Synamedia

PowerVu Professional Receiver (D9800) Release 07.80 RESTful Web Services API Reference Guide

First Published: 2024-12-19

Contents

Preface

This section describes the audience and conventions of the PowerVu Professional Receiver API Reference Guide. It also references related documentation and describes how to obtain documentation and submit a service request.

Audience

The audience of this guide includes developers who will build a client application to interface with the PowerVu Professional Receiver.

Conventions

This guide uses the following conventions.

Conventions	Indication
bold font	Commands and keywords and user-entered text appear in bold font.
italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in italic font.
[ ]	Elements in square brackets are optional.
{x | y | z }	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
`courier font`	Terminal sessions and information the system displays appear in courier font.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Note Means reader take note. Notes contain helpful suggestions or references to material not covered in the manual.

Caution Means reader be careful. In this situation, you might perform an action that could result in equipment damage or loss of data.

Warning

IMPORTANT SAFETY INSTRUCTIONS
Means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device.
SAVE THESE INSTRUCTIONS

Related Documentation

The following lists the Synamedia Customer Portal URL for product documentation and software downloads: https://www.portal.synamedia.com.

Note To access this page, an account is required. Contact your Synamedia account manager or support representative for a registration code or complete the form here: https://www.synamedia.com/about/#contact.

Customer Service Contact Information

The following lists the Synamedia Customer Service Desk URL for product support: https://vss.service-now.com. Alternatively, you may contact Customer Service Desk using the following toll-free number: 1-855-605-8390.

CHAPTER 1

Web Services API

This chapter describes the web services API used to control the PowerVu Professional Receiver, including HTTP request and response methods, URI arguments, and port information.

About Web Services API

Web services API is a standard interface designed for the PowerVu Professional Receiver. It replaces the previous interfaces, such as TELNET. The web services API is designed to be flexible and address different functionality requirements of an interface, such as concurrent access, authentication, authorization, and so on. All APIs are implemented using HTTP protocol and run on standard web servers using TCP port 80 or 443 for HTTPS.

Accessing Web Services API

This document assumes that the IP address of the unit being accessed is 192.168.0.1. When copying or pasting into a cURL console, you must replace **192.168.0.1** in the message with the IP address of the target unit. By default, the PowerVu Professional Receiver web server redirects all incoming web traffic to HTTPS on port 443 and will respond to all HTTP URI requests with a redirect response to the equivalent HTTPS URI.

The APIs in this document that has the **/ws/v1/** extension indicates a version 1 (Legacy) web services API, and the **/ws/v2/** extension indicates a version 2 web services API. For example, https://192.168.0.1**/ws/v2/**show/inventory.

To access a single API, you can use cURL. For more information, see https://curl.haxx.se/. Since SSL certificates on receiver units are self-signed, you can use "-k DEFAULT" options to bypass the security checks of the cURL command.

Example 1:

```
curl -k -i -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/show/id-eeprom"
```

Example 2:

```
curl -k "https://192.168.0.1/ws/v2/test/show/id-eeprom&session=$token"
```

Note The DEFAULT –k cipher option might not be available in older versions of the cURL command. An updated version of the tool is required to enable this option. Follow your operating systems instructions to update your cURL. For example, on CEL/Fedora, issue the following command as a root user and then restart:

```
yum update curl
```

The `-H "X-SESSION-ID: $token"` (or `session=$token`) part of the cURL syntax is required for each command in order to propagate the HTTP session_id. The `$token` represents a Linux (or Cygwin if that is the Host environment) environment variable that is based on the value of the XML <SESSION></SESSION> item from the successful Login response.

For an example of the login command, and setting the $token environment variable, see Access Level (Login and Logout), on page 13.

Access Level (Login and Logout)

When you successfully log in to a session, an access level is assigned. Access levels are confirmed on a per API basis, and depending on the access level restrictions, the behavior of the API may change.

The access level is assigned to the current user, based on the USER_ROLE value that was set for that user when the ADMIN user created the user account. The USER_ROLE value is returned in the Login response.

1. NotValid: No session available.
2. ADMIN: Admin session active (ability to create users, software, run all customer documented APIs).
3. USER: User session active (restricted so a subset of APIs, unable to change most target configuration settings).
4. GUEST: Guest session active (limited API availability to some read-only status APIs only).

The following is an example of the Login command.

Input:

```
curl -k https://192.168.0.1/ws/v1/table?t=return -X POST -d "<HDR><LOGIN>
<UID>admin</UID><USERPASS>localadmin</USERPASS></LOGIN></HDR>
" -H "Content-Type: text/xml; charset=UTF-8"
```

Expected output:

```
?xml version="1.0" encoding="ISO-8859-1"?>
<HDR><TABLE><ITEM><ID>STATUS</ID><VALUE>Please wait while the default page
loads up ....</VALUE></ITEM><LOGIN_INFO><USER_NAME>admin</USER_NAME>
<USER_ROLE>ADMIN</USER_ROLE><SESSION_ID>TokenValue</SESSION_ID></LOGIN_
INFO></TABLE></HDR>
```

Where Session `TokenValue` should be used in each subsequent Web Server API commands.

In each of the commands described in this document, the `-H "X-SESSION-ID: $token"` (or appended `"session=$token"`) portion of the curl request syntax passes the value of `$token` environment variable.

The value of the $token variable needs to be set based on the TokenValue from the XML response to the Login Command. This may be done by using the Linux (or Cyqwin) export command.

The example below assumes TokenValue = 869a010f-16e9-4b5a-b302-0097c779850f.

```
…<SESSION_ID>869a010f-16e9-4b5a-b302-0097c779850f </SESSION_ID>…
```

Usage Example (setting $token environment variable):

```
export token=869a010f-16e9-4b5a-b302-0097c779850f
```

An alternative to setting the environment variable is to replace $token with the literal string from the above (XML) output when calling the additional command(s). The following is an example of a logout command.

Input:

```
curl -k https://192.168.0.1/ws/v1/table?t=return -X POST -d
"<HDR><LOGOUT><ITEM><REASON>User Logout</REASON></ITEM></LOGOUT></HDR>
" -H "Content-Type: text/xml; charset=UTF-8" -H "X-SESSION-ID: $token"
```

Expected output:

```
?xml version="1.0" encoding="ISO-8859-1"?>
<HDR><TABLE><ITEM><ID>STATUS</ID><VALUE>Please wait while the default page
loads up ....</VALUE></ITEM><LOGIN_INFO><USER_NAME></USER_NAME>
<USER_ROLE>NotValid</USER_ROLE><SESSION_ID>NONE</SESSION_ID></LOGIN_
INFO></TABLE></HDR>
```

HTTP Request Methods

HTTP methods are defined by RFC 1945. We currently support GET and POST methods.

Get

GET is used for URI requests that have no payload to transfer in their bodies. GET can be formatted in either XML or JSON.

The following is an example of a Request in XML:

```
curl -k -i -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/service_cfg/input/rf?port=2"
```

Or

```
curl -k "https://192.168.0.1/ws/v2/service_
cfg/input/rf?port=2&session=$token"
```

The following is an example of a Request in JSON (by specifying js=1):

```
curl -k -i -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/service_cfg/input/rf?port=1&js=1"
```

Or

```
curl -k  "https://192.168.0.1/ws/v2/service_
cfg/input/rf?port=1&js=1&session=$token"
```

The resource URI must be in quotes if you pass in multiple query parameters separated by '&'. If you have spaces in the query values, you should encode them. For example, use the + symbol, or the %20 symbol, instead of a space (http://blogs.plexibus.com/2009/01/15/rest-esting-with-curl/).

Query parameters/URI arguments marked (key) in the API descriptions in this document are mandatory for the indicated command syntax.

A maximum of 26 arguments (after the ? symbol, and not including the session and json arguments) can be queried in a single GET operation from a Web Services API cURL command line request. If a GET request is made that specifies more than this amount, the response behavior is undefined (typically, the extra request parameters will be ignored).

The RFCs or Specifications for RESTful APIs, and most variants of client HTTP(s) software such as curl or Python request or C# HTTP, do not support passing a block of data in a GET call. The "-d" option is reserved only for POST operations, and in the current version of curl, it will override any -X (GET or POST) setting to always assume that it is a POST operation. Do not use -d or attempt to send data other than by Query Strings (referred to as "params" in Python Requests) in GET calls.

Multiple GET calls may be required in order to retrieve specific groups of elements greater than a count of 26 Query Strings (not including the js or session parameters). We highly recommend that you call the GET API to retrieve the entire response without attempting to use Query String or params filters in these cases.

Note Using Query String or params filters for Web Services Status APIs that have a "act", "rowstatus" or "cmdrow" parameter: This parameter is important to always be included in the response in order for the context of the response to be known. Otherwise, client software or test scripts may incorrectly take action based on stale or invalid data.

We highly recommend that you always include this parameter (when available) in the calls for GET APIs that use Query Strings. Omitting the "act", cmdrow" or "rowstatus" parameter may lead to false conclusions about the validity of the other parameters in the response.

Post

POST is used when a request body (in XML, JSON, or other formats) is present and is transferred to the unit with the request. The following lists three ways to use cURL to post parameters and arguments.

1. Using only arguments on command line separated by "&" (Maximum of 26 arguments, excluding JSON argument and session argument):

   ```
   curl -k -i -H "X-SESSION-ID: $token" -X POST
   "https://192.168.0.1/ws/v2/service_cfg/input/rf?port=
   [port#]&amp;dnlkfreq=&lt;Freq&gt;"
   ```

   ```
   curl -k -i -H "X-SESSION-ID: $token" -X POST
   "https://192.168.0.1/ws/v2/service_
   cfg/input/rf?port=2&amp;dnlkfreq=12.31"
   ```

   Or

   ```
   curl -k -X POST "https://192.168.0.1/ws/v2/service_cfg/input/rf?port=
   [port#]&DnlkFreq=<Freq>&session=$token"
   ```

2. Using xml data from command line:

   ```
   curl -k -i -H "X-SESSION-ID: $token" -X POST --header
   "Content-type: application/xml" -d
   "<Input><RF><Port>1</Port><Act>Yes</Act></RF</Input>
   " https://192.168.0.1/ws/v2/service_cfg/input?session=$token
   ```

   Or

   ```
   curl -k -i -X POST --header "Content-type: application/xml" -d
   "<Input><RF><Port>1</Port><Act>Yes</Act></RF</Input>"
   https://192.168.0.1/ws/v2/service_cfg/input?session=$token
   ```

3. Using json data from command line:

   Note The HTML body content data may alternatively be sent via the command line using JSON format by specifying "Content-type: application/json" instead of "Content-type: application/xml" in the --header or -H settings and using properly formatted JSON content in the -d string.

4. Using xml data from file:

   ```
   curl -k -i -H "X-SESSION-ID: $token" -X POST --header
   "Content-type: application/xml" -d @"input.xml"
   https://192.168.0.1/ws/v2/service_cfg/input
   ```

   Or

```
curl -k -i -X POST --header "Content-type: application/xml"
-d @"input.xml" "https://192.168.0.1/ws/v2/service_
cfg/input?session=$token"
```

Using json data from file:

> **Note** The HTML body content data may alternatively be sent from a file using JSON format by specifying "Content-type: application/json" instead of "Content-type: application/xml" in the --header or -H settings, naming the file with .json extension instead of .xml, and using properly formatted JSON content in the input.json file.

A maximum of 26 arguments (after the ?, and not including the session and json arguments) can be set in a single POST operation from a Web Services API cURL command line request. If a POST request is made that specifies more than the maximum, the response behavior is undefined (typically, the extra request parameters are ignored). We recommend that you use the alternate write from file method to set multiple arguments at the same time.

HTTP Responses

This section describes the HTTP status codes and content response types.

HTTP Status Codes

All HTTP commands respond with a status code that identifies if the client or server encountered any protocol or URL mapping issues. For the command to proceed (for example, return valid data), the HTTP status code must display the value 200 (HTTP_OK). Any other HTTP response code indicates an error and the data content will not be received. That is, if the client expects an XML (or JSON) response, then it must first check that the HTTP status code in the HTML header of the response is 200.

For client software or debugging purposes, you may want to dump the HTTP response header (which shows the status code) via the `curl -i` parameter.

The following is an example of an invalid command for which HTTP 404 error is returned:

```
curl -i -k -H "X-SESSION-ID: $token" -X
POST"https://192.168.0.1/ws/v2/invalid_command
HTTP/1.1 404 Not Found
Date: Fri, 23 Oct 2015 19:26:40 GMT
Server: Hiawatha v9.14
Connection: keep-alive
Transfer-Encoding: chunked
Status: 404
```

The following is a list of the common HTTP return codes when communicating with the PowerVu Professional Receiver (refer to RFC-1945 for the full list and descriptions):

```
200_HTTP_OK
400_HTTP_BAD_REQUEST
401_HTTP_UNAUTHORIZED
403_HTTP_FORBIDDEN
404_HTTP_NOT_FOUND
500_HTTP_INTERNAL_SERVER_ERROR
501_HTTP_NOT_IMPLEMENTED
502_HTTP_BAD_GATEWAY
503_HTTP_SERVICE_UNAVAILABLE
504_HTTP_GATEWAY_TIMEOUT
505_HTTP_VERSION_NOT_SUPPORTED
```

Content Response Types

When HTTP status code is HTTP_OK (value 200), there are two types of content responses available:

1. application/xml
2. application/json

When not specified, the response content is returned in XML format.

When js=1 is specified as an appended "&" parameter, then the response is returned in JSON format.

When a legacy API (for example, backup, restore_upload, cdt_upload, diagnostics) is being called, the response may be in command specific XML or HTML format only and does not support the Generic Response XML format or Json (js=1) options. Refer to the commands for specific descriptions.

XML or JSON Response Body

When a WS V2 or higher API is executed, and HTTP status code is HTTP_OK (value 200), a XML/JSON response body is returned, to indicate the API result.

The following is an example of the GET method API, used to query the receiver data. If the GET API is successfully executed, the response body displays the requested data.

```
$ curl -k -H "X_SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/input/rf?port=1&js=1"
{
    "input": {
        "rf": {
            "port": "1",
            "dnlkfreq": "28.3465",
            "symrate": "28.3465",
            "fec": "2/3",
            "mod": "QPSK DVB-S",
            "rolloff": ".35",
            "iq": "Normal",
            "pwr": "Off",
            "sel22khz": "Off",
            "lo1": "5.15",
            "lo2": "0.0",
            "pol": "Horizontal",
            "orbpos": "0.0",
            "ewflag": "N/A",
            "afcrange": "3.0",
            "acqmode": "Basic",
            "satlock": "No Lock",
            "inputrate": "0.0",
            "netname": "",
            "netid": "1",
            "transid": "",
            "Isttunerea": "User_Change",
            "tunereason": "User",
            "freqmode": "NIT",
            "swclstmode": "Rigorous",
            "baten": "No",
            "niten": "Yes",
            "sdten": "No",
            "paten": "No"
        }
    }
}
```

If an error occurs, a generic "failed" response body is provided to indicate the API failed, and a text message to indicate the failure reason. For example:

```
$ curl -k -H "X_SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/input/rf?port=1&js=1"
{
    "response": {
        "code": "10",
        "result": "failed",
        "message": "Access denied (must be admin or user)"
    }
}
```

The following is an example of the POST method API. It displays a generic "success" response. An API requires a specific response body when a POST API is executed successfully.

```
$ curl -k -H "X_SESSION-ID: $token" -X POST
"https://192.168.0.1/ws/v2/service_cfg/input/rf?port=1&&dnlkfreq=3.51&js=1"
{
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
}
```

If an error occurs, a generic "failed" response is given. For example:

```
$ curl -k -H "X_SESSION-ID: $token" -X POST
"https://192.168.0.1/ws/v2/service_cfg/input/rf?js=1"
{
    "response": {
        "code": "10",
        "result": "failed",
        "message": "Missing Key"
    }
}
```

Note The "code" in the generic "failed or success" response body is not used (most command responses set it to 10). We highly recommend that you do not change the field, unless otherwise indicated in the command descriptions.

Base URI

The base URI for an API indicates a specific or a group of functionalities, which is used in conjunction with URI parameters and options.

URI Parameters

Each web service API can have one or more URI parameters that can alter the semantics of the API. URI parameters are appended to the base URI. For instance, a request to Show API can change the type of information requested by providing a URI parameter such as "inventory" or "id-eeprom":

https://192.168.0.1/ws/v2/test/show/inventory

https://192.168.0.1/ws/v2/test/show/id-eeprom

URI parameters should be used separately, and are separated from the base URI by "/" character.

URI Arguments

Options are additional variables that an API can accept as input. Options are separated from URI parameters or base URI by a "?" character and are separated from each other by the "&" character.

Each option has a name and a value. For instance, in the URL "**https://192.168.0.1/ws/v2/test/show/id-eeprom?card=NDM**", the string "card" indicates an option with its value set to "NDM".

- Argument Description

  A brief description of the argument item.

- Argument Type

  Currently 5 different data types are supported for query options:

  1. Integer
  2. String
  3. Boolean
  4. IP Address
  5. Float

- Argument Values

  Values define a specific or a range of acceptable values for the option. The following list shows some examples of how the values or the range of values are identified in this document:

  - Integer: [1 ... 40] identifies range of values from 1 to 40.
  - String: * is used as wildcard.
  - Boolean: 0 is false and 1 is true.
  - IP Address: e.g. 192.160.0.1
  - Float: 30.5

- Argument Applies To

  Identifies which URI parameters a certain URI argument applies to. For example, the string "*Applies:* **id-eeprom, status-eeprom**" for the argument "**card**"should be interpreted as: "card argument can be used in conjunction only with URI parameters **id-eeprom, status-eeprom**".

Access Type

This field indicates the type and direction of interaction with the web services API. The valid types are:

- Read: Reading parameters from the PowerVu Professional Receiver.
- Write: Write parameters to the PowerVu Professional Receiver.

- Read/Write: Read or write parameters from or to the PowerVu Professional Receiver.

Port Information

There are three types of NTC boards:

Table 1.1 NTC Board Types

NTC Board	Port Description
NTC Basic	One Ethernet port: ▪ Management
NTC MOIP	Three Ethernet ports: ▪ Management ▪ DATA1 ▪ DATA2
NTC-MS	Five Ethernet ports: ▪ Management ▪ DATA1 and DATA2 (first data port pairs) ▪ DATA3 and DATA4 (second data port pairs)

When using "port" as a URI argument for web services API commands, the ports are referred to as port 0, port 1, port 2, and so on.

"Management port" is port 0. This can be used on all three types of NTC boards.

DATA1 is port 1. This can be used on NTC MOIP and NTC-MS boards.

DATA2 is port 2. This can be used on NTC MOIP and NTC-MS boards.

DATA3 and DATA4 is port 3 and port 4 respectively. They can only be used on the NTC-MS board.

Note When connecting to a unit to run web services API commands, the Management port (port 0) is used to establish and maintain the connection.

The ports are labeled on the backplane of an assembled unit. The Management port (port 0) is labeled Management. The DATA ports are labeled DATA1, DATA2, DATA3, and DATA4.

Note The Management port is also referred to as "Control port" in this document.

CHAPTER 2

API Definitions

This section describes the API definitions for controling the PowerVu Professional Receiver.

Legacy API

The legacy API commands use the ws/v1 (Web Services v1 API) or legacy command syntax (for the cases of the /restore_upload.html and /cdt_upload.html usage) and are similar in behavior to those that were implemented in previous D98xx products.

Backup Command

Table 2.1 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v1/backup?t=*option*
Command Information	Backup specific types of database data from the device to your local machine.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin

Table 2.2 Options

Option	Description
STD	Backs up standard data.
UD	Backs up user changes to factory data (customer presets).
EXT	Backs up all standard data and user data.

Example of issuing the backup command (assuming the STD option is specified):

Input:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v1/backup?t=STD"
```

Expected output (may not be exactly as shown):

```
<BKPRST>
<FILE_INFO><PLAT_TYPE>D9800</PLAT_TYPE><DESIGNATION>10</DESIGNATION>
<VERSION>1.00</VERSION>
<DATE_TIME>Date YYYY/DD/MM 2015/07/20 Time HR:MIN:SEC 20:57:33</DATE_TIME>
<FILE_NAME>file name</FILE_NAME></FILE_INFO>
<SETTINGS>
<BKPRST>
    </SETTINGS>
        <RECORD>
            <OID>1.3.6.1.4.1.1482.20.1.9.1.1.2.1</OID>
            <VALUE>6C410A0A</VALUE>
        </RECORD>
        …other XML records …
    </SETTINGS>
    …other XML data …
</BKPRST>
```

Note The XML response is not immediate and may take several seconds to complete. While the Backup command is in progress, the BKPRSTSTAT UIC table should not be queried by other curl clients other than the UI (if at all), in order to reduce the risk of timeout due to inter process call overhead. Prior to the completion of the command, the BKPRSTSTAT UIC table is incremented, with internal state changes and incremental progress percent completions, up to 100%. However, these state changes may occur faster than they are polled. When the command is completed, the BKPRSTSTAT UIC table Operation Status state will set back to Idle, and the Progress percentage will set back to 0%, in preparation for any subsequent Backup or Restore operation. The availability of the new Backup XML data (or file Save As dialog in the case of using the UI Browser) is the valid indicator for the successful completion of this command. If there is an issue with generating the Backup file, then the BKPRSTSTAT UIC table Operation Status will indicate the Fail state, with the Detailed Status field, indicating the reason for the Failure. Check this data only after the command returns. If there is a problem with file transfer, timeout, or invoking the command, then the XML response to this command will indicate a failure response through the legacy command Status form.

Restore Command

Table 2.3 Command Details

Command Detail	Description
Command URL	-F restorefile=@"file_path/file_name" "https://192.168.0.1/restore_upload.html"
Command Information	Restore database file(s) onto the device.
HTTP Method	POST
Access Type	Write
Access Level	User, Admin

Options: N/A

Example of issuing the restore command (assuming the filename is D9800.bkp):

Input:

```
curl -k -i -H "X-SESSION-ID: $token" -X POST -F
restorefile=@"C:/projects/d9800/D9800.bkp" "https://192.168.0.1/restore_
upload.html"
```

Expected output:

```
<html><body></body></html>
```

Note The XML response is almost immediate (a few seconds at most) and does not indicate that the restore command operation is completed.

Prior to initial run of this command after startup (prior to any Backup or Diagnostics operation), the BKPRSTSTAT table may indicate that the operationStatus="Pass" while detailedStatus="Idle" and percentCompleted=0".

Upon successful completion of the Restore operation, the BKPRSTSTAT table will indicate that the operationStatus="Pass" while detailedStatus="Idle" and percentCompleted=100".

In order to successfully start the command from either of these states without receiving a race-logic condition in which false "Pass" will be indicated, any poll loop that runs after starting the /restore command must consider that

ALL three items ("Pass" plus "Idle" plus "100") must coincide in order to consider Restore operation to be Successfully completed.

The progress and results of the Restore command may be determined by calling the BKPRSTSTAT command every few seconds until it reports that the file restore is completed at 100 percent, or that it was aborted (operationStatus="Fail") due to errors.

This operation may take over one minute, depending on the amount of data to validate and restore.

Versions Command

Table 2.4 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/test/versions
Command Information	Reports Installed App and Third Party or Open Source Package versions.
HTTP Method	GET
Access Type	Read, Test command
Access Level	User, Admin
GET Syntax Examples	GET "https://192.168.0.1/ws/v2/test/versions"

Table 2.5 URI Parameters (extension to the Command URL separated by /)

URI Argument	Description
default (no subsection specified)	Version 3.50 or earlier: Displays only apps subsection of version info strings. Version 3.75 or later: Displays all of (in this order) the data for apps, top, and platform subsections. Version 4.75 or later: Displays all of (in this order) the data for apps, top, platform, and distro subsections.
other	Version 3.50 or earlier: Only apps extension was available. Anything else as an extension was ignored and still only return apps subsection of versions as described below. Version 3.75 or later: The following extensions are available: apps, top, and platform and GET ws/test/versions will return all as subsections of versions as described below. Using anything else as an extension will result in a failure indicating "Invalid URI Parameter Name". Version 4.75 or later: The following extensions are available: apps, top, platform, and distro and GET ws/test/versions will return all as subsections of versions as described below. Using anything else as an extension will result in a failure indicating "Invalid URI Parameter Name".

Note Prior to Version 2.50, the use of the board, row, and slot options to filter apps results may result in incomplete response, crash of the internal process and subsequent restart of the server session. Workaround is for client to request all apps (no filter options), and for client to parse the desired result from that response.

Table 2.6
URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
session	Alternate way of specifying the logon session token id instead of using –H syntax
js	Format output using JSON standard

URI Argument	Description
	Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML

Note This command retrieves a large block of data and therefore takes an extended time (60 or more seconds in most cases) to complete in older builds. We recommend that browsers and client software that use these commands configure HTTP protocol timeouts to 4 minutes (240 seconds) or higher. HTTP timeout default of most commercial browsers (for example, Chrome) is typically set to 300 seconds (5 minutes) and it is not normally changed (to verify, check the Advanced Options tab in the browser). If the client software responds with HTTP 500 errors when attempting this API, increase the HTTP protocol timeout setting and try again.

Note In Version 3.75 or later, the response time for all data returned is 10 to 15 seconds, instead of the previous several minutes.

Intended Algorithm for Determining Current Application Version on unknown build:

1. Execute the GET ws/v2/test/versions/top API, and check for the HTTP Return Code. When using curl, HTTP Return code may be retrieved by adding the -i option to the curl command line.

   If this call returns HTTP_200 (OK), then the successful response will be formed as per the ws/v2/test/versions/top section below. In that case, if the http body response includes the "top" tag within the "versions" tag hierarchy, then the "version" tag (under "top" hierarchy) will contain the current app "version" string value.

   If that "version" tag string (when converted to a number) has a value that is greater than or equal to 4.25, then an additional "safeapp" tag should exist and the value of the "safeapp" tag corresponds to the top level safeapp platform available in the platform protected storage.

   Summary:

   — Since Version 3.75, the available ws/v2/test/versions/top API is the most efficient means to determine currently running Application version for the remote NTRP target

   — Since Version 4.25, the available ws/v2/test/versions/top API is also the most efficient means to determine the Safe Application for the remote NTRP target

   — Prior to Version 3.75, the ws/v2/test/versions/top API was not available and other follow up steps are needed to retrieve the currently running Application version for the remote receiver target.

2. If the call in Step 1 fails with either a HTTP_50x (where x = 0 to 4) HTTP return code or with a generic response failure (code=x, result=failure, message="detailed failure reason") then

proceed to Step 3. Otherwise, caller should already have obtained a "version" tag value and may exit.

3. If "version" was not retrieved via Step 1 and Step 2, the target code is running a version older than Version 3.50. Execute the deprecated ws/v2/test/versions/apps call with board=128&row=1&slot=0 as QueryParameters to retrieve the "rev" tag string value corresponding to the NTC Version "ident" tag. The combination of these three parameters only works for Version 3.75 or earlier.

> **Note** If Steps 2 and 3 were not adhered to and the software version is later than Version 3.75, then executing the GET ws/v2/test/apps API with all of board=128, row=1, and slot=0 specified as a Query Parameters filter with some newer releases will return unexpected results (more than the expected single row). In those cases, the command will return the expected single row result by limiting the Query Parameters to only specify "row=1" or "ident=NTC%20Version", but not both.

Examples (issue the ws/v2/test/versions command):

Input (get all version info):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions"
```

Expected output (Version 3.50 or earlier):

All version strings info will be returned in XML format by default or JSON format (when js Query Argument is used)

Expected output (Version 3.75 or later):

> **Note** Specifying apps or ntrp in the URI is now required)

```
<?xml version="1.0" encoding="ISO-8859-1"
?><versions><apps>…</apps><top>…</top><platform>…</platform></versions>
```

Retrieve All Software Versions (apps) Command

> **Note** Retrieving all software versions is time consuming and not recommended.

Table 2.7 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/test/versions/apps
Command	Reports Installed App and Third Party or Open Source Package versions.

Command Detail	Description
Information	See the *Intended Algorithm for determining Current Application Version(s) on unknown build* section in Versions Command, on page 26 before attempting this command.
HTTP Method	GET
Access Type	Read, Test command
Access Level	User, Admin
GET Syntax Examples	GET "https://192.168.0.1/ws/v2/test/versions/apps?row=1" GET "https://192.168.0.1/ws/v2/test/versions/apps?ident=NTC%20Version"

Table 2.8 URI Parameters (extension to the Command URL separated by /)

URI Argument	Description
apps	Displays all apps version info strings (for all software running on the target including third party and open source applications)

Note Prior to Version 2.50, use of the board, row, and slot options to filter apps results may result in incomplete response, crash of the internal process and subsequent restart of the server session. Workaround is for client to request all apps (no filter options), and for client to parse the desired result from that response.

Table 2.9 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
board (key)	Selects item based on board number (along with row (and slot for apps)) as assigned by the Application software. The value assigned to any specific board is subject to change as new hardware is supported (for example, in newer builds) and should not be relied on to always be a specific value. For example, although typically 128=NTC this is not (and was never) guaranteed to be its value for every future build. Type: Integer Values: 0..N Applies: apps Not recommended to be used prior to Version 2.50. Prior to Version 3.75, this key item is required to be specified in the command line when row, and slot are also specified. With Version 3.75 or later, this key is always shown in output but is no longer a mandatory input when a unique enhanced Query

URI Argument	Description
	Argument filter is used. See examples.
row (key)	Selects item based on row number (along with board and slot for apps). This value is an arbitrary ordinal to provide a unique index per board. It is dynamically assigned and is therefore not guaranteed to be the same consistent value for every build or boot. Type: Integer Values: 1..N Applies: apps Not recommended to be used prior to Version 2.50. Prior to Version 3.75, this key item is required to be specified in the command line when board, and slot are also specified. With Version 3.75 or later, this key is always shown in output but is no longer a mandatory input when a unique enhanced Query Argument filter is used. See examples.
slot (key)	Selects item based on slot number (along with board and row) Type: Integer Values: 0..15 Applies: apps Not recommended to be used prior to Release 2.50. Prior to Release 3.75, this key item is required to be specified in the command line when board, and row are also specified. With Release 3.75 or later, this key is always shown in output but is no longer a mandatory input when a unique enhanced Query Argument filter is used. See examples.
session	Alternate way of specifying the logon session token id instead of using –H syntax.
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Note Prior to Release 4.75, this command retrieves a large block of data and therefore takes an extended time (60 or more seconds in most cases) to complete in older builds . We recommend that browsers and client software that use these commands configure HTTP protocol timeouts to 4 minutes (240 seconds) or higher. HTTP timeout default of most commercial browsers (for example, Chrome) is typically set to 300 seconds (5 minutes) and should not normally need to be changed (check under advanced options tab in the browser to verify). If client software responds with HTTP 500 errors when attempting this API, increase the HTTP protocol timeout setting and try again.

Table 2.10 Output Field Descriptions

Output Field	Description
board (key)	Selects item based on board number (along with row and slot for apps) as assigned by the Application software. The value assigned to any specific board is subject to change as new hardware is supported (for example, in newer builds) and should not be relied on to always be a specific value. For example, although typically 128=NTC this is not (and was never) guaranteed to be its value for every future build. Type: Integer Value: 0 …. 255 Applies: apps
row (key)	Selects item based on row number (along with board and slot for apps). This value is an arbitrary ordinal to provide a unique index per board. It is dynamically assigned and is therefore not guaranteed to be the same consistent value for every build or boot. Since it is a key, this item is always presented in the output. Type: Integer Values: 1..N Applies: apps
ident	Unique identifier string for the software version. This item or value may be optionally matched or filtered out of the output via Query Parameters in the input call. Type: String Values: Each software item in the output has a unique ident name string Applies: apps
rev	Revision of the specific software app. This item or value may be optionally matched or filtered out of the output via Query Parameters in the input call. Type: String Values: Each software item in the output will have its own rev string Applies: apps

Output Field	Description
slot (key)	The reported ident and rev are specific to running on the processor hardware installed in this slot number (along with board and row). Since it is a key, this item is always presented in the output. Type: Integer Values: 0..15 Applies: apps

Examples (issue the ws/v2/test/versions/apps command):

Input (get all version info for all apps and third party or open source software):

```
curl -k -H "X-SESSION-ID: $token"
 "https://192.168.0.1/ws/v2/test/versions/apps"
```

Expected output (* actual output is very long so XML shown below is truncated to show only NTC Version and base files version). Values shown in output are for example purposes only:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<versions>
       <apps>
               <board>128</board>
               <row>1</row>
               <ident>NTC Version</ident>
               <rev>2.50</rev>
               <slot>0</slot>
       </apps>
       <apps>
               <board>0</board>
               <row>4</row>
               <ident>base-files</ident>
               <rev>3.0.14-r89.12</rev>
               <slot>0</slot>
       </apps>
</versions>
```

All version strings info will be returned in XML format by default (use &js to return data in JSON)

Examples (issue the ws/v2/test/versions/apps command with known key based filter):

Input (get all version info for app on specific card):

Note Requesting ws/v2/test/versions/apps with items after the ? (filter QueryParms) may crash the internal process and lose the session (due to process recovery restart) prior to Version 2.50. This command is therefore not recommended to be run with older software releases.

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions/apps?board=128&row=1&slot=0"
```

Expected output (values shown in output are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1"
?><versions><apps><board>128</board><row>1</row><slot>0</slot><ident>NTC
Version</ident><rev>2.50</rev></apps></versions>
```

All version strings info will be returned in XML format by default (use &js to return data in JSON).

Examples (issue the ws/v2/test/versions/apps command with enhanced QueryParms filter matching the unique ident(ifier) name instead of using known keys):

Input (get version info for specific app identifier):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions/apps?ident=NTC%20Version"
```

Expected output:

```
<?xml version="1.0" encoding="ISO-8859-1"
?><versions><apps><board>128</board><row>1</row><ident>NTC
Version</ident><rev>E3.75B1+DD6(3.51)</rev><slot>0</slot></apps></versions>
```

All version strings info will be returned in XML format by default (use &js to return data in JSON).

Examples (issue the ws/v2/test/versions/apps command with enhanced QueryParms filter matching the physical slot number):

Input (get all version infos for specific slot number):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions/apps?slot=0"
```

Expected output:

```
<?xml version="1.0" encoding="ISO-8859-1"
?><versions><apps><board>128</board><row>1</row><ident>NTC
Version</ident><rev>E3.75B1+DD6(3.51)</rev><slot>0</slot></apps></versions>
```

All version strings info will be returned in XML format by default (use &js to return data in JSON).

Retrieve Only Platform Specific Top Level Software Version Command

Note This command is supported in Version 3.75 or later.

Table 2.11 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/test/versions/top
Command Information	Reports top version for only currently running installed Apps and/or FPGA versions that are not from Third Parties or Open Source.
HTTP Method	GET
Access Type	Read, Test command
Access Level	User, Admin
GET Syntax Examples	GET "https://192.168.0.1/ws/v2/test/versions/top"

Table 2.12 URI Parameters (extension to the Command URL separated by /)

URI Argument	Description
top	Version 3.75 or later: Displays active running top level version string. Version 4.25 or later: Also displays top level safeapp version string.

Table 2.13 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
session	Alternate way of specifying the logon session token id instead of using –H syntax.
js	Format output using JSON standard. Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) **Note** Omitting this argument formats the output by default in XML.

Examples (issue the ws/v2/test/versions/top command):

Input (get top version info for platform):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions/top"
```

Expected output for Version 3.75 to Version 4.01 inclusive (value is for example purpose only):

```
<?xml version="1.0" encoding="ISO-8859-1"
?><versions><top><version>E3.75B1+DD6(3.51)</version></top></versions>
```

Expected output for Version 4.25 or later (values are for example purpose only):

```
<?xml version="1.0" encoding="ISO-8859-1"
?><versions><top><version>4.25</version><safeapp>3.01</safeapp></top></versio
ns>
```

Retrieve Only Platform Specific Software and/or FPGA Versions Command

Note This command is supported in Version 3.75 or later.

Table 2.14 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/test/versions/platform
Command Information	Reports only currently running installed Apps and/or FPGA versions that are not from Third Parties or Open Source
HTTP Method	GET
Access Type	Read, Test command
Access Level	User, Admin
GET Syntax Examples	GET "https://192.168.0.1/ws/v2/test/versions/platform" GET "https://192.168.0.1/ws/v2/test/versions/platform?slot=0&boardname=NTC&displayname&appversion" GET "https://192.168.0.1/ws/v2/test/versions/ntrp?slot=5&boardname&fpgaid&fpgaversion" GET "https://192.168.0.1/ws/v2/test/versions/platform?displayname=Tuner%201" GET "https://192.168.0.1/ws/v2/test/versions/platform?displayname&appversion"

Table 2.15 URI Parameters (extension to the Command URL separated by /)

URI Argument	Description
platform	Displays active running version info strings.

Table 2.16 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
boardname key (must be used along with slot)	Selects item based on board name (along with slot) as assigned by the boot firmware. Type: String Values: The available values depend on the installed hardware for the target system. These values correspond to the same values used in other test commands such as the GET /ws/v2/test/show or cardalive command: NBP, NTC, NFE, NCI, NDM, NTM, NTB Applies: platform When specified, the name of this item must be used in conjunction with the slot item name. Either or both keys may be used with a known specific value to further restrict/filter output. When filtering data with displayname, the slot and boardname is not required (and should not be specified).
slot key (must be used along with boardname)	Selects item based on slot number (along with boardname) Type: Integer Values: 0..15 Applies: platform When specified, the name of this item must be used in conjunction with the slot item name. Either or both keys may be used with a known specific value to further restrict/filter output. When filtering data with displayname, the slot and boardname is not required (and should not be specified).
displayname key (may be used instead of slot and boardname to filter output)	Unique identifier string for the friendly board displayname including its ordered instance identifier number (as used by WEBUI). Since this is a unique value for each board in slot, specifying a value for this item in the input Query Arguments may be used to return only a specific table row. When filtering via slot and boardname, the name of this item should NOT be included in the Query Arguments (it will always be included in the output data for the isolated table row(s)). Type: String

URI Argument	Description
	Values: Each board is identified by a friendly name by the software with an instance number assigned in the order that it is discovered consistent with the name displayed by the user interface. Applies: platform When filtering data with slot & boardname, the displayname is not required (and should not be specified).
session	Alternate way of specifying the logon session token id instead of using –H syntax
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML

In addition to the key items shown here, some combinations of the other output items in the following table (without values) may be used as Query Arguments on input to restrict the items included in the output.

Note Starting with Version 3.75, any combination of these element names (but without values) may also be specified as Query Argument(s) for enhanced filtering to limit output. The boardname and slot arguments are keys and thus when either are used both are needed to uniquely identify a processor on the system. Alternatively, displayname with a value may be used to uniquely identify the table row. Support for matching values for non-key items for this API is not supported. See examples.

Table 2.17 Output Field Descriptions

Output Field	Description
slot (key)	Selects item based on slot (along with boardname) to uniquely identify the provisioned hardware. Type: String Values: The available values depend on the installed hardware for the target system. These values correspond to the same values used in other test commands such as the GET /ws/v2/test/show or cardalive command. Since it is a key, this item is always presented in the output. Applies: platform This item will always be included in the output.
boardname (key)	Selects item based on board name (along with slot) as assigned by the boot firmware.

Output Field	Description
	Type: String Values: The available values depend on the installed hardware for the target system. These values correspond to the same values used in other test commands such as the GET /ws/v2/test/show or cardalive command. Since it is a key, this item is always presented in the output. Applies: platform This item will always be included in output.
displayname key (may be used instead of slot and boardname to filter output)	Unique identifier string for the friendly board displayname including its ordered instance identifier number (as used by WEBUI). Since this is a unique value for each board in slot, specifying a value for this item in the input Query Arguments may be used to return only the associated table row. When filtering via slot and boardname, this item's name should NOT be included in the Query Arguments (it will always be included in the output data for the isolated table row(s)). Type: String Values: Each board is identified by a friendly name by the software with an instance number assigned in the order that it is discovered consistent with the name displayed by the user interface. Applies: platform This item will always be included in output.
boardgen	Generation string for the unique board in slot as set by manufacturing in the non-volatile storage of the hardware. When filtering via slot and boardname or displayname, the name of this item (but not any value) must be specifically included in the Query Parameters in order to be included in the output call. Type: String Values: An example boardgen value would be "1G" or "2G" Applies: platform
appversion	Version of the platform application that is running on the main processor that controls the hardware installed in this board in slot . When filtering via slot and boardname or displayname, the name of this item (but not any value) must be specifically included in the Query Parameters in order to be included in the output call. Type: String Applies: platform
fpgaid	Identifier of the fpga image type that is installed in this board in slot.

Output Field	Description
	When filtering using the slot and boardname or displayname, the name of this item (but not any value) must be specifically included in the Query Parameters in order to be included in the output call. Type: String Applies: platform
fpgaversion	Version of the fpga image that is running on the hardware installed in this board in slot. When filtering via slot and boardname or displayname, this item's name (but not any value) must be specifically included in the Query Parameters in order to be included in the output call. Type: String Applies: platform
boltrev	Revision of the BOLT firmware used to aid booting of the application software on Broadcom processors. Applies: Expected to return data when boardname=NDM or boardname=NTB. Otherwise, this field will read back a value of "Unavailable".
ubootrev	Revision of the UBOOT firmware used to aid booting of the application software. Applies: Expected to return data for all boardname(s) that include processors that have booted application software: NTC, NDM, NTM, NTB. Otherwise, this field will read back a value of "Unavailable".
distrorev	Revision of the Linux distribution that is currently running on the processor. Applies: Expected to return data for all boardname(s) that include processors that have booted application software: NTC, NDM, NTM, NTB. Otherwise, this field will read back a value of "Unavailable".
distromin	Revision of the Linux distribution that is the minimum compatible revision suitable for running the current application version on this processor. If the distrorev does not meet the requirements of the distromin string, it may be a cause for unexpected behavior of some distro dependent software features. Applies: Expected to return data for all boardname(s) that include processors that have booted application software: NTC, NDM, NTM, NTB. Otherwise, this field will read back a value of "Unavailable".

Note The web UI may drop the instance number from the displayname when referring to a single instance of a populated card type on its versions page. Such is the case for "Tuner%201" where web UI refers to it as "Tuner".

Table 2.18 Available values of displayname

Value	Description
System	Applies: All Platforms This name is the friendly display name for the main controller card (NTC type for the PowerVu Professional Receiver) in the system. It corresponds to the processor card that is installed in slot 0 of the chassis.
Chassis%20Back%20Plane	Applies: All Platforms This name is the friendly display name for the chassis (NBP type) in the system that may connect other cards. It does not have any associated runnable firmware (fpgaversion) or software (appversion) but does typically have a boardgen value.
Tuner%201	Applies: Platform with Tuner support This name is derived from the detection of the NFE type card populated in the order in which the slots are discovered. Currently there is support for only one tuner card, so it will always have a displayname of "Tuner%201". It may have its own fpga and the associated software version is typically the main System controller card.
Transcoder%201 Transcoder%202	Applies: Multi-stream platform with Transcoding support This name is derived from the detection of the NTM type card populated in the order in which the slots are discovered. So if the first NTM card is populated in slot 3 and the next NTM card is populated in slot 4, then the displayname for the card in slot 3 will be "Transcoder %201" whereas the displayname for the card in slot 4 will be "Transcoder%202".
HEVC%20 Preprocessor%201 HEVC%20 Preprocessor%202	Applies: Multi-stream platform with HEVC Transcoding support This name is derived from the detection of the NTB type card populated in the order in which the slots are discovered. So if the first NTB card is populated in slot 2 and the next NTB card is populated in slot 5, then the displayname for the card in slot 2 will be "HEVC%20 Preprocessor%201" whereas the displayname for the card in slot 5 will be "HEVC%20 Preprocessor%202".
CI%20Option	Applies: Platform with CI capability This name is the friendly display name for the NCI type card in the system that may be populated in the Front Panel CI Socket. It may have its own FPGA and the associated software version is typically the main System controller card.
Decoder	Applies: Platform with Decoder capability This name is the friendly display name for the NDM type card in the system. Currently there is only support for one Decoder card instance.

Examples (issue the ws/v2/test/versions/platform command with the slot and boardname keys):

Input (get version info for specific slot):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions/platform?
slot=2&boardname&boardgen&appversion&fpgaid&fpgaversion"
```

Expected output:

```
<?xml version="1.0" encoding="ISO-8859-1"
?><versions><platform><slot>2</slot><boardname>NTB</boardname><displayname>HE
VC Preprocessor 1</displayname><appversion>E3.75B1+DD6
(3.51)</appversion><fpgaid>NTB-
F10</fpgaid><fpgaversion>R0.0.3</fpgaversion></platform></versions>
```

All version strings info will be returned in XML format by default (use &js to return data in JSON)

Examples (issue the ws/v2/test/versions/platform command with enhanced QueryParms filter matching the unique displayname instead of using the slot and boardname keys):

Input (get version info for specific displayname):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions/platform?displayname=HEVC%20Preproce
ssor%201"
```

Expected output:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<versions><platform>
<slot>2</slot>
<boardname>NTB</boardname>
<displayname>HEVC Preprocessor 1</displayname>
<appversion>E3.75B1+DD6(3.51)</appversion>
</platform></versions>
```

All version strings information will be returned in XML format by default (use &js to return data in JSON).

Retrieve Only Distro Specific Software Versions Command

Note This command is supported in Version 4.75 or later. Prior to Version 4.75, the distro related rows were returned along with the apps related rows when calling the GET ws/v2/test/versions/apps API. Beginning with Version 4.75, the distro related rows are no longer returned with the GET ws/v2/test/versions/apps API and are instead available by calling the new GET ws/v2/test/versions/distro API.

Table 2.19 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/test/versions/distro
Command Information	Reports Third Party or Open Source Package versions
HTTP Method	GET
Access Type	Read, Test command
Access Level	User, Admin
GET Syntax Examples	GET "https://192.168.0.1/ws/v2/test/versions/distro" GET "https://192.168.0.1/ws/v2/test/versions/distro?board=128&row=1&slot=0" GET "https://192.168.0.1/ws/v2/test/versions/distro?row=1" GET "https://192.168.0.1/ws/v2/test/versions/distro?ident=yajl"

Table 2.20 URI Parameters (extension to the Command URL separated by /)

URI Argument	Description
distro	Displays all apps version info strings (for all software running on the target including third party and open source applications).

Note In addition to the key items shown in the table below, some combinations of the other output items (without values) may be used as Query Arguments on input to restrict the items included in the output.

Table 2.21 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
board (key)	Selects item based on board number (along with row (and slot for apps)) as assigned by the Application software. The value assigned to any specific board is subject to change as new hardware is supported (for example, in newer builds) and should not be relied on to always be a specific value. For example, although typically 128=NTC this is not (and was never) guaranteed to be its value for every future build. Type: Integer Values: 0..N Applies: distro This key is always shown in output but is no longer a mandatory input when a unique enhanced Query Argument filter is used. See examples.

URI Argument	Description
row (key)	Selects item based on row number (along with board (and slot for apps)). This value is an arbitrary ordinal to provide a unique index per board. It is dynamically assigned and is therefore not guaranteed to be the same consistent value for every build or boot. Type: Integer Values: 1..N Applies: distro This key is always shown in output but is no longer a mandatory input when a unique enhanced Query Argument filter is used. See examples.
slot (key)	Selects item based on slot number (along with board and row). Type: Integer Values: 0..15 Applies: distro This key is always shown in output but is no longer a mandatory input when a unique enhanced Query Argument filter is used. See examples.
session	Alternate way of specifying the logon session token id instead of using –H syntax.
js	Format output using JSON standard. Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Note Any combination of the element names below (and optionally specific values for one or more of them) may also be specified as Query Argument(s) for enhanced filtering to limit output in lieu of using any previously cached values of the board, row, and slot keys. See examples.

Table 2.22 Output Field Descriptions

Output Field	Description
board (key)	Selects item based on board number (along with row and slot for apps) as assigned by the Application software. The value assigned to any specific board is subject to change as new hardware is supported (for example, in newer builds) and should not be relied on to always be a specific value. For example, typically 128=NTC is not (and was never) guaranteed to be its value for every future build. Type: Integer Value: 0 …. 255

Output Field	Description
	Applies: distro
row (key)	Selects item based on row number (along with board and slot for apps). This value is an arbitrary ordinal to provide a unique index per board. It is dynamically assigned and is therefore not guaranteed to be the same consistent value for every build or boot. Since it is a key, this item is always presented in the output. Type: Integer Values: 1..N Applies: distro
ident	Unique identifier string for the software version. This item or value may be optionally matched or filtered out of the output via Query Parameters in the input call. Type: String Values: Each software item in the output has a unique ident name string Applies: distro
rev	Revision of the specific software app. This item or value may be optionally matched or filtered out of the output via Query Parameters in the input call. Type: String Values: Each software item in the output will have its own rev string Applies: distro
slot key	The reported ident and rev are specific to running on the processor hardware installed in this slot number (along with board and row). Since it is a key, this item is always presented in the output. Type: Integer Values: 0..15 Applies: distro

Examples (issue the ws/v2/test/versions/distro command):

Input (get all version info for all apps and third party or open source software):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions/distro"
```

Expected output (* actual output is very long so XML shown below is truncated to show only base files version). Values shown in output are for example purposes only:

```
<?xml version="1.0" encoding="UTF-8"?>
<versions>
  <distro>
     <board>128</board>
     <row>3</row>
     <ident>base-files</ident>
     <rev>3.0.14-r89.18</rev>
     <slot>0</slot>
  </distro>
</versions>
```

All versions strings info will be returned in XML format by default (use &js to return data in JSON).

Examples (issue the ws/v2/test/versions/distro command with known key based filter):

Input (get all version info for app on specific card):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions/distro?board=128&row=3&slot=0"
```

Expected output (values shown in output are for example purposes only):

```
<?xml version="1.0" encoding="UTF-8"?>
<versions>
   <distro>
      <board>128</board>
      <row>3</row>
      <ident>base-files</ident>
      <rev>3.0.14-r89.18</rev>
      <slot>0</slot>
   </distro>
</versions>
```

All versions strings info will be returned in XML format by default (use &js to return data in JSON)

Examples (issue the ws/v2/test/versions/distro command with enhanced QueryParms filter matching the unique ident(ifier) name instead of using known keys):

Input (get version info for specific app identifier):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions/distro?ident=yajl"
```

Expected output:

```
<?xml version="1.0" encoding="UTF-8"?>
<versions>
<distro>
```

```
<board>128</board>
<row>328</row>
<ident>yajl</ident>
<rev>2.1.0-r6</rev>
<slot>0</slot>
</distro>
<distro>
<board>135</board>
<row>636</row>
<ident>yajl</ident>
<rev>2.1.0-r6</rev>
<slot>2</slot>
</distro>
<distro>
<board>134</board>
<row>945</row>
<ident>yajl</ident>
<rev>2.1.0-r6</rev>
<slot>3</slot>
</distro>
<distro>
<board>134</board>
<row>1254</row>
<ident>yajl</ident>
<rev>2.1.0-r6</rev>
<slot>4</slot>
</distro>
<distro>
<board>135</board>
<row>1562</row>
<ident>yajl</ident>
<rev>2.1.0-r6</rev>
<slot>5</slot>
</distro>
</versions>
```

All versions strings info will be returned in XML format by default (use &js to return data in JSON).

