05-09T14:37:05.284Z[UTC]",

"carrierStop": "2023-05-09T14:37:07.530Z[UTC]",

"baud": 300,

"goodPhasePct": 98.7,

"freqOffset": -0.4,

"signalStrength": 34.6,

"phaseNoise": 3.3,

"xmitTime": null,

"momsn": null,

"mtmsn": null,

"cdrReference": null,

"sessionStatus": null,

"base64": "Q0UzMjM0Q0MyMzEyOTE0MzcwNUczNS0wTk4xNjFFTjIwMDA1NGJCMUdASUBAcHlARk5TQFdASHZAcHlARk5TQFlASGpAcHlARk5TQFtASGlAcHlARk5TQF1PIA=="

},

{

"flags": 596053,

"platformId": "CE344292",

"sequenceNum": 58214,

"localRecvTime": "2023-05-09T14:04:56.862Z[UTC]",

"carrierStart": "2023-05-09T14:04:54.047Z[UTC]",

"carrierStop": "2023-05-09T14:04:56.005Z[UTC]",

"baud": 300,

"goodPhasePct": 99.5,

"freqOffset": 8.4,

"signalStrength": 37.9,

"phaseNoise": 2.25,

"xmitTime": null,

"momsn": null,

"mtmsn": null,

"cdrReference": null,

"sessionStatus": null,

"base64": "Q0UzNDQyOTIyMzEyOTE0MDQ1NEczOCswTk4wNDlFVUIwMDA0NWBCMURATkRATkVATkVATkRATkdATkVATkVATkZBU3FBU3FBU3FBU3FAVHNHIA=="

},

{

"flags": 71765,

"platformId": "CE3941F4",

"sequenceNum": 69414,

"localRecvTime": "2023-05-09T14:36:18.531Z[UTC]",

"carrierStart": "2023-05-09T14:36:20.286Z[UTC]",

"carrierStop": "2023-05-09T14:36:21.833Z[UTC]",

"baud": 300,

"goodPhasePct": 100,

"freqOffset": 1.5,

"signalStrength": 44.9,

"phaseNoise": 1.63,

"xmitTime": null,

"momsn": null,

"mtmsn": null,

"cdrReference": null,

"sessionStatus": null,

"base64": "Q0UzOTQxRjQyMzEyOTE0MzYyMEc0NSswTk4xNjFFTjEwMDAzMGJCMUZASG9ASG9ASG9ASG9ASXZASXFASXFASXZOIA=="

}

],

"moreToFollow": true

}

### Retrieve LRGS Status

GET URL Example

<http://localhost:8080/odcsapi/lrgsstatus?source=cdabackup>

If the “source” argument is missing, it will use the data source named by TSDB property “api.datasource”, as described above for message retrieval.

An example of the returned data structure is shown below:

{

"hostname": "covesw4",

"systemStatus": "Running",

"isUsable": true,

"systemTime": "2023-05-15T18:38:22.000Z[UTC]",

"maxClients": 150,

"currentNumClients": 4,

"lrgsVersion": "9.5.OpenDCS-6.8 RC12 (Jul 27, 2021)",

"arcDirOldest": 0,

"arcDirNext": 846181,

"arcDirWrap": 0,

"arcDirSize": 7385773,

"arcOldestOffset": 0,

"arcOldestMsgTime": "2023-05-09T00:00:02.000Z[UTC]",

"arcLastSeqNum": 60546,

"procStatus": [

{

"slot": 0,

"pid": 893924,

"name": "-",

"type": "DDS-CLI",

"user": "covetest",

"status": "running",

"lastSeqNum": 0,

"lastPollTime": "1970-01-20T11:49:35.880Z[UTC]",

"lastMsgTime": "1970-01-01T00:00:00.000Z[UTC]",

"staleCount": 0

},

{

"slot": 1,

"pid": 893930,

"name": "-",

"type": "DDS-CLI",

"user": "dcpmon",

"status": "running",

"lastSeqNum": 199768,

"lastPollTime": "1970-01-20T11:49:35.902Z[UTC]",

"lastMsgTime": "1970-01-20T11:49:35.874Z[UTC]",

"staleCount": 0

},

{

"slot": 2,

"pid": 893928,

"name": "-",

"type": "DDS-CLI",

"user": "dcpmon",

"status": "running",

"lastSeqNum": 81,

"lastPollTime": "1970-01-20T11:49:35.902Z[UTC]",

"lastMsgTime": "1970-01-20T11:49:35.765Z[UTC]",

"staleCount": 0

},

{

"slot": 3,

"pid": 893942,

"name": "-",

"type": "DDS-CLI",

"user": "covetest",

"status": "running",

"lastSeqNum": 0,

"lastPollTime": "1970-01-20T11:49:35.902Z[UTC]",

"lastMsgTime": "1970-01-01T00:00:00.000Z[UTC]",

"staleCount": 0

}

],

"downlinkStatus": [

{

"slot": 1,

"name": "DDS-Recv:Main",

"type": "DL\_DDS",

"status": "Active",

"lastMsgRecvTime": "2023-05-15T18:38:20.000Z[UTC]",

"group": "",

"lastSeqNum": 0,

"hourlyQuality": [

{

"hour": 0,

"numGood": 47704,

"numBad": 207,

"numRecovered": 0

},

{

"hour": 1,

"numGood": 47543,

"numBad": 205,

"numRecovered": 0

},

{

"hour": 2,

"numGood": 47304,

"numBad": 221,

"numRecovered": 0

},

{

"hour": 3,

"numGood": 47560,

"numBad": 186,

"numRecovered": 0

},

{

"hour": 4,

"numGood": 47285,

"numBad": 225,

"numRecovered": 0

},

{

"hour": 5,

"numGood": 47318,

"numBad": 191,

"numRecovered": 0

},

{

"hour": 6,

"numGood": 47494,

"numBad": 236,

"numRecovered": 0

},

{

"hour": 7,

"numGood": 47372,

"numBad": 503,

"numRecovered": 0

},

{

"hour": 8,

"numGood": 47442,

"numBad": 439,

"numRecovered": 0

},

{

"hour": 9,

"numGood": 47368,

"numBad": 222,

"numRecovered": 0

},

{

"hour": 10,

"numGood": 47265,

"numBad": 277,

"numRecovered": 0

},

{

"hour": 11,

"numGood": 47408,

"numBad": 294,

"numRecovered": 0

},

{

"hour": 12,

"numGood": 46008,

"numBad": 198,

"numRecovered": 0

},

{

"hour": 13,

"numGood": 43966,

"numBad": 232,

"numRecovered": 0

},

{

"hour": 14,

"numGood": 43987,

"numBad": 174,

"numRecovered": 0

},

{

"hour": 15,

"numGood": 44158,

"numBad": 239,

"numRecovered": 0

},

{

"hour": 16,

"numGood": 44062,

"numBad": 177,

"numRecovered": 0

},

{

"hour": 17,

"numGood": 44082,

"numBad": 190,

"numRecovered": 0

},

{

"hour": 18,

"numGood": 29681,

"numBad": 197,

"numRecovered": 0

},

{

"hour": 19,

"numGood": 47607,

"numBad": 206,

"numRecovered": 0

},

{

"hour": 20,

"numGood": 47562,

"numBad": 185,

"numRecovered": 0

},

{

"hour": 21,

"numGood": 47705,

"numBad": 195,

"numRecovered": 0

},

{

"hour": 22,

"numGood": 47788,

"numBad": 197,

"numRecovered": 0

},

{

"hour": 23,

"numGood": 47860,

"numBad": 212,

"numRecovered": 0

}

]

},

{

"slot": 5,

"name": "DDS:NLRGS1",

"type": "DDSCON",

"status": "Real-Time",

"lastMsgRecvTime": "2023-05-15T18:38:20.000Z[UTC]",

"group": "Primary",

"lastSeqNum": 0,

"hourlyQuality": [

{

"hour": 0,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 1,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 2,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 3,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 4,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 5,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 6,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 7,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 8,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 9,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 10,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 11,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 12,

"numGood": 32852,

"numBad": 139,

"numRecovered": 0

},

{

"hour": 13,

"numGood": 43966,

"numBad": 232,

"numRecovered": 0

},

{

"hour": 14,

"numGood": 43987,

"numBad": 174,

"numRecovered": 0

},

{

"hour": 15,

"numGood": 44158,

"numBad": 239,

"numRecovered": 0

},

{

"hour": 16,

"numGood": 44062,

"numBad": 177,

"numRecovered": 0

},

{

"hour": 17,

"numGood": 44082,

"numBad": 190,

"numRecovered": 0

},

{

"hour": 18,

"numGood": 29681,

"numBad": 197,

"numRecovered": 0

},

{

"hour": 19,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 20,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 21,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 22,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 23,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

}

]

},

{

"slot": 6,

"name": "DDS:EDDN1",

"type": "DDSCON",

"status": "Ready",

"lastMsgRecvTime": "1970-01-01T00:00:00.000Z[UTC]",

"group": "primary",

"lastSeqNum": 0,

"hourlyQuality": [

{

"hour": 0,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 1,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 2,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 3,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 4,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 5,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 6,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 7,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 8,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 9,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 10,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 11,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 12,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 13,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 14,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 15,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 16,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 17,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 18,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 19,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 20,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 21,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 22,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

},

{

"hour": 23,

"numGood": 0,

"numBad": 0,

"numRecovered": 0

}

]

},

"hourlyArchiveQuality": [

{

"hour": 0,

"numGood": 44123,

"numBad": 170,

"numRecovered": 0

},

{

"hour": 1,

"numGood": 44016,

"numBad": 141,

"numRecovered": 0

},

{

"hour": 2,

"numGood": 43846,

"numBad": 166,

"numRecovered": 0

},

{

"hour": 3,

"numGood": 44029,

"numBad": 148,

"numRecovered": 0

},

{

"hour": 4,

"numGood": 43825,

"numBad": 155,

"numRecovered": 0

},

{

"hour": 5,

"numGood": 43815,

"numBad": 143,

"numRecovered": 0

},

{

"hour": 6,

"numGood": 43984,

"numBad": 167,

"numRecovered": 0

},

{

"hour": 7,

"numGood": 43885,

"numBad": 230,

"numRecovered": 0

},

{

"hour": 8,

"numGood": 43897,

"numBad": 242,

"numRecovered": 0

},

{

"hour": 9,

"numGood": 43886,

"numBad": 155,

"numRecovered": 0

},

{

"hour": 10,

"numGood": 43797,

"numBad": 172,

"numRecovered": 0

},

{

"hour": 11,

"numGood": 43865,

"numBad": 182,

"numRecovered": 0

},

{

"hour": 12,

"numGood": 43942,

"numBad": 163,

"numRecovered": 0

},

{

"hour": 13,

"numGood": 43969,

"numBad": 161,

"numRecovered": 0

},

{

"hour": 14,

"numGood": 43988,

"numBad": 140,

"numRecovered": 0

},

{

"hour": 15,

"numGood": 44151,

"numBad": 179,

"numRecovered": 0

},

{

"hour": 16,

"numGood": 44060,

"numBad": 142,

"numRecovered": 0

},

{

"hour": 17,

"numGood": 44083,

"numBad": 156,

"numRecovered": 0

},

{

"hour": 18,

"numGood": 29689,

"numBad": 137,

"numRecovered": 0

},

{

"hour": 19,

"numGood": 44097,

"numBad": 152,

"numRecovered": 0

},

{

"hour": 20,

"numGood": 44066,

"numBad": 135,

"numRecovered": 0

},

{

"hour": 21,

"numGood": 44185,

"numBad": 143,

"numRecovered": 0

},

{

"hour": 22,

"numGood": 44110,

"numBad": 161,

"numRecovered": 0

},

{

"hour": 23,

"numGood": 44157,

"numBad": 153,

"numRecovered": 0

}

]

}

## Test Decoding

Example URL for HTTP POST method:

[http://localhost:8080/odcsapi/decode?token=73168ed6c9c49870&script=ST](http://localhost:8080/OHydroJson/decode?token=73168ed6c9c49870&script=ST)

As shown, the arguments must include:

* A valid security token
* The script name to use in decoding the message (if omitted, the first script in the config will be used).