Examples (issue the ws/v2/test/versions/distro command with enhanced QueryParms filter matching the physical slot number). Output truncated to preserve space in document:

Input (get all distro version infos for specific slot number):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/test/versions/distro?slot=0"
```

Expected output:

```
<?xml version="1.0" encoding="UTF-8"?>
<versions>
<distro>
<board>128</board>
<row>1</row>
<ident>avahi-daemon</ident>
<rev>0.6.32-r0.5</rev>
<slot>0</slot>
</distro>
<distro>
<board>128</board>
<row>2</row>
<ident>avahi-utils</ident>
<rev>0.6.32-r0.5</rev>
<slot>0</slot>
</distro>
<distro>
<board>128</board>
<row>3</row>
<ident>base-files</ident>
<rev>3.0.14-r89.18</rev>
<slot>0</slot>
</distro>
<distro>
<board>128</board>
<row>4</row>
<ident>base-passwd</ident>
<rev>3.5.29-r0</rev>
<slot>0</slot>
</distro>
</versions>
```

All version strings info will be returned in XML format by default (use &js to return data in JSON).

Download User Defaults (Customer Presets) Command

Table 2.23 Command Details

Command Detail	Description
Command URL	-F restorefile=@"ud_file_path/ud_file_name" "https://192.168.0.1/restore_upload.html"
Command Information	Download customer presets (user defaults) onto the device. This command is a variation of the Restore command. The XML format of the ud_file_name determines whether to change the user defaults data. To make the user defaults data persistent, this command must be followed by a Factory Reset command (see Reset Device Control Command, on page 484).

Command Detail	Description
HTTP Method	POST
Access Type	Write
Access Level	Guest, User, Admin

Options: N/A

Example of issuing the restore user defaults command (assuming the filename is UD9800.bkp):

Input:

```
curl -k -i –H "X-SESSION-ID: $token" -X POST -F
restorefile=@"C:/projects/d9800/UD9800.bkp" "https://192.168.0.1/restore_
upload.html"
```

Expected output:

```
<html><body></body></html>
```

The XML response is immediate and does not indicate that the restore command operation is completed.

The progress and results of the Restore command may be determined by calling the BKPRSTSTAT command every few seconds until it reports that the file restore is completed at 100 percent, or that it was aborted due to any possible error.

This operation may take over one minute, depending on the amount of data to validate and restore.

BKPRST Status Command

Table 2.24 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v1/table?t=BKPRSTSTAT
Command Information	Return information about Backup/Restore Status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin

Example of issuing the BKPRST status command:

Input:

```
curl -k -H "X-SESSION-ID: $token"
https://192.168.0.1/ws/v1/table?t=BKPRSTSTAT
```

Expected output (example only, subject to change):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><HDR><TABLE><RECORD><ITEM>
<ID>0x002D1247</ID><VALUE><![CDATA[file name]]></VALUE>
<HELP_STR_1><![CDATA[Filename of the last backup performed]]></HELP_STR_1>
<HELP_STR_2><![CDATA[This text field can hold 119 characters.]]></HELP_STR_
2></ITEM>
<ITEM><ID>0x002D1248</ID><VALUE><![CDATA[2007/11/17 21:06:53]]></VALUE>
<HELP_STR_1><![CDATA[Date and time of the last backup performed]]></HELP_STR_
1>
<HELP_STR_2><![CDATA[Item attribute: min(), max(2900/12/31
23:59:59).]]></HELP_STR_2></ITEM>
<ITEM><ID>0x002D1249</ID><VALUE><![CDATA[file name]]></VALUE>
<HELP_STR_1><![CDATA[Filename of the last restore performed]]></HELP_STR_1>
<HELP_STR_2><![CDATA[This text field can hold 119 characters.]]></HELP_STR_
2></ITEM>
<ITEM><ID>0x002D124A</ID><VALUE><![CDATA[2007/09/10 01:57:23]]></VALUE>
<HELP_STR_1><![CDATA[Date and time of the last restore performed]]></HELP_
STR_1>
<HELP_STR_2><![CDATA[Item attribute: min(), max(2900/12/31
23:59:59).]]></HELP_STR_2></ITEM>
<ITEM><ID>0x002D124B</ID><VALUE><![CDATA[Pass]]></VALUE>
<HELP_STR_1><![CDATA[Indicates whether the most recent operation passed,
failed,
or is in progress]]></HELP_STR_1>
<HELP_STR_2><![CDATA[This item has 3 options available.]]></HELP_STR_
2></ITEM>
<ITEM><ID>0x002D124C</ID><VALUE><![CDATA[Idle]]></VALUE>
<HELP_STR_1><![CDATA[State of the current backup or restore
operation]]></HELP_STR_1>
<HELP_STR_2><![CDATA[This item has 21 options available.]]></HELP_STR_
2></ITEM>
<ITEM><ID>0x002D124D</ID><VALUE><![CDATA[0]]></VALUE>
<HELP_STR_1><![CDATA[Progress of the current backup or restore operation as a
percent]]></HELP_STR_1>
<HELP_STR_2><![CDATA[Item attribute: min(0), max(100), step size(1), unit
()]]></HELP_STR_2>
</ITEM></RECORD></TABLE></HDR>
```

Note See the notes in the backup and restore commands for more information on intended use. See Backup Command, on page 24 and Restore Command, on page 25.

Legacy CDT Upload Command

Table 2.25 Command Details

Command Detail	Description
Command URL	-F cdtfile=@"cdt_file_path/cdt_file_name" "https://192.168.0.1/ws/v1/cdt_upload.html"
Command Information	Upload a CDT file to the device.
HTTP Method	POST
Access Type	Write
Access Level	Guest, User, Admin

Example of issuing the upload CDT file command (assuming the filename is NTRP.cdt):

Input:

```
curl -k -X POST -H "X-SESSION-ID: $token" -F cdtfile=@"C:/myfiles/NTRP.cdt"
"https://192.168.0.1/cdt_upload.html?cdtfile=NTRP.cdt"
```

Expected output:

```
<html><body></body></html>
```

Chunky CDT Upload Command

Note Due to memory constraints, in Version 2.75 or later, the traditional single command line curl variant of the CDT Upload command may fail with some builds. The following describes the support for "ChunkyDL" which requires the Client software to break up the source file into more manageable chunks, and issue sequential POST commands to achieve the full CDT File Upload without incurring significant memory allocation on the target.

Curl command line only supports the –F option to send multi-part HTML form body content. The command line does not have a built-in method of streaming single file content in chunks of data. The –F option puts the entire file into a single buffer, before sending the content out in a single POST operation.

Note Unlike bash scripting with the Curl command line, client software that uses libcurl native C library, Java, Python bindings, PHP, or Java Script does not have the limitation or requirement to split the source file up into many chunks. In these cases, the language support allows using byte array buffers and HTML header creation methods to format and send the chunked multi-part HTML form POST commands.

In order to overcome the command line limitation, client software which runs from a bash shell may use a bash shell based script to implement the following algorithm:

- Source file must be broken up into many files, with a maximum size of 10 MB (10*1024*1024) each. This requirement may be met by pre-processing the source cdt file with the bash shell split command. The "split.exe" command is available and installed on Windows client systems, using the default cygwin64 distribution.
- Each POST operation maintains the same –F file command line syntax, with the addition of a new header that must be patched by the client script for each POST call. This header has three inputs per chunk (for example, POST command): start offset byte (within the source file), end offset byte (included in the size of the current chunk), and total source file size (in bytes).
- Multi-POST command sequencing uses this new Content-Range header to be added to the command line syntax that indicates to the target that the received "chunk" is one of many. The Content-Range header is added to the command line with the –H option.
- Override of the default Content-Length header using –H option may be required to compensate for internal MIME boundary overhead (182 bytes in addition to range size), within the body of each transmitted chunk.

Table 2.26 Command Details

Command Detail	Description

Algorithm and Commands
 	
This description uses pseudo code to illustrate logic requirements. A working bash script will be available from Engineering, prior to Version 2.75.
	Determine support for Chunky Download feature. If not supported, revert to use the legacy CDT upload command. For more information, see Legacy CDT Upload Command, on page 51.
	
	`mkdir split_cdt_files_path`
	`cd split_cdt_files_path`
	`cp cdt_files_path/cdt_file_name`
	`split -d -b 10485760 cdt_file_name`
	
	result will be files x00, x01, x02, and so on.
	
	`$src_file-size=size of cdt_file_name contents (in bytes)`
	`$start_offset=0`
	
	for each file x## in split_cdt_files_path
	
	`{`
	`$end_offset=$start_offset+(size of x## file in bytes – 1)`
	`$compensated_content_length=182 + ($end_offset + 1) – $start_offset`
	
	`POST`

Command Detail	Description
	`-H "Content-Range: [start_offset]-[end_offset]/[src_file_size]" -H "Content-Length: [compensated-content-length] -F cdt file=@"split_cdt_files_path/current_chunk_file_name" "https://192.168.0.1/ws/v1/cdt_upload.html"` where a new POST command is formatted and called for each file chunk. `$start_offset=$end_offset+1 }` If using chunky method from command line (requires bash script), contact Synamedia Customer Service Desk for a working bash script example. Synamedia Engineering or Test personnel may use internal downloader tools employing one or more high level languages that do not have the same restrictions as the command line syntax requirements described here. Those languages support source file streaming into a buffer, and the contents of the buffer is an input to the client application specific HTTP(S) POST API implementation.
Command Information	Upload a CDT file to the device using Chunky algorithm.
HTTP Method	POST
Access Type	Write
Access Level	Guest, User, Admin

Example of the POST content that needs to be sent for each chunk is as follows:

```
POST / HTTP/1.1..Host: 10.1.1.1\r\n
  Accept: */*\r\n
  X-SESSION-ID: 1234567890\r\n // actual value of logon sessionid would be
here
  Content-Range: bytes 0-10485759/12659933\r\n // different for each chunk
  Content-Length: 10485942\r\n  // range end_ size plus fixed 182 byte
overhead
  Content-Type: multipart/form-data;boundary=----------------------------
9d4ffbe5a6c7\r\n\r\n
  ------------------------------9d4ffbe5a6c7\r\n
```

```
Content-Disposition: form-data; name="cdtfile"\r\n
Content-Type: application/octet-stream\r\n\r\n
a lot of binary data where ((end_chunk + 1) - start_chunk) is
number of bytes size (ie. Content-Range is inclusive)
\r\n-----------------------------9d4ffbe5a6c7--\r\n
```

Determining Support for Chunky CDT Upload Command

Table 2.27 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/device_ctl/download
Command Information	Show information about download command.
HTTP Method	GET, POST
Access Type	Read, Write
Access Level	Admin

Table 2.28 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
dlmode (optional)	Filter output based on the dlmode value. Type: String Values: Always (default), Never, Once Select how to handle unforced over-air downloads: ■ Always - Accept download, but not reboot. ■ Once - Accept one download then reboot and change the mode to Never. ■ Never - Reject unforced over-air downloads. A POST change of this setting will be persistent in non-volatile memory (after reboots). A Factory Reset will reset this setting to the default value. **Note** We recommend that you do not change this setting from its default value.
dlhttpchunk (optional) (Read Only)	Inform external tool or curl based command line script if CDT upload should use the multi-chunk algorithm. Type: String Values: Enabled (default), Disabled (for testing only)

URI Argument	Description
	This value is read-only and intended only for use by specific internal tools in Synamedia test and manufacturing. It is not used by the PowerVu Professional Receiver web UI, which is integrated within the target application software during fie upload.

Example (Retrieve Download Feature support using Version 2.75 or earlier):

Input:

```
curl -i -k -H "X-SESSION-ID: $token" "http://192.168.0.1/ws/v2/device_
ctl/download"
```

Expected output (full HTTP response must be checked, where 200 indicates that command is implemented and all other numbers indicate older software, where command is not implemented.)

```
HTTP/1.1 404 Not Found
Date: Mon, 25 Sep 2017 13:30:40 GMT
Server: Hiawatha v10.3
Connection: keep-alive
Transfer-Encoding: chunked
```

In this case, we always expect the client software to use the Legacy CDT Upload command.

Example (Retrieve Download Feature support using Version 2.75 or later):

Input (XML case):

```
curl -i -k -H "X-SESSION-ID: $token" "http://192.168.0.1/ws/v2/device_
ctl/download"
```

Expected output (Full HTTP response must be checked for 200 OK):

```
HTTP/1.1 200 OK
Date: Mon, 25 Sep 2017 13:33:04 GMT
Server: Hiawatha v10.3
Connection: keep-alive
Transfer-Encoding: chunked
Content-type: application/xml

<?xml version="1.0" encoding="ISO-8859-1" ?><devicectl><download><settings>

<dlmode>Always</dlmode><dlhttpchunk>Enabled</dlhttpchunk></settings></downloa
d></devicectl>
```

Input (JSON case):

```
curl -i -k -H "X-SESSION-ID: $token" "http://192.168.0.1/ws/v2/device_
ctl/download&js"
```

Expected output (Full HTTP response must be checked for 200 OK):

```
HTTP/1.1 200 OK
Date: Mon, 25 Sep 2017 13:39:30 GMT
Server: Hiawatha v10.3
Connection: keep-alive
Transfer-Encoding: chunked
Content-type: application/json

{
    "devicectl": {
        "download": {
            "settings": {
                "dlmode": "Always",
                "dlhttpchunk": "Enabled"
            }
        }
    }
}
```

Logic for interpreting the command response:

```
If the HTTP response is 200 (OK):
{
the client software must check the dlhttpchunk value.
    if the dlhttpchunk value is Enabled
    {
           the client code may use the Chunky CDT Upload command.
    }
    else
    {
           the client software must use the Legacy CDT Upload command.
    }
}
else
{
    the client software must use the Legacy CDT Upload command.
}
```

Upload Software License File Command

Table 2.29 Command Details

Command Detail	Description
Command URL	-F cdtfile=@"lic_file_path/lic_file_name" "https://192.168.0.1/cdt_upload.html"
Command Information	Software license files can be uploaded to the unit, the same as a CDT file.
HTTP Method	POST
Access Type	Write
Access Level	Guest, User, Admin

Example of issuing the upload license file command (assuming the filename is D9800_lic.cdt):

Input:

```
curl -k -X POST -H "X-SESSION-ID: $token" -F cdtfile=@"C:/myfiles/D9800_
lic.cdt"
"https://192.168.0.1/cdt_upload.html?cdtfile=D9800_lic.cdt"
```

Expected output:

```
<html><body></body></html>
```

Diagnostics Package (Export Debug File) Commands

Client Software Requirements:

Diagnostics package preparation may take several minutes, depending on hardware configuration, licensed features, and log complexity. You can no longer use one command to perform a request, wait for the results of the package generation, and then perform a file transfer.

In Version 2.50 or higher, the required client software operation is as follows:

1. Trigger a request for a diagnostic package creation (Diagnostics Trigger Command).
2. Wait for the status (Diagnostics Operation Status API).
3. Transfer the file (Diagnostics Retrieval command).

Diagnostics Package Trigger Command

Table 2.30 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v1/diagnostics/trigger
Command Information	Generate the diagnostics package file.
HTTP Method	POST

Command Detail	Description
Access Type	Write
Access Level	User, Admin Restricted to lock level 0 only.

Options: N/A

Example of issuing the diagnostics trigger command:

Input:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v1/diagnostics/trigger"
```

Expected output (for success case, which indicates only the specified Diag Package Preparation was started):

```
<?xml version="1.0" encoding="ISO-8859-1"
?><SESSIONINFO><STATUS>Success</STATUS></SESSIONINFO>
```

Expected output (for fail case, which indicates that the Diag Package Preparation was not started):

Note You may request additional failure reason details by using the DIAGSTAT API.

```
<?xml version="1.0" encoding="ISO-8859-1" ?><SESSIONINFO><STATUS>response
failure, rc=192512</STATUS></SESSIONINFO>
```

Diagnostics Package Operation Status Command

Table 2.31 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v1/table?t=DIAGSTAT
Command Information	Return information about Export Debug File (Diagnostics Package preparation) status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin

Example of issuing the DIAG Status command:

Input:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v1/table?t=DIAGSTAT"
```

Expected output (example of when diagnostics package preparation is in progress):

```
<?xml version="1.0" encoding="ISO-8859-1"
 ?><HDR><TABLE><RECORD><ITEM><ID>0x002D25CD</ID><VALUE><![CDATA
[Inprogress]]></VALUE><N
AME><![CDATA[OPERSTATUS]]></NAME><HELP_STR_1><![CDATA[Indicates whether the
most
 recent operation passed, failed, or is in
 progress]]></HELP_STR_1><HELP_STR_2><![CDATA[This item has 3 options
 available.]]></HELP_STR_2></ITEM><ITEM><ID>0x002D25CE</ID><VALUE><![CDATA
[Processing]]
></VALUE><NAME><![CDATA[DETAILEDSTATUS]]></NAME><HELP_STR_1><![CDATA[State of
the
 current Export Debug Support Data operation]]></HELP_STR_1><HELP_STR_2><!
[CDATA[This
 item has 6 options
 available.]]></HELP_STR_2></ITEM><ITEM><ID>0x002D25CF</ID><VALUE><![CDATA
[43]]></VALUE
><NAME><![CDATA[PERCENTCOMPLETE]]></NAME><HELP_STR_1><![CDATA[Progress of the
current
 Export Debug Support Data operation as a
percent]]></HELP_STR_1><HELP_STR_2><![CDATA[Item attribute: min(0), max(100),
step
 size(1), unit()]]></HELP_STR_2></ITEM></RECORD></TABLE></HDR>
```

Expected output (example of when diagnostics package preparation is 100% completed successfully):

```
<?xml version="1.0" encoding="ISO-8859-1"
 ?><HDR><TABLE><RECORD><ITEM><ID>0x002D25CD</ID><VALUE><![CDATA
[Pass]]></VALUE><NAME><!
[CDATA[OPERSTATUS]]></NAME><HELP_STR_1><![CDATA[Indicates whether the most
recent
 operation passed, failed, or is in progress]]></HELP_STR_1><HELP_STR_2><!
[CDATA[This
 item has 3 options available.]]></HELP_STR_
2></ITEM><ITEM><ID>0x002D25CE</ID><VALUE><![CDATA[Done]]>
</VALUE><NAME><![CDATA[DETAILEDSTATUS]]></NAME><HELP_STR_1><![CDATA[State of
the current
Export Debug Support Data operation]]></HELP_STR_1><HELP_STR_2><![CDATA[This
item has
6 options available.]]></HELP_STR_2></ITEM><ITEM><ID>0x002D25CF</ID><VALUE><!
[CDATA[100]]>
</VALUE><NAME><![CDATA[PERCENTCOMPLETE]]></NAME><HELP_STR_1><![CDATA[Progress
of the
current Export Debug Support Data operation as a percent]]></HELP_STR_
1><HELP_STR_2>
<![CDATA[Item attribute: min(0), max(100), step size(1), unit()]]></HELP_STR_
2></ITEM>
</RECORD></TABLE></HDR>
```

Expected output (example of when diagnostics package preparation fails, and detailed reason why it failed):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><HDR><TABLE><RECORD><ITEM>
<ID>0x002D25CD</ID><VALUE><![CDATA[Fail]]></VALUE><NAME><![CDATA
[OPERSTATUS]]>
</NAME><HELP_STR_1><![CDATA[Indicates whether the most recent operation
passed,
failed, or is in progress]]></HELP_STR_1><HELP_STR_2><![CDATA[This item has 3
options available.]]></HELP_STR_2></ITEM><ITEM><ID>0x002D25CE</ID><VALUE>
<![CDATA[Lock level too high for Export Debug Support Data
operation]]></VALUE><NAME>
<![CDATA[DETAILEDSTATUS]]></NAME><HELP_STR_1><![CDATA[State of the current
Export
Debug Support Data operation]]></HELP_STR_1><HELP_STR_2><![CDATA[This item
has 6
options available.]]></HELP_STR_2></ITEM><ITEM><ID>0x002D25CF</ID><VALUE><!
[CDATA[20]]>
</VALUE><NAME><![CDATA[PERCENTCOMPLETE]]></NAME><HELP_STR_1><![CDATA[Progress
of the
current Export Debug Support Data operation as a percent]]></HELP_STR_
1><HELP_STR_2>
<![CDATA[Item attribute: min(0), max(100), step size(1), unit()]]></HELP_STR_
2></ITEM>
</RECORD></TABLE></HDR>
```

Client software is expected to poll the status by calling this API every 5 seconds, while the Inprogress state is active, and until the OPERSTATUS is Pass and the PERCENTCOMPLETE is 100 or the OPERSTATUS is Fail. The first call should wait a minimum of 10 seconds, after the diagnostics trigger API, to allow the internal states to clear. If, at any time, the OPERSTATUS is Fail, then the client software should abort additional checking and report the last DETAILEDSTATUS.

Diagnostics Package Retrieval Command

Table 2.32 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v1/diagnostics
Command Information	Retrieves the diagnostics package file.

Command Detail	Description
HTTP Method	GET
Access Type	Read
Access Level	User, Admin Restricted to Lock Level 0 only.

Options: NA

Example of issuing the diagnostics package retrieval command:

Input:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v1/diagnostics" -o
diagnostics
```

Expected output (example for success case, which indicates that Diag Package Retrieval was completed):

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 5599k    0 5599k    0     0  4691k      0 --:--:--  0:00:01 --:--:--
4693k


File name for retrieved file will be as per the –o parameter string (eg. file
is
called "diagnostics" in this example) in the command invokation.
```

Expected output (example for fail case, which indicates that Diag Package Retrieval could not be completed):

Note Additional failure reason details may be determined by calling the DIAGSTAT API.

```
<?xml version="1.0" encoding="ISO-8859-1" ?><SESSIONINFO><STATUS>response
failure,
rc=192512</STATUS></SESSIONINFO>
```

Status API

This section describes the commands used to obtain input, output, and decode status information on the receiver.

RF Input Status Command

Table 2.33 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/input/rf
Command Information	Return information about RF input.
HTTP Method	GET
Access Type	Read
Access Level	Guest, User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/input/rf?js=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.34 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
port (Key)	Input port status to be responded. Type: Integer Values: 1..16 [NFE1 : 1..4, NFE2: 5..8, NFE3: 9..12, NFE4: 12..16] **Note** This does not refer to the NTC Ethernet ports, it refers specifically to the RF ports.
stream (Key) Optional	Multi Input Path Stream ID for RF. If not specified, default value of 1 is used. Type: Integer Values: 1 (D9800 Single-Stream), 1..6 (D9800 Multi-Stream)
js	Format output using JSON standard. Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.35 Output Field Descriptions

Output Field	Description
card	NFE Card number Type: Integer

Output Field	Description
	Value: 1 Default: 1
port	Input Port status to be responded Type: Integer Values: 1..4 (For R1) 1..16 [NFE1 : 1..4, NFE2: 5..8, NFE3: 9..12, NFE4: 12..16] (Post R1) **Note** This does not refer to the NTC Ethernet ports, it refers specifically to the RF ports.
stream	Multi Input Path Stream ID. If not specified, default value of 1 is used (For R1, only 1 is valid) Type: Integer Values: 1..6
act	Input is activated Type: String Values: "Yes" or "No" When Yes, the values of all other items in the response for the current row may be considered to be valid. When No, the values of all other items in the response for the current row may not be considered to be valid. We highly recommend you to always specify the name of this item in GET requests that use Query Strings for filtering (limiting) the output response.
dnlkfreq	Downlink Frequency Type: Float Values: 0.0..15.0
symrate	Symbol Rate Type: Float Values: 1.0..45.0
fec	FEC Mode Type: String Values: "Auto", "1/2", "2/3", "3/4", "5/6", "7/8", "8/9"
mod	Modulation Type Type: String

Output Field	Description
	Values: "N/A", "QPSK DVB-S", “QPSK DVB-S2", "8PSK DVB-S2", "16APSK DVB-S2", "32APSK DVB-S2" The following additional values are available when using hardware option (NFE-2G) that supports DVB-S2X (also requires Release 3.51 or later): "QPSK DVB-S2X", "8PSK DVB-S2X", "16APSK DVB-S2X", "32APSK DVB-S2X"
rolloff	Rolloff Factor Type: String Values: ".20", ".25", ".35", "Auto"
iq	IQ Mode Type: String Values: "Normal" or "Opposite"
siglevel	Signal Level Type: String Values: " -XX" where XX indicates strength of the received signal level, in dBm.
cndisp	Carrier to Noise Ratio Type: String Values: "XX.X" where XX.X indicates carrier to noise ratio, in dB. May be empty if no satlock is achieved for this port.
cnmargin	Carrier to Noise Ratio Margin Type: String Values: "XX.X" where XX.X indicates carrier to noise ratio margin in dB. The margin is the distance that the Carrier to Noise ratio is above the noise threshold. May be empty if no satlock is achieved for this port.
pwr	LNB Power Type: String Values: "Off", "13V", "18V", "H-NIT", "V-NIT"
sel22khz	22 KHz Selection Type: String Values: "Off", "On", "Auto"
lO1	LO Select 1 Type: Float

Output Field	Description
	Values: 0.0 … 15.0
lO2	LO Select 2 Type: Float Values: 0.0 … 15.0
xover	Cross Over Frequency Type: Float Values: 0.0 … 15.0
pol	Polarization Type: String Values: “Horizontal”, “Vertical”, “Auto”
orbpos	Orbital Position Type: Float Values: 0.0 ... 360.0
ewflag	East West Flag Type: String Values: "N/A", "East", "West"
afcrange	AFC Range Value Type: Float Values: 0.0 ... 5.0
acqmode	PSI Acquisition Mode Type : String Values : "Auto", "Basic", "Custom"
satlock	Satellite Lock Type: String Values: "No Lock", "Lock-Sig" or "Lock+Sig" When satlock "Lock+Sig" is not yet achieved, the values of several other parameters in the record row can not be considered to be valid. We highly recommend that you always specify the name of this item in GET requests that use Query Strings for filtering (limiting) the output response.
inputrate	Input Rate Type: Integer

Output Field	Description
	Values: 0…4294967295
netname	Network Name Type: String Values: Up to 64 characters
netid	Network ID Type: Integer Values: 0…65535
transid	Transport ID Type: Integer Values: 0…65535
lsttunerea	Last Tune Reason Type: String Values: "No_Change", "DR Change", "User_Change", "Preset_Change", "Uplink_ Change", or "NIT_TS_Change"
tunereason	Last Tune Reason Type: String Values: Up to 24 characters
freqmode	Frequency Tuning Mode Type: String Values: "NIT", "User Cfg"
svclstmode	Service List Mode Type: String Values: "Rigorous", "Relaxed"
baten	Enable BAT Reception Type: String Values: "Yes", "No"
niten	Enable NIT Reception Type: String Values: "Yes", "No", "N/A"
sdten	Enable SDT Reception

Output Field	Description
	Type: String Values: "Yes", "No", "N/A"
paten	Enable PAT Reception Type: String Values: "Yes", "No", "N/A"

Input Example:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/input/rf?js=1"
```

Expected output (when RF port 1 is active and tuned to signal, and values are for example purposes only):

Note The "siglevel", "cndisp", and "cnmargin" parameters are available on Version 3.75 or later. The values of these parameters may be empty or unreliable if satlock is not set to Lock+Sig for that port.

```
{
    "input": {
        "rf": [
            {
                "port": "1",
                "dnlkfreq": "12.31003",
                "symrate": "30.0",
                "fec": "9/10",
                "mod": "8PSK DVB-S2",
                "rolloff": ".35",
                "iq": "Normal",
                "siglevel": "",
                "cndisp": "",
                "cnmargin": "",
                "pwr": "Off",
                "sel22khz": "Off",
                "lo1": "10.75",
                "lo2": "0.0",
                "xover": "0.0",
                "pol": "Horizontal",
                "orbpos": "0.0",
                "ewflag": "N/A",
                "afcrange": "3.0",
                "acqmode": "Basic",
                "satlock": "No Lock",
                "inputrate": "0.0",
                "netname": "",
                "netid": "",
                "transid": "",
                "lsttunerea": "User_Change",
                "tunereason": "User",
                "freqmode": "NIT",
                "svclstmode": "Rigorous",
                "baten": "No",
                "niten": "Yes",
                "sdten": "Yes",
                "paten": "Yes"
            },
            {
                "port": "2",
                "dnlkfreq": "12.31003",
                "symrate": "30.0",
                "fec": "9/10",
                "mod": "8PSK DVB-S2",
                "rolloff": ".35",
                "iq": "Normal",
                "siglevel": " -58",
                "cndisp": "29.8",
                "cnmargin": "18.6",
                "pwr": "Off",
                "sel22khz": "Off",
                "lo1": "10.75",
                "lo2": "0.0",
                "xover": "0.0",
                "pol": "Horizontal",
                "orbpos": "0.0",
                "ewflag": "N/A",
                "afcrange": "3.0",
                "acqmode": "Basic",
                "satlock": "Lock+Sig",
                "inputrate": "0.0",
                "netname": "Effete",
                "netid": "1",
                "transid": "201",
                "lsttunerea": "User_Change",
                "tunereason": "User",
                "freqmode": "NIT",
                "svclstmode": "Rigorous",
                "baten": "No",
                "niten": "Yes",
                "sdten": "No",
                "paten": "No"
            },
            {
                "port": "3",
                "dnlkfreq": "12.31003",
                "symrate": "30.0",
                "fec": "9/10",
                "mod": "8PSK DVB-S2",
                "rolloff": ".35",
                "iq": "Normal",
                "siglevel": "",
                "cndisp": "",
                "cnmargin": "",
                "pwr": "Off",
                "sel22khz": "Off",
                "lo1": "5.15",
                "lo2": "0.0",
                "xover": "0.0",
                "pol": "Horizontal",
                "orbpos": "0.0",
                "ewflag": "N/A",
                "afcrange": "3.0",
                "acqmode": "Basic",
                "satlock": "No Lock",
                "inputrate": "0.0",
                "netname": "",
                "netid": "",
                "transid": "",
                "lsttunerea": "User_Change",
                "tunereason": "User",
                "freqmode": "NIT",
                "svclstmode": "Rigorous",
                "baten": "No",
                "niten": "Yes",
                "sdten": "Yes",
                "paten": "Yes"
            },
            {
                "port": "4",
                "dnlkfreq": "12.31003",
                "symrate": "30.0",
                "fec": "9/10",
                "mod": "8PSK DVB-S2",
                "rolloff": ".35",
                "iq": "Normal",
                "siglevel": "",
                "cndisp": "",
                "cnmargin": "",
                "pwr": "Off",
                "sel22khz": "Off",
                "lo1": "5.15",
                "lo2": "0.0",
                "xover": "0.0",
                "pol": "Horizontal",
                "orbpos": "0.0",
                "ewflag": "N/A",
                "afcrange": "3.0",
                "acqmode": "Basic",
                "satlock": "No Lock",
                "inputrate": "0.0",
                "netname": "",
                "netid": "",
                "transid": "",
                "lsttunerea": "User_Change",
                "tunereason": "User",
                "freqmode": "NIT",
                "svclstmode": "Rigorous",
                "baten": "No",
                "niten": "Yes",
                "sdten": "Yes",
                "paten": "Yes"
            }
        ]
    }
}
```

ASI Input Status Command

Table 2.36 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/input/asi
Command Information	Return information about ASI input.
HTTP Method	GET
Access Type	Read
Access Level	Guest, User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/input/asi?js=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.37 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
port	Input Port status to be returned. Type: Integer Values: 1..2 **Note** This does not refer to the NTC Ethernet ports, it refers specifically to the ASI ports.
stream (optional)	Multi Input Path. If not specified, default value of 1 is used. Type: Integer Values: 1
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.38 Output Field Descriptions

Output Field	Description
card	ASI Card number (Always 1) Type: Integer Value: 1 Default: 1
port	Input Port status to be responded Type: Integer Values: 1 (single-stream unit), 1..2 (multi-stream unit)
stream	Multi Input Path Stream ID for ASI. This id will always be 1 and default to 1 if not specified. Type: Integer Values: 1 Default: 1

Output Field	Description
act	Input is activated Type: String Values: "Yes" or "No" When Yes, the values of all other items in the response for the current row may be considered to be valid. When No, the values of all other items in the response for the current row may not be considered to be valid. We highly recommend that you always specify the name of this item in GET requests that use Query Strings for filtering (limiting) the output response.
asilock	ASI Signal Lock Type: String Values: "Lock" or "NoLock" When asilock "Lock" is not yet achieved, the values of several other parameters in the record row can not be considered to be valid. We highly recommend that you always specify the name of this item in GET requests that use Query Strings for filtering (limiting) the output response.
inputrate	Input Rate Type: Integer Values: 0…4294967295
netname	Network Name Type: String Values: Up to 64 characters
netid	Network ID Type: Integer Values: 0…65535
transid	Transport ID Type: Integer Values: 0…65535
scrambmode	Scrambling Mode Type: String Values: "Unk", "DES", "DVB", "BISS1", "BISS2", or "BISS3"
transerr	Transport Error

Output Field	Description
	Type: String Values: "Ok", "Error", or "N/A"
asilinkerr	ASI Link Error Type: String Values: "Ok", "Error", or "N/A"
asipktsize	ASI Packet Size Type: String Values: "188", "204", or "N/A"
lsttunerea	Last Tune Reason Type: String Values: "No_Change", "DR Change", "User_Change", "Preset_Change", "Uplink_ Change", or "NIT_TS_Change"
tunereason	Last Tune Reason Type: String Values: Up to 24 characters
freqmode	Frequency Tuning Mode Type: String Values: "NIT", "User Cfg"
svclstmode	Service List Mode (Version 2.75 and later) - Type: String - Values: "Rigorous", "Relaxed" Service List Mode (Version 2.50 and earlier) - Type : String - Values : "Rigorous", "Degraded"
baten	Enable BAT Reception Type: String Values: "Yes", "No"
niten	Enable NIT Reception Type: String Values: "Yes", "No", "N/A"

Output Field	Description
sdten	Enable SDT Reception Type: String Values: "Yes", "No", "N/A"
paten	Enable PAT Reception Type: String Values: "Yes", "No", "N/A"

Input Example:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/input/asi?js=1"
```

Expected output (values are for example purposes only):

```
{
    "input": {
        "asi": {
            "port": "1",
            "asilock": "NoLock",
            "inputrate": "0.0",
            "netname": "",
            "netid": "",
            "transid": "",
            "scrambmode": "Unk",
            "transerr": "Error",
            "asilinkerr": "Error",
            "asipktsize": "N/A",
            "lsttunerea": "User_Change",
            "tunereason": "User",
            "freqmode": "User Cfg",
            "svclstmode": "Rigorous",
            "baten": "No",
            "niten": "No",
            "sdten": "Yes",
            "paten": "Yes"
        }
    }
}
```

IP Input Status Command

Table 2.39 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/input/moip
Command Information	Return information about MOIP input.
HTTP Method	GET
Access Type	Read
Access Level	Guest, User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/input/moip?js=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.40 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
stream (optional)	Multi Input Path Stream ID for IP. If not specified, default value of 1 is used. Type: Integer Values: 1..32
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.41 Output Field Descriptions

Output Field	Description
card	IP Card number Type: Integer Value: 1 Default: 1
port	Input Port status to be returned Type: Integer Value: 1

Output Field	Description
	Default: 1
stream	MOIP Stream ID Type: Integer Values: 1 (single-stream unit), 1..32 (multi-stream unit)
act	Input is activated Type: String Values: "Yes" or "No" When Yes, the values of all other items in the response for the current row may be considered valid. When No, the values of all other items in the response for the current row may not be considered valid. We highly recommend that you always specify the name of this item in GET requests that use Query Strings for filtering (limiting) the output response.
moiplock	MOIP Signal Lock Type: String Values: "Lock", "Lock+Sig", or "NoLock"
inputrate	Input Rate Type: Integer Values: 0…4294967295
netid	Network ID Type: Integer Values: 0…65535
transid	Transport ID Type: Integer Values: 0…65535
scrambmode	Scrambling Mode Type: String Values: "Unk", "DES", "DVB”", "BISS1", "BISS2", or "BISS3"
transerr	Transport Error Type: String Values: "Ok", "Error", or "N/A"

Output Field	Description
iplinkstat	IP Link Status Type: String Values: "Down" or "Up"
ipsignal	IP Signal Type: String Values: "None" or "Active"
feclock	FEC Lock Type: String Values: "None", "Column", "Row", or "Collision"
pcrlock	PCR Lock Type: String Values: "Yes" or "No"
dejitlaten	Dejitter Latency, in ms Type: Integer Values: 0…4294967295
datasrc1	Data Source 1 (Primary Stream) Type: String Values: IP address in the dot format, for example, 192.131.244.10
datasrc2	Data Source 2 (Backup Stream) Type: String Values: IP address in the dot format, for example, 192.131.244.10
data1tstyp	Data 1 transport Type (Primary Stream) Type: String Values: "Unk", "UDP", or "RTP"
data2tstyp	Data 2 transport Type (Backup Stream) Type: String Values: "Unk", "UDP", or "RTP"
lsttunerea	Last Tune Reason Type: String Values: "No_Change", "DR Change", "User_Change", "Preset_Change", "Uplink_

Output Field	Description
	Change", or "NIT_TS_Change"
tunereason	Last Tune Reason Type: String Values: Up to 24 characters
freqmode	Frequency Tuning Mode Type: String Values: "NIT", "User Cfg"
svclstmode	Service List Mode (Version 2.75 and later) - Type : String - Values : "Rigorous", "Relaxed" Service List Mode (Version 2.50 and earlier) - Type : String - Values : "Rigorous", "Degraded"
baten	Enable BAT Reception Type: String Values: "Yes", "No", or "N/A"
niten	Enable NIT Reception Type: String Values: "Yes", "No", "N/A"
sdten	Enable SDT Reception Type: String Values: "Yes", "No", "N/A"
paten	Enable PAT Reception Type: String Values: "Yes", "No", "N/A"
cmdrow	MOIP Stream data status Type: String Values: "Active", "Inactive" When Active, the values of all other items in the response for the current row may be considered valid.

Output Field	Description
	When Inactive, the values of all other items in the response for the current row may not be considered valid. We highly recommend that you always specify the name of this item in GET requests that use Query Strings for filtering (limiting) the output response.
inputid	Input ID Type: String Values: “MOPI1" (single-stream unit), “MOIP1”-“MOIP32” (multi-stream unit)
pkterrcnt	Error Packets Counter Type: Integer Values: 0…4294967295
fecrate	FEC Input Rate Type: Integer
feccorrerr	Counter of Corrected FEC Error Packets Type: Integer Values: 0…4294967295
fecerrrate	FEC Error Rate Type: Integer
feccols	FEC Columns Present Type: String Values: "Yes", "No"
fecrows	FEC Rows Present Type : String Values : "Yes", "No"
feclval	FEC L-value Type: integer
fecdval	FEC D-value Type: integer
fecoverhead	FEC overhead, in percent Type: integer
feclatency	FEC latency, ms Type: integer

Output Field	Description
portinuse	ETH interface currently in use Type: string Values : "None", "DATA1", "DATA2", "DATA3" (D9800 MS), "DATA4" (D9800 MS), "DATA1/DATA2", "DATA3/DATA4" (D9800 MS), "Need Source" For more information, see MOIP Input Configuration Command, on page 286.
portinuse	ETH interface currently in use Type: string Values: "None", "DATA1", “DATA2”, “DATA3” (multi-stream unit), “DATA4” (multi-stream unit)
switchreas	Current redundancy switch reason Type: string Values: "None", "Setup", “EthLinkStatus”, “TS Status”, “ProgStatus”
switchtime	Current redundancy switch date and time Type: string Values: format “YYYY/MM/DD HH:MM:SS”
drstate	Disaster Recovery (D/R) status Type: string Values: "N/A", "Ready", “Need Config”, “Active”
destip	User or D/R-selected destination IP address Type: String Values: IP address, in dot format (for example, 225.1.1.10)

Input Example:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/input/moip?stream=1"
```

Expected output (values are for example purposes only):

```
{
    "input": {
        "moip": {
            "stream": "1",
            "moiplock": "Lock+Sig",
            "inputrate": "10.000038",
            "netid": "",
            "transid": "101",
            "scrambmode": "Unk",
            "transerr": "N/A",
            "iplinkstat": "Up",
            "ipsignal": "Active",
            "feclock": "Column",
            "pcrlock": "Yes",
            "dejitlaten": "105",
            "datasrc1": "192.168.1.22",
            "datasrc2": "192.168.1.22",
            "data1tstyp": "RTP",
            "data2tstyp": "RTP",
            "lsttunerea": "User_Change",
            "tunereason": "User",
            "freqmode": "User Cfg",
            "svclstmode": "Rigorous",
            "baten": "N/A",
            "niten": "N/A",
            "sdten": "Yes",
            "paten": "Yes",
            "cmdrow": "Active",
            "inputid": "MOIP1",
            "pkterrcnt": "0",
            "fecrate": "11.0",
            "feccorrerr": "0",
            "fecerrrate": "0.00e-0",
            "feccols": "Yes",
            "fecrows": "No",
            "feclval": "4",
            "fecdval": "10",
            "fecoverhead": "10",
            "feclatency": "84",
            "portinuse": "DATA1",
            "switchreas": "None",
            "switchtime": "1901/01/01 00:00:00",
            "drstate": "MOIP1:Need Config",
            "destip": "225.1.1.1"
        }
    }
}
```

ABR Input Status Command

Table 2.42 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/input/abr

Command Detail	Description
Command Information	Return information about ABR input.
HTTP Method	GET
Access Type	Read
Access Level	Guest, User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/input/abr?js=1"

Table 2.43 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
stream (key)	Multi Input Path Stream ID Type: Integer Values: 1...2
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output, by default, in XML.

Table 2.44 Output Field Descriptions

Output Field	Description
stream (key)	Multi Input Path Stream ID Type: Integer Values: 1...2
act	Input is activated Type: String Values: "Yes" or "No"
abrlock	ASI Signal Lock Type: String Values: "No Lock", "Lock-Sig", "Lock+Sig"
iplinkstat	ABR Ethernet link status Type: String

Output Field	Description
	Values: "Down", "Up"
ipsignal	ABR Signal Encapsulation Lock Status Type: String Values: "None", "Active"
inputrate	Bit rate of the input transport stream Type: Float Values: 0…4294.967295 Mbps
netname	Network Name Type: String Values: Up to 64 characters
netid	Network ID Type: Integer Values: 0…65535
transid	Transport ID Type: Integer Values: 0…65535
scrambmode	Scrambling Mode Type: String Values: "Unk", "DES", "DVB", "BISS1", "BISS2", or "BISS3"
lsttunerea	Last Tune Reason Type: String Values: "No_Change", "DR Change", "User_Change", "Preset_Change", "Uplink_Change", or "NIT_TS_Change"
tunereason	Last Tune Reason Type: String Values: Up to 24 characters
freqmode	Frequency Tuning Mode Type: String Values: "NIT", "User Cfg"
svclstmode	Service List Mode (Version 2.75 and later)

Output Field	Description
	■ Type: String ■ Values: "Rigorous", "Relaxed" Service List Mode (Version 2.50 and earlier) ■ Type: String ■ Values: "Rigorous", "Degraded"
baten	Enable BAT Reception Type: String Values: "Yes", "No", or "N/A"
niten	Enable NIT Reception Type: String Values: "Yes", "No", "N/A"
sdten	Enable SDT Reception Type: String Values: "Yes", "No", "N/A"
paten	Enable PAT Reception Type: String Values: "Yes", "No", "N/A"
rate	Bit rate of the input transport stream Type: Float Values: 0…4294.967295 Mbps
dlrate	Dowload bitrate of the input transport stream Type: Float Values: 0…4294.967295 Mbps
buflvlcurq1	Buffer level current queue 1 Type: Float Values: 0…4294.967295 Mb
buflvlcurq2	Buffer level current queue 2 Type: Float Values: 0…4294.967295 Mb
manidur	Manifest duration

Output Field	Description
	Type: Integer Values: 0…4294967295 seconds
manierr	Manifest parsing errors Type: Integer Values: 0…4294967295
dlmanicnt	Manifest download count Type: Integer Values: 0…4294967295
dlmanierr	Manifest download error Type: Integer Values: 0…4294967295
dlfragcnt	Download fragment count Type: Integer Values: 0…4294967295
dlfragerr	Download fragment errors Type: Integer Values: 0…4294967295
fragerr	Fragment parsing errors Type: Integer Values: 0…4294967295
dlcurminlat	Download stop time from the end of manifest Type: Integer Values: 0…4294967295
dlcurtrglat	Download start time from the beginning of manifest Type: Integer Values: 0…4294967295
cmdrow	ABR Stream data status Type : String Values : "Active", "Inactive" When Active, the values of all other items in the response for the current row

Output Field	Description
	may be considered valid. When Inactive, the values of all other items in the response for the current row may not be considered valid. We highly recommend that you always specify the name of this item in GET requests that use Query Strings for filtering (limiting) the output response.

Input Example:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/input/abr?js=1"
```

Expected output (values are for example purposes only):

```
{
    "input": {
        "abr": [
            {
                "stream": "1",
                "act": "Yes",
                "abrlock": "No Lock",
                "iplinkstat": "Down",
                "ipsignal": "None",
                "inputrate": "0.0",
                "netname": "",
                "netid": "1",
                "transid": "",
                "scrambmode": "Unk",
                "lsttunerea": "No_Change",
                "tunereason": "User",
                "freqmode": "User Cfg",
                "svclstmode": "Relaxed",
                "baten": "N/A",
                "niten": "N/A",
                "sdten": "No",
                "paten": "No",
                "acqstatus": "Failed",
                "name": "ABR1",
                "rate": "0",
                "dlrate": "0.0",
                "buflvlcurq1": "0.0",
                "buflvlcurq2": "0.0",
                "manidur": "0",
                "manierr": "0",
                "dlmanicnt": "0",
                "dlmanierr": "0",
                "dlfragcnt": "0",
                "dlfragerr": "0",
                "fragerr": "0",
                "dlcurminlat": "0",
                "dlcurtrglat": "0",
                "cmdrow": "Active"
            },
            {
                "stream": "2",
                "act": "Yes",
                "abrlock": "No Lock",
                "iplinkstat": "Down",
                "ipsignal": "None",
                "inputrate": "0.0",
                "netname": "",
                "netid": "113",
                "transid": "",
                "scrambmode": "Unk",
                "lsttunerea": "User_Change",
                "tunereason": "User",
                "freqmode": "User Cfg",
                "svclstmode": "Relaxed",
                "baten": "N/A",
                "niten": "N/A",
                "sdten": "No",
                "paten": "No",
                "acqstatus": "Failed",
                "name": "ABR2",
                "rate": "0",
                "dlrate": "0.0",
                "buflvlcurq1": "0.0",
                "buflvlcurq2": "0.0",
                "manidur": "0",
                "manierr": "0",
                "dlmanicnt": "0",
                "dlmanierr": "0",
                "dlfragcnt": "0",
                "dlfragerr": "0",
                "fragerr": "0",
                "dlcurminlat": "0",
                "dlcurtrglat": "0",
                "cmdrow": "Active"
            }
        ]
    }
```

```
}
```

Zixi Input Status Command

Table 2.45 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/input/zixi
Command Information	Return information about ZIXI input.
HTTP Method	GET
Access Type	Read
Access Level	Guest, User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/input/zixi?js=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.46 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
stream (key)	Multi Input Path Stream ID Type: Integer Values: 1...4
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.47 Output Field Descriptions

Output Field	Description
stream (key)	Multi Input Path Stream ID Type: Integer Values: 1...4
act	Input is activated Type: String

Output Field	Description
	Values: "Yes" or "No"
abrlock	Zixi Signal Lock Type: String Values: "No Lock", "Lock-Sig", "Lock+Sig"
iplinkstat	Zixi Ethernet link status Type: String Values: "Down", "Up"
ipsignal	Zixi Signal Encapsulation Lock Status Type: String Values: "None", "Active"
inputrate	Bit rate of the input transport stream Type: Float Values: 0…4294.967295 Mbps
netname	Network Name Type: String Values: Up to 64 characters
netid	Network ID Type: Integer Values: 0…65535
transid	Transport ID Type: Integer Values: 0…65535
scrambmode	Scrambling Mode Type: String Values: "Unk", "DES", "DVB", "BISS1", "BISS2", or "BISS3"
lsttunerea	Last Tune Reason Type: String Values: "No_Change", "DR Change", "User_Change", "Preset_Change", "Uplink_ Change", or "NIT_TS_Change"
tunereason	Last Tune Reason

Output Field	Description
	Type: String Values: Up to 24 characters
freqmode	Frequency Tuning Mode Type: String Values: "NIT", "User Cfg"
svclstmode	Service List Mode Type: String Values: "Rigorous", "Degraded"
baten	Enable BAT Reception Type: String Values: "N/A", "Yes", "No"
niten	Enable NIT Reception Type: String Values: "N/A", "Yes", "No"
sdten	Enable SDT Reception Type: String Values: "N/A", "Yes", "No"
paten	Enable PAT Reception Type String Values: "N/A", "Yes", "No"
acqstatus	Acquisition State Type: String Values: "Full", "Partial", "None"
inputid	Input Id Type: String Values: "ZIXI1", "ZIXI2"
reconn	Zixi reconnections Type: Integer Values: 0…65535
arqrec	Zixi packets recovered by ARQ (Automatic Repeat Request)

Output Field	Description
	Type: Integer Values: 0…65535
fecrec	Zixi packets recovered by FEC (Forward Error Correction) Type: Integer Values: 0…65535
notrec	Zixi packets not recovered Type: Integer Values: 0…65535
rowstatus	Zixi input row status Type: String Values: "Active", "Inactive" When Active, the values of all other items in the response for the current row may be considered valid. When Inactive, the values of all other items in the response for the current row may not be considered valid. We highly recommend that you always specify the name of this item in GET requests that use Query Strings for filtering (limiting) the output response.
cursrc	Zixi source currently connected Type: String Values: "zixi://tor02lab-s-002.synamedia.com:2077/synamedia-BBB", for example

Example:

Input:

```
curl -k -H "X-SESSION-ID: $token"
https://192.168.0.1/ws/v2/status/input/zixi?stream=1&js=1"
```

Expected output (values are for example purposes only):

```
{
    "input": {
        "zixi": {
            "stream": "1",
            "act": "Yes",
            "lock": "Lock+Sig",
            "iplinkstat": "Up",
            "ipsignal": "Active",
            "inputrate": "7.17246",
            "netname": "",
            "netid": "",
            "transid": "1",
            "scrambmode": "Unk",
            "lsttunerea": "User_Change",
            "tunereason": "User",
            "freqmode": "User Cfg",
            "svclstmode": "Relaxed",
            "baten": "N/A",
            "niten": "N/A",
            "sdten": "No",
            "paten": "Yes",
            "acqstatus": "Partial",
            "inputid": "ZIXI1",
            "reconn": "0",
            "arqrec": "0",
            "fecrec": "0",
            "notrec": "0",
            "rowstatus": "Active",
            "cursrc": "zixi://tor02lab-s-002.synamedia.com:2077/synamedia-
BBB"
        }
    }
}
```

SRT Input Status Command

Table 2.48 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/input/srt
Command Information	Return information about SRT input.
HTTP Method	GET
Access Type	Read
Access Level	Guest, User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/input/SRT?js=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.49 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
stream (key)	Multi Input Stream ID For GET operations, specify this key=value (stream=1 to 2) to only return the row of interest; when not specified the response will return all rows. Type: Integer Values: 1
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.50 Output Field Descriptions

Output Field	Description
card (Key 0)	Card Identifier Will show in GET Response but not needed when specifying input parameters. Type: Integer Values: 1
port (Key 1)	Input Port Number on this Card Will show in GET Response but not needed when specifying input parameters. Type: Integer Values: 1
stream (Key 2)	Multi Input Stream ID For GET operations, specify this key=value (stream=1 to 2) to only return the row of interest; when not specified the response will return all rows. Type: Integer Values: 1..2
act	Input is activated Type: String

Output Field	Description
	Values: "Yes" or "No"
lock	SRT Signal Lock Type: String Values: "No Lock", "Lock-Sig", "Lock+Sig"
iplinkstat	SRT Ethernet link status Type: String Values: "Down", "Up"
ipsignal	SRT Signal Encapsulation Lock Status Type: String Values: "None", "Active"
inputrate	Bit rate of the input transport stream Type: Float Values: 0…4294.967295 Mbps
netname	Network Name Type: String Values: Up to 64 characters
netid	Network ID Type: Integer Values: 0…65535
transid	Transport ID Type: Integer Values: 0…65535
scrambmode	Scrambling Mode Type: String Values: "Unk", "DES", "DVB", "BISS1", "BISS2" or "BISS3"
lsttunerea	Last Tune Reason Type: String Values: No_Change", "DR Change", "User_Change", "Preset_Change", "Uplink_ Change" or "NIT_TS_Change"
tunereason	Last Tune Reason

Output Field	Description
	Type: String Values: Up to 24 characters
freqmode	Frequency Tuning Mode Type : String Values : "NIT", "User Cfg" **Note** Due to the space character in "User Cfg", it requires URL encoding when sending the data via the URL inline parameter list method (such as via Query String for curl using "User%20Cfg" or parms parameter for Python, first converting the payload with the urlencode method). Note also that since the number of URL in-line parameters (or Query Strings in curl parlance) are limited to 26 at a time, filtering parameters should be limited to only a few entries if used at all.
svclstmode	Service List Mode Type : String Values : "Rigorous", "Relaxed"
baten	Enable BAT Reception Type : String Values : "N/A", "Yes", "No"
niten	Enable NIT Reception Type : String Values : "N/A", "Yes", "No"
sdten	Enable SDT Reception Type : String Values : "N/A", "Yes", "No"
paten	Enable PAT Reception Type : String Values : "N/A", "Yes", "No"
acqstatus	Acquisition State Type : String Values : "Full", "Partial", "None"
inputid	Input Id

Output Field	Description
	Type : String Values : "ZIXI1", "ZIXI2"
curintf	SRT settings physical interface Type : String Values : "Mgmt", "Data1", "Data2", "Data3", "Data4"
rowstatus	SRT Input Row Status Type : String Values : "Active", "Inactive" When Active, the values of all other items in the response for the current row may be considered to be valid. When Inactive, the values of all other items in the response for the current row may not be considered to be valid. We strongly recommend that you always specify the name of this item in GET requests that use Query Strings for filtering (limiting) the output response.
cursrc	SRT Source Currently Connected The string will consist of a prefix (Main, Backup) followed by a colon and then the ip address. The prefix indicates if the current source is the main or redundant backup source. Type : String Values : for example, "Main:10.89.123.456 "
deldelay	Time stamp based Packet Delivery Delay from current SRT input (in milli-seconds) Type : Integer Values : 0…65535
totallost	Total Lost Packets from current SRT Input Type : Integer Values : 0…65535
switchreas	SRT Redundancy Switch Over Reason Type : String Values : "None", "Setup", "EthLinkStatus", "TS Status", "ProgSTatus"
switchtime	SRT Redundancy Switch Over Date and Time

Output Field	Description
	Note For this status item to function correctly, the NTP feature must be in use and synchronized to an NTP server. Type : String Values : example, default is: "1900/12/31 19:00:00"
pkterrcnt	Indicates packet error rate of the SRT Input Type : Integer Values : 0…65535
drstate	Disaster Recovery state of the SRT Input Type : String Values : up to 48 characters
acqmode	Indicates which Acquisition Mode is being used for this Input Type : String Values : "Basic", "Auto", "Custom", "FixPID"
bissprofen	BISS Profile enabled or disabled for SRT Type : String Values : "Yes", "No"
bissprof	BISS Profile value for SRT Type : Integer Values : 1..32
tsprot	Transport Stream Protocol for this SRT Input Stream This item will only be valid when these status parameters are the following values: act="Yes" and lock="Lock+Sig". Type : String Values : "RTP", "UDP", "Unknown"
sendrate	SRT Send Rate Type : Integer Values : 0…65535
recvrate	SRT Receive Rate Type : Integer Values : 0…65535

Output Field	Description
rtt	SRT Round Trip Time (in milli-seconds) Type : Integer Values : 0…65535

Example:

Input:

```
curl -k -H "X-SESSION-ID: $token"
https://192.168.0.1/ws/v2/status/input/zixi?stream=1&js=1"
```

Expected output (values are for example purposes only):

```
{
    "input": {
        "srt": {
                "card": "1",
                "port": "1",
                "stream": "1",
                "act": "Yes",
                "lock": "Lock+Sig",
                "iplinkstat": "Up",
                "ipsignal": "Active",
                "inputrate": "14.999996",
                "netname": "",
                "netid": "",
                "transid": "101",
                "scrambmode": "Unk",
                "lsttunerea": "User_Change",
                "tunereason": "User",
                "freqmode": "User Cfg",
                "svclstmode": "Relaxed",
                "baten": "N/A",
                "niten": "N/A",
                "sdten": "Yes",
                "paten": "Yes",
                "acqstatus": "Full",
                "inputid": "SRT1",
                "curintf": "Mgmt",
                "rowstatus": "Active",
                "cursrc": "Main:10.89.123.456",
                "deldelay": "0",
                "totallost": "0",
                "switchtime": "1900/12/31 19:00:00",
                "pkterrcnt": "0",
                "drstate": "SRT1:Need Config",
                "acqmode": "Auto",
                "bissprofen": "Yes",
                "bissprof": "1",
                "tsprot": "UDP",
                "sendrate": "0",
                "recvrate": "15",
                "rtt": "0"
        }
    }
}
```

IP 2022-7 Input Status Command

Table 2.51 Command Details

Command Detail	**Description**
Command URL	https://192.168.0.1/ws/v2/status/input/moip2022-7
Command Information	Return extra information about MOIP input when using hitless merge. SMPTE 2022-7 hitless merge requires a configuration with mirrored MOIP input data ports (for example, using MOIP input on two Data ports with upstream MOIP outputs configured for mirrored mode), and enabling "reddir=2022-7%20Merge" in the POST ws/v2/service_cfg/input/moip API.
HTTP Method	GET
Access Type	Read
Access Level	Guest, User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/input/moip2022_7?stream=x&js" where x is the stream ID number of interest

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.52 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	**Description**
stream (optional)	Multi Input Path Stream ID for IP. If not specified, records for all streams are returned. For single-stream unit, only stream 1 is available. For multi-stream units, up to 32 streams are available. We recommend that you only retrieve records of interest in the output by specifying only the expected stream value. Type: Integer Values: 1..32
js	Format output using JSON standard

URI Argument	Description
	Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML

Table 2.53 Output Field Descriptions

Output Field	Description
card (key)	IP Card number Type: Integer Value: 1 Default: 1
port (Key)	Input Port status to be returned Type: Integer Value: 1 Default: 1
stream (Key)	MOIP Stream ID Type: Integer Values: 1 (D9800 SS), 1..32 (D9800 MS) Default: When not specified, all possible stream records will be returned by default when no other Query String filter items limit the output.
instance	Instance of IP Input Stream This is a unique identifier for User Interface (UI) display purposes that has been applied to this stream record (typically, uses MOIP%d where %d is the integer value of the MOIP Stream Id). Type: String
validpkts	Valid Packet Count Number of valid packets on both Main and Backup streams. The range is from "0" to the maximum value of an unsigned 64 bit integer. Type: String Max Characters in String: 20 (plus one for null terminator) Value: "0" to "18446744073709551615"
duplpkts	Duplicate Packet Count Number of duplicated packets. The range is from "0" to the maximum value of an unsigned 64 bit integer.

Output Field	Description
	Type: String Max Characters in String: 20 (plus one for null terminator) Value: "0" to "18446744073709551615"
missedpkts	Missed Packet Count Number of missed packets. The range is from "0" to the maximum value of an unsigned 64 bit integer. Type: String Max Characters in String: 20 (plus one for null terminator) Value: "0" to "18446744073709551615"
outrngpkts	Out of Range Packet Count Number of out of range packets. The range is from "0" to the maximum value of an unsigned 64 bit integer. Type: String Max Characters in String: 20 (plus one for null terminator) Value: "0" to "18446744073709551615"
reordpkts	Reordered Packet Count Number of missed packets. The range is from "0" to the maximum value of an unsigned 64 bit integer. Type: String Max Characters in String: 20 (plus one for null terminator) Value: "0" to "18446744073709551615"
reordwnpkts	Reordered Window Size, Packet Count Number of reordered window size packets Type: Integer Value: 0 to 65535
reordwndly	Reordering window delay, in milliseconds Type: Integer Value: 0 to 65535
resetcnt	Hardware Reset Counter Type: Integer

Output Field	Description
	Value: 0 to 65535
rowstatus	MOIP 2022-7 Stream data status Type : String Values : "Active", "Inactive" When Active, the values of all other items in the response for the current row may be considered to be valid. When Inactive, the values of all other items in the response for the current row may not be considered to be valid. We strongly recommend that you always specify the name of this item in GET requests that use Query Strings for filtering (limiting) the output response.

Example:

Input:

```
curl -k -H "X-SESSION-ID: $token"
https://192.168.0.1/ws/v2/status/input/moip2022_7?stream=1&js=1"
```

Expected output (values are for example purposes only):

```
{
     "input": {
        "moip2022_7": {
        "card": "1",
        "port": "1",
        "stream": "1",
        "instance": "MOIP1",
        "validpkts": "1237534",
        "duplpkts": "0",
        "missedpkts": "3",
        "outrngpkts": "0",
        "reordpkts": "1",
        "reordwnpkts": "0",
        "reordwndly": "0",
        "resetcount": "0",
        "rowstatus": "Active"
      }
   }
}
```

PE Status Command

Table 2.54 Command Details

Command Detail	Description
Command URL	ttps://192.168.0.1/ws/v2/status/pe
Command Information	This command is used to display PE status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax Examples	GET "https://192.168.0.1/ws/v2/status/pe", GET "https://192.168.0.1/ws/v2/status/pe?peid=2", or GET "https://192.168.0.1/ws/v2/status/pe?peid=2&chn&chname"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.55 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
peid (key)	Program Entry Identifier Type: Integer Value: 1..32
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.56 Output Field Descriptions

Output Field	Description
Chn	Channel Number to be configured on this Program Entry. Type: Integer Values: 0..65535
chname	Channel Name Type: String

Output Field	Description
	Values: String used to describe the channel name, less than 64 characters
caauth	CA authorized Type: String Values: "Yes", "No"
caencrypt	CA Encrypted Type: String Values: "Yes", "No"
cascramble	CA Scrambled Type: String Values: "Yes", "No"
caid	CA identifier Type: String Values: "Unknown", "FTA", "SA", "BISS1", "BISS2", "BISS3", "Standardized", "Canal Plus", "CCETT", "Deutsche Tel", "Eurodec", "France Tel", "Irdeto", "Jerrold GI", "Matra Comm", "NDS", "Nokia", "Norwegian Tel", "Ntl", "Philips", "Sony", "Tandberg Tv", "Thomson", "TV Com", "HPT Croatian", "HRT Croatian", "IBM", "NERA", "Beta Technik", "Kudelski", "Titan IS", "Telfonica", "Stentor", "Tadiran Scopus", "Barco", "Starguide DN", "Mentor DS", "European", "Polycipher", "General Ins", "Telemann", "Cryptoworks", "Tsinghua", "Easycas", "Alphacrypt", "DVN Holdings", "Shanghai ADT", "Shenzhen King", "Sky", "Dreamcrypt", "Thalescrypt", "Runcom", "Sidsa", "Beijing Com", "Latens", "Xcrypt" , "Beijing Dig", "Widevine Tec", "SK Tel", "Enigma Sys", "Reserved", "Ruscrypt", "Other"
prgmstatus	program status Type: String Values: "Active", "Inactive"
pmtpid	PMT pid Type: Integer Values: 0…8192
pcrpid	PCR pid Type: Integer Values: 0…8192
srstatus	SR Status Type: String

Output Field	Description
	Values: "Not Started", "Primary", "Alternate"
srtype	SR Type Type: String Values: "None", "Scheduled", "CA", "Cue Trigger"
srstartime	SR start time Type: Time Values: for example, "2007/09/01 00:00:00"
srendtime	SR end time Type: Time Values: for example, "2007/09/01 00:00:00"
peenabled	PE enabled Type: String Values: "Yes", "No"
hdlicensed	HD Licensed Type: String Values: "Yes", "No"
chsource	Channel source Type: String Values: "User", "First Chan", "LEC", "ADP Force Tune", "DR", "SR", "Powerup"

Input Example (request XML output):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/pe?peid=1"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1"
?><pe><record><peid>1</peid><chn>801</chn>
<inp>RF2</inp><chname>Effete H.264
1080i</chname><caauth>Yes</caauth><caencrypt>Yes</caencrypt>
<cascramble>Yes</cascramble><caid>SA</caid><prgmstatus>Active</prgmstatus>

<pmtpid>5801</pmtpid><pcrpid>2100</pcrpid><srstatus>Primary</srstatus><srtype
>None</srtype>
<srstartime>2007/09/01 00:00:00</srstartime>
```

```
<srendtime>2007/09/01
00:00:00</srendtime><peenabled>Yes</peenabled><hdlicensed>No</hdlicensed>
<chsource>User</chsource></record></pe>
```

Input (request JSON output):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/pe?peid=1&js=1"
```

Expected output (values are for example purposes only):

```
{
    "pe": {
        "record": {
            "peid": "1",
            "chn": "801",
            "inp": "RF2",
            "chname": "Effete H.264 1080i",
            "caauth": "Yes",
            "caencrypt": "Yes",
            "cascramble": "Yes",
            "caid": "SA",
            "prgmstatus": "Active",
            "pmtpid": "5801",
            "pcrpid": "2100",
            "srstatus": "Primary",
            "srtype": "None",
            "srstartime": "2007/09/01 00:00:00",
            "srendtime": "2007/09/01 00:00:00",
            "peenabled": "Yes",
            "hdlicensed": "No",
            "chsource": "User"
        }
    }
}
```

CA PowerVu Status Command

Table 2.57 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/ca/pvu
Command Information	This command returns (or clears counters for) PowerVu Status information.
HTTP Methods	GET (All Access), POST (Restricted Access)
Access Type	Read (Display Details), Write (Clear Counters)

Command Detail	Description
Access Level	Guest (Read Only), User (Read/Write), Admin (Read/Write)
Clear Counters Syntax	POST "https://192.168.0.1/ws/v2/status/ca/pvu/clear"
Display Details Syntax Details of Read: ISE User Address, Enc. Data Pkts Passed, Enc. Data Pkts Rcvd, Non-Enc Data Pkts Pass, Non-Enc DataPkts Rcvd.	GET "https://192.168.0.1/ws/v2/status/ca/pvu"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.58 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
clear	Clear all counters.
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.59 Output Field Descriptions

Output Field	Description
iseuseradd	ISE User Address Type: String
encpktpass	Encrypted ADP Packets Passed Type: Integer Value: 0 … 4294967295
encpktrcv	Encrypted ADP Packets Received Type: Integer Value: 0 …4294967295
nencpktpass	Non-Encrypted ADP Packets Passed Type: Integer Value: 0 …4294967295
nencpktrcv	Non-Encrypted ADP Packets Received

Output Field	Description
	Type: Integer Value: 0 …4294967295

Input Example (request XML default format):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/status/ca/pvu"
```

Expected output (values are for example purposes only):

```
<ca>
               <pvu>

                               <iseuseradd>000-529-2431-6</iseuseradd>
                               <encpktpass>10490</encpktpass>
                               <encpktrcv>10490</encpktrcv>
                               <nencpktpass>8390</nencpktpass>
                               <nencpktrcv>8390</nencpktrcv>
               </pvu>
               <pvu>
                               <iseuseradd>000-529-2432-3</iseuseradd>
                               <encpktpass >10388</encpktpass>
                               <encpktrcv >10388</encpktrcv>
                               <nencpktpass >8392</nencpktpass>
                               <nencpktrcv >8392</nencpktrcv>

               </pvu>
       </ca>
```

Input Example (request JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/ca/pvu?js=1"
```

Expected output (values are for example purposes only):

```
{
    "ca": {
        "pvu": [
            {
                "iseuseradd": "000-529-2431-6",
                "encpktpass": "10490",
                "encpktrcv": "10490",
                "nencpktpass": "8390",
                "nencpktrvc": "8390"
            },
            {
                "iseuseradd": "000-529-2432-3",
                "encpktpass": "10388",
                "encpktrcv": "10388",
                "nencpktpass": "8392",
                "nencpktrvc": "8392"
            }
        ]
    }
}
```

CI (Common Interface) CAM Status Command

Table 2.60 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/ca/ci
Command Information	Return information about Common Interface (CI) CAMs.
HTTP Methods	GET (All Access), POST (Restricted Access)
Access Type	Read
Access Level	Guest, User, Admin
Display Details Syntax	GET "https://192.168.0.1/ws/v2/status/ca/ci"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.61 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
slot	CI CAM Slot Type: String Value: "TOP", "BOTTOM"
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.62 Output Field Descriptions

Output field	Description
camstatus	CAM Status in this slot Type: String Value: "Not Ready", "Ready"
sysname	CAM System Name Type: String Value: Character String (Varies based on type of CAM inserted in the slot) maxLength: 29 (not including null terminator)
compname	CAM Company Name Type: String Value: Character string of inserted CAM name maxLength: 39 (not including null terminator)
mfgcode	CAM Manufacture Code Type: Integer Value: 0…4294967295
mfgid	CAM Manufacture Identifier Type: Integer Value: 0…4294967295
serialno	CAM Serial Number Type: String Value: Inserted CAM Serial number in string format (max 63 characters)
hwver	CAM Hardware Version Type: String Value: Inserted CAM Hardware Version in string format maxLength: 63 (not including null terminator)
appver	CAM Application Version Type: String Value: Inserted CAM Application Version in string format maxLength: 255 (not including null terminator)
prodcode	CAM Product Code Name

Output field	Description
	Type: String Value: Character string of inserted CAM Product Code name maxLength: 39 (not including null terminator)

Input Example:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/status/ca/ci"
```

Expected output (values are for example purposes only):

```
<ca>
    <ci>
         <slot>TOP</slot>
               <camstatus>Ready</camstatus>
               <sysname>Conax Conditional Access</sysname>
               <compname>SMIT</compname>
               <mfgcode>47806</mfgcode>
               <mfgid>51966</mfgid>
               <serialno>
               </serialno>
               <hwver>
               </hwver>
                <appver>
                </appver>
                <prodcode>DVB CA Module</prodcode>
    </ci>
         <slot>
                <slotId>BOTTOM</slotid>
                <camstatus>Not Ready</camstatus>
                <sysname>
                </sysname>
                <compname>
                </compname>
                <mfgcode>0</mfgcode>
                <mfgid>0</mfgid>
                <serialno>
                </serialno>
                <hwver>
                </hwver>
                <appver>
                </appver>
                <prodcode>
                </prodcode>
          </slot>
```

```
      </ci>
</ca>
```

Get Status for TOP slot in JSON format:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/ca/ci?slot=TOP&js=1"
```

Expected output (values are for example purposes only):

```
{
    "ca": {
        "ci": {
            "slot": "TOP",
            "camstatus": "Ready",
            "sysname": "Sony_MSM Professional CAM",
            "compname": "SMIT",
            "mfgcode": "47806",
            "mfgid": "51966",
            "serialno": "Unknown",
            "hwver": "Unknown",
            "appver": "Firmware: 4.1.8 M, App-V12, May 30 2016",
            "prodcode": "DVB CA Module"
        }
    }
}
```

CISYS Status Command

Table 2.63 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/ca/cisys
Command Information	Return information about Common Interface (CI) CAMs supported System IDs.
HTTP Methods	GET (All Access), POST (Restricted Access)
Access Type	Read
Access Level	Guest, User, Admin
Display Details Syntax	GET "https://192.168.0.1/ws/v2/status/ca/cisys"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.64 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.65 Output Field Descriptions

Output Field	Description
slotid	CI CAM Slot Type: String Value: "TOP", "BOTTOM"
sysname	CI System Name Type: String Value: CAM Supported System Name
sysid	CAM Supported System ID Type: String Value: Inserted CAM supported System IDs
sysnameid	CAM Supported System ID with Name Type: String Value: Inserted CAM supported System ID + Name

Input example (request XML format by default):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/status/ca/cisys"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<ca>
       <cisys>
        <slotid>TOP</slotid>
        <sysids>2525</sysids>
        <sysname>News Dataco</sysname>
        <sysnameids>News Dataco (0x9dd)</sysnameids>
    </cisys>
</ca>
```

Input example (request JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/ca/cisys?js=1"
```

Expected output (values are for example purposes only):

```
{
    "ca": {
        "cisys": {
            "slotid": "TOP",
            "sysids": "2525",
            "sysname": "News Dataco",
            "sysnameids": "News Dataco (0x9dd)"
        }
    }
}
```

Useraddr Status Command

Table 2.66 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/ca/useraddr
Command Information	Return user addresses from all ISE chips.
HTTP Methods	GET (All Access), POST (Restricted Access)
Access Type	Read
Access Level	Guest, User, Admin
Display Details Syntax	GET "https://192.168.0.1/ws/v2/status/ca/useraddr"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.67 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid)

URI Parameter	Description
	Omitting this argument formats the output by default in XML.

Table 2.68 Output Field Descriptions

Output Field	Description
uaindex	User address index within given chip (iseindex) Type: Integer Value: 1 to number of user addresses within unit
useraddr	User Address Type: String Value: output in the form "nnn-nnn-nnnn-n"

Input Example (request XML format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/ca/useraddr"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<ca>
    <useraddr>
        <uaindex>1</uaindex>
        <useraddr>000-666-0462-4</useraddr>
    </useraddr>
</ca>
```

Expected output (values are for example purposes only):

```
{
    "ca": {
        "useraddr": {
            "uaindex": "1",
            "useraddr": "000-666-0462-4"
        }
    }
}
```

Output Status Command

Table 2.69 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/output
Command Information	This command can be used to get all output status parameters.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Display Details Syntax	GET "https://192.168.0.1/ws/v2/status/output"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.70 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (any of js=1, js, js=0, or js=9999 are valid) Omitting this argument formats the output by default in XML.

Input example (request XML format):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/status/output"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1"
?><output><asi><rate>78.535733</rate>

<free>0.0</free><pid><peid>1</peid><row>1</row><inpid>1500</inpid><outpid>150
0</outpid>
<inchan>1500</inchan><type>VID</type><received>No</received><pcr>No</pcr>
<scrambled>No</scrambled></pid><pid><peid>1</peid><row>2</row>

<inpid>3349</inpid><outpid>3349</outpid><inchan>3349</inchan><type>LSDT</typ
e>
<received>Yes</received><pcr>No</pcr><scrambled>No</scrambled></pid>
<pid><peid>1</peid><row>3</row><inpid>4001</inpid><outpid>4001</outpid>
<inchan>4001</inchan><type>MPE</type><received>
```

```
No</received><pcr>No</pcr><scrambled>No</scrambled></pid><pid><peid>1</peid><
row>4</row>
<inpid>4002</inpid><outpid>4002</outpid><inchan>4002</inchan><type>MPE</type>
<received>No</received><pcr>No</pcr><scrambled>No</scrambled></pid>
<pid><peid>1</peid><row>5</row><inpid>4003</inpid>

<outpid>4003</outpid><inchan>4003</inchan><type>MPE</type><received>No</recei
ved><pcr>No</pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>6</row><inpid>4004</in
pid><outpid>4004

</outpid><inchan>4004</inchan><type>MPE</type><received>No</received><pcr>No<
/pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>7</row><inpid>4005</in
pid><outpid>4005

</outpid><inchan>4005</inchan><type>MPE</type><received>No</received><pcr>No<
/pcr><scrambled>

No</scrambled></pid><pid><peid>1</peid><row>8</row><inpid>1550</inpid><outpid
>1550</outpid>

<inchan>1550</inchan><type>DPI</type><received>No</received><pcr>No</pcr><scr
ambled>No</scrambled>

</pid><pid><peid>1</peid><row>9</row><inpid>2906</inpid><outpid>2906</outpid>
<inchan>2906</inchan>

<type>VBI</type><received>No</received><pcr>No</pcr><scrambled>No</scrambled>
</pid><pid><peid>1

</peid><row>10</row><inpid>1511</inpid><outpid>1511</outpid><inchan>1511</inc
han><type>AUD</type>

<received>Yes</received><pcr>No</pcr><scrambled>No</scrambled></pid><pid><pei
d>1</peid>

<row>11</row><inpid>1501</inpid><outpid>1501</outpid><inchan>1501</inchan><ty
pe>AUD</type>

<received>Yes</received><pcr>No</pcr><scrambled>No</scrambled></pid><pid><pei
d>1</peid><row>12</row>
```

```
<inpid>1102</inpid><outpid>1102</outpid><inchan>1102</inchan><type>AUD</type>
<received>Yes

</received><pcr>No</pcr><scrambled>No</scrambled></pid><pid><peid>1</peid><ro
w>13</row><inpid>1113

</inpid><outpid>1113</outpid><inchan>1113</inchan><type>AUD</type><received>Y
es</received>

<pcr>No</pcr><scrambled>No</scrambled></pid><pid><peid>1</peid><row>14</row><
inpid>2750</inpid>

<outpid>2750</outpid><inchan>2750</inchan><type>SUBT</type><received>Yes</rec
eived><pcr>No</pcr>
<scrambled>No</scrambled></pid><collision><status>No Record in Dynamic
Table</status></collision>

</asi><moip><enginerr>No</enginerr><overflow>No</overflow><comrate>68.5</comr
ate><ts1rate>0.0

</ts1rate><ts2rate>0.0</ts2rate><pid><peid>1</peid><row>1</row><inpid>1500</i
npid><outpid>1500

</outpid><inchan>1500</inchan><type>VID</type><received>No</received><pcr>No<
/pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>2</row><inpid>3349</in
pid><outpid>3349

</outpid><inchan>3349</inchan><type>LSDT</type><received>Yes</received><pcr>N
o</pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>3</row><inpid>4001</in
pid><outpid>4001

</outpid><inchan>4001</inchan><type>MPE</type><received>No</received><pcr>No<
/pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>4</row><inpid>4002</in
pid><outpid>4002

</outpid><inchan>4002</inchan><type>MPE</type><received>No</received><pcr>No<
/pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>5</row><inpid>4003</in
pid><outpid>4003
```

```
</outpid><inchan>4003</inchan><type>MPE</type><received>No</received><pcr>No<
/pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>6</row><inpid>4004</in
pid><outpid>4004

</outpid><inchan>4004</inchan><type>MPE</type><received>No</received><pcr>No<
/pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>7</row><inpid>4005</in
pid><outpid>4005

</outpid><inchan>4005</inchan><type>MPE</type><received>No</received><pcr>No<
/pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>8</row><inpid>1550</in
pid><outpid>1550

</outpid><inchan>1550</inchan><type>DPI</type><received>No</received><pcr>No<
/pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>9</row><inpid>2906</in
pid><outpid>2906

</outpid><inchan>2906</inchan><type>VBI</type><received>No</received><pcr>No<
/pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>10</row><inpid>1511</i
npid><outpid>1511

</outpid><inchan>1511</inchan><type>AUD</type><received>Yes</received><pcr>No
</pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>11</row><inpid>1501</i
npid><outpid>1501

</outpid><inchan>1501</inchan><type>AUD</type><received>Yes</received><pcr>No
</pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>12</row><inpid>1102</i
npid><outpid>1102

</outpid><inchan>1102</inchan><type>AUD</type><received>Yes</received><pcr>No
</pcr>
```

```
<scrambled>No</scrambled></pid><pid><peid>1</peid><row>13</row><inpid>1113</i
npid><outpid>1113

</outpid><inchan>1113</inchan><type>AUD</type><received>Yes</received><pcr>No
</pcr>

<scrambled>No</scrambled></pid><pid><peid>1</peid><row>14</row><inpid>2750</i
npid><outpid>2750

</outpid><inchan>2750</inchan><type>SUBT</type><received>Yes</received><pcr>N
o</pcr>
<scrambled>No</scrambled></pid><collision><status>No Record in Dynamic
Table</status></collision>
</moip></output>
```

Input example (request JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output?js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "asi": {
            "rate": "78.535922",
            "free": "0.0",
            "pid": [
                {
                    "peid": "1",
                    "row": "1",
                    "inpid": "1500",
                    "outpid": "1500",
                    "inchan": "1500",
                    "type": "VID",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "2",
                    "inpid": "3349",
                    "outpid": "3349",
                    "inchan": "3349",
                    "type": "LSDT",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "3",
                    "inpid": "4001",
                    "outpid": "4001",
                    "inchan": "4001",
                    "type": "MPE",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "4",
                    "inpid": "4002",
                    "outpid": "4002",
                    "inchan": "4002",
                    "type": "MPE",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "5",
                    "inpid": "4003",
                    "outpid": "4003",
                    "inchan": "4003",
                    "type": "MPE",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "6",
                    "inpid": "4004",
                    "outpid": "4004",
                    "inchan": "4004",
                    "type": "MPE",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "7",
                    "inpid": "4005",
                    "outpid": "4005",
                    "inchan": "4005",
                    "type": "MPE",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "8",
                    "inpid": "1550",
                    "outpid": "1550",
                    "inchan": "1550",
                    "type": "DPI",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "9",
                    "inpid": "2906",
                    "outpid": "2906",
                    "inchan": "2906",
                    "type": "VBI",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "10",
                    "inpid": "1511",
                    "outpid": "1511",
                    "inchan": "1511",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "11",
                    "inpid": "1501",
                    "outpid": "1501",
                    "inchan": "1501",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "12",
                    "inpid": "1102",
                    "outpid": "1102",
                    "inchan": "1102",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "13",
                    "inpid": "1113",
                    "outpid": "1113",
                    "inchan": "1113",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "14",
                    "inpid": "2750",
                    "outpid": "2750",
                    "inchan": "2750",
                    "type": "SUBT",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                }
            ],
            "collision": {
                "status": "No Record in Dynamic Table"
            }
        },
        "moip": {
            "enginerr": "No",
            "overflow": "No",
            "comrate": "68.5",
            "ts1rate": "0.0",
            "ts2rate": "0.0",
            "pid": [
                {
                    "peid": "1",
                    "row": "1",
                    "inpid": "1500",
                    "outpid": "1500",
                    "inchan": "1500",
                    "type": "VID",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "2",
                    "inpid": "3349",
                    "outpid": "3349",
                    "inchan": "3349",
                    "type": "LSDT",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "3",
                    "inpid": "4001",
                    "outpid": "4001",
                    "inchan": "4001",
                    "type": "MPE",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "4",
                    "inpid": "4002",
                    "outpid": "4002",
                    "inchan": "4002",
                    "type": "MPE",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "5",
                    "inpid": "4003",
                    "outpid": "4003",
                    "inchan": "4003",
                    "type": "MPE",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "6",
                    "inpid": "4004",
                    "outpid": "4004",
                    "inchan": "4004",
                    "type": "MPE",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "7",
                    "inpid": "4005",
                    "outpid": "4005",
                    "inchan": "4005",
                    "type": "MPE",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "8",
                    "inpid": "1550",
                    "outpid": "1550",
                    "inchan": "1550",
                    "type": "DPI",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "9",
                    "inpid": "2906",
                    "outpid": "2906",
                    "inchan": "2906",
                    "type": "VBI",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "10",
                    "inpid": "1511",
                    "outpid": "1511",
                    "inchan": "1511",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "11",
                    "inpid": "1501",
                    "outpid": "1501",
                    "inchan": "1501",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "12",
                    "inpid": "1102",
                    "outpid": "1102",
                    "inchan": "1102",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "13",
                    "inpid": "1113",
                    "outpid": "1113",
                    "inchan": "1113",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "14",
                    "inpid": "2750",
                    "outpid": "2750",
                    "inchan": "2750",
                    "type": "SUBT",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                }
            ],
            "collision": {
                "status": "No Record in Dynamic Table"
            }
        }
    }
}
```

ASI Output Status Command

Table 2.71 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/output
Command Information	This command is used to get ASI Output status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/output/asi", GET "https://192.168.0.1/ws/v2/status/output/asi1", or

Command Detail	Description
	GET "https://192.168.0.1/ws/v2/status/output/asi2"

Table 2.72 URI Parameters (extension to the Command URL separated by /)

URI Parameter	Description
asi/asi1/asi2	ASI output status On NTC Basic or NTC MOIP supported values are asi or asi2 for the unique asi output. On NTC MS supported values are asi1 for bidirectional port 1 and asi2 for bidirectional port 2.

Table 2.73 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.74 Output Field Descriptions

Output Field	Description
rate	Output Rate (Mbps) Type: Float Values: 0.0 ….. 206.0
free	Free Bandwidth (Mbps) Type: Float Values: 0.0 ….. 206.0

Example on NTC single-stream unit, where no ASI output configured (request XML format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/asi"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<output>
    <asi>
    <rate>0.0</rate>
    <free>0.0</free>
        <pid>
        <status>No Record in Dynamic Table</status>
        </pid>
        <collision>
        <status>No Record in Dynamic Table</status>
    </collision>
</asi>
</output>
```

Example on NTC MS unit in JSON format:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/asi2?js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "asi2": {
            "rate": "78.535695",
            "free": "0.0",
            "pid": [
                {
                    "peid": "1",
                    "row": "1",
                    "inpid": "1500",
                    "outpid": "1500",
                    "inchan": "1500",
                    "type": "VID",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "2",
                    "inpid": "3349",
                "outpid": "3349",
                "inchan": "3349",
                "type": "LSDT",
                "received": "Yes",
                "pcr": "No",
                "scrambled": "No"
            },
            {
                "peid": "1",
                "row": "3",
                "inpid": "4001",
                "outpid": "4001",
                "inchan": "4001",
                "type": "MPE",
                "received": "Yes",
                "pcr": "No",
                "scrambled": "No"
            },
            {
                "peid": "1",
                "row": "4",
                "inpid": "4002",
                "outpid": "4002",
                "inchan": "4002",
                "type": "MPE",
                "received": "Yes",
                "pcr": "No",
                "scrambled": "No"
            },
            {
                "peid": "1",
                "row": "5",
                "inpid": "4003",
                "outpid": "4003",
                "inchan": "4003",
                "type": "MPE",
                "received": "Yes",
                "pcr": "No",
                "scrambled": "No"
            },
            {
                "peid": "1",
                "row": "6",
                "inpid": "4004",
                "outpid": "4004",
                "inchan": "4004",
                "type": "MPE",
                "received": "Yes",
                "pcr": "No",
                "scrambled": "No"
            },
            {
                "peid": "1",
                "row": "7",
                "inpid": "4005",
                "outpid": "4005",
                "inchan": "4005",
                "type": "MPE",
                "received": "Yes",
                "pcr": "No",
                "scrambled": "No"
            },
            {
                "peid": "1",
                "row": "8",
                "inpid": "1550",
                "outpid": "1550",
                "inchan": "1550",
                "type": "DPI",
                "received": "No",
                "pcr": "No",
                "scrambled": "No"
            },
            {
                "peid": "1",
                "row": "9",
                "inpid": "2906",
                "outpid": "2906",
                "inchan": "2906",
                "type": "VBI",
                "received": "No",
                "pcr": "No",
                "scrambled": "No"
            },
            {
                "peid": "1",
                "row": "10",
                "inpid": "1511",
                "outpid": "1511",
                "inchan": "1511",
                "type": "AUD",
                "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "11",
                    "inpid": "1501",
                    "outpid": "1501",
                    "inchan": "1501",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "12",
                    "inpid": "1102",
                    "outpid": "1102",
                    "inchan": "1102",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "13",
                    "inpid": "1113",
                    "outpid": "1113",
                    "inchan": "1113",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "14",
                    "inpid": "2750",
                    "outpid": "2750",
                    "inchan": "2750",
                    "type": "SUBT",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                }
            ],
            "collision": {
                "status": "No Record in Dynamic Table"
            }
        }
    }
}
```

MOIP Output Status

Table 2.75 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/output/moip
Command Information	This command is used to get MOIP output status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/output/moip", GET "https://192.168.0.1/ws/v2/status/output/moip1", or GET "https://192.168.0.1/ws/v2/status/output/moip2"

Table 2.76 URI Parameters (extension to the Command URL separated by /)

URI Parameter	Description
moip/moip1/moip2	MOIP output status. On NTC Basic, this call is not supported. On NTC MOIP, the supported values are moip or moip1 for the unique IP output. On NTC multi-stream unit, the supported values are moip1 for IP output 1 and moip2 for IP output 2.

Table 2.77 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard

URI Parameter	Description
	Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.78 Ouput Field Descriptions

Output Field	Description
enginerr	Critical MOIP Engine Error Type: Boolean Values: "Yes" or "No"
overflow	MOIP TS stream Overflow Type: Boolean Values: "Yes" or "No"
comrate	Combined user rate (Mbps) Type: Float Values: 0.0 ….. 206.0
ts1rate	Actual TS1 port user rate (Mbps) Type: Float Values: 0.0 ….. 206.0
ts2rate	Actual TS2 port user rate (Mbps) Type: Float Values: 0.0 ….. 206.0

Example on NTC single-stream unit:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/moip"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<output>
<moip>
    <enginerr>No</enginerr>
    <overflow>No</overflow>
    <comrate>68.5</comrate>
    <ts1rate>0.0</ts1rate>
    <ts2rate>0.0</ts2rate>
    <pid>
        <peid>1</peid>
        <row>1</row>
        <inpid>2100</inpid>
            <outpid>2100</outpid>
            <inchan>2100</inchan>
            <type>VID</type>
            <received>Yes</received>
            <pcr>Yes</pcr>
            <scrambled>No</scrambled>
        </pid>
        <pid>
            <peid>1</peid>
            <row>2</row>
            <inpid>2110</inpid>
            <outpid>2110</outpid>
            <inchan>2110</inchan>
            <type>AUD</type>
            <received>Yes</received>
            <pcr>No</pcr>
            <scrambled>No</scrambled>
        </pid>
        <collision>
            <status>No Record in Dynamic Table</status>
        </collision>
    </moip>
</output>
```

Example on NTC multi-stream unit:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/moip1"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<output>
<moip1>
    <enginerr>No</enginerr>
    <overflow>No</overflow>
    <comrate>68.5</comrate>
    <ts1rate>0.0</ts1rate>
    <ts2rate>0.0</ts2rate>
    <pid>
        <peid>1</peid>
        <row>1</row>
        <inpid>2100</inpid>
        <outpid>2100</outpid>
        <inchan>2100</inchan>
        <type>VID</type>
        <received>Yes</received>
        <pcr>Yes</pcr>
        <scrambled>No</scrambled>
    </pid>
    <pid>
        <peid>1</peid>
        <row>2</row>
        <inpid>2110</inpid>
        <outpid>2110</outpid>
        <inchan>2110</inchan>
        <type>AUD</type>
        <received>Yes</received>
        <pcr>No</pcr>
        <scrambled>Yes</scrambled>
    </pid>
    <collision>
        <status>No Record in Dynamic Table</status>
    </collision>
    </moip1>
</output>
```

DPM PID Output Status Command

Table 2.79 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/output/<level_1>/<level_2>
Command Information	This command can be used to read **all** ASI/MOIP DPM PMT PID Output Status parameters. **Note** Partial response is not supported. This command returns the status of **all** active peid and row key combinations for all output fields for the specified MOIP or ASI.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax for getting MOIP status output	GET "https://192.168.0.1/ws/v2/status/output/moip/pid",

Command Detail	Description
parameters	GET "https://192.168.0.1/ws/v2/status/output/moip1/pid", or GET "https://192.168.0.1/ws/v2/status/output/moip2/pid"
Syntax for getting ASI status output parameters	GET "https://192.168.0.1/ws/v2/status/output/asi/pid", GET "https://192.168.0.1/ws/v2/status/output/asi1/pid", or GET "https://192.168.0.1/ws/v2/status/output/asi2/pid"

Table 2.80 URI <level_1> URI Parameters (extension to the Command URL separated by /)

URI Level 1 Parameter	Description
asi/asi1/as2	Output type is ASI. On NTC Basic or NTC MOIP, the supported values are asi or asi2 for the unique asi output. On NTC MS, the supported values are asi1 for bidirectional port 1 and asi2 for bidirectional port 2.
moip/moip1/moip2	Output type is MOIP. On NTC Basic, this call is not supported On NTC MOIP, the supported values are moip or moip1 for the unique IP output. On NTC MS, the supported values are moip1 for IP output 1 and moip2 for IP output 2.

Table 2.81 URI <level_2> URI Parameters (extension to the Command URL separated by /)

URI Level 2 Parameter	Description
pid	DPM PMT PID information.

Table 2.82 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid)

URI Parameter	Description
	Omitting this argument formats the output by default in XML.

Table 2.83 Output Field Descriptions

Output Field	Description
peid (Key)	Program Entry ID Type: Integer Values: 1..32
row (Key)	Identity of service in PMT Type: Integer Values: 1..64
inpid	Input PMT Pid Type: Integer Values: 0 .. 8191
outpid	Output PMT Pid Type: Integer Values: 0 .. 8191
type	Stream Category Type: String Values: "INVL", "PCR", "VID", "AUD", "SUBT", "VBI", "DPI", "MPE", "TTX", "DATA", "LSDT", "CDT", "ETV", "UNKN"
received	Input Received Type: Boolean Values: "Yes", "No"
pcr	PCR Type: Boolean Values: "Yes", "No"
scrambled	Output Scrambled Type: Boolean Values: "Yes", "No"
status	Indicate the API output status

Output Field	Description
	Type: String Values: text to indicate the output status **Note** The PID information is dynamic data. If there is no PID information in UIC table, the "status" field will indicate the reason.

ASI Example on NTC single-stream unit (for which ASI output was not configured):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/asi/pid&js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "asi": {
            "pid": {
                "status": "No Record in Dynamic Table"
            }
        }
    }
}
```

ASI Example on NTC single-stream unit (for which ASI output was configured):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/asi/pid&js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "asi": {
            "pid": [
                {
                    "peid": "1",
                    "row": "1",
                    "inpid": "2200",
                    "outpid": "2200",
                    "type": "VID",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "2",
                    "inpid": "2210",
                    "outpid": "2210",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "3",
                    "inpid": "18",
                    "outpid": "18",
                    "type": "UNKN",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                }
            ]
        }
    }
}
```

MOIP example on NTC single-stream unit (for which MOIP output was configured):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/moip/pid&js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "moip": {
            "pid": [
                {
                    "peid": "1",
                    "row": "1",
                    "inpid": "2200",
                    "outpid": "2200",
                    "type": "VID",
                    "received": "Yes",
                    "pcr": "Yes",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "2",
                    "inpid": "2210",
                    "outpid": "2210",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "3",
                    "inpid": "18",
                    "outpid": "18",
                    "type": "UNKN",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                }
            ]
        }
    }
}
```

ASI Example on NTC multi-stream unit (for which ASI2 output was configured):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/asi2/pid&js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "asi2": {
            "pid": [
                {
                    "peid": "1",
                    "row": "1",
                    "inpid": "2200",
                    "outpid": "2200",
                    "type": "VID",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": ""
                },
                {
                    "peid": "1",
                    "row": "2",
                    "inpid": "2210",
                    "outpid": "2210",
                    "type": "AUD",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": ""
                },
                {
                    "peid": "1",
                    "row": "3",
                    "inpid": "18",
                    "outpid": "18",
                    "type": "UNKN",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": ""
                }
            ]
```

MOIP Example on NTC multi-stream unit (for which MOIP1 output was configured):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/moip1/pid&js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "moip1": {
            "pid": [
                {
                    "peid": "1",
                    "row": "1",
                    "inpid": "2200",
                    "outpid": "2200",
                    "type": "VID",
                    "received": "Yes",
                    "pcr": "Yes",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "2",
                    "inpid": "2210",
                    "outpid": "2210",
                    "type": "AUD",
                    "received": "Yes",
                    "pcr": "No",
                    "scrambled": "No"
                },
                {
                    "peid": "1",
                    "row": "3",
                    "inpid": "18",
                    "outpid": "18",
                    "type": "UNKN",
                    "received": "No",
                    "pcr": "No",
                    "scrambled": "No"
                }
            ]
        }
    }
}
```

DPM Active Collisions Status Command

Table 2.84 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/output/<level_1>/<level_2>
Command Information	This command is used to read **all** the ASI or MOIP DPM active collision statuses. **Note** Partial response is not supported. This command returns the status of **all** active peid and row key combinations for all output fields for the specified MOIP or ASI.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax for getting MOIP status output parameters	GET "https://192.168.0.1/ws/v2/status/output/moip/collision", GET "https://192.168.0.1/ws/v2/status/output/moip1/collision", or GET "https://192.168.0.1/ws/v2/status/output/moip2/collision"
Syntax for getting ASI status output parameters	GET "https://192.168.0.1/ws/v2/status/output/asi/collision", GET "https://192.168.0.1/ws/v2/status/output/asi1/collision", or GET "https://192.168.0.1/ws/v2/status/output/asi2/collision"

Table 2.85 URI <level_1> URI Parameters (extension to the Command URL separated by /)

URI Level 1 Parameter	Description
asi/asi1/as2	Output type is ASI. On NTC Basic or NTC MOIP, the supported values are asi or asi2 for the unique asi output. On NTC MS, the supported values are asi1 for bidirectional port 1 and asi2 for bidirectional port 2.
moip/moip1/moip2	Output type is MOIP. On NTC Basic, this call is not supported. On NTC MOIP, the supported values are moip or moip1 for the unique IP output/ On NTC MS, the supported values are moip1 for IP output 1 and moip2 for IP output 2/

Table 2.86 URI <level_2> URI Parameters (extension to the Command URL separated by /)

URI Level 2 Parameter	Description
collision	DPM Active collision information.

Table 2.87 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.88 Output Field Descriptions

URI Level 2 Parameter	Description
idx (Key)	Index Type: Integer

URI Level 2 Parameter	Description
	Values: 0 …65535
type	Conflict type Type: String Values: "None", "Channel collision", "PMT-PMT PID collision", "PMT-ES PID collision", "ES-ES PID collision", "Illegal Mode-i PMT PID"
peidx1	PE index-1 Type: Integer Values: 1..32
pmtrow1	PMT Row-1 Type: Integer Values: 1 … 65
peidx2	PE index-2 Type: Integer Values: 1..32
pmtrow2	PMT Row-2 Type: Integer Values: 1 … 65
value	Conflict Value Type: Integer Values: 0 … 65535
status	Indicate the API output status Type: String Values: text to indicate the output status **Note** DPM active collision is dynamic data. If there is no collision information in UIC table, the "status" field will indicate the reason.

ASI example on single-stream unit (for which ASI output was configured):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/asi/collision&js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "asi": {
            "collision": {
                "status": "No Record in Dynamic Table"
            }
        }
    }
}
```

MOIP example on single-stream unit (for which MOIP output was configured):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/moip/collision&js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "moip": {
            "collision": {
                "status": "No Record in Dynamic Table"
            }
        }
    }
}
```

ASI Example on NTC multi-stream unit (for which ASI1 output was configured):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/asi1/collision&js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "asi1": {
            "collision": {
                "status": "No Record in Dynamic Table"
            }
        }
    }
}
```

MOIP Example on NTC multi-stream unit (for which MOIP2 output was configured):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/output/moip2/collision&js=1"
```

Expected output (values are for example purposes only):

```
{
    "output": {
        "moip2": {
            "collision": {
                "status": "No Record in Dynamic Table"
            }
        }
    }
}
```

Video Decode Status Command

Table 2.89 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/decode/video
Command Information	This command can be used to read all video decode status. **Note** Partial response is not supported. This command returns the status of **all** video decode output fields.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/decode/video"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.90 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.91 Output Field Descriptions

Output Field	Description
stream	Input Stream Format Type: String Values: "SD480i/2997", "SD480p/2997", "SD480i/3000", "SD576i/2500", "SD576p/2500", "HD720p/5000", "HD720p/5994", "HD720p/6000", "HD1080i/2500", "HD1080i/2997", "HD1080i/3000", "Unsupported"
pvoformat	PV Output Format Type: String Values: "HD1080i", "HD720p", "SD", "HD1080p"
sdoformat	SD Video Output Format Type: String Values: "PAL-B/G/I/D", "PAL-M", "PAL-N (AR)", "NTSC"
bitrate	Bitrate in bps Type : Integer Values: 0.. 4294967295
pulldown32	3:2 pulldown Type: String Values: "Yes", "No", or "Recent"
frmpersec	Frames per second Type : Integer Values: 0.. 4294967295
syncmode	Synch mode Type: String Values: "Auto", "Manual"
encoding	Encoding standard Type: String Values: "MPEG1", "MPEG2", "H264", "VC1", "MPEG_P2", or "Unknown"

Example:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/video"
```

Audio Decode Status Command

Table 2.92 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/decode/audio
Command Information	This command is used to read **all** the audio decode statuses. **Note** Partial response is not supported. This command returns the status of **all** audio decode output fields and ignores unsupported Query Strings.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/decode/audio"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.93 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
device	Audio Device Instance Type: Integer Value: 1 .. 4
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Other than the device key, the following Output Field elements may NOT be used as Query Strings to filter output for this API:

Table 2.94 Output Field Descriptions

Output Field	Description
device	Audio Device Instance Type: Integer Value: 1 .. 4
mode	Audio Mode Type: String Value: "None", "Sine", "Pink", "Beep", "MPEG1L1", "MPEG1L2", "MPEG2L1", "MPEG2L2", "AC3", "LOAS AAC", "ADTS AAC", "LOAS HEAAC", "ADTS HEAAC", "DDP", "ST302"
bitrate	Bitrate in bps Type : Integer Values: 0.. 4294967295
bufferlvl	Buffer level Type : Integer Values: 0.. 4294967295
sfr	Sample rate of input audio stream Type: Double (one decimal point) Value: default: 0.0 minimum: 0.0 maximum: 99.9
pid	Transport stream packet identifier Type: String Value: Up to 4 digits. May be empty.
ddpmode	Dolby Digital Mode Type: String Value: "Trans", "Pass"
dualmono	Audio Dualmono Mode Type: String Value: "OFF", "DUAL-MONO"
lang	Language Value.