The data sent in the POST request must be:

{

"config": ***{ config as returned by GET config described above}***

"rawmsg": ***{ Raw Message as returned by GET message described above }***

}

Here is an example request body:

{

"config": {

"configId": 2,

"configSensors": [

{

"absoluteMax": 30,

"absoluteMin": 0.009999999776482582,

"dataTypes": {

"SHEF-PE": "HG",

"CWMS": "Stage"

},

"properties": {},

"recordingInterval": 900,

"recordingMode": "F",

"sensorName": "Stage",

"sensorNumber": 1,

"timeOfFirstSample": 0

},

{

"absoluteMax": 251,

"absoluteMin": -0.10000000149011612,

"dataTypes": {

"SHEF-PE": "PC",

"CWMS": "Precip"

},

"properties": {},

"recordingInterval": 900,

"recordingMode": "F",

"sensorName": "Precip",

"sensorNumber": 2,

"timeOfFirstSample": 0

},

{

"dataTypes": {

"SHEF-PE": "VB",

"CWMS": "Volt-Battery"

},

"properties": {},

"recordingInterval": 3600,

"recordingMode": "F",

"sensorName": "Battery",

"sensorNumber": 3,

"timeOfFirstSample": 0

}

],

"description": "",

"name": "ROWI4",

"numPlatforms": 0,

"scripts": [

{

"dataOrder": "U",

"formatStatements": [

{

"format": "4x,8(f(s,b,3,1),f(s,b,3,2)),24x,f(s,b,1,3)",

"label": "st",

"sequenceNum": 0

}

],

"name": "ST",

"scriptSensors": [

{

"sensorNumber": 1,

"unitConverter": {

"a": 0.01,

"algorithm": "usgs-standard",

"b": 0,

"c": 1,

"d": 0,

"e": 0,

"f": 0,

"fromAbbr": "raw",

"toAbbr": "ft"

}

},

{

"sensorNumber": 2,

"unitConverter": {

"a": 0.01,

"algorithm": "usgs-standard",

"b": 0,

"c": 1,

"d": 0,

"e": 0,

"f": 0,

"fromAbbr": "raw",

"toAbbr": "in"

}

},

{

"sensorNumber": 3,

"unitConverter": {

"a": 0.3125,

"algorithm": "usgs-standard",

"b": 0,

"c": 1,

"d": 0.311,

"e": 0,

"f": 0,

"fromAbbr": "raw",

"toAbbr": "v"

}

}

]

}

]

},

"rawmsg": {

"base64": "Q0UyREQ2MzIyMjE5MjE0MDYyM0c0Ni0xTk4wNDlFTjIwMDA3OCJCU1RARnZAVX1ARm5ARnhAVX1ARm5ARnhAVX1ARm5ARnhAVX1ARm5ARndAVX1ARm5ARnZAVX1ARm5ARnhAVX1ARm5ARnhAVX1ARm9nIA=="

}

}

The raw data will be decoded according to the instructions in the passed config. The returned data will include log messages generated to trace the script execution, and the decoded data from the message. Here is the return data for the above request. Note that for each decoded value, the position within the raw message is given.

{

"logMessages": [

{

"priority": "INFO ",

"text": "Header type 'GOES' length=37",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:ModulationIndex=N",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:Length=78",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:Quality=N",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:Channel=49",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:FrequencyOffset=-1",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:Spacecraft=E",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:FailureCode=G",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:Time=07/11/2022 14:06:23 UTC",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:GPS=1",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:UplinkCarrier=N2",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:DcpAddress=CE2DD632",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "INFO ",

"text": " PM:SignalStrength=46",

"timeStamp": "2022-07-14T17:14:27.332Z[UTC]"

},

{

"priority": "DBG1 ",

"text": "After decoding there are 17 decoded samples.",

"timeStamp": "2022-07-14T17:14:27.349Z[UTC]"

}

],

"messageTime": "2022-07-11T14:06:23Z[UTC]",

"timeSeries": [

{

"sensorName": "Stage",

"sensorNum": 1,

"units": "ft",

"values": [

{

"rawDataPosition": {

"end": 7,

"start": 4

},

"time": "2022-07-11T14:00:00Z[UTC]",

"value": "4.38"

},

{

"rawDataPosition": {

"end": 13,

"start": 10

},

"time": "2022-07-11T13:45:00Z[UTC]",

"value": "4.3"

},

{

"rawDataPosition": {

"end": 19,

"start": 16

},

"time": "2022-07-11T13:30:00Z[UTC]",

"value": "14.05"

},

{

"rawDataPosition": {

"end": 25,

"start": 22

},

"time": "2022-07-11T13:15:00Z[UTC]",

"value": "4.4"

},

{

"rawDataPosition": {

"end": 31,

"start": 28

},

"time": "2022-07-11T13:00:00Z[UTC]",

"value": "4.3"

},

{

"rawDataPosition": {

"end": 37,

"start": 34

},

"time": "2022-07-11T12:45:00Z[UTC]",

"value": "14.05"

},

{

"rawDataPosition": {

"end": 43,

"start": 40

},

"time": "2022-07-11T12:30:00Z[UTC]",

"value": "4.39"

},

{

"rawDataPosition": {

"end": 49,

"start": 46

},

"time": "2022-07-11T12:15:00Z[UTC]",

"value": "4.3"