Output Field	Description
	Type: String Value: Any valid 3 character language code, for example: "eng". May be empty.
bitspersample	Bits Per Sample Type: String Value: "Unknown", "16", "20", "24"

Get all Audio Status in XML format example:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/audio"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<decode>
<audio>
    <device>1</device>
    <mode>None</mode>
    <bitrate>0</bitrate>
    <bufferlvl>0</bufferlvl>
    <sfr>0.0</sfr>
    <pid>
    </pid>
    <lang>
    </lang>
    <ddpmode>OFF</ddpmode>
    <dualmono>OFF</dualmono>
</audio>
<audio>
    <device>2</device>
    <mode>None</mode>
    <bitrate>0</bitrate>
    <bufferlvl>0</bufferlvl>
    <sfr>0.0</sfr>
    <pid>
    </pid>
    <lang></lang>
    <ddpmode>OFF</ddpmode>
    <dualmono>OFF</dualmono>
</audio>
</decode>
```

Get Audio Device 1 status in JSON format:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/audio?device=1&js=1"
```

Expected output (values are for example purposes only):

```
{
    "decode": {
        "audio": {
            "device": "1",
            "mode": "None",
            "bitrate": "0",
            "bufferlvl": "0",
            "sfr": "0.0",
            "pid": "",
            "lang": "",
            "ddpmode": "OFF",
            "dualmono": "OFF"
        }
    }
}
```

Audio STC302 Decode Status Command

Table 2.95 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/decode/audio/st302
Command Information	This command can be used to read **all** ST302 audio decode status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/decode/aud_st302"

URI Parameters (extension to the Command URL separated by /): N/A

The following Output Field elements may NOT be used as Query Strings to filter output for this API:

Table 2.96 Output Field Descriptions

Output Field	Description
device	ST302 Device Instance Type: Integer Value: 1
streamformat	ST302 stream format

Output Field	Description
	Type: String Value: "Unknown", "Dolby-E"
numstreams	Number of ST302 Streams (each consisting of 2 channels) Type : String Values: "0", "1", "2", "3", "4"
packetsize	ST302 Stream Packet Size Type : String Values: "Unknown", "1920", "1620/1621"
dolbyconfig	Dolby-E Program Configuration Type: String Value: one of the following options: "Unknown", "5.1 + 2", "5.1 + 2x1", "4 + 4", "4 + 2x2", "4 + 2 + 2x1", "4 + 4x1", "4x2", "3x2 + 2x1", "2x2 + 4x1", "2 + 6x1", "8x1", "5.1", "4 + 2", "4 + 2x1", "3x2", "2x2 + 2x1", "2 + 4x1", "6x1", “4”, "2 + 2", "2 + 2x1", "4x1", "7.1", "7.1 Screen", "PCM Bypass".

Get Audio ST302 Status in JSON format:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/aud_st302
```

Expected output (values are for example purposes only):

```
{
    "decode": {
        "aud_st302": {
            "device": "1",
            "streamformat": "Unknown",
            "numstreams": "0",
            "packetsize": "Unknown",
            "dolbyconfig": "Unknown"
        }
    }
}
```

Closed Caption (CC) Decode Status Command

Table 2.97 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/decode/cc
Command Information	This command can be used to read **all** Closed Caption decode status. **Note** Partial response is not supported. This command returns the status of **all** closed caption decode output fields and ignores unsupported Query Strings..
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/decode/cc"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.98 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &):

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.99 Output Field Descriptions

Output Field	Description
ccoutact	Actual Closed Caption Output Type: String Value: "Auto", "SA Custom", "EIA 708", "Type 3", "Type 4 SA", "Type 4 ATSC", "Reserved", "DVS 157", "DirectTV Type 3"

Input example (request default XML format:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/cc"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<decode>
<cc>
    <ccoutact>Auto</ccoutact>
</cc>
</decode>
```

Input example (request JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/cc?js=1"
```

Expected output (values are for example purposes only):

```
{
    "decode": {
        "cc": {
            "ccoutact": "Auto"
        }
    }
}
```

VBI Decode Status Command

Table 2.100 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/decode/vbi
Command Information	This command is used to read **all** VBI decode statuses. **Note** Partial response is not supported. This command returns the status of **all** closed caption decode output fields and ignores unsupported Query Strings..
HTTP Method	GET
Access Type	Read

Command Detail	Description
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/decode/vbi"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.101 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.102 Output Field Descriptions

Output Field	Description
vitcstatus	VIT Status Type: String Values: "LTC", "VITC", "BOTH", "UNDEFINED"

Input Example (request default XML format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/vbi"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<decode>
<vbi>
    <vitcstatus>VITC</vitcstatus>
</vbi>
</decode>
```

Input Example (request JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/vbi?js=1"
```

Expected output (values are for example purposes only):

```
{
    "decode": {
        "vbi": {
            "vitcstatus": "VITC"
        }
    }
}
```

SDI Decode Status Command

Table 2.103 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/decode/<level_1_parm>
Command Information	This command is used to read **all** SDI global or SDI services decode status. **Note** Partial response is not supported. This command returns the status of **all** decode output fields for the specified level_1_parm SDI type fields and ignores unsupported Query Strings.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/decode/sdigglb" or GET "https://192.168.0.1/ws/v2/status/decode/sdisvc"

Table 2.104 URI Parameters (extension to the Command URL separated by /)

URI Parameter	Description
sdiglb	Return SDI Global status
sdisvc	Return SDI Services status

Table 2.105 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

The following Output Field elements may NOT be used as Query Strings to filter output for this API:

Table 2.106 Output Field Descriptions when requesting SDI Global Status (sdiglb)

Output Field	Description
interlaced	Indicates if video is interlaced Type: String Values: "Yes" or "No"
framessec	Video frames per second Type: Integer Values: 0 ... 4294967295
lines	Lines per video frame Type : Integer Values: 0 ... 4294967295
wordsline	VANC words per line Type : Integer Values: 0 ... 4294967295
firstline	VANC area: first line Type : Integer Values: 0 ... 4294967295
lastline	VANC area: last line Type: Integer Values: 0 ... 4294967295
switchline	VANC area: switch line Type: Integer Values: 0 ... 4294967295

Output Field	Description
multiline	Loop of service types Type: String Values: "Yes" or "No"

The following Output Field elements may NOT be used as Query Strings to filter output for this API:

Table 2.107 Output Field Descriptions when requesting SDI Services Status (sdisvc)

Output Field	Description
serviceid	VANC Service identifier "EIA-708", "AFD", "DPI", "SMPTE-2031", "SDP-OP47", or "MULTI-OP47"
active	Service is being output Type: String Values: "Yes" or "No"
adjline	Line number where VANC service is expected Type: Integer Values: 0 ... 4294967295
topline	Line number where VANC service is located (in top field or frame) Type: Integer Values: 0 ... 4294967295
bottomline	Line number where VANC service is located (in bottom field) Type: Integer Values: 0 ... 4294967295
maxlines	Maximum number of lines used by VANC service Type: Integer Values: 0 ... 4294967295
avgdata	Average size of VANC service data per frame Type: Integer Values: 0 ... 4294967295
avgtranpkt	Average number of transmitted VANC packets per frame Type: Integer Values: 0 ... 4294967295
avgdroppkt	Average number of dropped VANC packets per frame

Output Field	Description
	Type: Integer Values: 0 ... 4294967295

Request SDI Global status (with result in JSON format) example:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/sdiglb"
```

Expected output (values are for example purposes only):

```
{
    "decode": {
        "sdiglb": {
            "interlaced": "No",
            "framessec": "0.0",
            "lines": "0",
            "wordsline": "0",
            "firstline": "0",
            "lastline": "0",
            "switchline": "0",
            "multiline": "No"
        }
    }
}
```

Request SDI Services status (with result in JSON format) example:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/sdisvc"
```

Expected output (values are for example purposes only):

```
{
    "decode": {
        "sdisvc": [
            {
                "serviceid": "EIA-708",
                "active": "No",
                "adjline": "2",
                "topline": "0",
                "bottomline": "0",
                "maxlines": "0",
                "avgdata": "0",
                "avgtranpkt": "0",
                "avgdroppkt": "0"
            },
            {
                "serviceid": "AFD",
                "active": "No",
                "adjline": "2",
                "topline": "0",
                "bottomline": "0",
                "maxlines": "0",
                "avgdata": "0",
                "avgtranpkt": "0",
                "avgdroppkt": "0"
            },
            {
                "serviceid": "DPI",
                "active": "No",
                "adjline": "3",
                "topline": "0",
                "bottomline": "0",
                "maxlines": "0",
                "avgdata": "0",
                "avgtranpkt": "0",
                "avgdroppkt": "0"
            },
            {
                "serviceid": "SMPTE-2031",
                "active": "No",
                "adjline": "4",
                "topline": "0",
                "bottomline": "0",
                "maxlines": "0",
                "avgdata": "0",
                "avgtranpkt": "0",
                "avgdroppkt": "0"
            },
            {
                "serviceid": "SDP-OP47",
                "active": "No",
                "adjline": "5",
                "topline": "0",
                "bottomline": "0",
                "maxlines": "0",
                "avgdata": "0",
                "avgtranpkt": "0",
                "avgdroppkt": "0"
            },
            {
                "serviceid": "MULTI-OP47",
                "active": "No",
                "adjline": "6",
                "topline": "0",
                "bottomline": "0",
                "maxlines": "0",
                "avgdata": "0",
                "avgtranpkt": "0",
                "avgdroppkt": "0"
            }
        ]
    }
}
```

HDMI Decode Status Command

Table 2.108 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/decode/hdmi
Command Information	This command is used to read **all** or single device HDMI decode status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/decode/hdmi" or GET "https://192.168.0.1/ws/v2/status/decode/hdmi?hdmiindex=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.109 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
hdmiindex	HDMI device (instance) number Type: integer Values: 1 … 2
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML

Other than hdmiindex, the following Output Field elements may NOT be used as Query Strings to filter output for this API:

Table 2.110 Output Field Description

Output Field	Description
hdmiindex	HDMI device (instance) number Type: integer Values: 1 … 2
connected	HDMI output connected to monitor Type: String Values: "Yes", "No"
powered	HDMI device powered Type : String Values: "Yes", "No"
monitorname	Name of monitor name connected to HDMI output Type : String Values: Text, up to 40 characters
colorspace	HDMI Color Space Type: String Value: "Unknown", "YCbCr420", "YCbCr444", "YCbCr422", "RGB"
colordepth	HDMI Color Depth Type: String Value: "Unknown", "16bit", "12bit", "10bit", "8bit"
colorrange	HDMI Color Range Type: String Value: "Auto", "Full", "Limited"
matrixcoeff	HDMI Matrix Coefficients Type: String Value: "Unknown", "ITU-2020-CL", "ITU-2020-NCL", "ITU-470", "XvYCC-709", "XvYCC-601", "ITU-709", "SMPTE-170M"

Output Field	Description
eotf	HDMI EOTF (Electro-Optical Transfer Function) Type: String Value: "Unknown", "SMPTE-2084", "HDR", "SDR"

Input example (request JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/hdmi?js=1"
```

Expected output (values are for example purposes only):

```
{
    "decode": {
        "hdmi": [
            {
               "hdmiindex": "1",
               "connected": "No",
               "powered": "No",
               "monitorname": "Unknown",
               "colorspace": "Unknown",
               "colordepth": "Unknown",
               "colorrange": "Unknown",
               "matrixcoeff": "Unknown",
               "eotf": "Unknown"
               },
            {
               "hdmiindex": "2",
               "connected": "No",
               "powered": "No",
               "monitorname": "Unknown",
               "colorspace": "Unknown",
               "colordepth": "Unknown",
               "colorrange": "Unknown",
               "matrixcoeff": "Unknown",
               "eotf": "Unknown"
               }
       ]
       }
}
```

Input example (request JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/decode/hdmi?hdmiindex=1&js=1"
```

Expected output (values are for example purposes only):

```
{
    "decode": {
        "hdmi": {
            "hdmiindex": "1",
            "connected": "No",
            "powered": "No",
            "monitorname": "Unknown",
            "colorspace": "Unknown",
            "colordepth": "Unknown",
            "colorrange": "Unknown",
            "matrixcoeff": "Unknown",
            "eotf": "Unknown"
        }
    }
}
```

Identity Status Command

The indentity status command is enhanced in Version 3.25 or later.

Note The identity status command is supported, but we recommend that you use licenses, device identity, and hardware options commands instead.

Table 2.111 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/identity
Command Information	This command is used to read **all** identity details. **Note** Partial response is not supported prior to Version 3.25. In previous software releases, this command simultaneously returns the status of **all** identity details output fields and ignored any Query/Set Arguments other than “js” or “session”.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/identity"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.112 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

In Version 3.25 and later, the items in the first column of the following table may be used as Query Parameters to restrictively filter the output. For example, adding ?masterua&hostname to the command line will result in output that only provides the data corresponding to those fields.

Table 2.113 Output Field Descriptions

Output Field	Description
hostname	Host name Type: String Value: Maximum of 64 characters
trackingid	Tracking ID Type: String Value: Maximum of 13 characters
masterua	Master user address Type: String Value: Maximum of 15 characters
CtlMacAddr	Ethernet control port MAC address Type: String Value: Maximum of 18 characters (format “00:00:00:00:00:00”)
hwoptions	
mpoipout	MPEG Over IP Output Supported Type: String Value: "Yes" (if "No" then element will not be output)
mpoipin	MPEG Over IP Input Supported Type: String

Output Field	Description
	Value: "Yes" (if "No" then element will not be output)
moipfec	MPEG Over IP FEC Supported Type: String Value: "Yes" (if "No" then element will not be output)
sdi	Serial Digital Interface Supported Type: String Value: "Yes" (if "No" then element will not be output)
Licenses **Note** The field returned is maximum of 8 characters regardless of whether or not it represents a numeric value.	
HD Transcode License Count	Number of HD transcode licenses Type: String Value: 1..N (if 0 then element will not be output)
SD Transcode License Count	Number of SD transcode licenses Type: String Value: 0..N (if 0 then element will not be output)
HEVC Transcode License Count	Number of HEVC transcode licenses Type: String Value: 1..N (if 0 then element will not be output)
Tuner License Count	Number of tuner licenses Type: String Value: 1..N (if 0 then element will not be output)
Decrypt License	Licensed for decryption Type: String Value: "Yes" (if "No" then element will not be output)
DVB Decrypt	Licensed for DVB decryption Type: String Value: "Yes" (if "No" then element will not be output)
DVB_S2 License	Licensed for DVB S2 Type: String

Output Field	Description
	Value: "Yes" (if "No" then element will not be output)
Tuner APSK License	Licensed for tuner APSK Type: String Value: "Yes" (if "No" then element will not be output)
AVC Decode License	Licensed for AVC decode Type: String Value: "Yes" (if "No" then element will not be output)
HEVC Decode License	Licensed for HEVC decode Type: String Value: "Yes" (if "No" then element will not be output)
HD Output License	Licensed for HD output Type: String Value: "Yes" (if "No" then element will not be output)
Full HD Output License	Licensed for full HD output Type: String Value: "Yes" (if "No" then element will not be output)
UHD Output License	Licensed for UHD output Type: String Value: “Yes” (if “No” then element will not be output)
AUD3_4 License	Licensed for audio 3 and 4 output Type: String Value: “Yes” (if “No” then element will not be output)

licensefile

These elements are the same data as "licenses" above but only show what is in the original license file (s). For example, default licenses have not been added. Also, the element tag is as it appears in the license file.

Note The field returned is maximum of 8 characters regardless of whether or not it represents a numeric value.

HD_Transcode	Number of HD transcode licenses Type: String

Output Field	Description
	Value: 1..N (if 0 then element will not be output)
SD_Transcode	Number of SD transcode licenses Type: String Value: 1..N (if 0 then element will not be output)
HEVC_Xcode	Number of HEVC transcode licenses Type: String Value: 1..N (if 0 then element will not be output)
Tuner	Number of tuner licenses Type: String Value: 1..N (if 0 then element will not be output)
C_DECRYPT	Licensed for decryption Type: String Value: "1" (if "0" then element will not be output)
C_D_DECRYPT	Licensed for DVB decryption Type: String Value: "1" (if "0" then element will not be output)
C_DVB_S2	Licensed for DVB S2 Type: String Value: "1" (if "0" then element will not be output)
C_Tuner_APSK	Licensed for tuner APSK Type: String Value: "1" (if "0" then element will not be output)
C_AVC_Decode	Licensed for AVC decode Type: String Value: "1" (if "0" then element will not be output)
C_HEVC_Decode	Licensed for HEVC decode Type: String Value: "1" (if "0" then element will not be output)
C_HD_Output	Licensed for HD output Type: String

Output Field	Description
	Value: "1" (if "0" then element will not be output)
C_Full_HD_Output	Licensed for full HD output Type: String Value: "1" (if "0" then element will not be output)
C_UHD_Output	Licensed for UHD output Type: String Value: "1" (if "0" then element will not be output)
C_AUD3_4	Licensed for audio 3 and 4 output Type: String Value: "1" (if "0" then element will not be output)

Input example (request default XML format):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/status/identity"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<ident><hostname>User-cfg-name</host
name><trackingid>456JK123YX</trackingid><masterua>123-123-4567-
1</masterua><ctlm

acaddr>38:C8:5D:FF:AF:E3</ctlmacaddr><hwoptions><mpoipout>Yes</mpoipout><mpoi
pin

>Yes</mpoipin><moipfec>Yes</moipfec><sdi>Yes</sdi></hwoptions><licenses><feat
ure
>Tuner License Count</feature><enabled>4</enabled><feature>DVB_S2
License</featu
re><enabled>Yes</enabled><feature>AVC Decode
License</feature><enabled>Yes</enab
led><feature>HEVC Decode License</feature><enabled>Yes</enabled><feature>HD
Outp
ut License</feature><enabled>Yes</enabled><feature>Full HD Output
License</featu
re><enabled>Yes</enabled><feature>AUD3_4
License</feature><enabled>Yes</enabled>

</licenses><licensefile><feature>Tuner</feature><enabled>3</enabled><feature>
C_DVB
_S2</feature><enabled>1</enabled><feature>C_AVC_
```

```
Decode</feature><enabled>1</enable
d><feature>C_HEVC_Decode</feature><enabled>1</enabled><feature>C_HD_
Output</feature>
<enabled>1</enabled><feature>C_Full_HD_
Output</feature><enabled>1</enabled><featur
e>C_AUD3_4</feature><enabled>1</enabled></licensefile>
</ident>
```

Input (request JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/identity?js=1"
```

Expected output (values are for example purposes only):

```
{
    "ident": {
        "hostname": "User-cfg-name",
        "trackingid": "456JK123YX",
        "masterua": "123-123-4567-1",
        "ctlmacaddr": "38:C8:5D:FF:AF:E3",
        "hwoptions": {
            "mpoipout": "Yes",
            "mpoipin": "Yes",
            "moipfec": "Yes",
            "sdi": "Yes"
        },
        "licenses": {
            "feature": "Tuner License Count",
            "enabled": "4",
            "feature": "DVB_S2 License",
            "enabled": "Yes",
            "feature": "AVC Decode License",
            "enabled": "Yes",
            "feature": "HEVC Decode License",
            "enabled": "Yes",
            "feature": "HD Output License",
            "enabled": "Yes",
            "feature": "Full HD Output License",
            "enabled": "Yes",
            "feature": "AUD3_4 License",
            "enabled": "Yes"
        },
        "licensefile": {
            "feature": "Tuner",
            "enabled": "3",
            "feature": "C_DVB_S2",
            "enabled": "1",
            "feature": "C_AVC_Decode",
            "enabled": "1",
            "feature": "C_HEVC_Decode",
            "enabled": "1",
            "feature": "C_HD_Output",
            "enabled": "1",
            "feature": "C_Full_HD_Output",
            "enabled": "1",
            "feature": "C_AUD3_4",
            "enabled": "1"
        }
    }
}
```

Licenses Command

Table 2.114 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/licenses
Command Information	This command can be used to read information about installed licenses and license files.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/licenses"

Parameters (extension to the Command URL separated by /): N/A

Table 2.115 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in JSON

The items in the first column of the following table may be used as Query Parameters to restrictively filter the output. For example, adding ?masterua&hostname to the command line will result in output that only provides the data corresponding to those fields.

Table 2.116 Output Field Descriptions

Output Field	Description	
licenses Contains multiple pairs <"feature", "enabled">, one per installed license.	feature	Name of licensed feature, for example, "DVB_S2 License" Type: String Value: 1..30 characters
	enabled	Number of supported 'feature' element [s]. Type: String Value: "Yes", "No" (for single instance), or count 1..N
licensefiles These elements are the same data as "licenses" above but only show what is in the original license file(s). (for example, default licenses have not been added). Also, the element tag is as it appears in the license file.	feature	Name of licensed feature, e.g. "Tuner" Type: String Value: 1..128 characters
	enabled	Number of supported 'feature' element [s]. Type: Integer Value: count 1..N

Input Example (request default XML format):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/status/licenses"
```

Expected output (values are for example purposes only):

```
{
    "licenseinfo": {
        "licenses": [
            {
                "feature": "Tuner License Count",
                "enabled": "7"
            },
            {
                "feature": "DVB_S2 License",
                "enabled": "Yes"
            },
            {
                "feature": "Tuner APSK License",
                "enabled": "Yes"
            },
            {
                "feature": "AVC Decode License",
                "enabled": "Yes"
            },
            {
                "feature": "HEVC Decode License",
                "enabled": "Yes"
            },
            {
                "feature": "HD Output License",
                "enabled": "Yes"
            },
            {
                "feature": "Full HD Output License",
                "enabled": "Yes"
            },
            {
                "feature": "UHD Output License",
                "enabled": "Yes"
            },
            {
                "feature": "AUD3_4 License",
                "enabled": "Yes"
            },
            {
                "feature": "IPI TS License",
                "enabled": "6"
            }
        ],
        "licensefiles": [
            {
                "feature": "Tuner",
                "enabled": "6"
            },
            {
                "feature": "DVB_S2",
                "enabled": "1"
            },
            {
                "feature": "Tuner_APSK",
                "enabled": "1"
            },
            {
                "feature": "AVC_Decode",
                "enabled": "1"
            },
            {
                "feature": "HEVC_Decode",
                "enabled": "1"
            },
            {
                "feature": "HD_Output",
                "enabled": "1"
            },
            {
                "feature": "Full_HD_Output",
                "enabled": "1"
            },
            {
                "feature": "UHD_Output",
                "enabled": "1"
            },
            {
                "feature": "AUD3_4",
                "enabled": "1"
            },
            {
                "feature": "IPI_TS",
                "enabled": "5"
            }
        ]
    }
}
```

Device Identification Command

Table 2.117 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/ident
Command Information	This command can be used to read information about identication of the unit, including MAC addresses and User Address (UA) related data.
HTTP Method	GET

Command Detail	Description
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/ident"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.118 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &):

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in JSON

The items in the first column of the following table may be used as Query Parameters to restrictively filter the output. For example, adding ?masterua&hostname to the command line will result in output that only provides the data corresponding to those fields.

Table 2.119 Output Field Descriptions

Output Field	Description
hostname	Host name Type: String Value: Maximum of 64 characters
modelnum	Model number Type: String Value: Maximum of 50 characters
modelname	Model name Type: String Value: Maximum of 50 characters
catalognum	Catalog number Type: String Value: Maximum of 50 characters
serialnum	Serial number

Output Field	Description	
	Type: String Value: Maximum of 50 characters	
trackingid	Tracking ID Type: String Value: Maximum of 13 characters	
macinfo Contains multiple pairs <”intfname”, “macaddr”>, one per ETH interface.	intfname	Name of ETH interface Type: String Value: “Management”, “Data1”, “Data2”, “Data3”, “Data4”
	macaddr	MAC address of ETH interface Type: String Value: MAC address in format "00:00:00:00:00:00"
masterua Info on Master UA	address	Master User Address Type: String Value: Maximum of 15 characters
	eprom_match	Master UA matches EEPROM Programming Type: String Value: “Yes”, “No”
	isesigned	ISE Block Signed Type: String Value: “Yes”, “No”
certua Info on Certified UA	address	Certified User Address. Zero UA means “none” Type: String Value: Maximum of 15 characters
	certverif	Certified UA is verified Type: String Value: “Yes”, “No”
ualist	address	User Address

Output Field	Description	
Lists all user addresses for this unit. This includes Master and Certified User Addresses		Type: String Value: Maximum of 15 characters
	iseversion	ISE (Integrated Security Element) Version Type: String Value: 1..8 characters

Input Example (request default XML format):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/status/ident"
```

Expected output prior to Release 4.75 (values are for example purposes only):

```
{
    "ident": {
        "hostname": "User-cfg-name",
        "modelnum": "D9800",
        "modelname": "Network Transport Receiver",
        "catalognum": "D9800-SS-MPEGOIP",
        "serialnum": "FJZ190500U7",
        "trackingid": "FJZ190500U7",
        "macinfo": [
            {
                "intfname": "Management",
                "macaddr": "38:C8:5C:FF:AF:CC"
            },
            {
                "intfname": "Data1",
                "macaddr": "38:C8:5C:FF:B0:3C"
            },
            {
                "intfname": "Data2",
                "macaddr": "38:C8:5C:FF:B0:3D"
            }
        ],
        "masterua": {
            "address": "000-661-4351-2",
            "eprom_match": "Yes",
            "isesigned": "Yes"
        },
        "certua": {
            "address": "000-661-4351-2",
            "certverif": "yes"
        },
        "ualist":
       {
            "address": "000-661-4351-2",
            "iseversion": "5.1(2)"
       },
        {
               // other UA entries in UA list
    }
  ]
}
```

Expected output starting with Release 4.75 or later (values are for example purposes only):

```
{
    "ident": {
        "hostname": "User-cfg-name",
        "modelnum": "D9800",
        "modelname": "PowerVu Professional Receiver",
        "catalognum": "D9800-SS-MPEGOIP",
        "serialnum": "FJZ190500U7",
        "trackingid": "FJZ190500U7",
        "macinfo": [
            {
                "intfname": "Management",
                "macaddr": "38:C8:5C:FF:AF:CC"
            },
            {
                "intfname": "Data1",
                "macaddr": "38:C8:5C:FF:B0:3C"
            },
            {
                "intfname": "Data2",
                "macaddr": "38:C8:5C:FF:B0:3D"
            }
        ],
        "masterua": {
            "address": "000-661-4351-2",
            "eprom_match": "Yes",
            "isesigned": "Yes"
        },
        "certua": {
            "address": "000-661-4351-2",
            "certverif": "yes"
        },
        "ualist":
       {
```

```
                "address": "000-661-4351-2",
                "iseversion": "5.1(2)"
          },
           {
                   // other UA entries in UA list
      }
   ]
}
```

Hardware Options Command

Table 2.120 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/hwoptions
Command Information	This command obtains information on installed hardware options.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/hwoptions"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.121 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Parameter	Description
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in JSON

The items in the first column of the following table may be used as Query Parameters to restrictively filter the output. For example, adding ?masterua&hostname to the command line will result in output that only provides the data corresponding to those fields.

Table 2.122 Output Field Descriptions

Output Field	Description	
info Contains multiple pairs <”feature”, “installed”>, one per HW option	feature	Name of hardware option, for example, “Multi-Stream” Type: String

Output Field	Description	
		Value: 1..64 characters
	installed	Shows if hardware options is installed Type: String Value: 1..16 characters, most used: “Yes”, “No”

Input Example (request default XML format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/hwoptions"
```

Expected output for NTC MOIP Multi-Stream Units (values are for example purposes only):

```
{
    "hwoptions": {
        "info": [
            {
                "feature": "Multi-Stream",
                "installed": "Yes"
            },
            {
                "feature": "MPOIP In",
                "installed": "Yes"
            },
            {
                "feature": "MPOIP Out",
                "installed": "Yes"
            },
            {
                "feature": "MPOIP FEC",
                "installed": "Yes"
            },
            {
                "feature": "SDI",
                "installed": "No"
            },
            {
                "feature": "Transcoder Available",
                "installed": "16"
            },
            {
                "feature": "HEVC Processing",
                "installed": "12"
            },
            {
                "feature": "DVB-S2X",
                "installed": "No"
            }
        ]
    }
}
```

Expected output for NTC MOIP Single-Stream Units (values are for example purposes only):

```
{
    "hwoptions": {
        "info": [
            {
                "feature": "Multi-Stream",
                "installed": "No"
            },
            {
                "feature": "MPOIP In",
                "installed": "Yes"
            },
            {
                "feature": "MPOIP Out",
                "installed": "Yes"
            },
            {
                "feature": "MPOIP FEC",
                "installed": "Yes"
            },
            {
                "feature": "SDI",
                "installed": "Yes"
            },
            {
                "feature": "DVB-S2X",
                "installed": "No"
            }
        ]
    }
}
```

Device Status Command

Table 2.123 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/device

Command Detail	Description
	https://192.168.0.1/ws/v2/status/device/<level_1_extension> https://192.168.0.1/ws/v2/status/device/about/clear
Command Information	This command is used to read either all or a specific category of device details. It may also be used to clear.
HTTP Methods	GET, POST (see Syntax row)
Access Type	Read
Access Level	User, Admin
Syntax	One of the following: GET "https://192.168.0.1/ws/v2/status/device" GET "https://192.168.0.1/ws/v2/status/device/ecc" GET "https://192.168.0.1/ws/v2/status/device/eth" GET "https://192.168.0.1/ws/v2/status/device/power" GET "https://192.168.0.1/ws/v2/status/device/storage" GET "https://192.168.0.1/ws/v2/status/device/about" POST "https://192.168.0.1/ws/v2/status/device/about/clear

Table 2.124 URI <level_1_extension> Parameters (extension to the Command URL separated by /)

URI Parameter	Description
None	Reads and returns current value of all device categories and related items.
ecc	Reads and returns current value of ECC specific device items.
eth	Reads and returns current value of ethernet specific device items.
power	Reads and returns current value of power status device items for all populated card slots.
storage	Reads and returns current values of storage (memory module) related device items.
about	Reads and returns current value of about specific (uncategorized) device items.

Table 2.125 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js	Format output using JSON standard Type: exist Values: any value or empty (any of js=1, js, js=0, or js=9999 are valid) Omitting this argument formats the output by default in XML.

See descriptions and other arguments for each variant of this command in the following sections.

About Status Command

Table 2.126 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/device/about
Command Information	This command reads and returns uncategorized device items or zeros out the liferstclr counter.
HTTP Methods	GET, POST (see Syntax row)
Access Type	Read, Write (for POST operation only)
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/device/about", or POST "https://192.168.0.1/ws/v2/status/device/about/clear"

Table 2.127 URI Parameters (extension to the Command URL separated by /)

URI Parameter	Description
none	Read all (when no Arguments) uncategorized device items or specific item(s) (when Argument(s) are specified)
clear	Clears the life reset counter (liferstclr) in the relevant device/about output field. Requires the POST HTTP Method and only valid when specified after …/ws/v2/status/device/about.

Table 2.128 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
See the Output Field Descriptions table	Specifying any or combination of the output field values (separated by &) will filter the output to only return those fields.

URI Argument	Description
js	Format output using JSON standard Type: exist Values: any value or empty (any of js=1, js, js=0, or js=9999 are valid) Omitting this argument formats the output by default in XML.

Table 2.129 Output Field Descriptions

Output Field	Description
modelnum	Model number Type: String Value: Maximum of 50 characters
modelname	Model name Type: String Value: Maximum of 50 characters
catalognum	Catalogue number Type: String Value: Maximum of 50 characters
boardid	Board ID Type: String Value: Maximum of 50 characters
boardrev	Board revision Type: String Value: Maximum of 50 characters
fpgatype	FPGA type Type: String Value: Maximum of 50 characters
fpgaid	FPGA ID Type: String Value: Maximum of 20 characters
serialnum	Unit serial number Type: String Value: Maximum of 50 characters

Output Field	Description
petrans	Maximum number of program entries simultaneously transcoded Type: Integer Value: 0..N
ethports	Number of Ethernet ports Type: Integer Value: 0..N
copyright	Copyright statement Type: String Value: Maximum of 100 characters
proddate	Production date and time Type : String Values: string format "yyyy/mm/dd hh:mm:ss"
pwrupdate	Last power on date and time Type : String Values: string format "yyyy/mm/dd hh:mm:ss"
lifepower	Lifetime number of hours powered on Type : Integer Values: 0..N
lifereset	Lifetime number of resets Type : Integer Values: 0..N
liferstclr	Clearable count of number of resets Type : Integer Values: 0..N
hoursup	Number of hours powered since power-on/reset Type : Integer Values: 0..N
lastrsttxt	Reason for last reset (for example, power on) Type : String Values: Maximum of 128 characters

Output Field	Description
about1,2, 3, 4, 5, 6, or 7	Line # of 7 of "about" text (legal jargon) Type : String Values: Maximum of 256 characters

Input example (request JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/about?js=1"
```

Expected output - Release 4.50 or earlier (values are for example purposes only):

```
{
    "device": {
        "about": {
            "modelnum": "D9800",
            "modelname": "Network Transport Receiver",
            "catalognum": "D9800-SS-MPEGOIP",
            "boardid": "NTC",
            "boardrev": "A0",
            "fpgatype": "1.00",
            "fpgaid": "Wildcard FPGA ID",
            "serialnum": "FJZ202201G3",
            "petrans": "16",
            "ethports": "5",
            "copyright": "2019 Synamedia and/or its affiliates. All rights
reserved.",
            "proddate": "2016/11/28 15:20:16",
            "pwrupdate": "2019/10/11 05:55:02",
            "lifepower": "25119",
            "lifereset": "2450",
            "liferstclr": "2450",
            "hoursup": "116",
            "lastrsttxt": "DL Shutdown Request",
            "about1": "This Synamedia product (D9800 PowerVu Professional
ReceiUnavailab",
            "about2": ") contains software licensed under the following
licenses:",
           "about3": "<br>\"GNU General Public License, version 3\" provided
with
ABSOLUTELY NO WARRANTY under the terms of \"GNU General Public License,
Version 3\",
available here:http://www.gnu.org/licenses/gpl.html",
            "about4": "<br>\"GNU General Public License, version 2\" provided
with
ABSOLUTELY NO WARRANTY under the terms of \"GNU General Public License,
version 2\",
available here:http://www.gnu.org/licenses/old-licenses/gpl-2.0.html",
            "about5": "<br>\"GNU Lesser General Public License, version 2.1\"
provided with
ABSOLUTELY NO WARRANTY under the terms of \"GNU Lesser General Public
License,
version 2.1\", available here:http://www.gnu.org/licenses/old-licenses/lgpl-
2.1.html",
            "about6": "<br>\"GNU Library General Public License, version 2\"
provided with
ABOSOLUTELY NO WARRANTY under the terms of \"GNU Library General Public
License,
version 2\", available here:http://www.gnu.org/licenses/old-licenses/lgpl-
2.0.html",
            "about7": "<br>"
        }
    }
}
```

Expected output - Release 4.80 or later (values are for example purposes only):

```
{
    "device": {
        "about": {
            "modelnum": "D9800",
            "modelname": "PowerVu Professional Receiver",
            "catalognum": "D9800-SS-MPEGOIP",
            "boardid": "NTC",
            "boardrev": "A0",
            "fpgatype": "1.00",
            "fpgaid": "Wildcard FPGA ID",
            "serialnum": "FJZ202201G3",
            "petrans": "16",
            "ethports": "5",
            "copyright": "2019 Synamedia and/or its affiliates. All rights
reserved.",
            "proddate": "2016/11/28 15:20:16",
            "pwrupdate": "2019/10/11 05:55:02",
            "lifepower": "25119",
            "lifereset": "2450",
            "liferstclr": "2450",
            "hoursup": "116",
            "lastrsttxt": "DL Shutdown Request",
            "about1": "This Synamedia product (D9800 PowerVu Professional
ReceiUnavailab",
            "about2": ") contains software licensed under the following
licenses:",
           "about3": "<br>\"GNU General Public License, version 3\" provided
with
ABSOLUTELY NO WARRANTY under the terms of \"GNU General Public License,
Version 3\",
available here:http://www.gnu.org/licenses/gpl.html",
            "about4": "<br>\"GNU General Public License, version 2\" provided
with
ABSOLUTELY NO WARRANTY under the terms of \"GNU General Public License,
version 2\",
available here:http://www.gnu.org/licenses/old-licenses/gpl-2.0.html",
            "about5": "<br>\"GNU Lesser General Public License, version 2.1\"
provided with
ABSOLUTELY NO WARRANTY under the terms of \"GNU Lesser General Public
License,
version 2.1\", available here:http://www.gnu.org/licenses/old-licenses/lgpl-
2.1.html",
            "about6": "<br>\"GNU Library General Public License, version 2\"
provided with
ABOSOLUTELY NO WARRANTY under the terms of \"GNU Library General Public
License,
version 2\", available here:http://www.gnu.org/licenses/old-licenses/lgpl-
2.0.html",
            "about7": "<br>"
        }
    }
}
```

Input example (request JSON format, filtered output):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/about?pwrupdate&lastrsttxt&js=1"
```

Expected output (values are for example purposes only):

```
{
    "device": {
        "about": {
            "lastrsttxt": "DL Shutdown Request",
            "pwrupdate": "2017/03/20 22:01:12"
        }
    }
}
```

Input (request XML format, clear the liferstclr counter):

```
curl -k -H "X-SESSION-ID: $token" -X POST
"https://192.168.0.1/ws/v2/status/device/about/clear"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><response><code>10</code>
<result>success</result><message></message></response>
```

ECC Status Command

Table 2.130 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/device/ecc
Command Information	Returns status of ECC (environmental characteristics and control) items.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/device/ecc"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.131 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
slot For Version 3.51 or lower, this argument is mandatory when row is specified.	This is a key field for filtering output fields to a specified slot. When used with ECC URI extension, this argument also requires specifying the table row key. The user or client software must have previous knowledge of all of the ECC row indices for a specific software and hardware configuration. Filtering output for ECC related device data may be useful as part of an overall automated client command caching strategy (after retrieving the entire table at least once so that the row indices are already known). Type: Integer Values: 0 ...
row For Version 3.51 or lower, this argument is mandatory when slot is specified.	This is a key field for filtering output fields to a specified row. See description under slot. Type: Integer Values: 0 ...
js	Format output using JSON standard.

URI Argument	Description
	Type: exist Values: any value or empty (any of js=1, js, js=0, or js=9999 are valid) Omitting this argument formats the output by default in XML
session	Alternate way of specifying the logon session token ID instead of using –H syntax.

Note In Version 3.75 or later, these items may also be used as enhanced filtering Query Arguments for GET operations. Due to Query Arguments specification restriction, only the first eleven (11) characters of any input Query Argument name will be matched. For example, input of ?displayBoarDNA will still successfully match the first 11 characters of displayBoardName. Query Argument values may be matched up to 200 characters each case up to a maximum of 200 characters in any line, and maximum 26 Query Arguments per line. See examples.

Table 2.132 Output Field Descriptions

Output Field	Description
slot	Chassis slot number. Type: Integer Value: 1 to number of slots in chassis
row	Row number. Type: Integer Value: 1 to number of rows on unit
card	Chassis card acronym. Type: String Value: for example, "NBP, NTC, NFE, NDM, NTM, NTB, NCI"
displayBoardName	User friendly name of board in device slot. Type: String Value: "System", "Controller", "Tuner", "CI Option", or "Decoder" or "TranscoderX", or "HEVC ProcX", where the X is replaced with the unique instance number (counting, starting with one) of the particular board type that was detected by the software at target initialization time.
location	Internal string for uniquely identifying the location of the sensor (not always the same as the displayItemName). Type: String Value: For example, "curr_in_temp"

Output Field	Description
displayItemName	Friendly name identifying the ECC sensor item to the user. **Note** We recommend that you use these names in combination with specific slot number to filter or restrict output to only a single sensor of interest instead of slot or row combinations. Type: String Value: For example, "+12.0V" to read 12 volt supply current reading
type	Category of ECC item. Type: String Value: "Temperature", "Voltage", "Fan Status/Speed", or "N/A"
value	Actual value of given item. Type: String Value: For example, "+12.23V" (current +12.0V supply reading)
cardslot	Concatenation of card and slot fields. Type: String Value: For example, "NDM1"
displayorderflag	Value used for sorting order of the rows on the web UI display. Type: Integer Value: 1-16
rowStatus	ECC Row Status Type : String Values : "Active", "Inactive" If "Active", then the data in this row is current. If "Inactive", then the data returned in this row is stale (cached) and may not reflect an up-to-date recent sensor reading. The "Inactive" state is also set temporarily while an ECC row is in the process of being updated so that GET operations cannot interfere with background write operations on the current record.

Recommendations on interpretation of ECC data by client software or test scripts:

Sensor data is read periodically for each device being monitored at different rates per device by a low level sensor driver subsystem in the operating system, and polled and recorded at a much lower rate (once per 15 seconds) by the application software. The disparity of update rates and ECC device sensor data availability after startup is compounded with NTRPs having multiple transcoders with multiple processors.

During the rowStatus "Inactive" state, there is a chance that either none or some of the record fields for a particular row (corresponding to a particular sensor device) are returned. In the former case, the array row may contain empty data yet the output array "row" index field may continue to increment (potentially resulting in skipped row numbers). This is intentional and unavoidable behavior that indicates that the place-holder row data for that sensor record corresponding to the empty row/device has not yet been reported and is not yet available. ECC Data received for a snapshot of empty or Inactive rows is therefore unreliable for those rows. Only data for when ECC rowStatus is Active can be used to make confident decisions about fan/temperature/voltage status or Alarm/Warning notifications.

Client software that receives either empty (or skipped) rows or rowStatus with an "Inactive" value are advised to wait at least 15 seconds (which is the read/update interval for the software to poll for any new sensor data recorded by the sensor drivers) before retrying the GET API for the ECC data again.

To avoid receiving incomplete data, it is recommended that the first GET status device ECC API only be called after the system has already been running for a few minutes so that there is at least a chance for all device sensors on every card to have been polled and recorded at least once after a power up or restart.

In a lengthy test script scenario, the risk of receiving incomplete or "Inactive" ECC data can usually be mitigated by calling the GET status/device/ecc API near the end of the APIs called in the test sequence.

If the start-up sequence prior to the first GET status device ECC API call is too short, then additional delay/sleeps and/or retries may be required until all required valid ECC data is obtained. If needed, the rowStatus value may be useful in implementing such retry logic.

Note There is another (less likely) possibility for missing rows in ECC sensor data. In that case, sensor data for a particular device may not be available due to a hardware failure or malfunction of the I2C (a two-wire serial protocol commonly used in embedded systems design) bus that is connected to the device.

In the case of empty rows, a single retry of the GET status device ECC API (after 15 seconds) should suffice to rule out a hardware malfunction for that device. If the data is missing in back to back (after delay) retries or frequently missing in the same row index, then the hardware may be suspected. If the data for that sensor placeholder is available after that initial delay, and continues to be available (after additional delay and retry), then the hardware is likely not the root cause.

Examples:

Input example (request default XML format, all ECC data):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/ecc"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<device><ecc><slot>0</slot><row>0</row><card>NBP</card><displayBoardName>Syst
em</displayBoardName>
<location>curr_in_temp</location><displayItemName>System
Intake</displayItemName><type>Temperature</type>
<value>+26
C</value><cardslot></cardslot><displayOrderFlag>0</displayOrderFlag></ecc><ec
c><slot>0</slot><row>1</row>
<card>NBP</card><displayBoardName>System</displayBoardName><location>hist_
max_in_temp</location>
<displayItemName>Ambient Peak
Temp</displayItemName><type>Temperature</type><value>+37
C</value><cardslot></cardslot>

<displayOrderFlag>0</displayOrderFlag></ecc><ecc><slot>0</slot><row>2</row><c
ard>NBP</card>
<displayBoardName>System</displayBoardName><location>hist_avg_in_
temp</location>
<displayItemName>Ambient Temp
Average</displayItemName><type>Temperature</type><value>+26
C</value><cardslot></cardslot>

<displayOrderFlag>0</displayOrderFlag></ecc><ecc><slot>0</slot><row>3</row><c
ard>NBP</card>
<displayBoardName>System</displayBoardName><location>power
fan</location><displayItemName>PSU Fan</displayItemName>
<type>Fan Status/Speed</type><value>11428
RPM</value><cardslot>NBP0</cardslot>

<displayOrderFlag>255</displayOrderFlag></ecc><ecc><slot>0</slot><row>4</row>
<card>NBP</card>
<displayBoardName>System</displayBoardName><location>left
fan</location><displayItemName>Left Fan</displayItemName>
<type>Fan Status/Speed</type><value>11045
RPM</value><cardslot>NBP0</cardslot>

<displayOrderFlag>253</displayOrderFlag></ecc><ecc><slot>0</slot><row>5</row>
<card>NBP</card>
<displayBoardName>System</displayBoardName><location>right fan</location>
<displayItemName>Right Fan</displayItemName><type>Fan
Status/Speed</type><value>11430 RPM</value>

<cardslot>NBP0</cardslot><displayOrderFlag>251</displayOrderFlag></ecc><ecc><
slot>0</slot><row>6</row>
<card>NBP</card><displayBoardName>System</displayBoardName><location>max
```

```
temp</location>
<displayItemName>Current Maximum
Temp</displayItemName><type>Temperature</type><value>+71.4 C</value>

<cardslot>NBP0</cardslot><displayOrderFlag>1</displayOrderFlag></ecc><ecc><sl
ot>0</slot><row>7</row>
<card>NTC</card><displayBoardName>Controller</displayBoardName><location>FPGA
+1.0V</location>
<displayItemName>FPGA +1.0V</displayItemName><type>Voltage</type><value>+0.97
V</value>

<cardslot>NTC0</cardslot><displayOrderFlag>1</displayOrderFlag></ecc><ecc><sl
ot>0</slot><row>8</row>
<card>NTC</card><displayBoardName>Controller</displayBoardName><location>FPGA
+1.8V</location>
<displayItemName>FPGA +1.8V</displayItemName><type>Voltage</type><value>+1.81
V</value><cardslot>NTC0</cardslot>

<displayOrderFlag>1</displayOrderFlag></ecc><ecc><slot>0</slot><row>9</row><c
ard>NTC</card>
<displayBoardName>Controller</displayBoardName><location>FPGA
temp</location><displayItemName>FPGA temp</displayItemName>
<type>Temperature</type><value>+71.9
C</value><cardslot>NTC0</cardslot><displayOrderFlag>1</displayOrderFlag></ec
c>

<ecc><slot>0</slot><row>10</row><card>NTC</card><displayBoardName>Controller<
/displayBoardName><location>+12.0V</location>
<displayItemName>+12.0V</displayItemName><type>Voltage</type><value>+11.99
V</value><cardslot>NTC0</cardslot>

<displayOrderFlag>1</displayOrderFlag></ecc><ecc><slot>0</slot><row>11</row><
card>NTC</card>

<displayBoardName>Controller</displayBoardName><location>+3.3V</location><dis
playItemName>+3.3V</displayItemName>
<type>Voltage</type><value>+3.32
V</value><cardslot>NTC0</cardslot><displayOrderFlag>1</displayOrderFlag></ecc
><ecc>

<slot>0</slot><row>12</row><card>NTC</card><displayBoardName>Controller</disp
layBoardName><location>+2.5V</location>
<displayItemName>+2.5V</displayItemName><type>Voltage</type><value>+2.50
V</value><cardslot>NTC0</cardslot>
```

```
<displayOrderFlag>1</displayOrderFlag></ecc><ecc><slot>0</slot><row>13</row><
card>NTC</card>

<displayBoardName>Controller</displayBoardName><location>+1.05V</location><di
splayItemName>+1.05V</displayItemName>
<type>Voltage</type><value>+1.05
V</value><cardslot>NTC0</cardslot><displayOrderFlag>1</displayOrderFlag></ecc
><ecc><slot>0</slot>

<row>14</row><card>NTC</card><displayBoardName>Controller</displayBoardName><
location>+1.5V</location>
<displayItemName>+1.5V</displayItemName><type>Voltage</type><value>+1.51
V</value><cardslot>NTC0</cardslot>

<displayOrderFlag>1</displayOrderFlag></ecc><ecc><slot>0</slot><row>15</row><
card>NTC</card>

<displayBoardName>Controller</displayBoardName><location>+0.9V</location><dis
playItemName>+0.9V</displayItemName>
<type>Voltage</type><value>+0.90
V</value><cardslot>NTC0</cardslot><displayOrderFlag>1</displayOrderFlag></ec
c>

<ecc><slot>0</slot><row>16</row><card>NTC</card><displayBoardName>Controller<
/displayBoardName>
<location>+3.3V LDO</location><displayItemName>+3.3V
LDO</displayItemName><type>Voltage</type><value>+3.31 V</value>

<cardslot>NTC0</cardslot><displayOrderFlag>1</displayOrderFlag></ecc><ecc><sl
ot>0</slot><row>17</row>

<card>NTC</card><displayBoardName>Controller</displayBoardName><location>+1.0
V</location>
<displayItemName>+1.0V</displayItemName><type>Voltage</type><value>+0.98
V</value><cardslot>NTC0</cardslot>

<displayOrderFlag>1</displayOrderFlag></ecc><ecc><slot>0</slot><row>18</row><
card>NTC</card>

<displayBoardName>Controller</displayBoardName><location>+5.0V</location><dis
playItemName>+5.0V</displayItemName>
<type>Voltage</type><value>+5.07
V</value><cardslot>NTC0</cardslot><displayOrderFlag>1</displayOrderFlag></ec
c>
```

```
<ecc><slot>0</slot><row>19</row><card>NTC</card><displayBoardName>Controller<
/displayBoardName><location>+0.75V</location>
<displayItemName>+0.75V</displayItemName><type>Voltage</type><value>+0.76
V</value><cardslot>NTC0</cardslot>

<displayOrderFlag>1</displayOrderFlag></ecc><ecc><slot>0</slot><row>20</row><
card>NTC</card>

<displayBoardName>Controller</displayBoardName><location>+1.8V</location><dis
playItemName>+1.8V</displayItemName>
<type>Voltage</type><value>+1.81
V</value><cardslot>NTC0</cardslot><displayOrderFlag>1</displayOrderFlag></ec
c>

<ecc><slot>0</slot><row>21</row><card>NTC</card><displayBoardName>Controller<
/displayBoardName><location>+1.2V</location>
<displayItemName>+1.2V</displayItemName><type>Voltage</type><value>+1.16
V</value><cardslot>NTC0</cardslot>

<displayOrderFlag>1</displayOrderFlag></ecc><ecc><slot>0</slot><row>22</row><
card>NTC</card>
<displayBoardName>Controller</displayBoardName><location>exhaust
temp</location>
<displayItemName>Rear
Temp</displayItemName><type>Temperature</type><value>+41.2 C</value>

<cardslot>NTC0</cardslot><displayOrderFlag>1</displayOrderFlag></ecc><ecc><sl
ot>0</slot><row>23</row>

<card>NTC</card><displayBoardName>Controller</displayBoardName><location>inta
ke temp</location>
<displayItemName>Front
Temp</displayItemName><type>Temperature</type><value>+26.6
C</value><cardslot>NTC0</cardslot>

<displayOrderFlag>1</displayOrderFlag></ecc><ecc><slot>0</slot><row>24</row><
card>NTC</card>
<displayBoardName>Controller</displayBoardName><location>CPU
temp</location><displayItemName>CPU temp</displayItemName>
<type>Temperature</type><value>+42.2
C</value><cardslot>NTC0</cardslot><displayOrderFlag>1</displayOrderFlag></ec
c>

<ecc><slot>1</slot><row>0</row><card>NFE</card><displayBoardName>Tuner</displ
ayBoardName><location>FPGA +1.0V</location>
<displayItemName>FPGA +1.0V</displayItemName><type>Voltage</type><value>+0.99
```

```
V</value><cardslot>NFE1</cardslot>

<displayOrderFlag>2</displayOrderFlag></ecc><ecc><slot>1</slot><row>1</row><c
ard>NFE</card>
<displayBoardName>Tuner</displayBoardName><location>FPGA
+1.8V</location><displayItemName>FPGA +1.8V</displayItemName>
<type>Voltage</type><value>+1.79
V</value><cardslot>NFE1</cardslot><displayOrderFlag>2</displayOrderFlag></ec
c>

<ecc><slot>1</slot><row>2</row><card>NFE</card><displayBoardName>Tuner</displ
ayBoardName><location>FPGA temp</location>
<displayItemName>FPGA
temp</displayItemName><type>Temperature</type><value>+47.2
C</value><cardslot>NFE1</cardslot>

<displayOrderFlag>2</displayOrderFlag></ecc><ecc><slot>2</slot><row>0</row><c
ard>NDM</card>
<displayBoardName>Decoder</displayBoardName><location>FPGA
+1.0V</location><displayItemName>FPGA +1.0V</displayItemName>
<type>Voltage</type><value>+0.99
V</value><cardslot>NDM2</cardslot><displayOrderFlag>3</displayOrderFlag></ecc
><ecc>

<slot>2</slot><row>1</row><card>NDM</card><displayBoardName>Decoder</displayB
oardName><location>FPGA +1.8V</location>
<displayItemName>FPGA +1.8V</displayItemName><type>Voltage</type><value>+1.79
V</value><cardslot>NDM2</cardslot>

<displayOrderFlag>3</displayOrderFlag></ecc><ecc><slot>2</slot><row>2</row><c
ard>NDM</card>
<displayBoardName>Decoder</displayBoardName><location>FPGA temp</location>
<displayItemName>FPGA
temp</displayItemName><type>Temperature</type><value>+46.8
C</value><cardslot>NDM2</cardslot>

<displayOrderFlag>3</displayOrderFlag></ecc><ecc><slot>2</slot><row>3</row><c
ard>NDM</card>
<displayBoardName>Decoder</displayBoardName><location>+1.0V
CPU</location><displayItemName>+1.0V CPU</displayItemName>
<type>Voltage</type><value>+0.94
V</value><cardslot>NDM2</cardslot><displayOrderFlag>3</displayOrderFlag></ec
c>

<ecc><slot>2</slot><row>4</row><card>NDM</card><displayBoardName>Decoder</dis
playBoardName><location>+3.3V</location>
```

```
<displayItemName>+3.3V</displayItemName><type>Voltage</type><value>+3.32
V</value><cardslot>NDM2</cardslot>

<displayOrderFlag>3</displayOrderFlag></ecc><ecc><slot>2</slot><row>5</row><c
ard>NDM</card>
<displayBoardName>Decoder</displayBoardName><location>+2.5V</location>
<displayItemName>+2.5V</displayItemName><type>Voltage</type><value>+2.51
V</value><cardslot>NDM2</cardslot>

<displayOrderFlag>3</displayOrderFlag></ecc><ecc><slot>2</slot><row>6</row><c
ard>NDM</card>

<displayBoardName>Decoder</displayBoardName><location>+1.8V</location><displa
yItemName>+1.8V</displayItemName>
<type>Voltage</type><value>+1.81
V</value><cardslot>NDM2</cardslot><displayOrderFlag>3</displayOrderFlag></ec
c>

<ecc><slot>2</slot><row>7</row><card>NDM</card><displayBoardName>Decoder</dis
playBoardName><location>+1.5V</location>
<displayItemName>+1.5V</displayItemName><type>Voltage</type><value>+1.50
V</value><cardslot>NDM2</cardslot>

<displayOrderFlag>3</displayOrderFlag></ecc><ecc><slot>2</slot><row>8</row><c
ard>NDM</card>

<displayBoardName>Decoder</displayBoardName><location>+5.0V</location><displa
yItemName>+5.0V</displayItemName>
<type>Voltage</type><value>+4.98
V</value><cardslot>NDM2</cardslot><displayOrderFlag>3</displayOrderFlag></ec
c>

<ecc><slot>2</slot><row>9</row><card>NDM</card><displayBoardName>Decoder</dis
playBoardName>
<location>+3.3V LDO</location><displayItemName>+3.3V
LDO</displayItemName><type>Voltage</type><value>+3.32 V</value>

<cardslot>NDM2</cardslot><displayOrderFlag>3</displayOrderFlag></ecc><ecc><sl
ot>2</slot><row>10</row><card>NDM</card>
<displayBoardName>Decoder</displayBoardName><location>+1.0V</location>
<displayItemName>+1.0V</displayItemName><type>Voltage</type><value>+0.99
V</value><cardslot>NDM2</cardslot>

<displayOrderFlag>3</displayOrderFlag></ecc><ecc><slot>2</slot><row>11</row><
card>NDM</card>
```

```
<displayBoardName>Decoder</displayBoardName><location>+0.75V</location><displ
ayItemName>+0.75V</displayItemName>
<type>Voltage</type><value>+0.77
V</value><cardslot>NDM2</cardslot><displayOrderFlag>3</displayOrderFlag></ec
c>

<ecc><slot>2</slot><row>12</row><card>NDM</card><displayBoardName>Decoder</di
splayBoardName>

<location>+12.0V</location><displayItemName>+12.0V</displayItemName><type>Vol
tage</type><value>+11.84 V</value>

<cardslot>NDM2</cardslot><displayOrderFlag>3</displayOrderFlag></ecc><ecc><sl
ot>2</slot><row>13</row>
<card>NDM</card><displayBoardName>Decoder</displayBoardName><location>intake
temp</location>
<displayItemName>Front
Temp</displayItemName><type>Temperature</type><value>+28.1 C</value>

<cardslot>NDM2</cardslot><displayOrderFlag>3</displayOrderFlag></ecc><ecc><sl
ot>2</slot>

<row>14</row><card>NDM</card><displayBoardName>Decoder</displayBoardName><loc
ation>CPU temp</location>
<displayItemName>CPU
temp</displayItemName><type>Temperature</type><value>+43.3
C</value><cardslot>NDM2</cardslot>
<displayOrderFlag>3</displayOrderFlag></ecc></device>
```

Input example (request JSON format, all ECC data):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/ecc?js=1"
```

Input example (request JSON format, ECC data only for slot 0, row 12):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/ecc?slot=0&row=12&js=1"
```

Expected output (values are for example purposes only):

```
{
    "device": {
        "ecc": {
            "slot": "0",
            "row": "12",
            "card": "NTC",
                "displayBoardName": "Controller",
                "location": "+2.5V",
                "displayItemName": "+2.5V",
                "type": "Voltage",
                "value": "+2.50 V",
                "cardslot": "NTC0",
                "displayOrderFlag": "1"
            }
        }
}
```

Input (multistream target, request XML format, ECC data only for displayBoardName=Transcoder1&displayItemName=CPU%20temp):

```
curl -k -H "X-SESSION-ID: $token"

"https://192.168.0.1/ws/v2/status/device/ecc?displayBoardName=Transcoder1&dis
playItemName=CPU%20temp"
```

Expected output (values are for example purposes only):

Note Identifying keys (slot, row) are always output and cannot be filtered.

Note Using specific value filters returns all items in the matched row.

```
<?xml version="1.0" encoding="ISO-8859-1"
?><device><ecc><slot>3</slot><row>22</row><card>NTM</card>
<displayBoardName>Transcoder1</displayBoardName><location>CPU
temp</location><displayItemName>CPU temp</displayItemName>
<type>Temperature</type><value>+31.3 C</value>

<cardslot>NTM3</cardslot><displayOrderFlag>4</displayOrderFlag></ecc></devic
e>
```

Input (multistream target, request XML format, ECC data for displayItemName for all slots/rows):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/ecc? displayItemName"
```

Expected output (values are for example purposes only):

Note Identifying Keys (slot, row) are always output and cannot be filtered.

Note Using no value filters returns that item for all matched rows.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<device>
<ecc><slot>0</slot><row>0</row><displayItemName>System
Intake</displayItemName></ecc>
<ecc><slot>0</slot><row>1</row><displayItemName>Ambient Peak
Temp</displayItemName>
</ecc>
<ecc><slot>0</slot><row>2</row><displayItemName>Ambient Temp
Average</displayItemName></ecc>
<ecc><slot>0</slot><row>3</row><displayItemName>PSU
Fan</displayItemName></ecc>
<ecc><slot>0</slot><row>4</row><displayItemName>Current Maximum
Temp</displayItemName></ecc>
<ecc><slot>0</slot><row>5</row><displayItemName>Left
Fan1</displayItemName></ecc>
<ecc><slot>0</slot><row>6</row><displayItemName>Left
Fan2</displayItemName></ecc>
<ecc><slot>0</slot><row>7</row><displayItemName>Middle
Fan1</displayItemName></ecc>
<ecc><slot>0</slot><row>8</row><displayItemName>Middle
Fan2</displayItemName></ecc>
<ecc><slot>0</slot><row>9</row><displayItemName>Right
Fan1</displayItemName></ecc>
<ecc><slot>0</slot><row>10</row><displayItemName>Right
Fan2</displayItemName></ecc>

<ecc><slot>0</slot><row>11</row><displayItemName>+12.0V</displayItemName></ec
c>

<ecc><slot>0</slot><row>12</row><displayItemName>+3.3V</displayItemName></ec
c>

<ecc><slot>0</slot><row>13</row><displayItemName>+2.5V</displayItemName></ec
c>

<ecc><slot>0</slot><row>14</row><displayItemName>+1.05V</displayItemName></ec
c>

<ecc><slot>0</slot><row>15</row><displayItemName>+1.5V</displayItemName></ec
c>

<ecc><slot>0</slot><row>16</row><displayItemName>+0.9V</displayItemName></ec
c>
<ecc><slot>0</slot><row>17</row><displayItemName>+3.3V
LDO</displayItemName></ecc>
```

```
<ecc><slot>0</slot><row>18</row><displayItemName>+1.0V</displayItemName></ec
c>

<ecc><slot>0</slot><row>19</row><displayItemName>+5.0V</displayItemName></ec
c>

<ecc><slot>0</slot><row>20</row><displayItemName>+0.75V</displayItemName></ec
c>

<ecc><slot>0</slot><row>21</row><displayItemName>+1.8V</displayItemName></ec
c>

<ecc><slot>0</slot><row>22</row><displayItemName>+1.2V</displayItemName></ec
c>
<ecc><slot>0</slot><row>23</row><displayItemName>Rear
Temp</displayItemName></ecc>
<ecc><slot>0</slot><row>24</row><displayItemName>PCIe-switch
temp</displayItemName></ecc>
<ecc><slot>0</slot><row>25</row><displayItemName>Front
Temp</displayItemName></ecc>
<ecc><slot>0</slot><row>26</row><displayItemName>CPU
temp</displayItemName></ecc>
<ecc><slot>0</slot><row>27</row><displayItemName>FPGA
+1.0V</displayItemName></ecc>
<ecc><slot>0</slot><row>28</row><displayItemName>FPGA
+1.8V</displayItemName></ecc>
<ecc><slot>0</slot><row>29</row><displayItemName>FPGA
temp</displayItemName></ecc>
<ecc><slot>1</slot><row>0</row><displayItemName>FPGA
+1.0V</displayItemName></ecc>
<ecc><slot>1</slot><row>1</row><displayItemName>FPGA
+1.8V</displayItemName></ecc>
<ecc><slot>1</slot><row>2</row><displayItemName>FPGA
temp</displayItemName></ecc>
<ecc><slot>2</slot><row>0</row><displayItemName>CPU1
temp</displayItemName></ecc>
<ecc><slot>2</slot><row>1</row><displayItemName>CPU2
temp</displayItemName></ecc>
<ecc><slot>2</slot><row>2</row><displayItemName>FPGA
+1.0V</displayItemName></ecc>
<ecc><slot>2</slot><row>3</row><displayItemName>FPGA
+1.8V</displayItemName></ecc>
<ecc><slot>2</slot><row>4</row><displayItemName>FPGA
temp</displayItemName></ecc>
<ecc><slot>2</slot><row>5</row><displayItemName>+1.0V
CPU1</displayItemName></ecc>
```

```
<ecc><slot>2</slot><row>6</row><displayItemName>+3.3V</displayItemName></ecc>
<ecc><slot>2</slot><row>7</row><displayItemName>+2.5V</displayItemName></ecc>
<ecc><slot>2</slot><row>8</row><displayItemName>+1.8V</displayItemName></ecc>
<ecc><slot>2</slot><row>9</row><displayItemName>+1.5V</displayItemName></ecc>
<ecc><slot>2</slot><row>10</row><displayItemName>+3.3V
LDO</displayItemName></ecc>
<ecc><slot>2</slot><row>11</row><displayItemName>+1.0V
CPU2</displayItemName></ecc>
<ecc><slot>2</slot><row>12</row><displayItemName>+0.75V
A</displayItemName></ecc>
<ecc><slot>2</slot><row>13</row><displayItemName>+0.75V
B</displayItemName></ecc>

<ecc><slot>2</slot><row>14</row><displayItemName>+1.0V</displayItemName></ec
c>

<ecc><slot>2</slot><row>15</row><displayItemName>+12.0V</displayItemName></ec
c>

<ecc><slot>2</slot><row>16</row><displayItemName>+1.65V</displayItemName></ec
c>
<ecc><slot>2</slot><row>17</row><displayItemName>Front
Temp</displayItemName></ecc>

*** Output truncated to save space ***

</device>
```

Input (multistream target, request JSON format, ECC data for specific displayItemName for specific slot):

```
curl -k -H "X-SESSION-ID: $token"

"https://192.168.0.1/ws/v2/status/device/ecc?slot=3&displayItemName=XCODE%20C
ore%201%20temp&js"
```

Expected output (values are for example purposes only):

Note Identifying keys (slot, row) are always output and cannot be filtered.

Note Using specific value filters returns only all items in the matched row.

```
{
    "device": {
        "ecc": {
            "slot": "3",
            "row": "18",
            "card": "NTM",
            "displayBoardName": "Transcoder1",
            "location": "XCODE Core 1 temp",
            "displayItemName": "XCODE Core 1 temp",
            "type": "Temperature",
            "value": "+56.9 C",
            "cardslot": "NTM3",
            "displayOrderFlag": "4"
        }
    }
}
```

Ethernet Status Command

Table 2.133 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/device/eth
Command Information	Return status of Ethernet ports. This command uses the ports referenced in Port Information, on page 22. However, in this API, all of the port numbers have +1 added. As a result, the Management Port is port1, DATA1 is port2, and DATA2 is port3.
HTTP Method	GET
Access Type	Read
Access Level	Guest, User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/device/eth"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.134 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
port (optional)	This is a key field for filtering output fields to a specified port. Ethernet port number Type: Integer Value: NTC Basic: 1 NTC MOIP: 1-3 NTC MS: 1-5

URI Argument	Description
js (optional)	Format output using JSON standard. Type: exist Values: any value or empty (any of js=1, js, js=0, or js=9999 are valid) Omitting this argument formats the output by default in XML

Table 2.135 Output Field Descriptions: these items may also be used as enhanced filtering Query Arguments for GET operations. See examples.

Output Field	Description
link	Ethernet link status Type: String Value: "Link Up" or "Link Down"
speed	Ethernet link speed Type: String Value: "10 Mbps", "100 Mbps", "1 Gbps", or "N/A"
duplex	Ethernet link speed Type: String Value: "Half", "Full", or "N/A"
xover	Ethernet MDI status (crossover) Type: String Value: "MDI", "MDIX", or "N/A"
name	Ethernet Port Name Type: String Value: "Management", "Data1", "Data2", "Data3", "Data4" NTC Basic: only Management port 1 available, NTC MOIP: only first three ports available, NTC Multi-Stream: all five ports available
ipv4addr	IP V4 Address Type: String Value: IP address, for example, 192.168.0.1
ipv4mask	IP V4 mask

Output Field	Description
	Type: digit Value: 8..30
ipv4defgw	IP V4 Default Gateway Type: String Value: IP address, for example, 192.131.244.254
dhcpenabled	Indicates if DHCP protocol is enabled or not Type: String Value: "No", "Yes" Default: "No"

Examples:

Input example (request XML output format for data for all Ethernet ports):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/eth"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<device><eth><port>1</port><link>Link Up</link><speed>1

Gbps</speed><duplex>Full</duplex><xover>N/A</xover></eth><eth><port>2</port><
link>Link

Down</link><speed>N/A</speed><duplex>N/A</duplex><xover>N/A</xover></eth><eth
><port>3</port>
<link>Link
Down</link><speed>N/A</speed><duplex>N/A</duplex><xover>N/A</xover></eth></de
vice>
```

Input example (get status of Ethernet port 1 in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/eth?port=1&js=1"
```

Expected output (values are for example purposes only):

```
{
    "device": {
        "eth": {
            "port": "1",
            "link": "Link Up",
            "speed": "1 Gbps",
            "duplex": "Full",
            "xover": "N/A"
        }
    }
}
```

Expected output, Release 4.50 or later (values are for example purposes only):

```
{
    "device": {
        "eth": {
            "port": "1",
            "link": "Link Up",
            "speed": "1 Gbps",
            "duplex": "Full",
            "xover": "N/A",
            "name": "Management",
            "ipv4addr": "169.168.0.1",
            "ipv4mask": "24",
            "ipv4defgw": "192.168.0.254",
            "dhcpenabled": "No"
        }
    }
}
```

Input (get only the speed of Ethernet port 1 in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/eth?port=1&speed&js=1"
```

Expected output (values are for example purposes only):

```
{
    "device": {
        "eth": {
            "port": "1",
            "speed": "1 Gbps"
        }
    }
}
```

Power Status Command

Table 2.136 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/device/power
Command Information	Return power status of all or individual slot(s) within the chassis.
HTTP Method	GET
Access Type	Read
Access Level	Guest, User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/device/power"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.137 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
slot	Slot number within the chassis Type: Integer Value: 0 to (number of slots in chassis – 1)
js (optional)	Format output using JSON standard Type: exist Values: any value or empty (any of js=1, js, js=0, or js=9999 are valid) Omitting this argument formats the output by default in XML.

In Release 3.75 or later, the output field items below may also be used as enhanced filtering Query Arguments for GET operations. See examples.

Table 2.138 Output Field Descriptions

Output Field	Description
slot	Slot number within the chassis Type: Integer Value: 0 to (number of slots in chassis – 1)
good	Status of "power good" signal Type: Boolean Value: "Yes" or "No"
displayBoardName	User Friendly Name of board in device slot

Output Field	Description
	Type: String Value: "Controller", "Tuner", "CI Option", or "Decoder" or "TranscoderX", or "HEVC ProcX", or "MemModule" where the X is replaced with the unique instance number (counting, starting with 1) of the particular board type that was detected by the software at target initialization time.
status	Status of slot check on power up Type: Boolean Value: "Ok" or "Fault"

Input example (request output in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/power?js=1"
```

Expected output (values are for example purposes only):

```
{
    "device": {
        "power": [
            {
                "slot": "0",
                "good": "Yes",
                "displayBoardName": "Controller",
                "status": "Ok"
            },
            {
                "slot": "1",
                "good": "Yes",
                "displayBoardName": "Tuner",
                "status": "Ok"
            },
            {
                "slot": "2",
                "good": "Yes",
                "displayBoardName": "HEVC Proc1",
                "status": "Ok"
            },
            {
                "slot": "3",
                "good": "Yes",
                "displayBoardName": "Transcoder1",
                "status": "Ok"
            },
            {
                "slot": "4",
                "good": "Yes",
                "displayBoardName": "Transcoder2",
                "status": "Ok"
            },
            {
                "slot": "5",
                "good": "Yes",
                "displayBoardName": "HEVC Proc2",
                "status": "Ok"
            },
            {
                "slot": "6",
                "good": "Yes",
                "displayBoardName": "CI option",
                "status": "Ok"
            }
        ]
    }
}
```

Expected output when Storage Card or Memory Module (NSD) is populated (slot 7) :

```
{
   "device": {
        "power": [
           {
                "slot": "0",
                "good": "Yes",
                "displayBoardName": "Controller",
                "status": "Ok"
           },
           {
                "slot": "1",
                "good": "Yes",
                "displayBoardName": "Tuner",
                "status": "Ok"
           },
           {
                "slot": "2",
                "good": "Yes",
                "displayBoardName": "HEVC Proc1",
                "status": "Ok"
           },
           {
                "slot": "3",
                "good": "Yes",
                "displayBoardName": "Transcoder1",
                "status": "Ok"
            },
            {
                "slot": "4",
                "good": "Yes",
                "displayBoardName": "Transcoder2",
                "status": "Ok"
            },
            {
                "slot": "5",
                "good": "Yes",
                "displayBoardName": "HEVC Proc2",
                "status": "Ok"
            },
            {
                "slot": "6",
                "good": "Yes",
                "displayBoardName": "CI option",
                "status": "Ok"
            },
            {
                "slot": "7",
                "good": "Yes",
                "displayBoardName": "MemModule",
                "status": "Ok"
            }
        ]
    }
}
```

Input example (Get status of slot power for slot 1 in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/power?slot=1&j
```

Expected output (values are for example purposes only):

```
{
    "device": {
        "power": {
            "slot": "1",
            "good": "Yes",
            "displayBoardName": "Tuner",
            "status": "Ok"
        }
    }
}
```

Input (multistream target, request JSON format, power data for specific displayBoardName):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/ecc?
displayBoardName=HEVC%20Proc2&js"
```

Expected output (values are for example purposes only):

Note Identifying keys (slot) are always output and cannot be filtered.

Note Using specific value filters returns only all items in the matched row.

```
{
    "device": {
        "power": {
            "slot": "5",
            "good": "Yes",
            "displayBoardName": "HEVC Proc2",
            "status": "Ok"
        }
    }
}
```

Storage Status Command

Table 2.139 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/device/storage
Command Information	Return storage status details when a Storage (Memory Module) card is populated.
HTTP Method	GET
Access Type	Read
Access Level	Guest, User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/storage"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.140 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js (optional)	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML

The following items may also be used as enhanced filtering Query Arguments for GET operations. See examples.

Table 2.141 Output Field Descriptions

Output Field	Description
present	NSD Storage card present or not Type: Boolean Value: "Yes" or "No" **Note** Other fields may be empty string or 0 values when this value is "No".
type	NSD Storage card type Type: String Value: "SATA", "PCIe", or "None" Currently only PCIe type will be available and supported.
sector	NSD Storage sectors Type: Integer Value: 0 … 65535
size	NSD Storage size Type: String Value: Size of storage device, typically in GB
model	NSD Storage model Type: String Value: Model of the storage device
serial	NSD Storage serial Type: String

Output Field	Description
	Value: Serial alpha-numeric id of the storage device
fwrev	NSD Storage firmware revision Type: String Value: Firmware revision of the storage device
vendor	NSD Storage vendor Type: String Value: Vendor identifier of the storage device (may be empty)

Examples:

Input (request output in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/device/storage?js=1"
```

Expected output when no storage card present (values are for example purposes only):

```
{
     "device": {
        "storage": {
                "present": "No",
                "type": "None",
                "sector": "0",
                "size": "",
                "model": "",
                "serial": "",
                "fwrev": "",
                "vendor": ""
        }
     }
}
```

Expected output when storage card is present (values are for example purposes only):

```
{
     "device": {
        "storage": {
                "present": "Yes",
                "type": "PCIe",
                "sector": "781422768",
                "size": "372.6GB",
                "model": "ST400KN001",
                "serial": "ZC3020B0",
                "fwrev": "NYMV0202",
                "vendor": ""
            }
        }
}
```

Alarms and Warnings Status Commands

This section describes the status commands related to alarms and warnings.

Fault Details Status Command

Table 2.142 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/faults
Command Information	Returns all faults status details information. **Note** No status record information is returned if the sum of values of actalarms and actwarns from the summary section is 0.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/faults"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.143 Command Control URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js (optional)	Format output using JSON standard. Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.
session (optional)	Alternate method of setting Login Token session ID. Use either one of this method or -H "X-SESSION-ID: $token" where $token represents value of session ID from Login command response); do not use both methods simultaneously. Type: String

Table 2.144 Output Field Descriptions and URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

Output Field	Description
type	Fault type. Indicates whether the fault is an alarm or warning. Type: String Value: "Alarm", "Warning"
name	Fault name. Unique name of the fault (includes instance id if applicable). This string is padded with spaces to always fill the indicated maxLength. Type: String minLength: 23 (not including NULL terminator) maxLength: 23 (not including NULL terminator)
details	Fault details. Additional detailed description for the fault beyond that available in the fault name field. Type: String maxLength: 63 (not including NULL terminator)
setsince	Date and time that the fault was triggered (format "yyyy/mm/dd hh:mm:ss"). Type: String Format: Modified version of RFC3339 datetimeonly format (replace the T with space), including leap-year validation. Regex pattern is: ^((((19|20)(([02468][048])|([13579][26]))[\\/.]02[\\/.]29))| ((20[0-9][0-9])|(19[0-9][0-9]))[\\/.] ((((0[1-9])|(1[0-2]))[\\/.] ((0[1-9])|(1\\d)|(2[0-8])))((((0[13578])|(1[02]))[\\/.]31| (((0[1,3-9])|(1 [0-2]))[\\/.](29|30)))))) (?:(?:([01]?\\d|2[0-3]):)?([0-5]?\\d):)?([0-5]?\\d)$
setdatetim	Date and time that the fault was triggered (format "yyyy/mm/dd hh:mm:ss"). Type: String Format: Modified version of RFC3339 datetimeonly format (replace the T with space), including leap-year validation. See setsince description for regex pattern.
rstdatetim	Date and time that the fault was cleared (format "yyyy/mm/dd hh:mm:ss"). Type: String Format: Modified version of RFC3339 datetimeonly format (replace the T with space), including leap-year validation. See setsince description for regex pattern.
state	Fault state. type: String

Output Field	Description
	Values: "Clear", "Set", "Clear By Reset"
actalarms	Number of active alarms. Type: Integer Minimum: 0 Maximum: 4294967295
actwarns	Number of active warnings. Type: Integer Minimum: 0 Maximum: 4294967295
clrfaults	Number of cleared faults. Type: Integer Minimum: 0 Maximum: 4294967295
textid ("key" for "settings" section only)	Unique internal text ID of the fault. When the fault classification can only have one instance, this will be a string representation of the enumerated integer id (FAULT_CLASS) for that classification. When the fault classification can have multiple instances, then the format of this field will be a string representation of two integers seperated by a space character (%20 in URL syntax). The first integer in the textid will be the enumerated id value for the fault classification (FAULT_CLASS) whereas the second integer is the instance value (INDX). The combined FAULT_CLASS and INDX (if any) value is used to generate the unique part of the fault name field value that distinguishes it from other faults. Type: String Format: Regex pattern is: ^(\d+\s){0,1}\d+$
faultnum	Unique numeric code for the fault. The classification id (FAULT_CLASS) and instance value (INDX) are base inputs to the fault number (FAULT_NUM) used for tracking and configuration settings of the unique fault. The conversion formula is FAULT_NUM = (FAULT_CLASS << 16) | (INDX << 8).

Output Field	Description
	Type: Integer Minimum: 0 Maximum: 4294967295
severity	Indicates whether the fault is major, minor, warning, information, indeterminate, or critical Type: String Value: "Major", "Minor", "Warning", "Information", "Indeterminate", or "Critical"
lastdatetim	Indicates when the fault was created or cleared (format "yyyy/mm/dd hh:mm:ss"). Type: String Format: Modified version of RFC3339 datetimeonly format (replace the T with space), including leap-year validation. Regex pattern is: ^((((19|20)(([02468][048])|([13579][26]))[\\/.]02[\\/.]29))| ((20[0-9][0-9])|(19[0-9][0-9]))[\\/.] ((((0[1-9])|(1[0-2]))[\\/.] ((0[1-9])|(1\\d)|(2[0-8])))((((0[13578])|(1[02]))[\\/.]31| (((0[1,3-9])|(1 [0-2]))[\\/.](29|30)))))) (?:(?:([01]?\\d|2[0-3]):)?([0-5]?\\d):)?([0-5]?\\d)$
trapstate	Trap state. Type: String Value: "Clear" or "Set"
relay	Indicates which relay number this fault should activate. Type: Integer Minimum: 0 Maximum: 8
mibpriority	SNMP MIB Priority of the fault. Lower number is higher priority. Type: Integer Minimum: 0 Maximum: 1024
priority ("key" when used with "status" section only)	Priority of the fault. Lower number is higher priority. Type: Integer Minimum: 0 Maximum: 1024

Note All indicated QueryString arguments are optional and read-only when used with the ws/v2/status/faults APIs. Indicating one or more Query Arguments without a value will filter output to only record items matching each argument. Indicating one or more Query Arguments with a value will filter output to only the record(s) with a match for the argument=value request. Arguments marked as "key" will always be returned in the indicated sections of the output schema.

Input example (read back all faults info in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults?js=1"
```

Expected output (values are for example purposes only):

```
{
    "faults": {
        "summary": {
            "actalarms": "1",
            "actwarns": "1",
            "clrfaults": "151"
        },
        "status": [
            {
                "priority": "9",
                "textid": "6 1",
                "name": "Input  1 Signal Status",
                "type": "Alarm",
                "details": "RF3 Signal is lost",
                "setsince": "2019/03/27 07:02:16",
                "mibpriority": "9"
            },
            {
                "priority": "186",
                "textid": "53 1",
                "name": "Transport Error        ",
                "type": "Warning",
                "details": "RF3 Transport Error Indicator",
                "setsince": "2019/03/27 07:02:32",
                "mibpriority": "186"
            }
        ],
        "history": [
            {
                "sequence": "3351",
                "name": "Input  1 Signal Status",
                "type": "Alarm",
                "details": "RF3 Signal is locked",
                "setdatetim": "2019/03/27 06:59:37",
                "rstdatetim": "2019/03/27 07:01:13",
                "state": "Clear"
            },
            {
                "sequence": "3352",
                "name": "Transport Error        ",
                "type": "Warning",
                "details": "RF3 Continuity Count Error Cleared",
                "setdatetim": "2019/03/27 07:01:28",
                "rstdatetim": "2019/03/27 07:01:44",
                "state": "Clear"
            },
            {
                "sequence": "3353",
                "name": "Transport Error        ",
                "type": "Warning",
                "details": "RF3 Continuity Count Error Cleared",
                "setdatetim": "2019/03/27 07:02:00",
                "rstdatetim": "2019/03/27 07:02:16",
                "state": "Clear"
            },
            {
                "sequence": "3354",
                "name": "Input  1 Signal Status",
                "type": "Alarm",
                "details": "RF3 Signal is lost",
                "setdatetim": "2019/03/27 07:02:16",
                "rstdatetim": "",
                "state": "Set"
            },
            {
                "sequence": "3355",
                "name": "Transport Error        ",
                "type": "Warning",
                "details": "RF3 Transport Error Indicator",
                "setdatetim": "2019/03/27 07:02:32",
                "rstdatetim": "",
                "state": "Set"
            }
        ],
        "settings": [
            {
                "textid": "10 1",
                "faultnum": "655360",
                "name": "PE  1: ISE Not Auth    ",
                "type": "Alarm",
                "severity": "Minor",
                "lastdatetim": "2019/03/27 02:23:59",
                "trapstate": "Clear",
                "details": "",
                "relay": "1",
                "mibpriority": "43",
                "priority": "43"
            },
            {
                "textid": "10 10",
                "faultnum": "657664",
                "name": "PE 10: ISE Not Auth    ",
                "type": "Alarm",
                "severity": "Minor",
                "lastdatetim": "2019/03/27 02:23:59",
                "trapstate": "Clear",
                "details": "",
                "relay": "1",
                "mibpriority": "52",
                "priority": "52"
            },
            {
                "textid": "10 11",
                "faultnum": "657920",
                "name": "PE 11: ISE Not Auth    ",
                "type": "Alarm",
                "severity": "Minor",
                "lastdatetim": "2019/03/27 02:23:59",
                "trapstate": "Clear",
                "details": "",
                "relay": "1",
                "mibpriority": "53",
                "priority": "53"
            },
            {
                "textid": "10 12",
                "faultnum": "658176",
                "name": "PE 12: ISE Not Auth    ",
                "type": "Alarm",
                "severity": "Minor",
                "lastdatetim": "2019/03/27 02:23:59",
                "trapstate": "Clear",
                "details": "",
                "relay": "1",
                "mibpriority": "54",
                "priority": "54"
            },
            {
                               // Several array entries removed from example
                               // in order to save space in document
            },
            {
                "textid": "92 1",
                "faultnum": "6029312",
                "name": "Xcode License Fault    ",
                "type": "Alarm",
                "severity": "Major",
                "lastdatetim": "2019/03/27 02:23:59",
                "trapstate": "Clear",
                "details": "",
                "relay": "1",
                "mibpriority": "261",
                "priority": "261"
            }
        ]
    }
}
```

Fault Actions Status Command

Table 2.145 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/faults/actions
Command Information	Clears fault history, alarms, or warnings.
HTTP Method	POST
Access Type	Write
Access Level	User, Admin
Syntax	One of the following: POST "https://192.168.0.1/ws/v2/status/faults/actions/clralarms" POST "https://192.168.0.1/ws/v2/status/faults/actions/clrwarns" POST "https://192.168.0.1/ws/v2/status/faults/actions/clrhistory"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.146 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js (optional)	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML
session (optional)	Alternate method of setting Login Token ID. Use either one of this method or -H "X-SESSION-ID: $token" (where $token represents value of session id from Login command response); do not use both methods simultaneously. Type: String

Input example (Clear Alarm Faults):

```
curl -k -X POST -H "Accept: application/json" -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/actions/clralarms"
```

Table 2.147 Expected output (values are for example purposes only)

Output Field	Description
clralarms	Clear Alarm Faults
clrwarns	Clear Warnings
clrhistory	Clear Fault History

Examples:

Input: (Clear Alarm Faults):

```
curl -k -X POST -H "Accept: application/json" -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/actions/clralarms"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><response><code>10</code>
<result>success</result><message></message></response>Expected output (values
are for example purposes only):
```

Input (Clear Warnings):

```
curl -k -X POST -H "Accept: application/json" -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/actions/clrwarns"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><response>
<code>10</code><result>success</result><message></message></response>
```

Input (Clear Fault History):

```
curl -k -X POST -H "Accept: application/json" -H
"X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/actions/clrhistory"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><response>
<code>10</code><result>success</result><message></message></response>
```

Faults Summary Status Command

Table 2.148 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/faults/summary
Command Information	Returns all faults status summary information.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/faults/summary"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.149 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js (optional)	Format output using JSON standard. Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.
session	Alternate method of setting Login Token session ID.

URI Argument	Description
(optional)	Use either one of this method or -H "X-SESSION-ID: $token" (where $token represents value of session id from Login command response); do not use both methods simultaneously. Type: String

Table 2.150 Output Field Descriptions and URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

Output Field	Description
actalarms	Number of active alarms Type: Integer Value: 0 ... 4294967295
actwarns	Number of active warnings Type: Integer Value: 0 ... 4294967295
clrfaults	Number of cleared faults Type: Integer Value: 0 ... 4294967295

Note All indicated QueryString arguments are optional and read-only when used with the ws/v2/status/faults APIs. Indicating one or more Query Arguments without a value will filter output to only record items matching each argument. Indicating one or more Query Arguments with a value will filter output to only the record(s) with a match for the argument=value request. Arguments marked as "key" will always be returned in the indicated sections of the output schema.

Input example (read back all faults summary in default XML format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/summary"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<faults>
<summary>
```

```
                <actalarms>1</actalarms>
                <actwarns>3</actwarns>
                <clrfaults>144 </clrfaults>
</summary>
</faults>
```

Input example (read back all faults summary in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/summary?js=1"
```

Expected output (values are for example purposes only):

```
{
    "faults": {
        "summary": {
            "actalarms": "1",
            "actwarns": "3",
            "clrfaults": "144"
        }
    }
}
```

Faults Status Command

Table 2.151 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/faults/status
Command Information	Returns only faults status information. **Note** No status record information is returned if the sum of values of actalarms and actwarns from the result of GET ws/v2/status/faults/summary is 0.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/faults/status"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.152 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js (optional)	Format output using JSON standard. Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.
session (optional)	Alternate method of setting Login Token session ID. Use either one of this method or -H "X-SESSION-ID: $token" (where $token represents value of session id from Login command response); do not use both methods simultaneously. Type: String.

Table 2.153 Output Field Descriptions and URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

Output Field	Description
priority (key)	Priority of the fault. Lower number is higher priority. Type: Integer Minimum: 0 Maximum: 1024
textid	Unique internal text ID of the fault. When the fault classification can only have one instance, this will be a string representation of the enumerated integer id (FAULT_CLASS) for that classification. When the fault classification can have multiple instances, then the format of this field will be a string representation of two integers seperated by a space character (%20 in URL syntax). The first integer in the textid will be the enumerated id value for the fault classification (FAULT_CLASS) whereas the second integer is the instance value (INDX). The combined FAULT_CLASS and INDX (if any) value is used to generate the unique part of the fault name field value that distinguishes it from other faults. Type: String

Output Field	Description
	Format: Regex pattern is: ^(\d+\s){0,1}\d+$
type	Fault Type Type: String Value: "Alarm", "Warning"
name	Fault name. Unique name of the fault (includes instance id if applicable). This string is padded with spaces to always fill the indicated maxLength. Type: String minLength: 23 (not including NULL terminator) maxLength: 23 (not including NULL terminator)
details	Fault details. Additional detailed description for the fault beyond that available in the fault name field. Type: String maxLength: 63 (not including NULL terminator)
setsince	Date and time that the fault was triggered (format "yyyy/mm/dd hh:mm:ss"). Type: String Format: Modified version of RFC3339 datetimeonly format (replace the T with space), including leap-year validation. Regex pattern is: ^((((19|20)(([02468][048])|([13579][26]))[\\/.]02[\\/.]29))| ((20[0-9][0-9])|(19[0-9][0-9]))[\\/.]((((0[1-9])|(1[0-2]))[\\/.] ((0[1-9])|(1\\d)|(2[0-8])))((((0[13578])|(1[02]))[\\/.]31| (((0[1,3-9])|(1[0-2]))[\\/.](29|30)))))) (?:(?:([01]?\\d|2[0-3]):)?([0-5]?\\d):)?([0-5]?\\d)$
mibpriority	SNMP MIB Priority of the fault. Lower number is higher priority. Type: Integer Minimum: 0 Maximum: 1024

Examples:

Input (read back faults status in default XML format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/status"
```

Expected output (values are for example purposes only):

```
?xml version="1.0" encoding="UTF-8"?>
<faults>
  <status>
    <priority>9</priority>
    <textid>6 1</textid>
    <name>Input  1 Signal Status</name>
    <type>Alarm</type>
    <details>RF3 Signal is lost</details>
    <setsince>2019/03/27 07:02:16</setsince>
    <mibpriority>9</mibpriority>
  </status>
  <status>
    <priority>198</priority>
    <textid>64 1</textid>
    <name>TDT warning            </name>
    <type>Warning</type>
    <details>RF3 TDT is lost</details>
    <setsince>2019/03/27 07:03:12</setsince>
    <mibpriority>198</mibpriority>
  </status>
</faults>
```

Expected output (values are for example purposes only), when no warning or alarms are currently active on the system:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<response><code>10</code><result>success</result>
<message>No status records found - must have active alarms and/or warnings
(see faults/summary)</message></response>
```

Input (read back faults status in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/status?js=1"
```

Expected output (values are for example purposes only), when any warning or alarms are currently active on the system:

```
{
    "faults": {
        "status": [
            {
                "priority": "9",
                "textid": "6 1",
                "name": "Input  1 Signal Status",
                "type": "Alarm",
                "details": "RF3 Signal is lost",
                "setsince": "2019/03/27 07:02:16",
                "mibpriority": "9"
            },
            {
                "priority": "198",
                "textid": "64 1",
                "name": "TDT warning            ",
                "type": "Warning",
                "details": "RF3 TDT is lost",
                "setsince": "2019/03/27 07:03:12",
                "mibpriority": "198"
            }
        ]
    }
}
```

Expected output (values are for example purposes only), when no warning or alarms are currently active on the system:

```
{
    "response": {
        "code": "10",
        "result": "success",
        "message": "No status records found - must have active alarms and/or
warnings (see faults/summary)"
    }
}
```

Faults History Status Command

Table 2.154 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/faults/history
Command Information	Returns only faults history information.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin

Command Detail	Description
Syntax	GET "https://192.168.0.1/ws/v2/status/faults/history"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.155 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js (optional)	Format output using JSON standard. Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.
session (optional)	Alternate method of setting Login Token session ID. Use either one of this method or -H "X-SESSION-ID: $token" (where $token represents value of session id from Login command response); do not use both methods simultaneously. Type: String

The following table lists the output fields:

Table 2.156 Output Field Descriptions and URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

Output Field	Description
type	Fault Type Type: String Values: "Alarm" or "Warning"
name	Fault name. Unique name of the fault (includes instance id if applicable). This string is padded with spaces to always fill the indicated maxLength. Type: String minLength: 23 (not including NULL terminator) maxLength: 23 (not including NULL terminator)
details	Fault details. Additional detailed description for the fault beyond that available in the fault name field. Type: String

Output Field	Description
	maxLength: 63 (not including NULL terminator)
setdatetim	Date and time fault was triggered Type : String Values: string format "yyyy/mm/dd hh:mm:ss"
rstdatetim	Date and time fault was cleared Type : String Values: string format "yyyy/mm/dd hh:mm:ss"
state	State of Fault Type: String Values: "Clear", "Set", "Clear By Reset"

Input example (Read back faults history in default XML format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/history"
```

Expected output (values are for example purposes only):

```
<faults>
<history>
                        <record>
                                <type>Alarm</type>
                                <name>Signal Status</name>
                                <details>ASI Signal - No Content </details>
                                <setdatetim>2010/11/12 23:55:34 </setdatetim>
                                <rstdatetim>2010/11/12 23:51:58 </rstdatetim>
                        </ record >
                        < record >
                                <type>Alarm</type>
                                <name>PE 1: Loss of Input</name>
                                <details>Loss of input detected</details>
                                <setdatetim>2010/11/12 23:53:41</ondatetim>
                                <rstdatetim> </rstdatetim>
                        </ record >
</ history >
</faults>
```

Input example (Read back faults history in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/history?js=1"
```

Expected output (values are for example purposes only):

```
{
    "faults": {
        "history": [
            {
                "name": "System Startup         ",
                "type": "Alarm",
                "details": "Auto reset after timeout",
                "setdatetim": "2016/12/19 17:33:46",
                "rstdatetim": "2016/12/19 17:33:47",
                "state": "Clear"
            },
            {
                "name": "System Startup         ",
                "type": "Alarm",
                "details": "Auto reset after timeout",
                "setdatetim": "2016/12/23 15:09:12",
                "rstdatetim": "2016/12/23 15:09:13",
                "state": "Clear"
            },
            {
                "name": "System Startup         ",
                "type": "Alarm",
                "details": "Auto reset after timeout",
                "setdatetim": "2016/12/24 07:48:23",
                "rstdatetim": "2016/12/24 07:48:24",
                "state": "Clear"
            },
            {
                "name": "System Startup         ",
                "type": "Alarm",
                "details": "Auto reset after timeout",
                "setdatetim": "2016/12/24 09:21:32",
                "rstdatetim": "2016/12/24 09:21:33",
                "state": "Clear"
            },
            {
                "name": "System Startup         ",
                "type": "Alarm",
                "details": "Auto reset after timeout",
                "setdatetim": "2016/12/24 09:21:31",
                "rstdatetim": "2016/12/24 09:21:32",
                "state": "Clear"
                }
            ]
        }
}
```

Faults Settings Status Command

Table 2.157 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/faults/settings
Command Information	Returns only faults settings information (Read-Only).
HTTP Method(s)	GET
Access Type	Read
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/faults/settings"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.158 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js (optional)	Format output using JSON standard. Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.
session (optional)	Alternate method of setting Login Token session ID. Use either one of this method or -H "X-SESSION-ID: $token" (where $token represents value of session id from Login command response); do not use both methods simultaneously. Type: String

Table 2.159 Output Field Descriptions and URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

Output Field	Description
textid (key)	Unique internal text ID of the fault.

Output Field	Description
	When the fault classification can only have one instance, this will be a string representation of the enumerated integer id (FAULT_CLASS) for that classification. When the fault classification can have multiple instances, then the format of this field will be a string representation of two integers seperated by a space character (%20 in URL syntax). The first integer in the textid will be the enumerated id value for the fault classification (FAULT_CLASS) whereas the second integer is the instance value (INDX). The combined FAULT_CLASS and INDX (if any) value is used to generate the unique part of the fault name field value that distinguishes it from other faults. Type: String Format: Regex pattern is: ^(\d+\s){0,1}\d+$
faultnum	Unique numeric code for the fault. The classification id (FAULT_CLASS) and instance value (INDX) are base inputs to the fault number (FAULT_NUM) used for tracking and configuration settings of the unique fault. The conversion formula is FAULT_NUM = (FAULT_CLASS << 16) | (INDX << 8). Type: Integer Minimum: 0 Maximum: 4294967295
name	Fault name. Unique name of the fault (includes instance id if applicable). This string is padded with spaces to always fill the indicated maxLength. Type: String minLength: 23 (not including NULL terminator) maxLength: 23 (not including NULL terminator)
type	Fault type. Indicates whether the fault is an alarm or warning. Type: String Value: "Alarm", "Warning"
severity	Indicates whether the fault is major, minor, warning, information, indeterminate, or critical. Type: String Value: "Major", "Minor", "Warning", "Information", "Indeterminate", or "Critical"

Output Field	Description
lastdatetim	Indicates when the fault was created or cleared (format "yyyy/mm/dd hh:mm:ss"). Type: String Format: Modified version of RFC3339 datetimeonly format (replace the T with space), including leap-year validation. Regex pattern is: ^((((19|20)(([02468][048])|([13579][26]))[\\/.]02[\\/.]29))| ((20[0-9][0-9])|(19[0-9][0-9]))[\\/.] ((((0[1-9])|(1[0-2]))[\\/.] ((0[1-9])|(1\\d)|(2[0-8])))((((0[13578])|(1[02]))[\\/.]31| (((0[1,3-9])|(1 [0-2]))[\\/.](29|30)))))) (?:(?:([01]?\\d|2[0-3]):)?([0-5]?\\d):)?([0-5]?\\d)$
trapstate	Trap state. Type: String Value: "Clear" or "Set"
details	Fault details. Additional detailed description for the fault beyond that available in the fault name field. Type: String maxLength: 63 (not including NULL terminator)
relay	Indicates which relay number this fault should activate. Type: Integer Minimum: 0 Maximum: 8
mibpriority	SNMP MIB Priority of the fault. Lower number is higher priority. Type: Integer Minimum: 0 Maximum: 1024
priority	Priority of the fault. Lower number is higher priority. Type: Integer Minimum: 0 Maximum: 1024

Note All indicated QueryString arguments are optional and read-only when used with the ws/v2/status/faults APIs. Indicating one or more Query Arguments without a value will filter output to only record items matching each argument. Indicating one or more Query Arguments with a value will filter output to only the record(s) with a match for the argument=value request. Arguments marked as "key" will always be returned in the indicated sections of the output schema.

Input: (Read back faults history in default XML format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/settings"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="UTF-8"?>
<faults>
  <settings>
    <textid>10 1</textid>
    <faultnum>655360</faultnum>
    <name>PE  1: ISE Not Auth    </name>
    <type>Alarm</type>
    <severity>Minor</severity>
    <lastdatetim>2019/03/27 02:23:59</lastdatetim>
    <trapstate>Clear</trapstate>
    <details>
    </details>
    <relay>1</relay>
    <mibpriority>43</mibpriority>
    <priority>43</priority>
  </settings>
  <settings>
    <textid>10 10</textid>
    <faultnum>657664</faultnum>
    <name>PE 10: ISE Not Auth    </name>
    <type>Alarm</type>
    <severity>Minor</severity>
    <lastdatetim>2019/03/27 02:23:59</lastdatetim>
    <trapstate>Clear</trapstate>
    <details>
    </details>
    <relay>1</relay>
    <mibpriority>52</mibpriority>
    <priority>52</priority>
  </settings>
  <settings>
    <textid>10 11</textid>
    <faultnum>657920</faultnum>
    <name>PE 11: ISE Not Auth    </name>
    <type>Alarm</type>
    <severity>Minor</severity>
    <lastdatetim>2019/03/27 02:23:59</lastdatetim>
    <trapstate>Clear</trapstate>
    <details>
    </details>
    <relay>1</relay>
    <mibpriority>53</mibpriority>
    <priority>53</priority>
  </settings>
  <settings>
    <textid>10 12</textid>
    <faultnum>658176</faultnum>
    <name>PE 12: ISE Not Auth    </name>
    <type>Alarm</type>
    <severity>Minor</severity>
    <lastdatetim>2019/03/27 02:23:59</lastdatetim>
    <trapstate>Clear</trapstate>
    <details>
    </details>
    <relay>1</relay>
    <mibpriority>54</mibpriority>
    <priority>54</priority>
  </settings>
  <settings>
                             // Several array entries removed from example
                             // in order to save space in document
  </settings>
  <settings>
    <textid>92 1</textid>
    <faultnum>6029312</faultnum>
    <name>Xcode License Fault    </name>
    <type>Alarm</type>
    <severity>Major</severity>
    <lastdatetim>2019/03/27 02:23:59</lastdatetim>
    <trapstate>Clear</trapstate>
    <details>
    </details>
    <relay>1</relay>
    <mibpriority>261</mibpriority>
    <priority>261</priority>
  </settings>
</faults>
```

Input: (Read back faults history in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/status/faults/settings?js"
```

Expected output (values are for example purposes only):

```
{
    "faults": {
        "settings": [
            {
                "textid": "10 1",
                "faultnum": "655360",
                "name": "PE  1: ISE Not Auth    ",
                "type": "Alarm",
                "severity": "Minor",
                "lastdatetim": "2019/03/27 02:23:59",
                "trapstate": "Clear",
                "details": "",
                "relay": "1",
                "mibpriority": "43",
                "priority": "43"
            },
            {
                "textid": "10 10",
                "faultnum": "657664",
                "name": "PE 10: ISE Not Auth    ",
                "type": "Alarm",
                "severity": "Minor",
                "lastdatetim": "2019/03/27 02:23:59",
                "trapstate": "Clear",
                "details": "",
                "relay": "1",
                "mibpriority": "52",
                "priority": "52"
            },
            {
                "textid": "10 11",
                "faultnum": "657920",
                "name": "PE 11: ISE Not Auth    ",
                "type": "Alarm",
                "severity": "Minor",
                "lastdatetim": "2019/03/27 02:23:59",
                "trapstate": "Clear",
                "details": "",
                "relay": "1",
                "mibpriority": "53",
                "priority": "53"
            },
            {
                "textid": "10 12",
                "faultnum": "658176",
                "name": "PE 12: ISE Not Auth    ",
                "type": "Alarm",
                "severity": "Minor",
                "lastdatetim": "2019/03/27 02:23:59",
                "trapstate": "Clear",
                "details": "",
                "relay": "1",
                "mibpriority": "54",
                "priority": "54"
            },
            {
                               // Several array entries removed from example
                               // in order to save space in document
            },
            {
                "textid": "92 1",
                "faultnum": "6029312",
                "name": "Xcode License Fault    ",
                "type": "Alarm",
                "severity": "Major",
                "lastdatetim": "2019/03/27 02:23:59",
                "trapstate": "Clear",
                "details": "",
                "relay": "1",
                "mibpriority": "261",
                "priority": "261"
            }
        ]
    }
}
```

Disaster Recovery (D/R) Status Command

Table 2.160 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr https://192.168.0.1/ws/v2/status/dr/<level_1_extension>
Command Information	This command can be used to read either all or a specific category of D/R details.
HTTP Method	GET (See Syntax row)
Access Type	Read

Command Detail	Description
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/status/dr", GET "https://192.168.0.1/ws/v2/status/dr/global", GET "https://192.168.0.1/ws/v2/status/dr/backupsglb", GET "https://192.168.0.1/ws/v2/status/dr/backupsrf", GET "https://192.168.0.1/ws/v2/status/dr/backupsmoip", GET "https://192.168.0.1/ws/v2/status/dr/backupszixi", GET "https://192.168.0.1/ws/v2/status/dr/backupsabr", GET "https://192.168.0.1/ws/v2/status/dr/backupsasi", GET "https://192.168.0.1/ws/v2/status/dr/backupnodes", GET "https://192.168.0.1/ws/v2/status/dr/origrftune", or GET "https://192.168.0.1/ws/v2/status/dr/origchans"

Table 2.161 URI <level_1_extension> Parameters (extension to the Command URL separated by /)

URI Argument	Description
None	Reads and returns current value of all D/R categories and related items.
global	Reads and returns current value of D/R global status.
backupsglb	Reads and returns current content of D/R backups (profiles) pool.
backupsrf	Reads and returns current content of D/R RF backups.
backupsmoip	Reads and returns current content of D/R MOIP backups.
backupszixi	Reads and returns current content of D/R Zixi backups.
backupsabr	Reads and returns current content of D/R ABR backups.
backupsasi	Reads and returns current content of D/R ASI backups.
backupnodes	Reads and returns current content of D/R backup nodes for all program entries.
origrftune	Reads and returns current content of D/R RF origin tuning.
origchans	Reads and returns current value of D/R origin channels.