}

]

},

{

"sensorName": "Precip",

"sensorNum": 2,

"units": "in",

"values": [

{

"rawDataPosition": {

"end": 10,

"start": 7

},

"time": "2022-07-11T14:00:00Z[UTC]",

"value": "14.05"

},

{

"rawDataPosition": {

"end": 16,

"start": 13

},

"time": "2022-07-11T13:45:00Z[UTC]",

"value": "4.4"

},

{

"rawDataPosition": {

"end": 22,

"start": 19

},

"time": "2022-07-11T13:30:00Z[UTC]",

"value": "4.3"

},

{

"rawDataPosition": {

"end": 28,

"start": 25

},

"time": "2022-07-11T13:15:00Z[UTC]",

"value": "14.05"

},

{

"rawDataPosition": {

"end": 34,

"start": 31

},

"time": "2022-07-11T13:00:00Z[UTC]",

"value": "4.4"

},

{

"rawDataPosition": {

"end": 40,

"start": 37

},

"time": "2022-07-11T12:45:00Z[UTC]",

"value": "4.3"

},

{

"rawDataPosition": {

"end": 46,

"start": 43

},

"time": "2022-07-11T12:30:00Z[UTC]",

"value": "14.05"

},

{

"rawDataPosition": {

"end": 52,

"start": 49

},

"time": "2022-07-11T12:15:00Z[UTC]",

"value": "4.38"

}

]

},

{

"sensorName": "Battery",

"sensorNum": 3,

"units": "V",

"values": [

{

"rawDataPosition": {

"end": 77,

"start": 76

},

"time": "2022-07-11T14:00:00Z[UTC]",

"value": "12.4985"

}

]

}

]

}

# Time Series Methods

## GET Time Series References

The tsrefs method returns a list of time series defined in the database. You have the option to filter out inactive time series by passing “active=true” argument.

Examples:

http://localhost:8080/odcsapi/tsrefs

http://localhost:8080/odcsapi/tsrefs?active=true

This returns an array of Time Series Identifiers. The numeric Key of a time series identifier may be used in subsequent calls to get the complete specification for the time series (GET tsspec) or to retrieve time series data (GET tsdata). The format of the returned data is as follows:

[

{

"uniqueString": "OKVI4.Stage.Inst.15Minutes.0.raw",

"key": 1,

"description": null,

"storageUnits": "ft",

"active": true

},

{

"uniqueString": "OKVI4.Stage.Ave.1Day.1Day.CO",

"key": 2,

"description": null,

"storageUnits": "ft",

"active": true

},

{

"uniqueString": "OKVI4.Stage.Ave.1Day.1Day.CC",

"key": 4,

"description": null,

"storageUnits": "ft",

"active": true

},

. . .

]

## GET Time Series Specification

The tsspec method returns a complete specification for a time series identified by the “key” parameter.

Example:

http://localhost:8080/odcsapi/tsspec?key=532

The format of the returned data is as follows:

{

"tsid": {

"uniqueString": "MROI4.Stage.Inst.30Minutes.0.raw",

"key": 11,

"description": "MROI4 - Stage (created by DECODES)",

"storageUnits": "ft",

"active": true

},

"location": "MROI4",

"param": "Stage",

"statCode": "Inst",

"interval": "30Minutes",

"duration": "0",

"version": "raw",

"siteId": 6,

"datatypeId": 48,

"intervalId": 12,

"durationId": 36,

"lastModified": "2020-05-11T20:50:55.795Z[UTC]",

"active": true,

"allowDSTVariation": false,

"utcOffset": 0,

"offsetErrorAction": "IGNORE",

"storageType": "N",

"storageTable": 2,

"numValues": 144,

"min": {

"sampleTime": "2010-12-22T20:00:00.000Z[UTC]",

"value": 8.18,

"flags": 0

},

"max": {

"sampleTime": "2010-12-20T14:00:00.000Z[UTC]",

"value": 14.07,

"flags": 0

},

"oldest": {

"sampleTime": "2010-12-19T20:30:00.000Z[UTC]",

"value": 8.43,

"flags": 0

},

"newest": {

"sampleTime": "2010-12-22T20:00:00.000Z[UTC]",

"value": 8.18,

"flags": 0

}

}

## GET Time Series Data

The tsdata method returns data for a time series over a specified time range. The method takes 3 arguments:

* key (required) – the numeric key identifying the time series. It is contained within a Time Series Identifier described above.
* start – Optionally specifies the start of the time range for retrieval. If omitted, the oldest data in the database is returned. See below for time format.
* end – Optionally specifies the end of the time range for retrieval. If omitted, the newest data in the database is returned. See below for time format.

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

Examples:

<http://localhost:8080/odcsapi/tsdata?key=12>

The format of the returned data is as follows:

{

"tsid": {

"uniqueString": "MROI4.Precip.Inst.30Minutes.0.raw",

"key": 12,

"description": "MROI4 - Precip (created by DECODES)",

"storageUnits": "in",

"active": true

},

"values": [

{

"sampleTime": "2010-12-19T20:30:00.000Z[UTC]",

"value": 11,

"flags": 0

},

{

"sampleTime": "2010-12-19T21:00:00.000Z[UTC]",

"value": 11,

"flags": 0

},

{

"sampleTime": "2010-12-19T21:30:00.000Z[UTC]",

"value": 11,

"flags": 0

},

*. . .*

]

}

## Interval Methods

Time Intervals are stored in the database for OpenTSDB. They are hardcoded for CWMS and HDB.

### GET intervals

Returns a list of time intervals defined in the database.

Example: http://localhost:8080/odcsapi/intervals

* The token argument is optional. If supplied it will reset the timer on the token.

An array of data structures representing all known time intervals will be returned as shown below.