Table 2.162 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js	Format output using JSON standard. Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.

See descriptions and other Arguments for each variant of this command in the sections below.

Disaster Recover (D/R) Global Status Command

Table 2.163 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/global
Command Information	Get of D/R global status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	All D/R global status GET "https://192.168.0.1/ws/v2/status/dr/global"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.164 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
status	D/R Status Type: String Value: "D/R in progress", "No Disaster - D/R Ready", "D/R disabled - Maintenance Mode", "NIT Retune Recovery in progress"
siglock	D/R Signal Lock Timeout Value Type: Integer Value: 5 … 255
siglost	D/R Signal Loss Timeout Vaule

URI Argument	Description
	Type: Integer Value: 5 … 2160000
verify	D/R Verification Timeout Value Type: Integer Value: 10 … 255
cfgsource	D/R Configuration Source Type: String Value: "None", "Uplink", "User"
maint	D/R Maintenance window status Type: String

GET Examples

Example 1: Retrieve the full D/R global status

The following example assumes that the user has successfully logged on to the unit, received the session ID, and set the unit to a variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP address to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/status/dr/global"
```

If successful, the following is the return body:

```
 "dr": {
        "global": {
            "status": "No Disaster - D/R Ready",
            "siglost": "120",
            "siglock": "40",
            "verify": "60",
            "cfgsource": "User","
            "maint" : ""
        }
    }
```

Disaster Recovery (D/R) Search Status Command

Note This is not supported in the current release.

Table 2.165 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/searchstate
Command Information	Get D/R Search Status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	All D/R global status GET "https://192.168.0.1/ws/v2/status/dr/searchstate"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.166 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
actinpname	Name of input which is currently being processed by the D/R Type: String Value: "None" (if D/R is not active), or name of input being processed (for example, “MOIP1”, “ZIXI1”, and so on).
tuningsrc	Node on the Search Path being currently processed Type: String Value: “N/A” (if DR is not active), “Origin”, “Backup 1”, “Backup 2”, or “Backup 3”
state	D/R processing state as applicable to the “tuningsrc” Type: String Value: “N/A” (if DR is not active), “Searching”, “Tuning to signal”, “Locking signal”, “Acquiring signal”, “Switching channel”, or “Locking channel”
numpendinp	Number of inputs pending D/R processing Type: Integer Value: 0 … 32
nextinpname	Name of the first pending input to be processed by the D/R Type: String Value: "None" (if there is no pending inputs), or name of input to be processed next (for example, “MOIP2”, “ZIXI2”, and so on)

Example 1: Retrieve the D/R search status

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H
"X-SESSION-ID: $token" -k "https://192.168.0.1/ws/v2/status/dr/searchstate"
```

If successful, the following is the return body:

```
"dr": {
        "searchstate": {
            "actinpname": "MOIP1",
            "tuningsrc": "Origin",
            "state": "Locking signal",
            "numpendinp": "1",
            "nextinpname": "MOIP2"
        }
    }
```

Disaster Recovery (D/R) Backups (Profiles) Global Status Command

Table 2.167 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/backupsglb
Command Information	Get of D/R Profiles status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	All D/R backup tuning status GET "https://192.168.0.1/ws/v2/status/dr/backupsglb" Single D/R backup tuning status, for example, GET "https://192.168.0.1/ws/v2/status/dr/backupsglb?recid=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.168 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
recid (Key)	Record index of D/R profile status

URI Argument	Description
	Type: Integer Values: 1 … 96
name	D/R Profile name Type: String
inputtype	D/R Profile input type Type: String Values: “NONE”, “RF”, “MOIP”, “ZIXI”, “ABR”, “ASI”
csiid	D/R Profile ID in the corresponding D/R Profile Tuning storage Type: Integer Values: range depends on input type
cfgsource	D/R Profile tuning source Type: String Values: "None", "Local", “Uplink”
rowstatus	D/R Profile record status Type: String Values: "Active", "Inactive"

GET Examples:

Example 1: Retrieve D/R Profile 1 status

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must hange the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/status/dr/backupsglb?recid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupsglb": {
            "recid": "1",
            "name": "Zixi_1",
            "inputtype": "ZIXI",
            "csiid": "1",
            "cfgsource": "None",
            "rowstatus": "Active"
        }
    }
```

Disaster Recover (D/R) RF Backup Status Command

Note This is supported on single-stream units only.

Table 2.169 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/backupsrf
Command Information	Get of D/R RF backup (profile) status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	All D/R backup tuning status. GET "https://192.168.0.1/ws/v2/status/dr/backupsrf" Single D/R RF backup tuning status, for example, GET "https://192.168.0.1/ws/v2/status/dr/backupsrf?csiid=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.170 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R RF backup ID Type: Integer Value: 1 … 16
csiidx (Key)	D/R RF backup IDX Type: Integer Value: 0 … 16 **Note** This is not supported in the current release.
inputname	D/R RF backup input name Type: String

URI Argument	Description
	Value: "RF1", "RF2", "RF3", or "RF4"
netid	D/R RF backup network id Type: Integer Value: 0 ~ 65535
streamid	D/R RF backup Stream ID Type: Integer Value: 1..6
mod	D/R RF backup modulation system Type: String Value: for example, "QPSK” or "DVB-S"
dnlkfreq	D/R RF backup Frequency [GHz] Type: Float Value: 0.0 … 15.0
symrate	D/R RF backup symbol rate Type: Float Value: 1.0 ... 45.0
fec	D/R RF backup FEC mode Type: String Value: "Auto", "1/2", "2/3", "3/4", "5/6", "7/8", or "8/9"
orbpos	D/R RF backup orbital position, in deg Type: Float Values: 0.0 ... 360.0
ewflag	D/R RF backup West/East Flag Type: String Value: "N/A", "East", or "West"
pol	D/R RF backup Orbital Polarization Type: String Value: "Horiz", "Vert", "Circ_l", "Circ_r", or "Auto"
rolloff	D/R RF backup roll-off Type: String

URI Argument	Description
	Value: ".20", ".25", ".35", or "Auto"
cfgsource	D/R RF backup tuning Source Type: String Value: "None", "Local", or "Uplink"
rowstatus	D/R RF backup record status Type: String Value: "Yes" or "No"

GET Example

Example 1: Retrieve D/R RF backup 1 status

The following example assumes that the user has successfully logged on to the unit, received the session ID, and set the unit to a variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must hange the IP address to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/status/dr/backupsrf?csiid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupsrf": {
            "csiid": "1",
            "csiidx": "0",
            "inputname": "RF1",
            "netid": "1",
            "streamid": "1",
            "mod": "N/A",
            "dnlkfreq": "3.4",
            "symrate": "2.0",
            "fec": "N/A",
            "orbpos": "0.0",
            "ewflag": "N/A",
            "pol": "Horiz",
            "rolloff": ".35",
            "cfgsource": "None",
            "rowstatus": "No"
        }
    }
}
```

Disaster Recovery (D/R) MOIP Backup Status Command

Note This is supported on multi-stream units only.

Table 2.171 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/backupsmoip
Command Information	Get of D/R MOIP backup (profile) status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	All D/R MOIP backups tuning status GET "https://192.168.0.1/ws/v2/status/dr/backupsmoip" Single D/R MOIP backup tuning status, for example, GET "https://192.168.0.1/ws/v2/status/dr/backupsmoip?csiid=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.172 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R MOIP backup ID Type: Integer Value: 1 … 32
cfgsource	D/R MOIP backup tuning source Type: String Value: "None", "Local", "Uplink"
name	D/R MOIP backup name Type: String Value: up to 64 characters
interface	D/R MOIP backup ETH interface Type: String Value: "Data1", "Data2", "Data3" (multi-stream units only), or "Data4" (multi-stream units only)

URI Argument	Description
usemcast	D/R MOIP backup 'Is Multicast Destination' flag Type: String Value: "Yes" or "No"
mcastaddr	D/R MOIP backup destination IP address Type: IPv4 address in standard notation
fecmode	D/R MOIP backup FEC Mode Type: String Value: "None", "1D", or "2D"
tsdestport	D/R MOIP backup TS destination UDP port Type: Integer Values: 1..65535 (some ports are reserved and cannot be used)
feccolport	D/R MOIP backup FEC Columns destination UDP port Type: Integer Values: 1..65535 (some ports are reserved and cannot be used)
fecrowport	D/R MOIP backup FEC Rows destination UDP port Type: Integer Values: 1..65535 (some ports are reserved and cannot be used)
dejalg	D/R MOIP backup de-jittering algorithm Type: String Value: "VBR" or "CBR"
dejlat	D/R MOIP de-jittering buffer latency, in ms Type: Integer Value: 40..150
rowstatus	D/R MOIP backup record status Type: String Value: "Active" or "Inactive"

GET Example

Example 1: Retrieve D/R MOIP backup 1 status

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must hange the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/status/dr/backupsmoip?csiid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupsmoip": {
            "csiid": "1",
            "cfgsource": "None",
            "name": "Moip_1",
            "interface": "Data1",
            "usemcast": "Yes",
            "mcastaddr": "225.1.1.1",
            "fecmode": "None",
            "tsdestport": "49152",
            "feccolport": "49154",
            "fecrowport": "49156",
            "dejalg": "VBR",
            "dejlat": "110",
            "rowstatus": "Active"
        }
    }
}
```

Disaster Recovery (D/R) Zixi Backup Status

Table 2.173 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/backupszixi
Command Information	Get of D/R Zixi backup (profile) status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	All D/R Zixi backups tuning status GET "https://192.168.0.1/ws/v2/status/dr/backupszixi" Single D/R Zixi backup tuning status, for example, GET "https://192.168.0.1/ws/v2/status/dr/backupszixi?csiid=1"

Command Detail	Description

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.174 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R Zixi backup ID Type: Integer Value: 1 … 4
name	D/R Zixi backup name Type: String Value: up to 64 characters
cfgsource	D/R Zixi backup tuning source Type: String Value: "None", "Local", or "Uplink"
netid	D/R Zixi backup network ID Type: Integer Value: 0 ~ 65535
type	D/R Zixi backup type Type: String Value: “Pull” (only supported option)
interface	D/R MOIP backup ETH interface Type: String Value: “Mgmt”, "Data1", "Data2", "Data3" (multi-stream units only), or "Data4" (multi-stream units only)
source	D/R Zixi backup source Type: String Value: up to 200 characters
sid	D/R Zixi backup Stream ID (name) Type: String Value: up to 30 characters

URI Argument	Description
port	D/R Zixi backup UDP port Type: Integer Value: 1..65535 (some ports are reserved and cannot be used)
latmode	D/R Zixi backup latency mode Type: String Value: “Static”, “Increasing”, or “Dynamic”
latency	D/R Zixi backup latency, in ms Type: Integer Value: 0..10000
fecmaxjitr	D/R Zixi backup FEC maximum jitter, in ms Type: Integer Value: 0 .. 10000
fecstufnul	D/R Zixi backup “FEC Stuffing NULL Packets Enabled” flag Type: String Value: “Yes” or “No”
decrkey	D/R Zixi backup decryption key Type: String Range: up to 64 characters
fecmode	D/R Zixi backup FEC mode Type: String Value: “Off”, “On”, or “Adaptive”
fecovhd	D/R Zixi backup FEC Overhead Type: Integer Value: 0 to 10000
fecblkms	D/R Zixi backup FEC block size, in ms Type: Integer Value: 0 to 10000
fecconawar	D/R Zixi backup FEC Content Aware Type: String

URI Argument	Description
	Value: “Yes” or “No”
rowstatus	D/R Zixi backup record status Type: String Value: "Active" or "Inactive"

GET Example

Example 1: Retrieve D/R Zixi backup 1 status

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must hange the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/status/dr/backupszixi?csiid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupszixi": {
            "csiid": "1",
            "name": "",
            "cfgsource": "None",
            "netid": "1",
            "type": "Pull",
            "interface": "Mgmt",
            "source": "",
            "sid": "",
            "port": "0",
            "latmode": "Static",
            "latency": "5000",
            "fecmaxjitr": "0",
            "fecstufnul": "No",
            "decrkey": "",
            "fecmode": "On",
            "fecovhd": "30",
            "fecblkms": "50",
            "fecconawar": "No",
            "rowstatus": "Inactive"
        }
    }
}
```

Disaster Recovery (D/R) ABR Backup Status Command

Note This is supported on multi-stream units only.

Table 2.175 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/backupsabr
Command Information	Get of D/R ABR backup (profile) status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	All D/R ABR backups tuning status GET "https://192.168.0.1/ws/v2/status/dr/backupsabr" Single D/R ABR backup tuning status, for example, GET "https://192.168.0.1/ws/v2/status/dr/backupsabr?csiid=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.176 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R ABR backup ID Type: Integer Values: 1 … 4
name	D/R ABR backup name Type: String Value: up to 64 characters
cfgsource	D/R ABR backup tuning source Type: String Value: "None", "Local", "Uplink"
interface	D/R ABR backup ETH interface Type: String Values: “Mngmt”, "Data1", "Data2", "Data3" (multi-stream units only), "Data4" (multi-stream units only)

URI Argument	Description
url	D/R ABR stream URL Type: String Values: up to 200 characters
latency	D/R ABR backup Target Latency, in ms Type: Integer Values: 0..4294967295
propdelay	D/R ABR backup Propagation Delay, in ms Type: Integer Values: 0..4294967295
vod	D/R ABR Live VOD Type: String Values: "Yes”, “No"
rowstatus	D/R ABR backup record status Type: String Values: "Active", "Inactive"

GET Examples

Example 1: Retrieve D/R ABR backup 1 status

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/status/dr/backupsabr?csiid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupsabr": {
            "csiid": "1",
            "name": "",
            "cfgsource": "None",
            "interface": "Mngmt",
            "url": "",
            "latency": "6000",
            "propdelay": "0",
            "vod": "No",
            "rowstatus": "Inactive"
        }
    }
}
```

Disaster Recovery (D/R) ASI Backup Status Command

Note This is supported on multi-stream units only.

Table 2.177 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/backupsasi
Command Information	Get of D/R ASI backup (profile) status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	All D/R ASI backups tuning status GET "https://192.168.0.1/ws/v2/status/dr/backupsasi" Single D/R ASI backup tuning status, for example, GET "https://192.168.0.1/ws/v2/status/dr/backupsasi?csiid=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.178 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R ASI backup ID Type: Integer Values: 1 (D9800 SS), 1..2 (D9800 MS)
name	D/R ASI backup name Type: String Value: up to 64 characters
port	D/R ASI port ID Type: Integer

URI Argument	Description
	Value: 1 (D9800 SS), 1..2 (D9800 MS)
cfgsource	D/R ASI backup tuning source Type: String Value: "None", "Local", "Uplink"
rowstatus	D/R ASI backup record status Type: String Values: "Active", "Inactive"

Example 1: Retrieve D/R ASI backup 1 status

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. Change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/status/dr/backupsasi?csiid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupsasi": {
            "csiid": "1",
            "name": "ASI1",
            "port": "1",
            "cfgsource": "None",
            "rowstatus": "Active"
        }
    }
}
```

Disaster Recover (D/R) Backup Nodes (Search Path) Status Command

Table 2.179 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/backupnodes
Command Information	Get of D/R backup node status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	All D/R backup nodes Status

Command Detail	Description
	GET "https://192.168.0.1/ws/v2/status/dr/backupnodes" Single D/R backup nodes status, for example, GET "https://192.168.0.1/ws/v2/status/dr/backupnodes?peid=PE1&recid=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.180 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
peid (Key)	D/R backup node Program Entry ID Type: String Values: "PE1" … "PE16" (single-stream units only), "PE1" … "PE32" (multi-stream units only)
recid (Key)	D/R backup node index for specific program entry Type: Integer Values: 1 … 16
csiid	D/R backup ID in the corresponding input-dependent backup status table Type: Integer Values: range is input-dependent
name	D/R backup name Type: String Value: up to 64 characters
bkpchan	D/R backup node channel number Type: Integer Values: 0 … 65535
bkpchandisp	D/R backup node channel display name Type: String Values: up to 8 characters

URI Argument	Description
rowstatus	D/R backup node status Type: String Values: "Yes", "No"

GET Examples

Example 1: Retrieve the status of D/R backup node 1 on PE1

The following example assumes that the user has successfully logged on to the unit, received the session ID, and set the unit to a variable token. In addition, it is assumed that the IP address of the unit is 192.168.0.1. You must change the IP address to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/status/dr/backupnodes?peid=PE1&recid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupnodes": {
            "peid": "PE1",
            "recid": "1",
            "csiid": "0",
            "name": "",
            "bkpchan": "0",
            "bkpchandisp": "",
            "rowstatus": "No"
        }
    }
}
```

Disaster Recover (D/R) RF Origin Tuning Command

Note This is supported on single-stream units only.

Table 2.181 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/origrftune
Command Information	Get of D/R RF Origin Tuning status.
HTTP Method	GET

Command Detail	Description
Access Type	Read
Access Level	User, Admin
Syntax	All D/R backup tuning status GET "https://192.168.0.1/ws/v2/status/dr/origrftune" Single D/R RF Origin Tuning parameter's status, for example, GET "https://192.168.0.1/ws/v2/status/dr/origrftune?mod"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.182 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
inputname	D/R RF backup input name Type: String Values: "RF1", "RF2", "RF3", "RF4"
netid	D/R RF backup network id Type: Integer Values: 0 ~ 65535
mod	D/R RF backup modulation system Type: String Values: for example, "QPSK", DVB-S"
dnlkfreq	D/R RF backup Frequency [GHz] Type: Float Values: 0.0 … 15.0
symrate	D/R RF backup symbol rate Type: Float Values: 1.0 ... 45.0
fec	D/R RF backup FEC Mode Type: String Values: "Auto", "1/2", "2/3", "3/4", "5/6", "7/8", "8/9"
orbpos	D/R RF backup Orbital Position [Deg] Type: Float

URI Argument	Description
	Values: 0.0 ... 360.0
ewflag	D/R RF backup West/East Flag Type: String Values: "N/A", "East", "West"
pol	D/R RF backup Orbital Polarization Type: String Values: "Horiz", "Vert", "Circ_l", "Circ_r", "Auto"
rolloff	D/R RF backup roll-off Type: String Values: ".20", ".25", ".35", "Auto"

GET Examples

Example 1: Retrieve D/R RF Origin Tuning status

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/status/dr/backupsrf?csiid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "origrftune": {
            "inputname": "RF1",
            "netid": "1",
            "mod": "8PSK DVB-S2",
            "dnlkfreq": "12.38",
            "symrate": "30.0",
            "fec": "AUTO",
            "orbpos": "0.0",
            "ewflag": "East",
            "pol": "Horiz",
            "rolloff": ".35"
        }
    }
}
```

Disaster Recover (D/R) Origin Channels Command

Table 2.183 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/status/dr/origchans
Command Information	Allows get of D/R origin channel status.
HTTP Method	GET
Access Type	Read
Access Level	User, Admin
Syntax	All D/R origin channels Status: GET "https://192.168.0.1/ws/v2/status/dr/origchans" Single D/R origin channels status, for example, GET "https://192.168.0.1/ws/v2/status/dr/origchans?peid=PE1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.184 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
peid (Key)	D/R origin channel Program Entry ID Type: String Values: "PE1" … "PE16" (single-stream units only), "PE1" … "PE16" (multi-stream units only)
origchan	D/R origin channel number Type: Integer Values: 0 … 65535
chandisp	D/R origin channel display name Type: String Values: up to 8 characters
inputname	D/R origin channel input name Type: String Values: up to 32 characters

GET Examples

Example 1: Retrieve origin channel of PE1

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/status/dr/origchans?peid=PE1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "origchans": {
            "peid": "PE1",
            "origchan": "101",
            "chandisp": "101",
            "inputname": "RF1"
        }
    }
}
```

Configuration API

This section lists all the configuration commands.

Input Configuration Command

Table 2.185 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/input
Command Information	Allows inputs to be read or configured.
HTTP Methods	GET, POST
Access Type	Read/Write (with complex input data)
Access Level	User, Admin
GET Syntax	GET "https://192.168.0.1/ws/v2/service_cfg/input"
POST Syntax	Either (using body data from command line) POST --header "Content-type:application/xml" –d "<input>some_XML_body_data</input>" "https://192.168.0.1/ws/v2/service_cfg/input"

Command Detail	Description
	OR (using body data from XML file) POST --header "Content-type:application/xml" –d @"C:/your_path/input.xml" "https://192.168.0.1/ws/v2/service_cfg/input"

Table 2.186 URI Parameters (extension to the Command URL separated by /)

URI Parameter	Description
default	By default (no additional URL extension), all data is returned in a GET, and any data may be set in a POST. See following command sections for API syntax that extends this command URL and allows returning or setting a categorized subset of the fields.

URI Query/Set Arguments (possible fields and values preceded by ? and separated by &): N/A

Input XML Field Descriptions:

The XML parameters are as described in each of the following input configuration API sections Arguments options

Output Fields:

Output fields returned in GET results are the same names used for the possible Arguments when appending each possible URI Parameter to this command.

POST (Settings Write) Examples:

Input with invalid setting (Setting Parameters using XML body data from command line):

```
curl -k -X POST -H "X-SESSION-ID: $token" --header
"Content-type:application/xml" -d "<input><rf><port>1</port>
<act>Yes</act><dnlkfreq>3.40></dnlkfreq></rf><rf><port>2</port>
<dnlkfreq>3.40</dnlkfreq></rf></input>" "https://192.168.0.1/ws/v2/service_
cfg/input"
```

Expected output (values are for example purposes only) when validation of input parameter(s) fail:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<response><code>10</code><result>failure</result>
<message>the downlink frequency band did not match the LO frequency
band</message></response>
```

Input with valid setting (Setting Parameters using XML body data from command line):

```
curl -k -X POST -H "X-SESSION-ID: $token" --header
"Content-type:application/xml" -d "<input><rf><port>1</port>

<act>Yes</act><dnlkfreq>12.31></dnlkfreq></rf><rf><port>2</port><dnlkfreq>3.4
0</dnlkfreq></rf></input>"
"https://192.168.0.1/ws/v2/service_cfg/input"
```

Expected output (values are for example purposes only) when validation of input parameter(s) pass:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<response><code>10</code><result>success</result><message></message></respons
e>
```

Input (Setting Parameters using body data from XML File):

```
curl -k -X POST -H "X-SESSION-ID: $token" --header
"Content-type:application/xml" -d @"C:/projects/input.xml"
"https://192.168.0.1/ws/v2/service_cfg/input"
```

Input File "input.xml" contains data as shown below:

```
<input>
    <rf>
        <port>1</port>
        <act>Yes</act>
           <dnlkfreq>12.31</dnlkfreq>
    </rf>
    <rf>
        <port>2</port>
       <act>No</act>
        <dnlkfreq>3.40</dnlkfreq>
     </rf>
    <asi>
       <port>1</port>
       <act>No</act>
   </asi>
</input>
```

Expected output (values are for example purposes only) when validation of input parameter(s) pass:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<response><code>10</code><result>success</result><message></message></respons
e>
```

GET (Settings Read) Examples:

Input (Request read back all input settings in XML format):

```
curl -k -H "X-SESSION-ID: $token" "https://192/168.0.1/ws/v2/service_
cfg/input"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<input><rf><port>1</port><act>Yes</act><dnlkfreq>12.31</dnlkfreq><symrate>30.
0</symrate>
<fec>Auto</fec><mod>DVB-
S2</mod><rolloff>Auto</rolloff><iq>Auto</iq><pwr>Off</pwr><sel22khz>Off</sel2
2khz>

<lo1>10.75</lo1><lo2>0.0</lo2><xover>0.0</xover><pol>Horizontal</pol><orbpos>
0.0</orbpos>

<ewflag>N/A</ewflag><afcrange>3.0</afcrange><strm1sel>Yes</strm1sel><strm2sel
>No</strm2sel>

<strm3sel>No</strm3sel><strm4sel>No</strm4sel><strm5sel>No</strm5sel><strm6se
l>No</strm6sel>

<acqmode>Basic</acqmode><netid>1</netid><freqmode>NIT</freqmode><svclstmode>R
igorous</svclstmode>

<niten>Yes</niten><sdten>Yes</sdten><paten>Yes</paten></rf><rf><port>2</port>
<act>No</act>
<dnlkfreq>12.31</dnlkfreq><symrate>30.0</symrate><fec>Auto</fec><mod>DVB-
S2</mod>

<rolloff>Auto</rolloff><iq>Auto</iq><pwr>Off</pwr><sel22khz>Off</sel22khz><lo
1>10.75</lo1>

<lo2>0.0</lo2><xover>0.0</xover><pol>Horizontal</pol><orbpos>0.0</orbpos><ewf
lag>N/A</ewflag>

<afcrange>3.0</afcrange><strm1sel>Yes</strm1sel><strm2sel>No</strm2sel><strm3
sel>No</strm3sel>

<strm4sel>No</strm4sel><strm5sel>No</strm5sel><strm6sel>No</strm6sel><acqmode
>Basic</acqmode>

<netid>1</netid><freqmode>NIT</freqmode><svclstmode>Rigorous</svclstmode><nit
en>Yes</niten>
```

```
<sdten>Yes</sdten><paten>Yes</paten></rf><rf><port>3</port><act>No</act><dnlk
freq>12.31</dnlkfreq>
<symrate>30.0</symrate><fec>Auto</fec><mod>DVB-
S2</mod><rolloff>Auto</rolloff><iq>Auto</iq>

<pwr>Off</pwr><sel22khz>Off</sel22khz><lo1>5.15</lo1><lo2>0.0</lo2><xover>0.0
</xover>

<pol>Horizontal</pol><orbpos>0.0</orbpos><ewflag>N/A</ewflag><afcrange>3.0</a
fcrange>

<strm1sel>No</strm1sel><strm2sel>No</strm2sel><strm3sel>No</strm3sel><strm4se
l>No</strm4sel>

<strm5sel>No</strm5sel><strm6sel>No</strm6sel><acqmode>Basic</acqmode><netid>
1</netid>

<freqmode>NIT</freqmode><svclstmode>Rigorous</svclstmode><niten>Yes</niten><s
dten>Yes</sdten>

<paten>Yes</paten></rf><rf><port>4</port><act>No</act><dnlkfreq>12.31</dnlkfr
eq>
<symrate>30.0</symrate><fec>Auto</fec><mod>DVB-
S2</mod><rolloff>Auto</rolloff><iq>Auto</iq>

<pwr>Off</pwr><sel22khz>Off</sel22khz><lo1>5.15</lo1><lo2>0.0</lo2><xover>0.0
</xover>

<pol>Horizontal</pol><orbpos>0.0</orbpos><ewflag>N/A</ewflag><afcrange>3.0</a
fcrange>

<strm1sel>No</strm1sel><strm2sel>No</strm2sel><strm3sel>No</strm3sel><strm4se
l>No</strm4sel>

<strm5sel>No</strm5sel><strm6sel>No</strm6sel><acqmode>Basic</acqmode><netid>
1</netid>

<freqmode>NIT</freqmode><svclstmode>Rigorous</svclstmode><niten>Yes</niten><s
dten>Yes</sdten>

<paten>Yes</paten></rf><asi><port>1</port><act>No</act><acqmode>Basic</acqmod
e><netid>1</netid>
<freqmode>User
Cfg</freqmode><svclstmode>Rigorous</svclstmode><niten>No</niten><sdten>Yes</s
dten>
<paten>Yes</paten></asi><moip><port>1</port><act>No</act><ipmode>SW
```

```
Map</ipmode>

<fecmode>None</fecmode><usemcast>Yes</usemcast><mcastaddr>225.1.1.1</mcastadd
r>

<tsdestport>49152</tsdestport><feccolport>49154</feccolport><fecrowport>49156
</fecrowport>

<usrsrcaddr>Yes</usrsrcaddr><srcfltmode>None</srcfltmode><dejalg>VBR</dejalg>
<acqmode>Basic</acqmode>
<netid>1</netid><freqmode>User
Cfg</freqmode><svclstmode>Rigorous</svclstmode><niten>No</niten>
<sdten>Yes</sdten><paten>Yes</paten></moip></input>
```

Note The parameter list for the configuration APIs in this and following sections are compliant with the UIC tables in the PowerVu Professional Receiver software application, which may use some special characters that are non-compliant with the allowed URL character set.

When using curl, URL strings that include arguments (the options after the ?) must be in quotes, as discussed in HTTP Request Methods, on page 14. In this case, the conversion of special characters is implicit.

However, when using a web browser, argument strings must be first converted to substitute URL safe "escape" replacements for special, reserved characters (as discussed in rfc2396).

For example, in URL safe format, spaces that are embedded in the URL string would typically need to be replaced by %20 (percent sign followed by the asci hex value equivalent of a space).

Since some parameters may have spaces or forward slashes or other reserved URL characters in them. We recommend that the web UI, and other similar implementations, pass the full URL string through a safe-url character converter before issuing the command.

For more information about safe URL encoding, see http://www.ietf.org/rfc/rfc2396.txt, or http://www.skorks.com/2010/05/what-every-developer-should-know-about-urls/.

RF Input Configuration Command

Table 2.187 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/input/rf
Command Information	Allows RF tuning input settings to be read or configured.

Command Detail	Description
HTTP Methods	GET, POST
Access Type	Read/Write (with complex input data)
Access Level	User, Admin
GET Syntax	One of the following: GET "https://192.168.0.1/ws/v2/service_cfg/input/rf", GET "https://192.168.0.1/ws/v2/service_cfg/input/rf?port=1", or GET "https://192.168.0.1/ws/v2/service_cfg/input/rf?port=1&dnlkfreq"
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/input/rf?port=[port#]&dnlkfreq=<Freq>..."

URI Parameters (extension to the Command URL separated by /): N/A

Note Setting Parameters using command line arguments are limited to maximum of 26 arguments after the ?.

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values, except the port which must be specified. For POST the port must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

Table 2.188 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
port (Key)	Input Port to be configured Type: Integer Values: 1 .. 16, range as shown below: NFE1 : 1..4 NFE2: 5..8 NFE3: 9..12 8 NFE4: 12..16 8 **Note** This does not refer to the NTC Ethernet ports, it refers specifically to the RF ports.
stream	Multi Input Stream ID. If not specified, default value of 1 is used

URI Argument	Description
(Key) (optional) This is not applicable to Version 1.x	Type: Integer Values: 1..6 (RF)
act	Activate Input Type: String Values: "Yes" or "No"
dnlkfreq	Downlink Frequency Type: Float Values: 0.0 .. 15.0
symrate	Symbol Rate Type: Float Values: 1.0 .. 45.0
fec	FEC Mode Type: String Values: "Auto", "1/2", "2/3", "3/4", "5/6", "7/8" Modulation Type of DVB-S2 must use a FEC value of "Auto"
mod	Modulation Type Type: String Values: "DVBS" or "DVB-S2"
rolloff	Rolloff Factor Type: String Values: ".20", ".25", ".35", "Auto"
Iq	IQ Mode Type: String Values: "Opposite", "Normal", "Auto"
pwr	LNB Power Type: String

URI Argument	Description
	Values: "Off", "13V", "18V", "H-NIT", "V-NIT"
sel22khz	22 KHz Selection Type: String Values: "Off", "On", "Auto"
lo1	LO Select 1 Type: Float Values: 0.0 … 15.0
lo2	LO Select 2 Type: Float Values: 0.0 … 15.0
xover	Cross Over Frequency Type: Float Values: 0.0 … 15.0
pol	Polarization Type: String Values: "Horizontal", "Vertical", "Auto"
orbpos	Orbital Position Type: Float Values: 0.0 .. 360.0
ewflag	East West Flag Type: String Values: "NA", "East", "West"
afcrange	AFC Range Value Type: Float Values: 0.0 .. 5.0
strm1Sel This is not applicable to Version 1.x	Select Stream 1 Type: String Values: "Yes" or "No"

URI Argument	Description
strm2sel This is not applicable to Version 1.x	Select Stream 2 Type: String Values: "Yes" or "No"
strm3sel This is not applicable to Version 1.x	Select Stream 3 Type: String Values: "Yes" or "No"
strm4sel This is not applicable to Version 1.x	Select Stream 4 Type: String Values: "Yes" or "No"
strm5sel This is not applicable to Version 1.x	Select Stream 5 Type: String Values: "Yes" or "No"
strm6sel This is not applicable to Version 1.x	Select Stream 6 Type: String Values: "Yes" or "No"
acqmode	PSI Acquisition Mode Type: String Values: "Auto", "Basic", "Custom"
camode	Select how programs that are marked as encrypted (and cannot be decrypted) are handled. Type String Values: "Std", "Open"
netid	Network Identifier Type: Integer

URI Argument	Description
	Values: 0 … 65535
freqmode	Frequency Tuning Mode Type: String Values: "NIT", "User Cfg" **Note** Due to the space character in "User Cfg", it requires URL encoding when sending the data via the URL inline parameter list method (such as via Query String for curl using "User%20Cfg" or parms parameter for Python, first converting the payload with the urlencode method). Alternatively, do not use URL in-line parameter lists and send the data via a request HTML body payload that has been pre-formatted as JSON object data (in conformance to the object hierarchy of the GET response output schema format). **Note** All JSON parameter values are to be sent as strings, regardless of the base type of the parameter value documented here. **Note** Since the number of URL in-line parameters (or Query Strings in curl parlance) are limited to 26 at a time, we recommend that you use the JSON request body payload alternative for setting many changes.
svclstmode	Service List Mode (Version 2.75 and later) **Note** "Degraded" is only available as a Write (POST) option. It is the same as "Relaxed". - Type : String - Values : "Rigorous", "Relaxed" Service List Mode (Version 2.50 and earlier) - Type : String - Values : "Rigorous", "Degraded"
niten	Enable NIT Reception Type: String Values: "Yes", "No"
sdten	Enable SDT Reception Type: String

URI Argument	Description
	Values: "Yes", "No"
paten	Enable PAT Reception Type: String Values: "Yes", "No"

Output Fields:

Output fields returned in GET results are the same names used for the possible Arguments for this command.

POST Examples

Example 1: Changing one RF Tuning Parameter

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes the iq mode of RF 3 to Normal.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/rf?port=3&verbose=1&iq=Normal"
```

If successful, the return body will be:

```
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 2: Changing multiple RF Tuning Parameters

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. Change the IP to the specific unit IP in use. This example changes several RF port 3 tuning parameters. It will set the IQ mode to Normal, sdten to Yes, paten to Yes and niten to No.

```
curl -X POST –i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/rf?port=3&verbose=1&iq=Normal&paten=Yes&sdten=Yes&niten=No"
```

If successful, the return body will be:

```
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples

Example 3: GET the full RF tuning values for a specific RF port

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example retrieves all of the RF tuning settings for RF port 3.

Note The user **MUST** always specify the port number in the URI.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/rf?port=3"
```

If successful, the return body will be:

```
"input": {
        "rf": {
            "port": "3",
            "act": "No",
            "dnlkfreq": "12.31",
            "symrate": "30.0",
            "fec": "Auto",
            "mod": "DVB-S2",
            "rolloff": ".35",
            "iq": "Normal",
            "pwr": "Off",
            "sel22khz": "Off",
            "lo1": "10.75",
            "lo2": "0.0",
            "xover": "0.0",
            "pol": "Horizontal",
            "orbpos": "0.0",
            "ewflag": "N/A",
            "afcrange": "3.0",
            "strm1sel": "Yes",
            "strm2sel": "No",
            "strm3sel": "No",
            "strm4sel": "No",
            "strm5sel": "No",
            "strm6sel": "No",
            "acqmode": "Basic",
            "netid": "1",
            "freqmode": "NIT",
            "svclstmode": "Rigorous",
            "niten": "No",
            "sdten": "Yes",
            "paten": "Yes"
        }
    }
```

Example 4: GET a specific RF tuning value for a specific RF port

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example retrieves one specific parameter value from the RF tuning settings for RF port 3.

Note The user **MUST** always specify the port number in the URI followed by the URI argument the user wishes to retrieve. This example retrieves the IQ Mode from RF Port 3.

Note In GET URIs, only the port argument must contain a value for the port, all other arguments **DO NOT** contain values.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/rf?port=3&iq"
```

If successful, the return body will be:

```
"input": {
        "rf": {
            "port": "3",
            "iq": "Normal"
        }
    }
```

Example 5: GET multiple RF tuning values for a specific RF port

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example retrieves multiple parameter value from the RF tuning settings for RF port 3.

Note Yhe user **MUST** always specify the port number in the URI followed by the URI arguments the user wishes to retrieve. This example retrieves the iq, sdten, niten and paten values from RF Port 3.

Note In GET URIs, only the port argument must contain a value for the port, all other arguments **DO NOT** contain values.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/rf?port=3&iq&paten&sdten&niten"
```

If successful, the return body will be:

```
"input": {
        "rf": {
            "port": "3",
            "iq": "Normal",
            "niten": "No",
            "paten": "Yes",
            "sdten": "Yes"
        }
    }
```

Miscellaneous Examples

Example 6: URI Parameters are misspelled or missing

```
curl -i -k -H "X-SESSION-ID: $token" -X POST
"https://192.168.0.1/ws/v2/service_
cfg/inputrf?port=1&act=Yes&dnlkfreq=1.421&symrate=32&mod=DVB-S2"
```

Expected Output:

```
1)      No XML output will be returned.
2)     HTTP return code is not value 200 (HTTP_OK).

HTTP/1.1 404 Not Found
Date: Fri, 23 Oct 2015 19:26:40 GMT
Server: Hiawatha v9.14
Connection: keep-alive
Transfer-Encoding: chunked
Status: 404
```

Remedy:

Client software must always first check HTTP return code.

Re-check that URI parameters portion of the command (the part before the ?) is correctly input

In the above example, /inputrf should be /input/rf.

Example 7: URI Arguments are misspelled or invalid for the command

```
curl -i -k -H "X-SESSION-ID: $token" -X POST
"https://192.168.0.1/ws/v2/service_
cfg/input/rf?port=1&act=Yes&dnlkfreq=1.421&symrate=32&mod=DVBS2"
```

Expected Output:

```
<?xml version="1.0" encoding="ISO-8859-1" ?><response>
<code>10</code><result>failed</result>
<message>Failed to set item Modulation value DVBS2</message></response>
```

Remedy:

Re-check that URI Arguments portion of the command (the part after the ?) is correctly input.

Re-check that the URI Arguments or combinations are valid for the command.

In the above example, DVBS2 should be DVB-S2.

Example 8: URI Parameters and Arguments are valid for the command – in this case POST command:

```
curl -i -k -H "X-SESSION-ID: $token" -X POST
"https://192.168.0.1/ws/v2/service_
cfg/input/rf?port=1&act=Yes&dnlkfreq=3.421&symrate=32&mod=DVB-S"
```

Expected Output:

```
<?xml version="1.0" encoding="ISO-8859-1" ?><response>
<code>10</code><result>success</result><message> </message></response>
```

ASI Input Configuration Command

Table 2.189 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/input/asi
Command Information	Allows ASI tuning input settings to be read or configured.
HTTP Methods	GET, POST
Access Type	Read/Write (with complex input data)
Access Level	User, Admin
GET Syntax	One of the following: GET "https://192.168.0.1/ws/v2/service_cfg/input/asi", or

Command Detail	Description
	GET "https://192.168.0.1/ws/v2/service_cfg/input/asi? port=[port#]&mip=<mip#>&act"
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/input/asi? port=[port#]&mip=<mip#>..."

URI Parameters (extension to the Command URL separated by /): N/A

Note Setting Parameters using command line arguments are limited to maximum of 26 arguments after the ?.

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values, except the port which must be specified. For POST the port must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

Table 2.190 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
port (Key)	Input Port to be configured Type: Integer Value: 2 **Note** This does not refer to the NTC Ethernet ports, it refers specifically to the ASI ports
stream (Key) (optional) This is not applicable to Version 1.x	Multi Input Stream ID. If not specified, default value of 1 is used. Type: Integer Value: 2
act	Activate Input Type: String Values: "Yes" or "No"
acqmode	PSI Acquisition Mode Type: String Values: "Auto", "Basic", "Custom"
camode	Select how programs that are marked as encrypted (and cannot be

URI Argument	Description
	decrypted) are handled. Type: String Values: "Std", "Open"
freqmode	Frequency Tuning Mode Type: String Values: "NIT", "User Cfg"
svclstmode	Service List Mode (Version 2.75 and later) **Note** "Degraded" is only available as a Write (POST) option. It is the same as "Relaxed". - Type : String - Values : "Rigorous", "Relaxed" Service List Mode (Version 2.50 and earlier) - Type: String - Values: "Rigorous", "Degraded"
niten	Enable NIT Reception Type: String Values: "Yes", "No"
sdten	Enable SDT Reception Type: String Values: "Yes", "No"
paten	Enable PAT Reception Type: String Values: "Yes", "No"

Output Fields:

Output fields returned in GET results are the same names used for the possible arguments for this command.

POST Examples:

Example 1: Changing one ASI Tuning Parameter

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes the acqmode of ASI port 1 to Auto.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/asi?port=1&acqmode=Auto"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 2: Changing multiple ASI Tuning Parameters

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes several ASI port 1 tuning parameters. It will set the acqmode to Auto, sdten to Yes, paten to Yes and niten to No.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/asi?port=1&sdten=Yes&acqmode=Auto&paten=Yes&niten=No"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 3: GET the full ASI tuning values for a specific ASI port

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example retrieves all of the ASI tuning settings for ASI port 1.

Note The user must always specify the port number in the URI.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/asi?port=1"
```

If successful, the return body will be:

```
"input": {
        "asi": {
            "port": "1",
            "act": "No",
            "acqmode": "Auto",
            "netid": "1",
            "freqmode": "NIT",
            "svclstmode": "Rigorous",
            "niten": "No",
            "sdten": "Yes",
            "paten": "Yes"
        }
    }
```

Example 4: GET a specific ASI tuning value for a specific ASI port

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example retrieves one specific parameter value from the ASI tuning settings for ASI port 1.

Note The user must always specify the port number in the URI followed by the URI argument the user wishes to retrieve. This example retrieves the acqmode from ASI Port 1.

Note In GET URIs only, the port argument must contain a value for the port, all other arguments do not contain values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/asi?port=1&acqmode"
```

If successful, the return body will be:

```
"input": {
        "asi": {
            "port": "1",
            "acqmode": "Auto"
        }
    }
```

Example 5: GET multiple ASI tuning value for a specific ASI port

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example retrieves multiple parameter value from the ASI tuning settings for ASI port 1.

Note The user must always specify the port number in the URI followed by the URI arguments the user wishes to retrieve. This example retrieves the acqmode, sdten, niten and paten values from ASI Port 1.

Note In GET URIs, only the port argument must contain a value for the port, all other arguments **DO NOT** contain values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/asi?port=1&sdten&acqmode&paten&niten"
```

If successful, the return body will be:

```
"input": {
        "asi": {
            "port": "1",
            "acqmode": "Auto",
            "niten": "No",
            "paten": "Yes",
            "sdten": "Yes"
        }
    }
```

MOIP Input Configuration Command

Table 2.191 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/input/moip
Command Information	Allows MOIP tuning input settings to be read or configured.
HTTP Methods	GET, POST
Access Type	Read/Write (with complex input data)
Access Level	User, Admin
GET Syntax	One of the following: GET "https://192.168.0.1/ws/v2/service_cfg/input/moip", or GET "https://192.168.0.1/ws/v2/service_cfg/input/moip? stream=[stream#]&act"

Command Detail	Description
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/input/moip? stream=[stream#]&act=Yes..."

URI Parameters (extension to the Command URL separated by /): N/A

> **Note** Setting Parameters using command line arguments are limited to maximum of 26 arguments after the ?.
>
> All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values, except the stream which must be specified. For POST the stream must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

Table 2.192 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
card (Key 0- Read Only)	Card Identifier. Will show in GET Response but not needed when setting parameters. Type: Integer Values: 1
port (Key 1 - Read Only)	Input Port Number on this Card. Will show in GET Response but not needed when setting parameters Type: Integer Values: 1
stream (Key 2 - Read Only)	Multi Input Stream ID. For GET operations, specify this key=value (stream=1 to 32) to only return the row of interest; when not specified the response will return all rows. For POST operations, the stream value is mandatory since it identifies the row to be acted upon. Type: Integer Values: 1..32
act	Activate Input. Type: String Values: "Yes" or "No"

URI Argument	Description
ipmode	IP address selection method. Type: String Values: "SW Map" or "UserCfg"
fecmode	FEC mechanism. Type: String Values: "None", "1D", or "2D"
usemcast	Use multicast address. Type: String Values: "Yes" or "No"
mcastaddr	Multicast address. Type: String Values: IP (v4) address in the dot format, etc 192.168.0.1
tsdestport	Transport destination port value. Type Integer Values: 1024..65534
feccolport	FEC column port value. Type: Integer Values: 1024..65534
fecrowport	FEC row port value. Type: Integer Values: 1024..65534
usrsrcaddr	User configured source IP address. Type String Values: IP (v4) address in the dot format, for example: 192.131.244.10
srcfltmode	Source IP address filtering mode. Type: String Values: "None", "White List", "Black List"
dejalg	Dejitter Algorithm. Type: String

URI Argument	Description
	Values: "CBR" or "VBR"
avlostrig	Redundancy Trigger on Audio/Video Loss. Type: String Values: "Yes" , "No"
pcrlostrig	Redundancy Trigger on PCR Loss. Type: String Values: "Yes" , "No"
pmtlostrig	Redundancy Trigger on PMT Loss. Type: String Values: "Yes" , "No"
lnklostrig	Redundancy Trigger on Link Loss. Type: String Values: "Yes" **Note** This value is read-only.
tslostrig	Redundancy Trigger on Transport Loss. Type: String Values: "Yes" **Note** This value is read-only.
acqmode	PSI acquisition mode. Type: String Values: "Auto", "Basic", "Custom"
camode	Select how programs that are marked as encrypted (and cannot be decrypted) are handled. Type: String Values: "Std", "Open"
freqmode	Frequency tuning mode. Type: String Values: "NIT", "User Cfg"

URI Argument	Description
	Note Due to the space character in "User Cfg", it requires URL encoding when sending the data via the URL inline parameter list method (such as via Query String for curl using "User%20Cfg" or parms parameter for Python, first converting the payload with the urlencode method). Alternatively, do not use URL in-line parameter lists and send the data via a request HTML body payload that has been pre-formatted as JSON object data (in conformance to the object hierarchy of the GET response output schema format). **Note** All JSON parameter values are to be sent as strings, regardless of the base type of the parameter value documented here. **Note** Since the number of URL in-line parameters (or Query Strings in curl parlance) are limited to 26 at a time, we recommend that you use the JSON request body payload alternative for setting many changes.
svclstmode	Service List Mode **Note** "Degraded" is only available as a Write (POST) option. It is the same as "Relaxed". - Type: String - Values: "Rigorous", "Relaxed"
niten	Enable NIT reception. Type: String Values: "Yes", "No"
sdten	Enable SDT reception. Type: String Values: "Yes", "No"
paten	Enable PAT reception. Type: String Values: "Yes", "No"
dejlat	Dejitter buffer latency. Type: Integer Values: 40 … 150

URI Argument	Description
ipiport	IPI port. Type: String Values: "Data1", "Data2", "Data3", "Data4"
redenb	Input redundancy mode Type: String Values: "Yes", "No"
reddir	IPI redundancy switchover direction. Type: String Values: "Legacy Non-Revertive", "Non-Revertive", "Revertive", "2022-7 Merge" For IPI Hitless Merge (SMPTE 2022-7), use the "reddir=2022-7%20Merge" option for enabling this case (note the requirement for URL encoding of special characters).
reddel	Direct redundancy switchover delay, in milliseconds. Type: Integer Values: 0 … 10000
redrev	Revertive redundancy switchover delay, in seconds. Type: Integer Values: 0 … 30
cmdrow	Add or delete an IP input instance. Type: String Values: "Add” (R/W), "Delete" (R/W), "Active" (RO), "Inactive" (RO)

Table 2.193 URI Query/Set Arguments for Independent Backup Stream Setup

URI Argument	Description
manbkpcfg	Enable manual backup configuration. Type: String Values: "Yes" or "No" **Note** This item has to be set explicitly in a separate call, prior to any Backup configuration changes for a specific stream.

URI Argument	Description
ipiportbkp	IPI port. Type: String Values: "Data1", "Data2", "Data3", "Data4" **Note** This value is read-only.
usemultbkp	Use Multicast address Type: String Values: "Yes" or "No"
multaddrbkp	Multicast address. Type: String Values: IP (v4) address in the dot format, for example, 192.168.0.1
tsportbkp	Transport destination port value. Type: Integer Values : 1024..65534
fecmodebkp	FEC mechanism. Type: String Values: "None", "1D", or "2D"
fec1portbkp	FEC column port value. Type: Integer Values: 1024..65534
fec2portbkp	FEC row port value. Type: Integer Values: 1024..65534

Output Fields:

Output fields returned in GET results are the same names used for the possible arguments for this command.

User Guidelines:

The following improvements and considerations were made for MOIP Input Configuration, Validation and Status behavior beginning with Software Release 5.50:

Stream configuration in "ipmode=UserCfg" source selection mode

1. Any change to the following stream configuration parameters will trigger cleanup of User Selected Sources:
   - a. Interface
   - b. Multicast
   - c. Address
   - d. Redundancy settings
2. MOIP Input Status ‘portinuse’ ('Current Intf' on WEBUI) status field will show ‘Need Source’, to prompt user to explicitly select source[s]
3. When Redundancy is disabled, one source must be selected (for the data port configured under ‘Interface’ field)
4. When Redundancy is enabled, in any mode, two sources must be selected (for each data interface in a pair)
5. Switching MOIP input stream Off and On will preserve previously selected sources.

Stream Status in "ipmode=UserCfg" source selection mode

1. 1. When stream is disabled, all MOIP Input Status fields will be reset
2. 2. When stream is enabled, the following improvements were made for MOIP Input Status ‘portinuse’ ('Current Intf' on WEBUI) field:
   - Not enough sources selected: ‘Need Source’ will be displayed
   - In ‘2022-7 Merge’ redundancy mode this field will display:
     - ‘DATA1/DATA2’, when ‘Interface’ is configured as Data1 or Data2
     - ‘DATA3/DATA4’, when ‘Interface’ is configured as Data3 or Data4

Specific validation requirements for "reddir=2022-7 Merge" redundancy mode

'2022-7 Merge' redundancy mode requires configuration of any pair of IP Input Data ports (DATA1/DATA2 may be paired, DATA3/DATA4 may alternatively be used as an input pair when available (D9800 MS Unit only)). Both physical interface connections in a pair are required to be sourced from a upstream unit configured in Mirrored mode.

On the D9800 (UUT) unit for which this feature applies:

1. ‘Source Selection’ (ipmode) value must be ‘UserCfg’
2. Only RTP sources must be selected
3. Each interface in a pair must be assigned a unique RTP source

   In case when less than 2 sources are selected, MOIP Input Status 'portinuse' (‘Current Intf’ on WEBUI) status field will display ‘Need Source’.

‘2022-7 Merge’ Warning Fault

A new Warning fault was introduced in Release 5.50 to indicate when the FEC Engine is not well (resets) while processing two selected RTP streams. The most probably reasons for that are:

1. Selected RTP streams are not related (not identical), e.g. are from different upstream units
2. Network delay for one of the identical RTP streams is higher than the max supported delay (calculated as ‘Reordering Window Delay’, in ms).

POST Examples:

Example 1: Changing one MOIP Tuning Parameter

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes the freqmode of MOIP stream 1 to NIT.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/moip?stream=1&freqmode=NIT"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 2: Changing multiple MOIP Tuning Parameters

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes several MOIP stream 1 tuning parameters. It will set the freqmode to NIT, sdten to Yes, paten to Yes and niten to No.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/moip?stream=1&paten=Yes&sdten=Yes&freqmode=NIT&niten=No"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 3: Changing MOIP Backup Configuration Mode

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes the manbkpcfg of MOIP stream 1 to Yes (Manual). This call has to be made prior to any subsequent changes in Backup Configuration:

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/moip?stream=1& manbkpcfg=Yes"
```

If successful, the return body will be:

```
   "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 4: GET the full MOIP tuning values for a specific MOIP stream

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example retrieves all of the MOIP tuning settings for MOIP stream 1.

Note The user must always specify the stream number in the URI.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/moip?stream=1"
```

If successful, the return body will be:

```
{
    "input": {
        "moip": {
            "stream": "1",
            "act": "Yes",
            "ipmode": "SW Map",
            "fecmode": "2D",
            "usemcast": "Yes",
            "mcastaddr": "225.1.1.1",
            "tsdestport": "49152",
            "feccolport": "49154",
            "fecrowport": "49156",
            "dat1srcaddr": "0.0.0.0",
            "dat2srcaddr": "0.0.0.0",
            "dat3srcaddr": "0.0.0.0",
            "dat4srcaddr": "0.0.0.0",
            "srcfltmode": "None",
            "dejalg": "VBR",
            "dejlat": "110",
            "acqmode": "Basic",
            "netid": "1",
            "freqmode": "User Cfg",
            "svclstmode": "Rigorous",
            "niten": "No",
            "sdten": "Yes",
            "paten": "Yes",
            "ipiport": "Data1",
            "redenb": "No",
            "reddir": "Revertive",
            "reddel": "0",
            "redrev": "0",
            "avlostrig": "Yes",
            "pcrlostrig": "Yes",
            "pmtlostrig": "Yes",
            "lnklostrig": "Yes",
            "tslostrig": "Yes",
            "cmdrow": "Active","manbkpcfg": "Yes"
            "ipportbkp": "Data2",
            "usemultbkp": "Yes",
            "multaddrbkp": "225.1.1.111",
            "fecmodebkp": "2D",
            "tsportbkp": "49172",
            "fec1portbkp": "49174",
            "fec2portbkp": "49176"
        }
    }
}
```

Example 5: GET a specific MOIP tuning value for a specific MOIP stream

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example retrieves one specific parameter value from the MOIP tuning settings for MOIP stream 1.