[

{

"intervalId": 1,

"name": "irregular",

"calConstant": "minute",

"calMultilier": 0

},

{

"intervalId": 2,

"name": "2Minutes",

"calConstant": "minute",

"calMultilier": 2

},

. . .

]

For each interval the system stores a numeric ID, a name, a Java Calendar Constant (one of second, minute, hour, day, week, month, year), and a multiplier for the constant.

### POST interval

Create a new, or update an existing Time Interval

Example URL for POST:

http://localhost:8080/odcsapi/interval?token=6b994be905e1fddf

This method requires a valid session token.

The POST data should contain a single time interval record as described above for the “intervals” list. As with other POST methods, to create a new record, omit the numeric ID. To update an existing record, include the “intervalId”.

For example, to create a interval “fortnight”, the data could be:

{

"name": "fortnight",

"calConstant": "day",

"calMultilier": 14

},

The returned data structure will be the same as the data passed, except that if this is a new interval the intervalId member will be added.

### DELETE interval

Delete an existing Time Interval record.

Example URL for DELETE:

http://localhost:8080/odcsapi/interval?token=6b994be905e1fddf&intervalid=1459

This deletes the Time Interval with ID 1459.

Use care with this method. The system needs to know about all of the “interval” and “duration” specifiers used for time series IDs.

## Time Series Group Methods

Time Series Groups are used to define a set of time series identifiers. Groups can contain:

* Explicit list of time series identifiers
* A list of attributes to flexibly define a set of time series identifiers, E.g. All time series at a particular with interval “30minutes”.
* A list of sub-groups that can be included, excluded, or intersected with the group being defined.

### GET tsgrouprefs

Provide a list of all groups defined in the database.

Example URL:

http://localhost:8080/odcsapi-0-7/tsgrouprefs

A security token may be supplied in the header or in the URL, but it is not required.

The returned list has the following structure:

[

{

"groupId": 8,

"groupName": "topgroup",

"groupType": "basin",

"description": ""

},

{

"groupId": 7,

"groupName": "subgroup-x",

"groupType": "data type",

"description": "testing for OPENDCS-15 issue"

},

{

"groupId": 2,

"groupName": "regtest\_017",

"groupType": "data-type",

"description": "Group for regression test 017"

},

{

"groupId": 3,

"groupName": "stageRate1Var",

"groupType": "basin",

"description": "Collection of TS IDs with stage to flow ratings"

}

]

### GET tsgroup

Provide a complete definition of a single group.

Example URL:

http://localhost:8080/odcsapi-0-7/tsgroup?groupid=9

A security token may be supplied in the header or in the URL, but it is not required.

The returned list has the following structure:

{

"groupId": 9,

"groupName": "junk",

"groupType": "basin",

"description": "",

"tsIds": [

{

"uniqueString": "OKVI4.Stage.Inst.15Minutes.0.raw",

"key": 1,

"description": null,

"storageUnits": "ft",

"active": true

},

{

"uniqueString": "OKVI4.Stage.Ave.1Day.1Day.CO",

"key": 2,

"description": null,

"storageUnits": "ft",

"active": true

}

],

"includeGroups": [

{

"groupId": 1,

"groupName": "MROI4-ROWI4-HG",

"groupType": "basin",

"description": "This is a group for the MROI4-ROWI4-HG Regression Test"

}

],

"excludeGroups": [

{

"groupId": 2,

"groupName": "regtest\_017",

"groupType": "data-type",

"description": "Group for regression test 017"

}

],

"intersectGroups": [

{

"groupId": 7,

"groupName": "subgroup-x",

"groupType": "data type",

"description": "testing for OPENDCS-15 issue"

}

],

"groupAttrs": [

"BaseLocation=TESTSITE2",

"BaseParam=ELEV",

"BaseVersion=DCP",

"Duration=0",

"Interval=1Hour",

"ParamType=Inst",

"SubLocation=Spillway2-Gate1",

"SubParam=PZ1B",

"SubVersion=Raw",

"Version=DCP-Raw"

],

"groupSites": [

{

"siteId": 2,

"sitenames": {

"CWMS": "ROWI4",

"USGS": "05449500"

},

"publicName": "IOWA RIVER NEAR ROWAN",

"description": "IOWA RIVER NEAR ROWAN 4NW"

}

],

"groupDataTypes": [

{

"id": 224,

"standard": "CWMS",

"code": "ELEV-PZ2A",

"displayName": "CWMS:ELEV-PZ2A"

}

]

}

Notes:

* “tsIds” is a list of explicit time series identifiers that are considered part of the group.
* “includedGroups” is a list of subgroups to be included in this group.
* “excludedGroups” is a list of subgroups. The TSIDs in the subgroup will be excluded from this group.
* “intersectedGroups” is a list of subgroups to be intersected with this group. Only TSIDs in both groups are considered part of this group.
* “groupSites” is a list of Site records. TSIDs in these Sites are considered members of this group.
* “groupDataTypes” is a list of fully-specified data types (a.k.a. “Param” in CWMS and OpenTSDB databases). TSIDs with a matching data type will be included in the group.
* “groupAttrs” is a list of attributes that are used to define the group. These are presented in “name=value” pairs where the name is one of the following:
  + BaseLocation – only the first part of Site (Location) before first hyphen
  + SubLocation – only trailing part of Site after first hyphen.
  + BaseParam – only first part of data type (Param) before first hyphen
  + SubParam – only trailing part of data type (Param) after first hyphen
  + ParamType
  + Interval
  + Duration
  + Version
  + BaseVersion
  + SubVersion

### POST tsgroup

Create a new, or update an existing time series group

Example URL for POST:

http://localhost:8080/odcsapi/tsgroup?token=6b994be905e1fddf

This method requires a valid session token.

The POST data is as described above for GET tsgroup

### DELETE tsgroup

Delete an existing Time Interval record.

Example URL for DELETE:

http://localhost:8080/odcsapi/delete?token=6b994be905e1fddf&groupid=9

This deletes the Time series group with ID 9.

This method requires a valid session token.

### POST tsgroupeval

This method evaluates the group passed in the POST data and returns an array of time series identifiers.

This method requires a valid session token.

The POST data is a group as defined above for the GET method. It does not need to be a group already saved in the database. This method does not saved the passed group definition.

The return data is an array of time series identifiers as shown below:

[

{

"uniqueString": "OKVI4.Stage.Inst.15Minutes.0.raw",

"key": 1,

"description": null,

"storageUnits": "ft",

"active": true

},

{

"uniqueString": "OKVI4.Precip.Inst.15Minutes.0.raw",

"key": 6,

"description": "OKVI4 - Precip (created by DECODES)",

"storageUnits": "in",

"active": true

},

{

"uniqueString": "OKVI4.Volt-Battery.Inst.1Hour.0.raw",

"key": 7,

"description": "OKVI4 - Volt-Battery (created by DECODES)",

"storageUnits": "V",

"active": true

}

]

# Network List Methods

## 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"}

}

}

# OpenDCS Process Monitor and Control (M&C)

## App M&C

Section 2.12 describes API methods for retrieving and manipulating “Loading Application” records. The concept of a “Loading App” has been generalized to include any application that is known by the OpenDCS software.

Applications each have a set of properties. The following properties are relevant to M&C:

* startCmd – A string containins a command used to start the application on this server. Most of the OpenDCS apps use lock records to ensure that only a single instance can run at a time.
* Monitor – A Boolean (true/false) value indicating whether this app should listen for “event clients.” The API can act as an event client. Event clients can connect to the app via a socket and pull a list of events generated by the app. This is typically used to provide a scrolling event window.
* EventPort – If set, this property determines the port that this app will listen on for event clients. If *not* set (the usual case), the port is determined by the formula:

port = (pid % 10000) + 20000

### GET appstat

Example

http://localhost:8080/odcsapi/appstat

The returned structure is an array with one element for each application returned by the “GET apprefs” method described in section 2.12.1.

If an application is currently running, the “pid” will by the system process ID, and “heartbeat” will be a valid date/time. Also “status” will be set to some relevant string for that application. For example, the compproc app sets its status to the number of computation runs and errors.

[

{

"appId": 1,

"appName": "decodes",

"appType": null,

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 4,

"appName": "compproc",

"appType": "computationprocess",

"hostname": "mmaloney3.local",

"pid": 12176,

"heartbeat": "2023-05-25T16:34:18.073Z[UTC]",

"eventPort": null,

"status": "Cmps: 0/0"

},

{

"appId": 5,

"appName": "compproc\_regtest",

"appType": null,

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 8,

"appName": "utility",

"appType": "utility",

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 18,

"appName": "StaleDataChecker",

"appType": null,

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 19,

"appName": "compedit",

"appType": null,

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 20,

"appName": "corrections",

"appType": null,

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 21,

"appName": "limits",

"appType": null,

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 22,

"appName": "statmon",

"appType": null,

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 23,

"appName": "dcpmon",

"appType": null,

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 25,

"appName": "CompEdit",

"appType": "gui",

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 26,

"appName": "RoutingScheduler",

"appType": "routingscheduler",

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

},

{

"appId": 27,

"appName": "compdepends",

"appType": "compdepends",

"hostname": null,

"pid": null,

"heartbeat": null,

"eventPort": null,

"status": "Inactive"

}

]

### GET appevents

Sample URL:

http://localhost:8080/odcsapi/appevents?appid=4&token=58a45adde8759bbb

This method requires a valid token, passed either in the URL (as shown above) or in the HTTP request header.

The returned data structure is an array of events. Each event contains:

* appId – the one requested on the header
* appName – from the loading app database record
* priority – one of DBG3, DBG2, DBG1, INFO, WARNING, FAILURE.
* eventTime – date/time of the event
* eventText – provided by the application

Returned data structure example shown below. Note that an empty array simply means that there are no new events from that application.

[

{

"appId": 4,

"appName": "compproc",

"priority": "DBG1",

"eventTime": "2023-05-25T17:26:13.000Z[UTC]",

"eventText": "OpenTimeSeriesDAO No new data for appId=4"

},

{

"appId": 4,

"appName": "compproc",

"priority": "DBG1",

"eventTime": "2023-05-25T17:26:14.000Z[UTC]",

"eventText": "OpenTimeSeriesDAO No new data for appId=4"

},

{

"appId": 4,

"appName": "compproc",

"priority": "DBG1",

"eventTime": "2023-05-25T17:26:15.000Z[UTC]",

"eventText": "OpenTimeSeriesDAO No new data for appId=4"

},

{

"appId": 4,

"appName": "compproc",

"priority": "DBG1",

"eventTime": "2023-05-25T17:26:16.000Z[UTC]",

"eventText": "OpenTimeSeriesDAO No new data for appId=4"

},

{

"appId": 4,

"appName": "compproc",

"priority": "DBG1",

"eventTime": "2023-05-25T17:26:17.000Z[UTC]",

"eventText": "OpenTimeSeriesDAO No new data for appId=4"

}

]

Errors will be returned if the application is not currently running or if its heartbeat has gone stale (i.e. 20 seconds have elapsed with no update). For example:

{

"status": 410,

"message": "appid 5 (compproc\_regtest) is not running."

}

### POST appstart

Sample URL:

http://localhost:8080/odcsapi/appstart?token=8ef5b2c8b9998b8d&appid=26

A valid token is required for this operation which may be passed in the HTTP header or command line.