Note The user must always specify the stream number in the URI followed by the URI argument the user wishes to retrieve.

This example retrieves the freqmode from MOIP stream 1. In GET URIs only the stream argument must contain a value for the stream, all other arguments do not contain values.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/moip?stream=1&freqmode"
```

If successful, the return body will be:

```
"input": {
        "moip": {
            "stream": "1",
            "freqmode": "NIT"
        }
    }
```

Example 6: GET multiple MOIP tuning value for a specific MOIP stream

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example retrieves multiple parameter value from the MOIP tuning settings for MOIP stream 1.

Note The user must always specify the stream number in the URI followed by the URI arguments the user wishes to retrieve.

This example retrieves the freqmode, sdten, niten and paten values from MOIP stream 1. In GET URIs only the stream argument must contain a value for the stream, all other arguments do not contain values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/moip?stream=1&paten&sdten&freqmode&niten"
```

If successful, the return body will be:

```
"input": {
        "moip": {
            "stream": "1",
            "freqmode": "NIT",
            "niten": "Yes",
            "paten": "Yes",
            "sdten": "Yes"
        }
    }
```

MOIP Input Source Filters Configuration Command

Table 2.194 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/input/moip/srcfilter
Command Information	Allows MOIP source filter settings to be read or configured.
HTTP Methods	GET, POST
Access Type	Read/Write (with complex input data)
Access Level	User, Admin
GET Syntax	One of the following: GET "https://192.168.0.1/ws/v2/service_cfg/input/moip", or GET "https://192.168.0.1/ws/v2/service_cfg/input/moip/srcfilter? idx=5"
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/input/moip/srcfilter? idx=3&ip=192.131.244.113&oper=Add"

URI Parameters (extension to the Command URL separated by /): N/A

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values, except the idx which must be specified. For POST the idx must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

Table 2.195 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
idx (Key)	Filter slot to be configured Type: Integer Values: 1 to 8
ip	IP Address Type : String Values : IP address in the dot format, for example, 192.168.0.1

URI Argument	Description
oper	Filter index operation Type: String Value: "Add", "Delete" Use "Add" to modify an existing entry

POST Examples:

Example 1: Changing both MOIP Source Filter Parameter

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes both the ip address of filter slot 1 and its filter index operation (oper).

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/moip/srcfilter?idx=1&ip=10.10.10.10&oper=Add"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Get full table of MOIP Source Filters

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example returns the full MOIP source filters for every index.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/moip/srcfilter"
```

If successful, the return body will be:

```
HTTP/1.1 200 OK
Date: Fri, 29 Jul 2016 02:43:54 GMT
Server: Hiawatha v9.14
Connection: keep-alive
```

```
Transfer-Encoding: chunked
Status: 200
Content-type: application/json

{
    "srcfilter": {
        "record": [
            {
                "idx": "1",
                "ip": "0.0.0.0",
                "oper": "Inactive"
            },
            {
                "idx": "2",
                "ip": "0.0.0.0",
                "oper": "Inactive"
            },
            {
                "idx": "3",
                "ip": "0.0.0.0",
                "oper": "Inactive"
            },
            {
                "idx": "4",
                "ip": "0.0.0.0",
                "oper": "Inactive"
            },
            {
                "idx": "5",
                "ip": "0.0.0.0",
                "oper": "Inactive"
            },
            {
                "idx": "6",
                "ip": "0.0.0.0",
                "oper": "Inactive"
            },
            {
                "idx": "7",
                "ip": "0.0.0.0",
                "oper": "Inactive"
            },
            {
                "idx": "8",
                "ip": "0.0.0.0",
                "oper": "Inactive"
            }
        ]
    }
}
```

To retrieve the table for a specific idx the CURL would look like:

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/moip/srcfilter?idx=1"
```

MOIP Input Source Select Configuration Command

Table 2.196 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/input/moip/srcselect
Command Information	Allows MOIP source select settings to be read or configured.
HTTP Methods	GET, POST
Access Type	Read/Write (with complex input data)
Access Level	User, Admin
GET Syntax	GET "https://192.168.0.1/ws/v2/service_cfg/input/moip/srcselect", or GET "https://192.168.0.1/ws/v2/service_cfg/input/moip/srcselect? stream=[stream#]&idx=[index#]" **Note** Since not specifying the key items (stream and idx) results in a significant amount of data transfer and potential for the client to timeout or have memory/buffering issues, we highly recommend that you always specify the stream (and preferably also idx) in the GET call. The caller is expected to know the stream ID of interest.
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/input/moip/srcselect? stream=[stream#]&dat1srcaddr=10.10.10.10..."

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.197 URI Query Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
stream (Key)	Stream ID

URI Argument	Description
	Type: Integer Values: 1 … 32 When not specified, data for all potential streams is returned.
idx (Key)	Index Type: Integer Values: 1 … 64 When not specified, data for all potential idx values for the stream id is returned.
ip	IP Address Type: String Values: IP address in the dot format, e.g. 192.168.0.1
data1sel	DATA1 selection state. DATA1 is either selected or not selected. Type: String Value: "Yes", "No"
data1avail	IP is present in the DATA1 stream. Type: String Value : "No", "UDP", "RTP" "UDP" and "RTP" indicates the protocol type detected in the stream.
data1Enabled	Allow DATA1 to be selected or not. Type: String Value: "Yes", "No"
data2sel	DATA2 selection state. DATA2 is either selected or not selected. Type: String Value: "Yes", "No"
data2avail	IP is present in the DATA2 stream. Type: String Value : "No", "UDP", "RTP" "UDP" and "RTP" indicates the protocol type detected in the stream.
data2Enabled	Allow DATA2 to be selected or not. Type: String Value: "Yes", "No"

URI Argument	Description
data3sel	DATA3 selection state. DATA3 is either selected or not selected. Type: String Value: "Yes", "No"
data3avail	IP is present in the DATA3 stream. Type: String Value : "No", "UDP", "RTP" "UDP" and "RTP" indicates the protocol type detected in the stream.
data3Enabled	Allow DATA3 to be selected or not. Type: String Value: "Yes", "No"
data4sel	DATA4 selection state. DATA4 is either selected or not selected. Type: String Value: "Yes", "No"
data4avail	IP is present in the DATA4 stream. Type: String Value : "No", "UDP", "RTP" "UDP" and "RTP" indicates the protocol type detected in the stream.
data4Enabled	Allow DATA4 to be selected or not. Type: String Value: "Yes", "No"

Table 2.198 URI Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
stream (Key)	Stream ID Type: Integer Values: 1 … 32
dat1srcaddr	Set the IP address to lock from DATA1. Type: String Value: IP address in the dot format, for example, 192.168.0.1

URI Argument	Description
dat2srcaddr	Set the IP address to lock from DATA2. Type: String Value: IP address in the dot format, for example, 192.168.0.1
dat3srcaddr	Set the IP address to lock from DATA3. Type: String Value: IP address in the dot format, for example, 192.168.0.1
dat4srcaddr	Set the IP address to lock from DATA4. Type: String Value: IP address in the dot format, for example, 192.168.0.1

POST Examples:

Example 1: Changing MOIP Data Source IP Addresses

This example changes the IP address to lock from DATA1 to 10.10.10.10 and the IP address to lock from DATA2 to 10.10.10.11.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/moip/srcselect?stream=1&dat1srcaddr=10.10.
10.10&dat2srcaddr=10.10.10.11"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Get the MOIP Source Select Configuration

This example returns the MOIP source select configuration for stream 1, index 1.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/moip/srcselect?stream=1&idx=1"
```

If successful, the return body will be:

```
{
    "srcselect": {
        "record": {
            "stream": "1",
            "idx": "1",
            "ip": "0.0.0.0",
            "data1sel": "No",
            "data1avail": "No",
            "data1Enabled": "No",
            "data2sel": "No",
            "data2avail": "No",
            "data2Enabled": "No",
            "data3sel": "No",
            "data3avail": "No",
            "data3Enabled": "No",
            "data4sel": "No",
            "data4avail": "No",
            "data4Enabled": "No"
        }
    }
}
```

ABR Input Configuration Command

Table 2.199 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/input/abr
Command Information	Allows ABR input settings to be read or configured.
HTTP Method(s)	GET, POST
Access Type	Read/Write (with complex input data)
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/service_cfg/input/abr", or GET "https://192.168.0.1/ws/v2/service_cfg/input/abr? stream=[stream#]"
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/input/abr? stream=[stream#]&cmdrow=Add..."

URI Parameters (extension to the Command URL separated by /): N/A

Note Setting Parameters using command line arguments are limited to maximum of 26 arguments after the ?.

Note All of the URI Arguments in the table apply to both GET and POST. For GET, the URI arguments do not need any values, except the stream which must be specified. For POST the stream must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

Table 2.200 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &):

URI Argument	Description
card (Key 0 - Read Only)	Card Identifier Will show in GET Response but not needed when setting parameters Type: Integer Values: 1
port (Key 1 - Read Only)	Input Port Number on this Card Will show in GET Response but not needed when setting parameters Type: Integer Values: 1
stream (Key 2 - Read Only)	Multi Input Stream ID For GET operations, specify this key=value (stream=1 to 2) to only return the row of interest; when not specified the response will return all rows. For POST operations, the stream value is mandatory since it identifies the row to be acted upon. Type: Integer Values: 1..2
act	Activate or Active Input **Note** This item is independent of the cmdrow item. We recommend that you set the Active Input to No before deleting the stream instance via cmdrow=Delete. Type: String Values: "Yes" or "No"
netid	Network Identifier Type: Integer Values: 0 … 65535
acqmode	Select which tables are required for service list creation.

URI Argument	Description
	Type: String Values: "Basic", "Auto", "Custom", "FixPID"
camode	Select how programs that are marked as encrypted (and can not be decrypted) are handled. Type: String Values: "Std", "Open"
inputsel	Input Selection Type: String Values: "UserCfg", "SW Map"
freqmode	Enable SDT Reception Type: String Values: "NIT", "User Cfg" **Note** Due to the space character in "User Cfg", it requires URL encoding when sending the data via the URL inline parameter list method (such as via Query String for curl using "User%20Cfg" or parms parameter for Python, first converting the payload with the urlencode method). Alternatively, do not use URL in-line parameter lists and send the data via a request HTML body payload that has been pre-formatted as JSON object data (in conformance to the object hierarchy of the GET response output schema format). **Note** All JSON parameter values are to be sent as strings, regardless of the base type of the parameter value documented here. **Note** Since the number of URL in-line parameters (or Query Strings in curl parlance) are limited to 26 at a time, we recommend that you use the JSON request body payload alternative for setting many changes.
svclstmode	Service List Mode (Version 2.75 or later) **Note** "Degraded" is only available as a Write (POST) option for backwards compatibility. It is the same as "Relaxed". - Type : String - Values : "Rigorous", "Relaxed"

URI Argument	Description
	Service List Mode (Version 2.50 and earlier) ■ Type : String ■ Values : "Rigorous", "Degraded"
baten	Enable BAT Reception Type: String Values: "Yes", "No"
niten	Enable NIT Reception Type: String Values: "Yes", "No”
sdten	Enable SDT Reception Type: String Values: "Yes", "No"
paten	Enable PAT Reception Type: String Values: "Yes", "No"
name	ABR settings name Type: String Value: Up to 12 characters
url	ABR settings URL Type: String Value: Up to 200 characters
latency	ABR settings target latency Type: Integer Values: 0.. 4294967295
phyport	ABR settings physical port Type: String Values: "Mngmt", "Data1", "Data2", "Data3", "Data4"
cmdrow	Add or Delete an instance of ABR Input For a GET operation, the value will read back the Read-Only (RO) value where the "Active" value means that a previous Add operation was successful for that stream ID.

URI Argument	Description
	For a POST operation, set the stream "cmdrow "value to "Add" to make that stream id instance "Active" or "Delete" to make an "Active" stream ID "Inactive". Type: String Values: "Add" (R/W), "Delete" (R/W), "Active" (RO), "Inactive" (RO)
minlatency	ABR settings minimum latency Type: Integer Values: 0.. 4294967295

Output Fields:

Output fields returned in GET results are the same names used for the possible Arguments for this command.

POST Examples:

Example 1: Changing one ABR Parameter

The following example assumes that the user has successfully logged onto the unit, received the session id, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example changes the name of an existing ABR entry to "Prog5".

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/abr?stream=2&name=Prog5
```

If successful, the return body will be:

```
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 2: Changing multiple ABR Parameters

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example changes several ABR parameters for an existing entry. It will set the netid to 12345, sdten to Yes, paten to Yes and niten to Yes.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/abr?stream=2&netid=12345&sdten=Yes&paten=Yes&niten=Yes"
```

If successful, the return body will be:

```
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 3: GET the full ABR tuning values for a specific ABR entry

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves all of the ABR settings for stream 2.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/abr?stream=2"
```

If successful, the return body will be:

```
"input": {
        "abr": {
            "stream": "2",
            "act": "No",
            "netid": "1",
            "camode": "Std",
            "inputsel": "UserCfg",
            "freqmode": "User Cfg",
            "svclstmode": "Rigorous",
            "baten": "No",
            "niten": "No",
            "sdten": "Yes",
            "paten": "Yes",
            "name": "ABR2",
            "url": "",
            "latency": "0",
            "phyport": "Mngmt",
            "cmdrow": "Inactive",
            "minlatency": "0"
    }
}
```

Example 4: GET a specific ABR value for a specific ABR entry

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves one specific parameter value, camode, from the ABR settings for ABR stream 2.

Note In GET URIs only, the entry selection argument (stream) must contain a value, all other arguments **DO NOT** contain values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/abr?stream=2&camode&js=1"
```

If successful, the return body will be:

```
"input": {
        "abr": {
            "stream": "2",
            "camode": "Std"        }
    }
```

Example 5: GET multiple ABR tuning value for a specific ABR stream

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example retrieves multiple parameter values from the ABR settings. This example retrieves the camode, sdten, niten and paten values from ABR stream 2.

Note In GET URI, only the entry selection argument (stream) must contain a value, all other arguments **DO NOT** contain values.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/abr?stream=2&camode&sdten&paten&niten&js=1"
```

If successful, the return body will be:

```
"input": {
        "abr": {
            "stream": "2",
            "camode": "Std",
            "niten": "No",
            "paten": "Yes",
            "sdten": "Yes"        }
   }
```

Zixi Input Configuration Command

Table 2.201 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/input/zixi
Command Information	Allows ZIXI input settings to be read or configured.
HTTP Methods	GET, POST
Access Type	Read/Write (with complex input data)
Access Level	User, Admin
GET Syntax	GET "https://192.168.0.1/ws/v2/service_cfg/input/zixi", or GET "https://192.168.0.1/ws/v2/service_cfg/input/zixi? stream=[stream#]"
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/input/zixi? stream=[stream#]&cmdrow=Add..."

URI Parameters (extension to the Command URL separated by /): N/A

Note Setting Parameters using command line arguments are limited to maximum of 26 arguments after the ?.

Note All of the URI Arguments in the table below apply to both GET and POST. For GET, the URI arguments do not need any values, except the stream which must be specified. For POST, the stream must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below, following the table.

Table 2.202 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
card (Key 0- Read Only)	Card Identifier Will show in GET Response but not needed when setting parameters Type: Integer Values: 1
port (Key 1 - Read Only)	Input Port Number on this Card Will show in GET Response but not needed when setting parameters Type: Integer Values: 1
stream (Key 2 - Read	Multi Input Stream ID

URI Argument	Description
Only)	For GET operations, specify this key=value (stream=1 to 4) to only return the row of interest; when not specified the response will return all rows. For POST operations, the stream value is mandatory since it identifies the row to be acted upon. Type: Integer Values: 1..4
act	Activate or Active Input **Note** This item is independent of the cmdrow item. We recommend that you set the Active Input to No before deleting the stream instance via cmdrow=Delete. Type: String Values: "Yes" or "No"
netid	Network Identifier Type: Integer Values: 0 … 65535
acqmode	Select which tables are required for service list creation. Type: String Values: "Basic", "Auto", "Custom", "FixPID"
camode	Select how programs that are marked as encrypted (and cannot be decrypted) are handled. Type: String Values: "Std", "Open"
inputsel	Input Selection Type: String Values: "UserCfg", "SW Map"
freqmode	Enable SDT Reception Type: String Values: "NIT", "User Cfg" **Note** Due to the space character in "User Cfg", it requires URL encoding when sending the data via the URL inline parameter list method (such as via

URI Argument	Description
	Query String for curl using "User%20Cfg" or parms parameter for Python, first converting the payload with the urlencode method). Alternatively, do not use URL in-line parameter lists and send the data via a request HTML body payload that has been pre-formatted as JSON object data (in conformance to the object hierarchy of the GET response output schema format). **Note** All JSON parameter values are to be sent as strings, regardless of the base type of the parameter value documented here. **Note** Since the number of URL in-line parameters (or Query Strings in curl parlance) are limited to 26 at a time, we recommend that you use the JSON request body payload alternative for setting many changes.
svclstmode	Service List Mode (Version 2.75 and later) **Note** "Degraded" is only available as a Write (POST) option for backwards compatibility (means the same as "Relaxed") ■ Type : String ■ Values : "Rigorous", "Relaxed" Service List Mode (Version 2.50 and earlier) ■ Type : String ■ Values : "Rigorous", "Degraded"
baten	Enable BAT Reception Type: String Values: "Yes", "No"
niten	Enable NIT Reception Type: String Values: "Yes", "No"
sdten	Enable SDT Reception Type: String Values: "Yes", "No"
paten	Enable PAT Reception

URI Argument	Description
	Type: String Values: "Yes", "No"
Name (RO)	ZIXI settings name Type: String Value: Up to 12 characters
type	ZIXI Type Type: String Value: "None", "Pull"
source	ZIXI input source such as a URL Type: String Value: Up to 200 characters
sid	Source Stream ID Type: String Value: Up to 200 characters
protport	Source Port Type: Integer Values: 0 … 65535
interface	ZIXI settings physical interface Type: String Values: "Mngmt", "Data1", "Data2", "Data3", "Data4"
cmdrow	Add or Delete an instance of ZIXI Input For a GET operation, the value will read back the Read-Only (RO) value where the "Active" value means that a previous Add operation was successful for that stream ID. For a POST operation, set the stream "cmdrow "value to "Add" to make that stream id instance "Active" or "Delete" to make an "Active" stream id "Inactive". Type: String Values: "Add" (R/W), "Delete" (R/W), "Active" (RO), "Inactive" (RO)
latency	ZIXI input stream latency Type: Integer Values: 0.. 4294967295

URI Argument	Description
pw	ZIXI input stream password Type: String Value: Up to 127 characters
decrkey	Descrambling key for ZIXI Input Stream Type: String Value: Up to 64 characters
redmode	Redundancy Mode Type: String Values: "Yes", "No"
dirdelay	Redundancy switchover delay Type: Integer Values: 0 … 10000
avtrig	Redundancy audio/video loss trigger Type: String Values: "Yes", "No"
pmttrig	Redundancy PMT loss trigger Type: String Values: "Yes", "No"
pcrtrig	Redundancy PCR loss trigger Type: String Values: "Yes", "No"
redintf	Redundancy Backup ZIXI physical interface Type: String Values: "Mngmt", "Data1", "Data2", "Data3", "Data4"
redsource	Redundancy Backup ZIXI input source such as a URL Type: String Value: Up to 200 characters
redsid	Redundancy Backup Source Stream ID Type: String

URI Argument	Description
	Value: Up to 200 characters
redprotport	Redundancy Backup Source Port Type: Integer Values: 0 … 65535
reddecrkey	Descrambling key for Redundancy Backup ZIXI Input Stream Type: String Value: Up to 64 characters
redpw	Redundancy Backup ZIXI input stream password Type : String Value : Up to 127 characters
fecmode	Forward Error Correction Mode Type: String Values: "Off", "On", "Adaptive"
fecovhd	Forward Error Correction Overhead Type: Integer Values: 0 … 10000
fecblkms	Forward Error Correction Block Type: Integer Values: 0 … 10000
fecconawar	Forward Error Correction Content Aware Type: String Values: "Yes", "No"
fecmaxjitr	Forward Error Correction Maximum Jitter Type: Integer Values: 0 … 10000
fecstufnul	Forward Error Correction Stuff Null Type: String Values: "Yes", "No"
latmode	Latency Mode

URI Argument	Description
	Type: String Values: "Static", "Increasing", "Dynamic"
adaptmode	Adaptive Mode Type: String Values: "None", "Encoder", "FEC"

Output Fields:

Output fields returned in GET results are the same names used for the possible Arguments for this command.

POST Examples:

The following examples assume that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use

Example 1: Changing one ZIXI Parameter

The following example changes the type of an existing ZIXI entry to Pull mode".

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/zixi?stream=2&type=Pull"
```

If successful, the return body will be:

```
 "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 2: Changing multiple ZIXI Parameters

This example changes several ZIXI parameters for an existing entry. It will set the netid to 12345, sdten to Yes, paten to Yes and niten to Yes.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/zixi?stream=2&netid=12345&sdten=Yes&paten
=Yes&niten=Yes"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 3: GET the full ZIXI tuning values for a specific ZIXI entry

This example retrieves all of the ZIXI settings for stream 2.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/zixi?stream=2"
```

If successful, the return body will be:

```
"input": {
        "zixi": {
            "stream": "2",
            "act": "Yes",
            "netid": "1",
            "acqmode": "Auto",
            "camode": "Std",
            "inputsel": "UserCfg",
            "freqmode": "User Cfg",
            "svclstmode": "Rigorous",
            "baten": "No",
            "niten": "No",
            "sdten": "Yes",
            "paten": "Yes",
            "name": "ZIXI1",
            "type": "Pull",
            "source": "zixi://tor02lab-s-002.cisco.com",
            "sid": "cisco-CAN",
            "protport": "2077",
            "interface": "Mngmt",
            "rowstatus": "Active",
            "latency": "5000",
            "decrkey": "",
            "redmode": "No",
            "redsource": "",
            "redsid": "",
            "reddecrkey": "",
            "fecmode": "On",
            "fecovhd": "30",
            "fecblkms": "50",
            "fecconawar": "No",
            "fecmaxjitr": "0",
            "fecstufnul": "No",
            "latmode": "Static",
            "adaptmode": "None"
        }
    }
```

Example 4: GET a specific ZIXI value for a specific ZIXI entry

This example retrieves one specific parameter value, camode, from the ZIXI settings for ZIXI stream 1.

Note In GET URIs, only the entry selection argument (stream) must contain a value, all other arguments **DO NOT** contain values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/zixi?stream=1&camode&js=1"
```

If successful, the return body will be:

```
"input": {
        "zixi": {
            "stream": "1",
            "camode": "Std"        }
    }
```

Example 5: GET multiple ZIXI tuning value for a specific ZIXI stream

This example retrieves multiple parameter values from the ZIXI settings. This example retrieves the camode, sdten, niten and paten values from ZIXI stream 1.

Note In GET URIs, only the entry selection argument (stream) must contain a value, all other arguments **DO NOT** contain values.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/zixi?stream=2&camode&sdten&paten&niten&js
=1"
```

If successful, the return body will be:

```
"input": {
        "zixi": {
            "stream": "1",
            "camode": "Std",
            "niten": "No",
            "paten": "Yes",
            "sdten": "Yes"        }
    }
```

SRT Input Configuration Command

Table 2.203 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/input/srt
Command Information	Allows SRT input settings to be read or configured.
HTTP Methods	GET, POST
Access Type	Read/Write (with complex input data)
Access Level	User, Admin
GET Syntax	GET "https://192.168.0.1/ws/v2/service_cfg/input/srt" Or GET "https://192.168.0.1/ws/v2/service_cfg/input/srt? stream=[stream#]"
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/input/srt?stream=[stream#]&cmdrow=Add..."

URI Parameters (extension to the Command URL separated by /): N/A

Note Setting Parameters using command line arguments are limited to maximum of 26 arguments after the ?.

Note All of the URI Arguments in the table below apply to both GET and POST. For GET, the URI arguments do not need any values, except the stream which must be specified. For POST, the stream must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below, following the table.

Table 2.204 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
card (Key 0-Read Only)	Card Identifier Will show in GET Response but not needed when setting parameters. Type: Integer

URI Argument	Description
	Values: 1
port (Key 1 - Read Only)	Input Port Number on this Card Will show in GET Response but not needed when setting parameters Type: Integer Values: 1
stream (Key 2 - Read Only)	Multi Input Stream ID For GET operations, specify this key=value (stream=1 to 2) to only return the row of interest; when not specified the response will return all rows. For POST operations, the stream value is mandatory since it identifies the row to be acted upon. Type: Integer Values: 1..2
act	Activate or Active Input **Note** This item is independent of the cmdrow item. We recommend that you set the Active Input to No before deleting the stream instance via cmdrow=Delete. Type: String Values: "Yes" or "No"
netid	Network Identifier Type : Integer Values : 0 … 65535
acqmode	Select which tables are required for service list creation. Type : String Values : "Basic", "Auto", "Custom","FixPID"
camode	Select how programs that are marked as encrypted --and cannot be decrypted-- are handled. Type : String Values : "Std", "Open"
inputsel	Input Selection Type : String

URI Argument	Description
	Values : "UserCfg", "SW Map"
freqmode	Enable SDT Reception Type : String Values : "NIT", "User Cfg" **Note** Due to the space character in "User Cfg", it requires URL encoding when sending the data via the URL inline parameter list method (such as via Query String for curl using "User%20Cfg" or parms parameter for Python, first converting the payload with the urlencode method). Alternatively, do not use URL in-line parameter lists and send the data via a request HTML body payload that has been pre-formatted as JSON object data (in conformance to the object hierarchy of the GET response output schema format). **Note** All JSON parameter values are to be sent as strings, regardless of the base type of the parameter value documented here. **Note** Since the number of URL in-line parameters (or Query Strings in curl parlance) are limited to 26 at a time, we recommend that you use the JSON request body payload alternative for setting many changes.
svclstmode	Service List Mode Type : String Values : "Rigorous", "Relaxed"
baten	Enable BAT Reception **Note** BAT is not supported for SRT in the current release. This parameter cannot be changed to "Yes". Type : String Values : "No"
niten	Enable NIT Reception Type : String Values : "Yes", "No"
sdten	Enable SDT Reception Type : String

URI Argument	Description
	Values : "Yes", "No"
paten	Enable PAT Reception Type : String Values : "Yes", "No"
Name (RO)	SRT settings name Type : String Value : Up to 12 characters (eg. SRT1, SRT2)
type	SRT Connection Type Type : String Value : "Listener", "Caller", "Rendezvous"
remoteip	SRT input source ip address Type : String Value : Up to 200 characters
remoteport	SRT input remote UDP ip port Type : String Value : Up to 200 characters
listenport	SRT input listen UDP ip port Type : Integer Values : 0 … 65535
curintf	SRT settings physical interface Type : String Values : "Mgmt", "Data1", "Data2", "Data3", "Data4"
reacquire	Reacquire SRT stream Type : String Values : "Yes", "No"
redmode	Redundancy Mode Type : String Values : "Yes", "No"
redremip	Redundancy Backup SRT input source ip address

URI Argument	Description
	Type : String Value : Up to 200 characters
redremport	Redundancy Backup SRT input source UDP ip port Type : Integer Values : 0 … 65535
redlisten	Redundancy Backup SRT input listen UDP ip port Type : Integer Values : 0 … 65535
redintf	Redundancy Backup SRT physical interface Type : String Values : "Mgmt", "Data1", "Data2", "Data3", "Data4"
bissprofen	BISS Profile enabled or disabled for SRT Type : String Values : "Yes", "No"
bissprof	BISS Profile value for SRT Type : Integer Values : 1..32
cmdrow	Add or Delete an instance of SRT Input For a GET operation, the value will read back the Read-Only (RO) value where the "Active" value means that a previous Add operation was successful for that stream ID. For a POST operation, set the stream "cmdrow "value to "Add" to make that stream ID instance "Active" or "Delete" to make an "Active" stream ID "Inactive". Type : String Values : "Add" (R/W), "Delete" (R/W), "Active" (RO), "Inactive" (RO)
latency	SRT input stream latency This value represents the duration of content that is buffered to allow for re-transmission of data. It is measured in milli-seconds. Type : Integer Values : 0.. 4294967295
overhdbw	SRT Recovery Bandwidth Overhead This value represents how much extra data is reserved for reliability of the SRT. It is the

URI Argument	Description
	recovery bandwidth overhead above the input rate, measured as a percent (%). Type : Integer Values : 5..100
pw	SRT input stream password The passphrase is the shared secret between the sender and the receiver. Type : String Value : 10 to 79 characters
sid	SRT Source Stream ID Type : String Value : Up to 200 characters
reddir	SRT Redundancy Direction Type : String Value : "Non-Revertive"
reddlydir	SRT Direct Redundancy Switch Over Delay Type : Integer Values : 0 … 10000
avtrig	SRT Redundancy AV loss trigger If set, Audio/Video loss will trigger switch over Type : String Values : "Yes", "No"
pmttrig	SRT Redundancy PMT loss trigger If set, PMT loss will trigger switch over Type : String Values : "Yes", "No"
pcrtrig	SRT Redundancy PCR loss trigger If set, PCR loss will trigger switch over Type : String Values : "Yes", "No"
linktrig	SRT Redundancy link loss trigger If set, link loss will trigger switch over Type : String Values : "Yes", "No"
tstrig	SRT Redundancy TS loss trigger If set, transport loss will trigger switch over

URI Argument	Description
	Type : String Values : "Yes", "No"
redconn	SRT Redundancy Connection Type Type : String Value : "Listener", "Caller", "Rendezvous"
redlatency	SRT Redundancy Backup input stream latency This value represents the duration of content that is buffered to allow for re-transmission of data. It is measured in milli-seconds. Type : Integer Values : 0.. 4294967295
redpw	SRT Redundancy Backup input stream password In Redundancy mode, the passphrase is the shared secret between the sender and the receiver. Type : String Value : 10 to 79 characters
redsid	SRT Redundancy Backup Source Stream ID Type : String Value : Up to 200 characters

Output Fields:

Output fields returned in GET results are the same names used for the possible Arguments for this command.

POST Examples:

The following examples assume that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1 and that the remote IP address and port are 10.89.123.456 and 9020 respectively. Please change the IP to the specific unit IP and remote ip and port in use

Example 1: Adding an SRT Stream Instance (making it Active)

```
curl -k -X POST "https://192.168.0.1 /ws/v2/service_
cfg/input/srt?session=$token&js&stream=2&act=Yes&cmdrow=Add&type=Caller
&remoteip=10.89.123.456&remoteport=9020"
```

If successful, the return body will be:

```
"response": {
       "code": "10",
       "result": "success",
       "message": ""
}
```

Example 2: Changing one SRT Parameter

This example changes the type of an existing SRT entry to Rendezvous mode".

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/srt?stream=1&type=Rendezvous"
```

If successful, the return body will be:

```
"response": {
       "code": "10",
       "result": "success",
       "message": ""
}
```

Example 3: Changing multiple SRT Parameters

This example changes several ZIXI parameters for an existing entry. It will set the netid to 12345, sdten to Yes, paten to Yes and niten to Yes

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/srt?stream=1&netid=12345&sdten=Yes&paten=Yes&niten=Yes"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
}
```

GET Examples:

Example 4: GET the full SRT tuning values for a specific SRT entry

This example retrieves all of the SRT settings for stream 1.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/srt?stream=1"
```

If successful, the return body will be:

```
{
       "input": {
               "srt": {
               "card": "1",
               "port": "1",
               "stream": "1",
               "act": "Yes",
               "netid": "12345",
               "acqmode": "Auto",
               "camode": "Std",
               "inputsel": "UserCfg",
               "freqmode": "User Cfg",
               "svclstmode": "Rigorous",
               "baten": "No",
               "niten": "No",
               "sdten": "Yes",
               "paten": "Yes",
               "name": "SRT1",
               "type": "Caller",
               "remoteip": "10.89.123.456",
               "remoteport": "9020",
               "listenport": "0",
               "curintf": "Mgmt",
               "reacquire": "No",
               "redmode": "No",
               "redremip": "",
               "redremport": "0",
               "redlisten": "0",
               "redintf": "Mgmt",
               "bissprofen": "Yes",
               "bissprof": "1",
               "cmdrow": "Active",
               "latency": "0",
               "ovrhdbw": "25",
               "pw": "",
               "sid": "",
               "reddir": "Non-Revertive",
               "reddlydir": "0",
               "avtrig": "Yes",
               "pcrtrig": "Yes",
               "pmttrig": "Yes",
               "linktrig": "Yes",
               "tstrig": "Yes",
               "redconn": "Caller",
               "redlatency": "0",
               "redovrhdbw": "25",
               "redpw": "",
               "redsid": ""
               }
       }
}
```

Example 5: GET a specific SRT value for a specific SRT entry

This example retrieves one specific parameter value, camode, from the SRT settings for SRT stream 1. Please note that in GET URI's only the entry selection argument (stream) must contain a value, all other arguments DO NOT contain values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/input/srt?stream=1&camode&js=1"
```

If successful, the return body will be:

```
"input": {
        "srt": {
                "stream": "1",
                "camode": "Std"        }
}
```

Example 6: GET multiple SRT tuning value for a specific SRT stream

This example retrieves multiple parameter values from the SRT settings. This example retrieves the camode, sdten, niten and paten values from SRT stream 1.

Note In GET URIs, only the entry selection argument (stream) must contain a value, all other arguments **DO NOT** contain values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/input/srt?stream=1&camode&sdten&paten&niten&js"
```

If successful, the return body will be:

```
"input": {
        "srt": {
                "stream": "1",
                "camode": "Std",
                "niten": "No",
                "paten": "Yes",
                "sdten": "Yes"        }
}
```

PE Configuration Command

Table 2.205 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/pe
Command Information	Allows set or get of PE information.
HTTP Methods	GET, POST
Access Type	Read/Write
Access Level	User, Admin
GET Syntax	One of the following: GET https://192.168.0.1/ws/v2/service_cfg/pe, or GET "https://192.168.0.1/ws/v2/service_cfg/pe? peid=[peid#]"
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/pe? peid=[peid#]&chn=[chn#]"

URI Parameters (extension to the Command URL separated by /): N/A

For GET, the URI arguments do not need any values to dump info for all PEs. To GET info for a specific PE, specify the peid argument and value. For POST, specify the peid argument and value, and the chn argument followed by the associated value the user would like to map. Muliple peids may be mapped by using the input file syntax for POST command. In-depth CURL examples can be seen below following the table.

Table 2.206 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
peid (key)	Program Entry Identifier Type: Integer Value: 1..32 When not specified on a GET, all PEs are returned.
chn	Channel Number to be configured on this Program Entry Type: Integer No value required on GET (but may be used as filter for requested output, eg &chn&…) POST Values: 0 .. 65535
inp	Input name Type: String

URI Argument	Description
	No value required on GET (but may be used as filter for requested output, eg &inp&…) This parameter is only applicable for Multi Stream targets It is ignored for Single Stream targets because the input in that case is the one for the currently Active port. POST Values: RF1..RF4, ASI1..AS2, MOIP1..MOIP32, ABR1..ABR2, ZIXI1..ZIXI4, SRT1..SRT2
js	Format output using JSON standard (applies to command line input only) Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML

Table 2.207 Output Field Descriptions

Output Field	Description
peid	Program Entry Identifier Type: Integer Value: 1..32
chn	Channel Number to be configured on this Program Entry Type: Integer Values: 0 .. 65535
inp	Input port name Type: String Values: Single-Stream target: RF1-RF4, MOIP, ASI Multi-Stream target: RF1-RF4, MOIP1,MOIP2, ASI1, ASI2

GET Method Examples

Use to display PE information in the format of XML or JSON, which is based on input parameters.

Input (GET all PE Info and request default XML format output):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_cfg/pe"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<pe>
<record><peid>1</peid><chn>802</chn><inp>RF1</inp></record>
<record><peid>2</peid><chn>0</chn><inp>  </inp></record>
<record><peid>3</peid><chn>0</chn><inp>  </inp></record>
<record><peid>4</peid><chn>0</chn><inp>  </inp></record>
<record><peid>5</peid><chn>0</chn><inp>  </inp></record>
<record><peid>6</peid><chn>0</chn><inp>  </inp></record>
<record><peid>7</peid><chn>0</chn><inp>  </inp></record>
<record><peid>8</peid><chn>0</chn><inp>  </inp></record>
<record><peid>9</peid><chn>0</chn><inp>  </inp></record>
<record><peid>10</peid><chn>0</chn><inp>  </inp></record>
<record><peid>11</peid><chn>0</chn><inp>  </inp></record>
<record><peid>12</peid><chn>0</chn><inp>  </inp></record>
<record><peid>13</peid><chn>0</chn><inp>  </inp></record>
<record><peid>14</peid><chn>0</chn><inp>  </inp></record>
<record><peid>15</peid><chn>0</chn><inp>  </inp></record>
<record><peid>16</peid><chn>0</chn><inp>  </inp></record>
<record><peid>17</peid><chn>0</chn><inp>  </inp></record>
<record><peid>18</peid><chn>0</chn><inp>  </inp></record>
<record><peid>19</peid><chn>0</chn><inp>  </inp></record>
<record><peid>20</peid><chn>0</chn><inp>  </inp></record>
<record><peid>21</peid><chn>0</chn><inp>  </inp></record>
<record><peid>22</peid><chn>0</chn><inp>  </inp></record>
<record><peid>23</peid><chn>0</chn><inp>  </inp></record>
<record><peid>24</peid><chn>0</chn><inp>  </inp></record>
<record><peid>25</peid><chn>0</chn><inp>  </inp></record>
<record><peid>26</peid><chn>0</chn><inp>  </inp></record>
<record><peid>27</peid><chn>0</chn><inp>  </inp></record>
<record><peid>28</peid><chn>0</chn><inp>  </inp></record>
<record><peid>29</peid><chn>0</chn><inp>  </inp></record>
<record><peid>30</peid><chn>0</chn><inp>  </inp></record>
<record><peid>31</peid><chn>0</chn><inp>  </inp></record>
<record><peid>32</peid><chn>0</chn><inp>  </inp></record>
</pe>
```

Input (GET all PE Info and request JSON format output):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_
cfg/pe?js=1"
```

Expected output (values are for example purposes only):

```
{
    "pe": {
        "record": [
            {
                 "peid": "1",
                 "chn": "802",
                 "inp": "RF1"
             },
                        // data for items peid: 2 to peid: 31 omitted to save space in docum
             {
                 "peid": "32",
                 "chn": "0",
                 "inp": "  "
             }
         ]
     }
}
```

Input (GET single PE Info and request default XML format output):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_
cfg/pe?peid=1"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<pe>
<record><peid>1</peid><chn>802</chn><inp>RF1</inp></record>
</pe>
```

Input (GET single PE peid and chn info only and request JSON format output):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_
cfg/pe?peid=1&chn&js=1"
```

Expected output (values are for example purposes only):

```
{
    "pe": {
        "record": {
            "peid": "1",
            "chn": "802"
        }
    }
}
```

POST Method Examples:

Use to Configure PE, XML and JSON are supported from the post file.

XML from file Example:

File contents of temp.xml:

```
<pe><record><peid>1</peid><chn>801</chn><inp>RF1</inp></record></pe>
```

Input (POST PE configuration info via file temp.xml):

```
curl -k -X POST -H "X-SESSION-ID: $token" --header
"Content-Type:application/xml" -d @"C:/projects/temp.xml"
"https://192.168.0.1/ws/v2/service_cfg/pe"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><response>
<code>10</code><result>success</result><message></message></response>
```

JSON from file Example:

File contents of temp.js:

```
{
    "pe": {
        "record": {
            "peid": "1",
            "chn": "802"
        }
    }
}
```

Input (POST PE configuration info via file temp.js):

```
curl -k -X POST -H "X-SESSION-ID: $token" --header
"Content-Type:application/json" -d @"C:/projects/temp.js"
"https://192.168.0.1/ws/v2/service_cfg/pe?js=1"
```

Expected output (values are for example purposes only):

```
{
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
}
```

To POST a channel change to a specific PE, in this case PE 1 and new channel will be 41 the following CURL command applies:

Input (POST PE specific peid configuration info via command line):

```
curl -X POST -i -H "Accept: application/json" -H
"X-SESSION-ID: $token" -k "https:// 192.168.0.1/ws/v2/service_
cfg/pe?peid=1&chn=41"
```

Expected output (values are for example purposes only):

```
{
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
}
```

Output Configuration Command

Table 2.208 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/output
Command Information	Allows set or get of output configuration parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	GET "https://192.168.0.1/ws/v2/service_cfg/output"
POST Syntax	POST --header "Content-type:application/xml" -d @"C:\projects\output.xml" "https://192.168.0.1/ws/v2/service_cfg/output? peid=[peid#]&chn=[chn#]"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.209 XML/JSON file input arguments and Output Fields

Port Name	Description
asi moip	Type: String Valid output port names for Phase 1 (single-stream) units only
asi1 asi2 moip1	Type: String Valid output port names for multi-stream units

Port Name	Description
moip2	

Table 2.210 ASI, ASI1, or ASI2 Next level names (after Port Name)

ASI, ASI1, or ASI2 Next Level Name	Description	
mode	Output Mode Type: String Values: "No Output", "Passthrough", "Service Channels Only", "Map Passthrough", "Map Service Channels Only", "Transcoding"	
dscrmode	Output Descrambling Mode Type: String Values: "Descrambled", "Scrambled"	
ratemode	Output Rate Control Mode Type: String Value: "User"	
rate	Output Rate Type: Float Values: 0.0 ….. 206.0	
insertnul	Insert Null Packet Type: String Values: "Yes", "No" insertnul="No" is not allowed with ratemode="User"	
pe	peid	Program Entry Identifier Type: String XML/JSON Format Value: PE1…PE32, P1A…P16A Command Line Input Value: 1…32 (Multistream), 1..16 (Phase I: Single Stream)
	action	PE Action Type: String

ASI, ASI1, or ASI2 Next Level Name	Description	
		Values: "Drop", "Pass", "Map", "XCode"
	pmtpid	Output PMT PID Type: Integer Values: 0..8191
outchan	Output Channel Number Type: Integer Values: 0..65535	

Table 2.211 MOIP, MOIP1, or MOIP2 Next level names (after Port Name)

MOIP, MOIP1, or MOIP2 Next Level Name	Description
mode	Output Mode Type: String Values: "No Output", "Passthrough", "Service Channels Only", "Map Passthrough", "Map Service Channels Only", "Transcoding"
dscrmode	Output Descrambling Mode Type: String Values: "Descrambled", "Scrambled"
ratemode	Output Rate Control Mode Type: String Value: "User"
rate	Output Rate Type: Float Values: 0.0 ….. 206.0
insertnul	Insert Null Packet Type: String Values: "Yes", "No" insertnul="No" is not allowed with ratemode="User"
mcastip	Set valid SAP destination IP address (multicast only)

MOIP, MOIP1, or MOIP2 Next Level Name	Description	
	Type: String Value: IP address in dot format, for example, 224.2.127.254	
destport	Set the SAP UDP destination port number Type: Integer Value: 1024..65534	
fecscheme	FEC arrangement : Block-aligned (0) or Non Block-aligned Type: String Values: "Block Aligned", "Non-Block Aligned"	
tpperframe	Set the number of transport packets to be sent per IP packet Type: Integer Value: 1..7	
redunmode	Redundancy Mode Type: String Values: "Mirroring", "Backup: First Port", "Backup: Second Port", "Manual: First Port", "Manual: Second Port"	
redundir	Switchover direction Type: String Values: "Non-Revertive", "Revertive"	
redundelfwd	Delay before switching from Main to Backup port, in ms Type: Integer Value: 0..10000	
redundelbck	Delay before switching from Backup to Main port, in sec Type: Integer Value: 0..120	
pe	peid	Program Entry Identifier Type: String XML/JSON Format Value: PE1…PE32, P1A…P16A Command Line Input Value: 1…32 (multi-

MOIP, MOIP1, or MOIP2 Next Level Name	Description	
		stream), 1..16 (single-stream)
	action	PE Action Type: String Values: "Drop", "Pass", "Map", "XCode"
	pmtpid	Output PMT PID Type: Integer Values: 0..8191
	outchan	Output Channel Number Type: Integer Values: 0..65535
ipo	ipoid	Program Entry Identifier Type: Integer Values: 1…40 for D9800-MS with 8 Transcoders, 1…48 for MS with 16 Transcoders, 1..16 for D9800-SS
iporate	Bitrate (in Mbps) of Transport output stream Type: Float Values: 0.0…800.0	
encaps	Encapsulation of Transport output stream Type: String Values: "RTP", "UDP"	
dest_ip	Destination IP Address Type: String Values: IP (v4) address in the dot format, for example, 192.168.0.1	
dest_port	Destination IP Port Type: Integer Values: 1024..65534	
toc	Traffic Class Type: Integer	

MOIP, MOIP1, or MOIP2 Next Level Name	Description
	Values: 0..255
ttl	Time To Live Type: Integer Values: 0..255
src_port	Source UDP Port Type: Integer Values: 0..65535 src_port = 0 means use default UDP port
ann_type	Announce Type Type: String Values: "None", "RFC 2327"
ann_src	Announce Title Source Type: String Values: "User String", "SDT Channel"
ann_title	Title of Announce User Type: String Values: Up to 31 characters (truncates if longer)
fec_mode	EC Mode Type: String Values: "None", "1D","2D"
fec_col	Fec Columns Depth Type: Integer Values: 1..20
fec_row	Fec Rows Depth Type: Integer Values: 4..20
fec_colport	FEC Columns UDP Port Type: Integer Values: 1024..65534

MOIP, MOIP1, or MOIP2 Next Level Name	Description
fec_rowport	FEC Rows UDP Port Type: Integer Values: 1024..65534

Table 2.212 Additional arguments (preceded by ? and separated by & on command line)

Argument	Description
resync	Resynchronize ALL PE Output (applies to POST syntax only) Type: Integer Value: 1..4 1: Resynchronize Program, PMT PID, ES list & ES PIDs 2: Resynchronize ES list 3: Resynchronize ES PIDs 4: Resynchronize Template ES List & PIDs **Note** Currently, only value 1 is supported. All the other inputs (2, 3, and 4) are handled as 1.
js	Format output using JSON standard (applies to command line input only). Type: exist Values: any value or empty (any of js=1, js, js=0, or js=9999 are valid) Omitting this argument formats the output by default in XML.