The appid parameter is required. The API will execute the “startCmd” property found in the database record for the app. The string will be expanded with the environment, so it may contain strings like:

$DCSTOOL\_HOME/scripts/startRoutSched

On most operating systems, when the command is executed, it becomes a child of the API process. Thus when/if the API process is terminated, the child app will also die. To prevent this on a Linux system, create a script that uses nohup to break the link between parent and child processes. For example the startRoutSched script looks like this:

#!/bin/bash

export DCSTOOL\_HOME=$HOME/test/odcsapi

nohup $DCSTOOL\_HOME/bin/routsched -d1 -l routsched.log &

### POST appstop

Sample URL:

http://localhost:8080/odcsapi/appstop?token=8ef5b2c8b9998b8d&appid=26

A valid token is required for this operation which may be passed in the HTTP header or command line.

The appid parameter is required. The API will remove the cp\_comp\_proc\_lock record for this process, if one exists. Well behaved OpenDCS programs will then clean up and exit within 10 seconds.

## Routing Monitor and Control

The following methods allow a user to view the status of all routing specs and to start/stop them.

### GET routingstatus

This method allows a developer to implement a web version of the OpenDCS Routing Monitor screen.

Sample URL:

http://localhost:8080/odcsapi/routingstatus

This method does not require a valid token. If one is passed, its last-used time is updated.

The returned data structure is shown below. Note the following:

* All routing specs are contained in the list regardless of whether they have a schedule entry assigned. No schedule entry is indicated by scheduleEntryId = null.
* Routing specs with a suffix of “-manual” and the “manual” attribute set to true indicate that the routing spec was run throught the “rs” command. Otherwise they were run by a Routing Scheduler from a schedule entry.
* A routing spec may be run either way. Note that the entry for “test” and “test-manual” are the same routing spec. “test” was run from the routing scheduler with scheduleEntryId=43, and “test-manual” was run from the command line “rs”.

[

{

"routingSpecId": 44,

"name": "rs-MROI4-ROWI4",

"scheduleEntryId": null,

"appId": null,

"appName": null,

"runInterval": null,

"lastActivity": null,

"lastMsgTime": null,

"numMessages": 0,

"numErrors": 0,

"enabled": false,

"manual": false

},

{

"routingSpecId": 58,

"name": "test",

"scheduleEntryId": 43,

"appId": 26,

"appName": "RoutingScheduler",

"runInterval": "5 minute",

"lastActivity": "2023-05-31T18:56:54.364Z[UTC]",

"lastMsgTime": "2023-05-31T18:56:53.099Z[UTC]",

"numMessages": 3362,

"numErrors": 3362,

"enabled": true,

"manual": false

},

{

"routingSpecId": 58,

"name": "test-manual",

"scheduleEntryId": 40,

"appId": 0,

"appName": null,

"runInterval": null,

"lastActivity": "2023-05-31T18:37:02.490Z[UTC]",

"lastMsgTime": "2023-05-31T18:37:02.458Z[UTC]",

"numMessages": 5700,

"numErrors": 5699,

"enabled": true,

"manual": true

},

{

"routingSpecId": 59,

"name": "goes1-manual",

"scheduleEntryId": 39,

"appId": 0,

"appName": null,

"runInterval": null,

"lastActivity": "2022-12-01T22:19:06.024Z[UTC]",

"lastMsgTime": "2022-12-01T22:19:05.939Z[UTC]",

"numMessages": 9,

"numErrors": 0,

"enabled": true,

"manual": true

},

{

"routingSpecId": 63,

"name": "periodic-10-min",

"scheduleEntryId": 38,

"appId": 26,

"appName": "RoutingScheduler",

"runInterval": "10 minute",

"lastActivity": "2023-05-22T13:28:17.631Z[UTC]",

"lastMsgTime": "2023-05-22T13:28:12.825Z[UTC]",

"numMessages": 8,

"numErrors": 0,

"enabled": true,

"manual": false

},

{

"routingSpecId": 63,

"name": "periodic-10-minute-manual",

"scheduleEntryId": 37,

"appId": 0,

"appName": null,

"runInterval": null,

"lastActivity": "2022-12-01T20:37:50.811Z[UTC]",

"lastMsgTime": "2022-12-01T20:37:50.710Z[UTC]",

"numMessages": 6,

"numErrors": 0,

"enabled": true,

"manual": true

},

{

"routingSpecId": 65,

"name": "last-5-min",

"scheduleEntryId": null,

"appId": null,

"appName": null,

"runInterval": null,

"lastActivity": null,

"lastMsgTime": null,

"numMessages": 0,

"numErrors": 0,

"enabled": false,

"manual": false

},

{

"routingSpecId": 65,

"name": "last-5-min-manual",

"scheduleEntryId": 42,

"appId": 0,

"appName": null,

"runInterval": null,

"lastActivity": "2023-05-31T18:49:49.453Z[UTC]",

"lastMsgTime": "2023-05-31T18:49:47.902Z[UTC]",

"numMessages": 3179,

"numErrors": 0,

"enabled": true,

"manual": true

}

]

### GET routingexecstatus

Sample URL

http://localhost:8080/odcsapi/routingexecstatus?scheduleentryid=38

Note in section 5.2.1 the “GET routingstatus” method returns a list of routing specs showing a unique scheduleEntryId for each entry. There may be more than one entry for each routing spec because:

* The same routing spec may be run in multiple schedule entries.
* A “manual” routing spec (i.e. run with the “rs” command) will appear as a separate schedule entry with the “manual” Boolean set to true.

Thus this method, GET routingexecstatus, takes a scheduleentryid as its argument. It returns all of the executions for the specified schedule entry. The returned data structure appears as follows:

[

{

"routingExecId": 568,

"scheduleEntryId": 38,

"routingSpecId": 63,

"runStart": "2023-06-01T17:20:00.516Z[UTC]",

"runStop": "2023-06-01T17:20:00.526Z[UTC]",

"numMessages": 0,

"numErrors": 0,

"numPlatforms": 0,

"lastMsgTime": null,

"lastActivity": "2023-06-01T17:20:00.527Z[UTC]",

"runStatus": "ERR-OutputInit",

"hostname": "mmaloney3.local",

"lastInput": null,

"lastOutput": null

},

{

"routingExecId": 565,

"scheduleEntryId": 38,

"routingSpecId": 63,

"runStart": "2023-06-01T17:10:00.841Z[UTC]",

"runStop": "2023-06-01T17:10:00.855Z[UTC]",

"numMessages": 0,

"numErrors": 0,

"numPlatforms": 0,

"lastMsgTime": null,

"lastActivity": "2023-06-01T17:10:00.855Z[UTC]",

"runStatus": "ERR-OutputInit",

"hostname": "mmaloney3.local",

"lastInput": null,

"lastOutput": null

},

{

"routingExecId": 562,

"scheduleEntryId": 38,

"routingSpecId": 63,

"runStart": "2023-06-01T17:00:00.259Z[UTC]",

"runStop": "2023-06-01T17:00:00.269Z[UTC]",

"numMessages": 0,

"numErrors": 0,

"numPlatforms": 0,

"lastMsgTime": null,

"lastActivity": "2023-06-01T17:00:00.270Z[UTC]",

"runStatus": "ERR-OutputInit",

"hostname": "mmaloney3.local",

"lastInput": null,

"lastOutput": null

}

]

The entries are sorted in descending order by the runStart time. “runStop” may be null if the execution was halted abnormally or if it is still running. If any messages were processed, the num Messages/Errors/Platforms will be non-zero.

### GET dacqevents

Sample URL:

http://localhost:8080/odcsapi/dacqevents?token=88d30903583e6751&appid=26

The “GET dacqevents” method returns events stored in the DACQ\_EVENT database table. These are events having to do with data acquisition (DACQ) events. That can be associated with …

* An execution of a routing spec (pass argument routingexecid)
* An application (pass argument appid)
* A specific platform (pass argument platformid)

The URL must contain a valid user token (either on URL or in HTTP header). It may contain any of the following additional argument. Each argument refines a filter that determines which events are to be returned:

* appid (long integer): only return events generated by a specific app.
* routingexecid (long integer): only return events generated during a specific execution of a routing spec. (The “GET routingexecstatus” method will return a list of executions, each with a unique ID.)
* platformid (long integer): only return events generated during the processing of a specific platform.
* backlog (string): either the word “last” or one of the valid interval names returned in GET intervals (see section 3.4.1). Only events generated since the specified interval are returned. The word “last” means only return events generated since the last “GET dacqevents” call within this session. It can be used to approximate a real-time stream.

The returned data looks like this:

[

{

"eventId": 181646,

"routingExecId": 607,

"platformId": null,

"eventTime": "2023-06-08T19:21:15.255Z[UTC]",

"priority": "INFO",

"appId": 26,

"appName": "RoutingScheduler",

"subsystem": null,

"msgRecvTime": null,

"eventText": "RoutingSpec(test) Connected to DDS server at www.covesw.com:-1, username='covetest'"

},

{

"eventId": 181647,

"routingExecId": 606,

"platformId": null,

"eventTime": "2023-06-08T19:21:15.281Z[UTC]",

"priority": "INFO",

"appId": 26,

"appName": "RoutingScheduler",

"subsystem": null,

"msgRecvTime": null,

"eventText": "RoutingSpec(periodic-10-minute) Connected to DDS server at www.covesw.com:-1, username='covetest'"

},

{

"eventId": 181648,

"routingExecId": 607,

"platformId": null,

"eventTime": "2023-06-08T19:21:15.284Z[UTC]",

"priority": "INFO",

"appId": 26,

"appName": "RoutingScheduler",

"subsystem": null,

"msgRecvTime": null,

"eventText": "RoutingSpec(test) Purging old DACQ\_EVENTs before Sat Jun 03 15:21:15 EDT 2023"

},

]

### GET platformstat

Sample URL:

http://localhost:8080/odcsapi/platformstat

Token is not required for this method but if passed in the header or URL, it will be updated.

Optional argument “netlistid” can be passed to only return platforms that have a transport medium in the referenced network list.

The returned data structure contains information about recent activity on each platform:

[

{

"platformId": 53,

"platformName": "OKVI4",

"siteId": 1,

"lastContact": "2023-06-09T18:30:53.086Z[UTC]",

"lastMessage": "2023-06-09T18:30:53.086Z[UTC]",

"lastError": null,

"lastMsgQuality": "G",

"annotation": null,

"lastRoutingExecId": 609,

"routingSpecName": "periodic-10-minute"

},

{

"platformId": 54,

"platformName": "MROI4",

"siteId": 6,

"lastContact": "2023-06-09T18:30:53.102Z[UTC]",

"lastMessage": "2023-06-09T18:30:53.102Z[UTC]",

"lastError": null,

"lastMsgQuality": "G",

"annotation": null,

"lastRoutingExecId": 609,

"routingSpecName": "periodic-10-minute"

},

{

"platformId": 55,

"platformName": "ROWI4",

"siteId": 2,

"lastContact": "2023-06-09T18:30:53.013Z[UTC]",

"lastMessage": "2023-06-09T18:30:53.013Z[UTC]",

"lastError": null,

"lastMsgQuality": "G",

"annotation": null,

"lastRoutingExecId": 609,

"routingSpecName": "periodic-10-minute"

}

]

This may be used to populate a GUI similar to the Java “Platform Monitor” GUI in OpenDCS. A pulldown list of network list IDs and names is recommended.

Note the following:

* “lastContact” is the last time that any communication from this platform was received.
* “lastMessage” is the time stamp that the last message from this platform was received.
* “lastError” is the time that a decoding or communications error last occurred with this platform.
* “lastRoutingExecId” indicates the specific execution of a routing spec that was last used to process this platform. (See Routing Exec Status above.)
* “annotation” is the text of the last error message generated from this platform.

To retrieve a list of Data Acquisition events for a platform, you can use the GET dacqevent method described in section 5.2.3, passing the “platformid” argument.