Example Read (GET) commands (results for single-stream unit shown):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_
cfg/output"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<output><asi><mode>Service Channels
Only</mode><dscrmode>Descrambled</dscrmode><ratemode>User
```

```
</ratemode><rate>68.5</rate><insertnul>Yes</insertnul><pe><peid>PE1</peid><ac
tion>Pass</action>

<pmtpid>5801</pmtpid><outchan>801</outchan></pe><pe><peid>PE2</peid><action>P
ass</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE3</peid><action>Pas
s</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE4</peid><action>Pas
s</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE5</peid><action>Pas
s</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE6</peid><action>Pas
s</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE7</peid><action>Pas
s</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE8</peid><action>Pas
s</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE9</peid><action>Pas
s</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE10</peid><action>Pa
ss</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE11</peid><action>Pa
ss</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE12</peid><action>Pa
ss</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE13</peid><action>Pa
ss</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE14</peid><action>Pa
ss</action>

<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE15</peid><action>Pa
ss</action>
```

```
<pmtpid>8191</pmtpid><outchan>0</outchan></pe><pe><peid>PE16</peid><action>Pa
ss</action>
<pmtpid>8191</pmtpid><outchan>0</outchan></pe></asi><moip><mode>Service
Channels Only</mode>
<dscrmode>Descrambled</dscrmode><ratemode>User</ratemode><rate>68.5</rate>

<insertnul>Yes</insertnul><mcastip>224.2.127.254</mcastip><destport>9875</des
tport>
<fecscheme>Block Aligned</fecscheme><tpperframe>7</tpperframe>
<redunmode>Backup: First Port</redunmode><redundir>Revertive</redundir>
<redundelfwd>0</redundelfwd><redundelbck>1</redundelbck><pe><peid>PE1</peid>
<action>Pass</action><pmtpid>5801</pmtpid><outchan>801</outchan></pe><pe>

<peid>PE2</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>0</outcha
n>

</pe><pe><peid>PE3</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>0
</outchan>

</pe><pe><peid>PE4</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>0
</outchan>

</pe><pe><peid>PE5</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>0
</outchan>

</pe><pe><peid>PE6</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>0
</outchan>

</pe><pe><peid>PE7</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>0
</outchan>

</pe><pe><peid>PE8</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>0
</outchan>

</pe><pe><peid>PE9</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>0
</outchan>

</pe><pe><peid>PE10</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>
0</outchan>

</pe><pe><peid>PE11</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>
0</outchan>

</pe><pe><peid>PE12</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>
0</outchan>
```

```
</pe><pe><peid>PE13</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>
0</outchan>

</pe><pe><peid>PE14</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>
0</outchan>

</pe><pe><peid>PE15</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>
0</outchan>

</pe><pe><peid>PE16</peid><action>Pass</action><pmtpid>8191</pmtpid><outchan>
0</outchan>
</pe></moip></output>
```

Example Write (POST) commands:

Setting parameters using XML body data from command line (for single-stream unit shown):

```
curl -k -X POST -H "X-SESSION-ID: $token" --header
"Content-type:application/xml" -d "<output><asi><port>1</port>
<mode>Pass</mode></asi><moip><mode>No Output</mode></moip></output>"
"https://192.168.0.1/ws/v2/service_cfg/output"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><response>
<code>10</code><result>success</result><message></message></response>
```

Setting Parameters using body data from XML File (for single-stream unit shown):

```
curl -k -X POST -H "X-SESSION-ID: $token" --header
"Content-type:application/xml" -d @"C:\projects\output.xml"
"https://192.168.0.1/ws/v2/service_cfg/output"
```

Output File "output.xml" contains data as shown below.

```
<output>
    <asi>
         <mode>Pass</mode>
       <dscrmode>Scramble</dscrmode>
    </asi>
    <moip>
         <mode>No Output</mode>
       <dscrmode>Descramble</dscrmode>
    </moip>
</output>
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><response>
<code>10</code><result>success</result><message></message></response>
```

ASI Global Output Configuration Command

Table 2.213 Command Details

Command Detail	Description
Command URL	Single-stream (NTC Basic, NTC MOIP), asi output port (only one output exists): https://192.168.0.1/ws/v2/service_cfg/output/asi Or, alternatively: Single-stream (NTC Basic, NTC MOIP), asi output port (only one exists): https://192.168.0.1/ws/v2/service_cfg/output/asi2 Multi-stream unit, asi output port 1: https://192.168.0.1/ws/v2/service_cfg/output/asi1 Multi-stream unit, asi output port 2: https://192.168.0.1/ws/v2/service_cfg/output/asi2
Command Information	Allows set or get of ASI output parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	Single-stream unit: GET "https://192.168.0.1/ws/v2/service_cfg/output/asi" Multi-stream unit: GET "https://192.168.0.1/ws/v2/service_cfg/output/asi1", or GET "https://192.168.0.1/ws/v2/service_cfg/output/asi12?mode"
POST Syntax	Single-stream unit: POST --header "Content-type:application/xml" -d @"C:\projects\output.xml" "https://192.168.0.1/ws/v2/service_cfg/output/asi? mode=<mode>" **Note** Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.

Command Detail	Description
	Multi-stream unit: POST --header "Content-type:application/xml" -d @"C:\projects\output.xml" "https://192.168.0.1/ws/v2/service_cfg/output/asi1? mode=<mode>

URI Parameters (extension to the Command URL separated by /): N/A

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values. For POST any of the below URI arguments can be supplied with their associated value the user would like to set. In-depth CURL examples can be seen below following the table.

Table 2.214 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
mode	Output Mode Type: String Values: "No Output", "Passthrough", "Service Channels Only", "Map Passthrough", "Map Service Channels Only", "Transcoding"
dscrmode	Output Descrambling Mode Type: String Values: "Descrambled", "Scrambled"
ratemode	Output Rate Control Mode Type: String Value: "User"
rate	Output Rate Type: Float Values: 0.0 ….. 206.0
insertnul	Insert Null Packet Type: String Values: "Yes", "No" insertnul="No" is not allowed with ratemode="User"

Table 2.215 Additional arguments (preceded by ? and separated by & on command line)

Argument	Description
resync	Resynchronize ALL PE Output Type: Integer Value: 1..4 1: Resynchronize Program, PMT PID, ES list & ES PIDs 2: Resynchronize ES list 3: Resynchronize ES PIDs 4: Resynchronize Template ES List & PIDs
js	Format output using JSON standard (applies to command line input only) Type: exist Values: any value or empty (any of js=1, js, js=0, or js=9999 are valid) Omitting this argument formats the output by default in XML.

Note For resync operation, only HTTP data body is supported. POST from XML/JSON file is not supported. As noted in Get, on page 14, in the URL input, the space in STRING should be replaced by %20, for example: "https://192.168.0.1/ws/v2/service_cfg/output/asi?mode=Map%20Service%20Channels%20Only.

POST Examples:

Example 1: Set Various Output ASI Parameters

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes both the output descrambling mode to Descrambled and the ratemode to User.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/output/asi?dscrmode=Descrambled&ratemode=User"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

This is the same example being executed on a NTC MS unit:

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/output/asi1?dscrmode=Descrambled&ratemode=Auto"
```

GET Examples:

Example 2: Get full table of ASI Output Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example returns the full ASI Output settings.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https:// 192.168.0.1/ws/v2/service_cfg/output/asi"
```

If successful, the return body will be (please note this is a snippet of the full return body):

```
"output": {
        "asi": {
            "mode": "No Output",
            "dscrmode": "Descrambled",
            "ratemode": "User",
            "rate": "68.5",
            "insertnul": "Yes",
            "pe": [
                {
                    "peid": "PE1",
                    "action": "Drop",
                    "pmtpid": "8191",
                    "outchan": "0"
                },
                {
                    "peid": "PE2",
                    "action": "Drop",
                    "pmtpid": "8191",
                    "outchan": "0"
                }
            ]
    }
}
```

This is the same example executed on a NTC multi-stream unit:

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https:// 192.168.0.1/ws/v2/service_cfg/output/asi2"
```

Example 3: Get specific ASI Output Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example returns the specific ASI Output setting of the descrambled mode (dscrmode) and Output Mode (mode) parameters.

Note For GET arguments do not supply values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/output/asi?&dscrmode&mode"
```

If successful, the return body will be:

```
"output": {
        "asi": {
            "dscrmode": "Descrambled",
            "mode": "No Output"
        }
    }
}
```

MOIP Global Output Configuration Command

Table 2.216 Command Details

Command Detail	Description
Command URL	Single-stream (NTC Basic): MOIP is not supported. Single-stream (NTC MOIP), one unique moip output available: https://192.168.0.1/ws/v2/service_cfg/output/moip, or https://192.168.0.1/ws/v2/service_cfg/output/moip1 Multi-stream unit, moip output port 1: https://192.168.0.1/ws/v2/service_cfg/output/moip1 Multi-stream unit, moip output port 2: https://192.168.0.1/ws/v2/service_cfg/output/moip2
Command Information	Allows set or get of MOIP output parameters.
HTTP	GET, POST

Command Detail	Description
Methods	
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	One of the following: GET "https://192.168.0.1/ws/v2/service_cfg/output/moip", GET "https://192.168.0.1/ws/v2/service_cfg/output/moip1", or GET "https://192.168.0.1/ws/v2/service_cfg/output/moip2?mode"
POST Syntax	Single-stream unit: POST --header "Content-type:application/xml" -d @"C:\projects\output.xml" "https://192.168.0.1/ws/v2/service_cfg/output/moip? mode=<mode>" **Note** Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?. Multi-stream unit: POST --header "Content-type:application/xml" -d @"C:\projects\output.xml" "https://192.168.0.1/ws/v2/service_cfg/output/moip1? mode=<mode>

URI Parameters (extension to the Command URL separated by /): N/A

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values. For POST any of the below URI arguments can be supplied with their associated value the user would like to set. In-depth CURL examples can be seen below following the table.

Table 2.217 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
mode	Output Mode Type: String Values: "No Output", "Passthrough", "Service Channels Only", "Map Passthrough", "Map Service Channels Only", "Full DPM Control", "Transcoding", "SPTS Service Channels Only", "SPTS MAP Service Channels Only", "SPTS Full DPM Control", "SPTS Transcoding"
dscrmode	Output Descrambling Mode Type: String

URI Argument	Description
	Values: "Descrambled", "Scrambled"
ratemode	Output Rate Control Mode Type: String Value: "User"
rate	Output Rate Type: Float Values: 0.0 ….. 206.0
insertnul	Insert Null Packet Type: String Values: "Yes", "No" insertnul="No" is not allowed with ratemode="User"
mcastip	Set valid SAP destination IP address (multicast only) Type: String Value: IP address in dot format, for example, 224.2.127.254
destport	Set the SAP UDP destination port number Type: Integer Value: 1024..65534
fecscheme	FEC arrangement : Block-aligned (0) or Non Block-aligned Type: String Values: "Block Aligned", "Non-Block Aligned"
tpperframe	Set the number of transport packets to be sent per IP packet Type: Integer Value: 1..7
redunmode	Redundancy Mode Type: String Values: "Mirroring", "Backup: First Port", "Backup: Second Port", "Manual: First Port", "Manual: Second Port"
redundir	Switchover direction Type: String Values: "Non-Revertive" or "Revertive"

URI Argument	Description
redundelfwd	Delay before switching from Main to Backup port, in ms Type: Integer Value: 0..10000
redundelbck	Delay before switching from Backup to Main port, in sec Type: Integer Value: 0..120

Table 2.218 Additional arguments (preceded by ? and separated by & on command line)

Argument	Description
resync	Resynchronize ALL PE Output Type: Integer Value: 1..4 1: Resynchronize Program, PMT PID, ES list & ES PIDs 2: Resynchronize ES list 3: Resynchronize ES PIDs 4: Resynchronize Template ES List & PIDs
js	Format output using JSON standard (applies to command line input only). Type: exist Values: any value or empty (any of js=1, js, js=0, or js=9999 are valid) Omitting this argument formats the output by default in XML

Note For resync operation, only HTTP data body is supported. POST from XML/JSON file is not supported.

As noted in Get, on page 14, in the URL input, the space in STRING should be replaced by %20, for example: "https://192.168.0.1/ws/v2/service_cfg/output/moip?mode=Map%20Service%20Channels%20Only".

POST Examples:

Example 1: Set Various MOIP Output Parameters

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes both the output descrambling mode to Descrambled and the ratemode to User.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/output/moip?dscrmode=Descrambled&ratemode=User"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

This is the same example except it is being executed on a NTC MS unit and the descrambling mode is set to Scrambled instead of Descrambled:

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/output/moip2?dscrmode=Scrambled&ratemode=User"
```

GET Examples:

Example 2: Get full table of MOIP Output Settings

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example returns the full MOIP Output settings.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/output/moip"
```

If successful, the return body will be (please note this is a snippet of the full return body):

```
"output": {
        "moip": {
            "mode": "No Output",
            "dscrmode": "Descrambled",
            "ratemode": "User",
            "rate": "68.5",
            "insertnul": "Yes",
            "pe": [
                {
                    "peid": "1",
                    "action": "Drop",
                    "pmtpid": "8191",
                    "outchan": "0"
                },
                {
                    "peid": "2",
                    "action": "Drop",
                    "pmtpid": "8191",
                    "outchan": "0"
                }
      ]
        }
}
```

This is the same example being executed on a NTC MS unit:

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/output/moip1"
```

Example 3: Get specific MOIP Output Settings

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example returns the specific MOIP Output setting of the descrambled mode (dscrmode) and Output Mode (mode) parameters.

Note For GET arguments do not supply values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/output/moip?&dscrmode&mode"
```

If successful, the return body will be:

```
"output": {
        "moip": {
            "dscrmode": "Descrambled",
            "mode": "No Output"
        }
    }
}
```

DPM ASI/MOIP PE Output Configuration Command

Table 2.219 Command Details

Command Detail	Description
Command URL	Single-stream (NTC Basic, NTC MOIP), asi output port (only one output exists): https://192.168.0.1/ws/v2/service_cfg/output/asi/pe Or alternatively, Single-stream (NTC Basic, NTC MOIP), asi output port (only one output exists): https://192.168.0.1/ws/v2/service_cfg/output/asi2/pe Single-stream (NTC Basic): MOIP is not supported. Single-stream (NTC MOIP), only one unique moip output available: https://192.168.0.1/ws/v2/service_cfg/output/moip/pe, or https://192.168.0.1/ws/v2/service_cfg/output/moip1/pe Multi-stream, asi or moip output port 1: https://192.168.0.1/ws/v2/service_cfg/output/asi1/pe https://192.168.0.1/ws/v2/service_cfg/output/moip1/pe Multi-stream unit, asi or moip output port 2: https://192.168.0.1/ws/v2/service_cfg/output/asi2/pe https://192.168.0.1/ws/v2/service_cfg/output/moip2/pe
Command Information	Allows set or get of ASI or MOIP DPM PE output parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	Single-stream NTC MOIP or NTC Basic unit: GET "https://192.168.0.1/ws/v2/service_cfg/output/asi/pe" GET "https://192.168.0.1/ws/v2/service_cfg/output/asi2/pe" Single-stream NTC MOIP unit: GET "https://192.168.0.1/ws/v2/service_cfg/output/moip/pe" GET "https://192.168.0.1/ws/v2/service_cfg/output/moip1/pe" Multi-stream unit:

Command Detail	Description
	GET "https://192.168.0.1/ws/v2/service_cfg/output/asi1/pe" GET "https://192.168.0.1/ws/v2/service_cfg/output/asi2/pe?mode" GET "https://192.168.0.1/ws/v2/service_cfg/output/moip1/pe" GET "https://192.168.0.1/ws/v2/service_cfg/output/moip2/pe?mode
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	Single-stream unit: POST "https://192.168.0.1/ws/v2/service_cfg/output/asi/pe? peid=[peid#]&mode=<mode>... " POST "https://192.168.0.1/ws/v2/service_cfg/output/moip/pe? peid=[peid#]&mode=<mode>... " Multi-stream unit: POST "https://192.168.0.1/ws/v2/service_cfg/output/asi1/pe? peid=[peid#]&mode=<mode>... " POST "https://192.168.0.1/ws/v2/service_cfg/output/moip1/pe? peid=[peid#]&mode=<mode>... "

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.220 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
peid (Key)	Program Entry Identifier Type: String XML/JSON Format Value: PE1…PE32, P1A…P16A Command Line Input Value: 1…32 (multi-stream), 1..16 (single-stream)
action	PE Action Type: String Values: "Drop", "Pass", "Map", "XCode"
pmtpid	Output PMT PID Type: Integer Values: 0..8191

URI Argument	Description
outchan	Output Channel Number Type: Integer Values: 0..65535

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values, except the peid argument which must be supplied with a program entry ID value. For POST any of the below URI arguments can be supplied with their associated value the user would like to set.

Note To use this command properly, a level 1 parameter must be specified such as asi (asi1/asi2 on a NTC MS unit) followed by a level 2 parameter which is pe followed by the peid argument that must be supplied with a value, and only then can the remaining arguments (as below) be used for GET/POST. In-depth CURL examples follow below.

POST Examples:

Note All POST examples will apply to the DPM ASI PE settings, but the same examples can be applied to the DPM MOIP PE settings, by simply replacing the ‘asi’ in the URI with ‘moip’

Example 1: Set Various DPM ASI Parameters for a specific PE

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes both the PE action to Pass and Output Channel Number to 105.

Note The asi/pe?peid=VAL must be provided before adding the specific arguments as seen in the curl command below (due to having to place level 1 parameter followed by level 2 followed by peid followed by optional arguments).

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/output/asi/pe?peid=1&action=Pass&outchan=105"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Note All GET examples will apply to the DPM ASI PE settings, but the same examples can be applied to the DPM MOIP PE settings, by simply replacing the ‘asi’ in the URI with ‘moip’.

Example 2: Get full table of DPM ASI Parameters for all PEs

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example returns the full DPM ASI Output Settings for all PEs.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/output/asi/pe"
```

If successful, the return body will be (please note this is a snippet of the full return body):

```
"output": {
        "asi": {
            "pe": [
                {
                    "peid": "PE1",
                    "action": "Pass",
                    "pmtpid": "8191",
                    "outchan": "105"
                },
                {
                    "peid": "PE2",
                    "action": "Drop",
                    "pmtpid": "8191",
                    "outchan": "0"
                },
            ]
        }
}
```

Example 3: Get the table of DPM ASI Parameters for a specific PE

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example returns the DPM ASI Output Settings for a specific PE, in this case PE 1.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/output/asi/pe?peid=1"
```

If successful, the return body will look like:

```
"output": {
        "asi": {
            "pe": {
                "peid": "PE1",
                "action": "Pass",
                "pmtpid": "8191",
                "outchan": "105"
            }
        }
    }
```

Example 4: Get specific parameters of the DPM ASI Settings for a specific PE

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example returns the pmtpid and output channel of the DPM ASI Output Settings for a specific PE, in this case PE 1.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/output/asi/pe?peid=1&pmtpid&outchan"
```

If successful, the return body will look like:

```
"output": {
        "asi": {
            "pe": {
                "peid": "PE1",
                "outchan": "105",
                "pmtpid": "8191"}
        }
}
```

MOIP Output Stream Configuration

Table 2.221 Command Details

Command Detail	Description
Command URL	Single-stream (Basic, NTC MOIP), moip output port pair 1/2 https://192.168.0.1/ws/v2/service_cfg/output/moip/ipo, or https://192.168.0.1/ws/v2/service_cfg/output/moip1/ipo Single-stream (Basic): MOIP is not supported Multi-stream unit, moip output port pair 1/2: https://192.168.0.1/ws/v2/service_cfg/output/moip1/ipo Multi-stream unit, moip output port pair 3/4:

Command Detail	Description
	https://192.168.0.1/ws/v2/service_cfg/output/moip2/ipo
Command Information	Allows set or get of MOIP Stream ID output parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	Single-stream NTC MOIP unit: GET "https://192.168.0.1/ws/v2/service_cfg/output/moip/ipo" GET "https://192.168.0.1/ws/v2/service_cfg/output/moip1/ipo" Multistream unit: GET "https://192.168.0.1/ws/v2/service_cfg/output/moip1/ipo" GET "https://192.168.0.1/ws/v2/service_cfg/output/moip2/ipo"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	Single-stream unit: POST "https://192.168.0.1/ws/v2/service_cfg/output/moip/ipo?ipoid=[Stream#]&iporate=<ipo rate>... " Multi-stream unit: POST "https://192.168.0.1/ws/v2/service_cfg/output/moip1/ipo?ipoid=[Stream#]&iporate=<ipo rate>... " POST "https://192.168.0.1/ws/v2/service_cfg/output/moip2/ipo?ipoid=[Stream#]&iporate=<ipo rate>... "

URI Parameters (extension to the Command URL separated by /): N/A

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values, except the peid argument which must be supplied with a program entry ID value. For POST, any of the below URI arguments can be supplied with their associated value the user would like to set.

Table 2.222 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
ipoid	Program Entry Identifier Type: Integer Values: 1…40 for multi-stream unit with 8 Transcoders, 1…48 for multi-stream unit with 16 Transcoders, 1..16 for single-stream unit

URI Argument	Description
iporate	Bitrate (in Mbps) of transport output stream Type: Float Values: 0.0…800.0
encaps	Encapsulation of transport output stream Type: String Values: "RTP", "UDP"
dest_ip	Destination IP Address Type: String Values: IP address in the dot format, for example, 192.168.0.1
dest_port	Destination IP Port Type: Integer Values: 1..65534
toc	Traffic Class Type: Integer Values: 0..255
ttl	Time To Live Type: Integer Values: 0..255
src_port	Source UDP Port Type: Integer Values: 0..65534 src_port = 0 means use default UDP port
ann_type	Announce Type Type: String Values: "None", "RFC 2327"
ann_src	Announce Title Source Type: String Values: "User String", "SDT Channel"
ann_title	Announce User’s Title

URI Argument	Description
	Type: String Values: Up to 32 characters
fec_mode	FEC Mode Type: String Values: "None", "1D","2D"
fec_col	Fec Columns Depth Type: Integer Values: 1..20
fec_row	Fec Rows Depth Type: Integer Values: 4..20
fec_colport	FEC Columns UDP Port Type: Integer Values: 1..65534
fec_ rowport	FEC Rows UDP Port Type: Integer Values: 1..65534

Note To use this command properly, firstly a level 1 parameter must be specified such as moip (moip1/moip2 on a NTC MS unit) followed by a level 2 parameter which is ipo followed by the ipoid argument that must be supplied with a value, and only then can the remaining arguments (as below) be used for GET/POST. In-depth CURL examples follow below.

POST Examples:

Example 1: Set Various MOIP Output Parameters for a specific Stream (PE)

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes both the MOIP output bitrate to 12.5 and Time To Live (TTL) to 54 for MOIP stream which belongs to PE4 (in SPTS) mode.

Note The moip/ipo?ipoid=VAL must be provided before adding the specific arguments as seen in the curl command below (due to having to place level 1 parameter followed by level 2 followed by ipoid followed by optional arguments).

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/output/moip/ipo"
```

If successful, the return body will be:

Note This is a snippet of the full return body.

```
        "output": {
        "moip": {
            "ipo": [
                {
                    "ipoid": "1",
                    "iporate": "68.5",
                    "encaps": "RTP",
                    "dest_ip": "225.1.1.101",
                    "dest_port": "49152",
                    "toc": "0",
                    "ttl": "64",
                    "src_port": "0",
                    "ann_type": "None",
                    "ann_src": "User String",
                    "ann_title": "Cisco Default SAP1_1",
                    "fec_mode": "2D",
                    "fec_col": "4",
                    "fec_row": "10",
                    "fec_colport": "49154",
                    "fec_rowport": "49156"
                },
                {
                    "ipoid": "2",
                    "iporate": "18.0",
                    "encaps": "RTP",
                    "dest_ip": "168.0.169.34",
                    "dest_port": "49152",
                    "toc": "0",
                    "ttl": "34",
                    "src_port": "0",
                    "ann_type": "None",
                    "ann_src": "User String",
                    "ann_title": "Cisco Default SAP1_2",
                    "fec_mode": "2D",
                    "fec_col": "4",
                    "fec_row": "10",
                    "fec_colport": "49154",
                    "fec_rowport": "49156"
                },
            }
```

Example 3: Get the table of MOIP Output Stream Parameters for a specific Stream (or PE)

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example returns the MOIP Output Settings for a specific Stream (PE), in this case PE 1.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/output/moip/ipo?ipoid=1"
```

If successful, the return body will be:

```
"output": {
        "moip": {
            "ipo": {
                "ipoid": "1",
                "iporate": "68.5",
                "encaps": "RTP",
                "dest_ip": "225.1.1.101",
                "dest_port": "49152",
                "toc": "0",
                "ttl": "64",
                "src_port": "0",
                "ann_type": "None",
                "ann_src": "User String",
                "ann_title": "Cisco Default SAP1_1",
                "fec_mode": "2D",
                "fec_col": "4",
                "fec_row": "10",
                "fec_colport": "49154",
                "fec_rowport": "49156"
            }
        }
    }
```

Example 4: Get specific parameters of the MOIP Output Settings for a specific Stream (PE)

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example returns the destination

IP address and the destination IP port of the MOIP Output Settings for a specific Stream (PE) in this case PE 1.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/output/moip/ipo?ipoid=1&dest_ip&dest_
port"
```

If successful, the return body will be:

```
"output": {
        "moip": {
            "ipo": {
                "ipoid": "1",
                "dest_ip": "225.1.1.101",
                "dest_port": "49152"
            }
        }
    }
```

Decode Configuration Commands

This section lists the decode configuration commands.

Control Decode Configuration Command

Table 2.223 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/ctrl
Command Information	Allows set or get of decoder enable/disable control parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	GET "https://192.168.0.1/ws/v2/service_cfg/decode/ctrl", or GET "https://192.168.0.1/ws/v2/service_cfg/decode/ctrl?decoder=[value]"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_cfg/decode/ctrl?decoder=video&enable=Yes"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.224 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
decoder	Decoder Identifier Type: String Values: "Video", "Audio1", "Audio2", "Audio3", "Audio4", "VBI", "Data","MPE1", "MPE2", "MPE3", "MPE4", "MPE5", "STT", "DPI"
enable	Enable/Disable selected Decoder Type: String Value: "Yes", "No"

Note For GET, the URI arguments below will not contain values, except for the decoder argument which if used must contain the value for the specific decoder users would like to retrieve information about. For POST, the arguments below will need to be specified with a value. In-depth CURL examples can be seen below.

POST Examples:

Example 1: Set a Specific Decoder to be enabled

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example sets the video decoder as enabled.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/ctrl?enable=Yes&decoder=Video"
```

If successful, the return body will look like:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Retrieve the full list of Decoders and whether they are enabled or disabled

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/ctrl"
```

If successful, the return body will look like (please note this is a snippet of the full return body):

```
"decode": {
        "ctrl": [
            {
                "decoder": "Video",
                "enable": "Yes"
            },
            {
                "decoder": "Audio1",
                "enable": "Yes"
            },
     ]
}
```

Example 3: Retrieve a specific Decoder and whether it is enabled or disabled

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves the video decoder to view the enabled status.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/ctrl?&decoder=Video"
```

If successful, the return body will look like:

```
"decode": {
        "ctrl": {
            "decoder": "Video",
            "enable": "Yes"
        }
    }
```

Video Decode Configuration Command

Table 2.225 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/video

Command Detail	Description
Command Information	Allows set or get of video decode settings parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	One of the following: GET "https://192.168.0.1/ws/v2/service_cfg/decode/video", or GET "https://192.168.0.1/ws/v2/service_cfg/decode/video?pvformat"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_cfg/decode/video?pvformat=[value]&sdformat=[value]"

URI Parameters (extension to the Command URL separated by /): N/A

Note For GET, the URI arguments below will not contain values. For POST, the arguments below will need to be specified with a value. In-depth CURL examples can be seen below

Note Trisync is not supported in the current release. It may still be accepted as an input Query/Set Argument and shown in the output return body data in software Version 2.27 or below. The client software should ignore and not use or display this field.

Table 2.226 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
pvformat	PV Output Format Type: String Values: "Auto", "HD1080i", "HD720p", "SD"
sdformat	SD Video Output Format Type: String Values: "Auto", "PAL-B/G/I/D", "PAL-M", "PAL-N (AR)", "NTSC"

POST Examples:

Example 1: Set various video arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example sets the PV Output Format to Auto and the SD Video Output Format to Auto.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/video?sdformat=Auto&pvformat=Auto"
```

If successful, the return body will look like:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Retrieve the full Video Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/video"
```

If successful, the following is an example of the return body (this is a snippet of the full return body):

```
"decode": {
        "video": {
            "pvformat": "Auto",
            "sdformat": "Auto",
        }
    }
```

Example 3: Retrieve specific video settings

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example will retrieve the PV Output Format and SD Video Output Format.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/video?&sdformat&pvformat"
```

If successful, the following is an example of the return body:

```
"decode": {
        "video": {
            "pvformat": "Auto",
            "sdformat": "Auto"
        }
```

Aspect Ratio (AR) Decode Configuration Command

Table 2.227 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/ar
Command Information	Allows set or get of aspect ratio decode settings parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	One of the following: GET "https://192.168.0.1/ws/v2/service_cfg/decode/ar", or GET "https://192.168.0.1/ws/v2/service_cfg/decode/ar?sdar"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_cfg/decode/ar?sdar=[value]&arselect=[value]"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.228 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
sdar	Standard Definition Aspect Ratio Type : String Value: "4:3", "16:9"
arselect	Selected Aspect Ratio Conversion Type: String Value: "None", "Auto", "Auto AFD", "16:9 L/B", "4:3 P/B", "14:9", "4:3 CCO", "16:9 SCALE"
wssmode	Wide Screen Signaling Mode

URI Argument	Description
	Type: String Values: "Passthrough", "Suppress", "Auto:Modify", "Auto:Create"

Note For GET, the URI arguments below will not contain values. For POST, the arguments below will need to be specified with a value. In-depth CURL examples can be seen below

POST Examples:

Example 1: Set various Aspect Ratio Setting Arguments

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example sets the Standard Definition Aspect Ratio to 4:3 and the Selected Aspect Ratio Conversion to Auto.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1 /ws/v2/service_cfg/decode/ar?arselect=Auto&sdar=4:3"
```

If successful, the return body will look like:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Retrieve the full Aspect Ratio Settings

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/ar"
```

If successful, the return body will look like (this is a snippet of the full return body):

```
"decode": {
        "ar": {
            "sdar": "4:3",
            "arselect": "Auto",
            "wssmode": "Passthrough"
        }
    }
```

Example 3: Retrieve specific Aspect Ratio Settings

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves the Standard Definition Aspect Ratio and the Selected Aspect Ratio Conversion.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/ar?&arselect&sdar"
```

If successful, the return body will look like:

```
"decode": {
        "ar": {
            "arselect": "Auto",
            "sdar": "4:3"
        }
    }
```

Subtitle (SUBT) Decode Configuration Command

Table 2.229 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/subt
Command Information	Allows set or get of subtitle decode settings parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	One of the following: GET "https://192.168.0.1/ws/v2/service_cfg/decode/subt", or GET "https://192.168.0.1/ws/v2/service_ cfg/decode/subt?opmode"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_ cfg/decode/ar?opmode=[value]"

URI Parameters (extension to the Command URL separated by /): N/A

Note For GET, the URI arguments below will not contain values. For POST, the arguments below will need to be specified with a value. In-depth CURL examples can be seen below.

Table 2.230 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
opmode	Operational Mode of Subtitle Type: String Value: "Off", "On", "Imitext", "DVB"
langmenu	Language Menu Options Type: String Value: "Language List", "Language Entry", "PMT Order"
langlist	Language List Options Type: String Value: "ara", "btk", "ben", "bul", "chi", "cze", "dan", "dut", "eng", "fin", "fre", "ger", "gre", "heb", "hin, "hun", "ice", "ind", "ita", "jpn", "kor", "may", "mul", "nor", "per", "pol", "por", "rum", "rus", "san", "scc", "sin", "slo", "som", "spa", "swe", "tai", "tam", "tha", "tur", "ukr", "vie"
langentry	Language Entry Type: String Value: 3 character language string
pmtorder	PMT Order of Subtitle PID Type: String Values: "First", "Second", "Third", "Fourth", "Fifth", "Sixth", "Seventh", "Eighth"
imitext	imitext Position Type: String Values: "Standard", "Extended"
foregndcol	Foreground Color Type: String Values: "Auto", "Yellow", "White"
backgndcol	Foreground Color

URI Argument	Description
	Type: String Values: "None", "Auto", "Shadow", "Opaque", "Semi"

POST Examples:

Example 1: Set various Subtitle Setting Arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example sets the imitext Position to Standard and the Foreground Color to Auto.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/subt?imitext=Standard&foregndcol=Auto"
```

If successful, the return body will look like:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Retrieve the full Subtitle Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/subt"
```

If successful, the return body will look:

```
"decode": {
        "subt": {
            "opmode": "Off",
            "langmenu": "Language Entry",
            "langlist": "eng",
            "langentry": "eng",
            "pmtorder": "First",
            "imitext": "Standard",
            "foregndcol": "Auto",
            "backgndcol": "Auto"
        }
    }
```

Example 3: Retrieve specific Subtitle Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves the imitext Position and the Foreground Color.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/subt?&imitext&foregndcol"
```

If successful, the return body will look like:

```
"decode": {
        "subt": {
            "foregndcol": "Auto",
            "imitext": "Standard"
        }
    }
```

Closed Caption (CC) Decode Configuration Command

Table 2.231 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/cc
Command Information	Allows set or get of closed caption decode settings parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	One of the following: GET "https://192.168.0.1/ws/v2/service_cfg/decode/cc", or GET "https://192.168.0.1/ws/v2/service_cfg/decode/cc?prefccmode"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_cfg/decode/cc?prefccmode=[value]"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.232 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
prefccmode	Preferred Closed Caption Mode Type: String Value: "Auto", "SA Custom", "EIA 708", "Type 3", "DirectTv Type 3", "Type 4 SA", "Type 4 ATSC", "Reserved", "DVS 157"

Note For POST, the value for the argument seen above must be specified, but for GET there are no values for arguments.

POST Examples:

Example 1: Set various CC Setting Arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example sets the prefccmode to Auto.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/cc?prefccmode=Auto"
```

If successful, the return body will look like:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Retrieve the full CC Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/cc"
```

If successful, the return body will look like:

```
"decode": {
          "cc": {
               "prefccmode": "Auto"
          }
     }
```

Note The full CC settings table is the same as just retrieving it with the prefccmode argument as the prefccmode argument is the only argument of CC.

Audio Decode Configuration Command

Note The audio decode configuration command is only supported in Version 1.x to 2.05.

Table 2.233 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/audio
Command Information	Allows set or get of audio decode settings parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	One of the following: GET "https://192.168.0.1/ws/v2/service_cfg/decode/audio", GET "https://192.168.0.1/ws/v2/service_cfg/decode/audio?device=[device#] &mode&ac3comp&left&right& pmtsrc&ddpmode&langmenu&lang", or GET "https://192.168.0.1/ws/v2/service_cfg/decode/audio? device=all" **Note** This option is available in Version 2.75 or later. This command format is currently equivalent to specifying the GET operation for service_cfg/device/audio without any Query arguments. It will return the complete data for all Audio device instances. Other filter parameters will currently be ignored when specified along with the GET device=all key for this command.
POST Syntax Setting Parameters using command line	POST "https://192.168.0.1/ws/v2/service_cfg/decode/audio?device=[device#] &mode=[value]&…", or

Command Detail	Description
arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_cfg/decode/audio?device=all&langmenu=Language%20List" **Note** This option is available in Version 2.75 or later. Unlike the device=[device#] format, this command variation allows values of specific fields to be set simultaneously for all of the available Audio device instances.

URI Parameters (extension to the Command URL separated by /): N/A

The following table lists the URI arguments:

Table 2.234 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
device (key)	Audio Device Instance Type: Integer Value: 1 .. 2, all
mode	Audio Mode Type: String Value: "Stereo", "Mixed", "L-MONO", "R-MONO"
ac3comp	AC3 Compression Type: String Value: "RF Mode", "Line Mode", "Custom 1", "Custom 0"
left	Left volume Type : Float Value: -6.0 .. 6.0
right	Right volume Type : Float Value: -6.0 .. 6.0
pmtsrc	PMT Audio Source Type: String Value: "AUD1", "AUD2", ……. "AUD64"
ddpmode	Dolby Digital Mode

URI Argument	Description
	Type: String Value: "Trans", "Pass"
langmenu	Language Selection Menu option Type: String Value: "Language List", "Language Entry", "PMT Order"
langlist	Language List Type: String Value: 3 character language string (for details, see Subtitle (SUBT) Decode Configuration Command, on page 373.
lang	Language Value. Type: String Value: Any valid 3 character language code, for example, "eng"

Note All of the URI Arguments below apply to both GET and POST and are the same names used in the GET output fields. For GET, the URI arguments do not need any values, except the device which must be specified. For POST, the device must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

POST Examples:

Example 1: Set various Audio Setting Arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example sets the ddpmode to Pass.

Note The device index must be specified in the URI. For this example the ddpmode will be set for audio device 1.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/audio?ddpmode=Pass&device=1"
```

If successful, the return body will look like:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Retrieve the full Audio Settings for a specific device

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. In this example the audio settings are retrieved for Device 1.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/audio?right&left&ddpmode&mode&device=1&
lang&ac3comp&pmtsrc&langmenu"
```

If successful, the return body will be:

```
"decode": {
        "audio": {
            "device": "1",
            "ac3comp": "RF Mode",
            "ddpmode": "Pass",
            "lang": "eng",
            "langmenu": "PMT Order",
            "left": "0.0",
            "mode": "Stereo",
            "pmtsrc": "AUD1",
            "right": "0.0"
        }
    }
```

Specific audio setting arguments per given audio device can be retrieved by including the specific argument in the URI. For example, to only retrieve the pmtscr and langmenu arguments of audio device 1, the CURL command is:

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/audio?device=1&pmtsrc&langmenu"
```

Example 3: Set Audio decoder langmenu option to all Audio devices

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

This example sets the langmenu option to PMT%20Order.

Note The device=all must be specified in the URI for PMT%20Order case because that setting value cannot be combined with any other "Language%20List" or "Language%20Entry" option value for other devices.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/audio?device=all&langmenu=PMT%20Order"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 4: Set Audio decoder langmenu option to Language List when previously set to PMT Order

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

This example is intended to be run only after Example 3 and sets the langmenu option to Language%20List.

Note The device=all must be specified in the URI for Language%20List case when the existing value was previously set to PMT%20Order because that setting value cannot be combined with any other "Language%20List" or "Language%20Entry" option value for other devices.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/audio?device=all&langmenu=Language%20Lis
t"
```

If successful, the return body will look like:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 5: Attempt to set Audio decoder langmenu option to PMT Order using device=# instead of device=all format

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

This example is intended to be run only after Example 4 and attempts to sets the langmenu option to PMT%20Order on only one device.

Note The device=all must be specified in the URI for PMT%20Order case because that setting value cannot be combined with any other "Language%20List" or "Language%20Entry" option value for other devices.

This example is not expected to be successful:

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/audio?device=1&langmenu=PMT%20Order"
```

The return body will look like:

```
{
    "response": {
        "code": "10",
        "result": "failure",
        "message": "Select by PMT Order and Language cannot be combined.
       Try changing settings for all Audio devices simultaneously."
    }
}
```

To fix this issue, use device=all in the command to set the same value for all devices first as per Example 3.

Example 6: Attempt to set Audio decoder langmenu option to Language Entry using device=# instead of device=all format after all Audio devices already have langmenu set to PMT Order

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

The following example is intended to be run only after Example 3 and attempts to sets the langmenu option to Language%20Entry on only one device.

Note The device=all must be specified in the URI for Language%20Entry case when the existing value was previously set to PMT%20Order because that setting value cannot be combined with any other "Language%20List" or "Language%20Entry" option value for other devices.

The following example is not expected to be successful:

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/audio?device=2&langmenu=Language%20Entry"
```

The return body will look like:

```
{
    "response": {
        "code": "10",
        "result": "failure",
        "message": "Select by PMT Order and Language cannot be combined.
       Try changing settings for all Audio devices simultaneously."
    }
}
```

To fix this issue, use device=all in the command to set the same value for all devices first as per below:

This example sets the langmenu option for ALL Audio devices to Language%20Entry.

Note The device=all must be specified in the URI for Language%20Entry case when the existing value was previously set to PMT%20Order because that setting value cannot be combined with any other "Language%20List" or "Language%20Entry" option value for other devices.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/audio?device=all&langmenu=Language%20Ent
ry"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

To then achieve, the device=# setting, please proceed to next example.

Example 7: Attempt to set Audio decoder langmenu option to Language List (or Language Entry) on single device in combination with other devices that are already set to use Language Entry or Language List

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

This example is intended to be run only after either all or some of current Audio devices are already set to using either Language%20Entry or Language%20List but not PMT%20Order. We recommend that you first set all the devices (using the "device=all" POST version of the command) to the same Language%20xxxx option value before changing the setting for individual devices.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/audio?device=2&langmenu=Language%20Entry"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Audio Device SDI Preference Setting Command

Note The audio decode configuration command is only supported in Version 2.26 or later.

Table 2.235 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/audio (for SDI Audio Preference Setting)
Command Information	Allows set or get of audio decode SDI preference setting parameter.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	GET "https://192.168.0.1/ws/v2/service_cfg/decode/audio?device=[device#]&sdiindex=[sdiindex#]&sdipref"
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/decode/audio?device=[device#]&sdiindex=[sdiindex#]&sdipref=

Command Detail	Description
Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	[sdiprefvalue]…"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.236 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
device (key)	Audio Device Instance Type: Integer Value: 1 .. 2
sdiindex (key)	Audio SDI Device Instance Type: Integer Value: 1 Note For this command, the only valid value for sdiindex is 1. This is not to be confused with the sdiindex value (1..2) range that applies for other commands (such as ws/v2/service_cfg/decode/sdiaudio).
sdipref	SDI Audio Preference/AES Output Control Type: String Values: "CM Samples", "Compressed"

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values, except the device and sdiindex which must be specified. For POST the device and sdiindex must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

POST Examples:

Example 1: Set Audio Device SDI Preference

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example sets the sdipref to Compressed.

Note The device index must be specified in the URI, as well as the sdiindex. For this example the sdipref will be set for audio device 1 and sdiindex 1.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/audio?device=1&sdiindex=1&sdipref=Compressed"
```

If successful, the return body will look like:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Retrieve the Audio Device SDI Preference setting

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. In this example the audio device is 1 and sdiindex is 1. Please note that the only argument of this command is sdipref hence it must be part of the URI.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/audio?device=1&sdiindex=1&sdipref"
```

If successful, the return body will look like:

```
"decode": {
        "audio": {
            "sdipref": "Compressed"
        }
}
```

Audio Device SDI1 or SDI2 Output Settings Command

Table 2.237 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/audio (for SDI1 or SDI2 Output Settings)
Command Information	Allows set or get of audio decode SDI1 or SDI2 output settings parameter.

Command Detail	Description
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	GET "https://192.168.0.1/ws/v2/service_cfg/decode/audio?sdi1output"
POST Syntax Setting parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_ cfg/decode/audio? sdi1output=[sdi1_stringvalue]…"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.238 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
sdi1output	SDI1 Output Mode Type: String Value: "PVO", "SD"
sdi2output	SDI2 Output Mode Type: String Value: "PVO", "SD"

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values. For POST at least one of the arguments must be specified along with the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

As seen above, the possible URIs are seen. The SDI 1 and 2 Output Modes are still a part of audio. Place the specific URI (see above) into the CURL command and the user will be able to POST and GET the SDI 1 and 2 Output Modes.

POST Example

Example 1: Set Audio Device SDI1Output:

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/audio?sdi1output=PVO"
```

If successful, the return body will look like:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Example

Example 2: Retrieve the Audio Device SDI1Output setting

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/audio?sdi1output"
```

If successful, the return body will look like:

```
"decode": {
        "audio": {
            "sdi1output": "PVO"
        }
}
```

Audio ST302 Device Settings Command

Table 2.239 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/aud_302
Command Information	Allows set or get of audio ST302 decoder settings.
HTTP Method(s)	GET, POST
Access Type	Read, Write
Access Level	User, Admin
Syntax	GET "https://192.168.0.1/ws/v2/service_cfg/decode/aud_st302
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_cfg/decode/aud_st302? streamid=2”

URI Parameters (extension to the Command URL separated by /): N/A

URI Query/Set Arguments (possible fields and values preceded by ? and separated by &):

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values. For POST at least one of the arguments must be specified along with the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

Table 2.240 URI Query/Set Arguments

URI Argument	Description
device	ST302 Device ID Type: String Value: "1"
streamid	ST302 Stream ID Type: String Value: "1", "2", "3", "4"

POST Example

Example 1: Set Audio ST302 Device Stream ID

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/aud_st302?streamid=2"
```

If successful, the return body will look like:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Example

Example 2: Retrieve the Audio ST302 Device Stream ID

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/aud_302"
```

If successful, the return body will look like:

```
"decode": {
        "aud_st302": {
            "device": "1",
            "streamid": "2"
        }
    }
```

SDI Decode Configuration Command

Table 2.241 Command Details

Command Detail	Description
Command URL	SDI Audio: https://192.168.0.1/ws/v2/service_cfg/decode/sdiaudio SDI VANC: https://192.168.0.1/ws/v2/service_cfg/decode/sdivanc SDI VII: https://192.168.0.1/ws/v2/service_cfg/decode/sdivii
Command Information	Allows set or get of decode settings for SDI parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	SDI Audio: GET "https://192.168.0.1/ws/v2/service_cfg/decode/sdiaudio", or GET "https://192.168.0.1/ws/v2/service_ cfg/decode/sdiaudio?group=1&slot=1&sdiindex=1&decoder" SDI VANC (Vertical Ancillary Data): GET "https://192.168.0.1/ws/v2/service_cfg/decode/sdivanc", or GET "https://192.168.0.1/ws/v2/service_ cfg/decode/sdivanc?serviceid=AFD&sdiindex=1&enable" OR (for 4.50 and up) GET "https://192.168.0.1/ws/v2/service_ cfg/decode/sdivanc?serviceid=AFD&sdiindex=1&enable&st2308sel SDI VII: GET "https://192.168.0.1/ws/v2/service_cfg/decode/sdivii", or GET "https://192.168.0.1/ws/v2/service_cfg/decode/sdivii?sdiindex=1"
POST Syntax Setting parameters using command line	SDI Audio: POST "https://192.168.0.1/ws/v2/service_cfg/decode/sdiaudio?

Command Detail	Description
arguments is limited to maximum of 26 arguments after the ?.	group=1&slot=1/2&sdiindex=1&decoder=PCM%20%Audio%201&chan=Left/Right" SDI VANC (Vertical Ancillary Data): POST "https://192.168.0.1/ws/v2/service_ cfg/decode/sdivanc?serviceid=AFD&enable=Yes" OR POST "https://192.168.0.1/ws/v2/service_ cfg/decode/sdivanc?serviceid=AFD&enable=Yes&st2308sel=Yes" SDI VII: POST "https://192.168.0.1/ws/v2/service_ cfg/decode/sdivii?sdiindex=1&enable=Enabled"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.242 SDIAUDIO URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
group (key)	Audio Group Type: Integer Value: 1..4
slot (key)	Group Slot Type: Integer Value: 1/2, 3/4
sdiindex (key)	Audio SDI Device Instance Type: Integer Value: 1..2
decoder	Audio Decoder Type: Integer Values: "PCM Audio 1", "PCM Audio 2", "PCM Audio 3", "PCM Audio 4", "Compressed Audio 1", "Compressed Audio 2", "Compressed Audio 3", "Compressed Audio 4", "Off" or exist (for GET output filtering) Type: Integer Value: 1..4
chan	Audio Channel

URI Argument	Description
	Type: String Value: "Left/Right" or "Right/Left", or exist (for GET output filtering)

Note All of the URI Arguments below apply to both GET and POST except where indicated. For GET, the URI arguments do not need any values, except the key arguments and values which must be specified. For POST the key arguments and values must be specified followed by any of the other URI arguments and the associated value the user would like to set. The GET result output field names re-use the argument names and possible values. This note applies to all of Audio URI Arguments, VANC URI Arguments, and VII URI Arguments.

Table 2.243 SDIVANC URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

SDI VANC URI Argument	Description
serviceid (key)	VANC Service Identifier. Type: String Value: "EIA-708", "AFD", "DPI", "SMPTE-2031", "SDP-OP47", "MULTI-OP47"
sdiindex (key)	Audio SDI Device Instance. Type: Integer Value: 1..2
enable	Enable or Disable VANC Service. Type: String Value: "Yes", "No", or exist (for GET output filtering)
maxoffset	Maximum Line Offset Allowed. Type: Integer Value: 2..18, or exist (for GET output filtering)
st2308sel	If set to “Yes”, SMPTE-2038 data will be used to override the specified VANC service normal source. Type: String Value: “No” (default), “Yes”

Table 2.244 SDIVII URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

SDI VII Argument	Description
sdiindex (key)	VII SDI Device Instance Type: Integer Value: 1..2
enable	**Specify for POST operation only** Enable or Disable VII Service Type: String Value: "Enabled", "Disabled"

Note The possible URI for both POST and GET for sdivanc, sdiaudio, and sdivii are seen above. You must add these URIs into the CURL command with a valid token and users will be able to POST and GET. Refer to the POST and GET syntax examples in the Command Details section above for examples on how to do GET and POST with CURL commands.

HDMI Decode Configuration Command

Table 2.245 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/hdmi
Command Information	Allows set or get of decode settings for HDMI parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All HDMI Video Settings for specified HDMI instance: GET "https://192.168.0.1/ws/v2/service_cfg/decode/hdmi?hdmiindex=1" Single HDMI Video Setting for specified HDMI instance: GET "https://192.168.0.1/ws/v2/service_cfg/decode/hdmi?hdmiindex=1&colorspace"
POST Syntax Setting parameters using command line arguments is limited to maximum of 26 arguments after the ?.	Single HDMI Video setting: POST "https://192.168.0.1/ws/v2/service_cfg/decode/hdmi?hdmiindex=1&colorspace=YCbCr420" Multiple HDMI Video settings:

Command Detail	Description
	POST "https://192.168.0.1/ws/v2/service_cfg/decode/hdmi?hdmiindex=1&colorspace=YCbCr422&colorrange=Full&eotf=HDR"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.246 HDMI URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
hdmiindex (key)	HDMI Device Instance Type: Integer Value: 1..2
colorspace	HDMI Color Space Type: String Value: "Auto", "YCbCr420", "YCbCr444", "YCbCr422", "RGB"
colordepth	HDMI Color Depth Type: String Value: "Auto", "16bit", "12bit", "10bit", "8bit"
colorrange	HDMI Color Range Type: String Value: "Auto", "Full", "Limited"
matrixcoeff	HDMI Matrix Coefficients Type: String Value: "Auto", "ITU-2020-CL", "ITU-2020-NCL", "ITU-470", "XvYCC-709", "XvYCC-601", "ITU-709", "SMPTE-170M"
Eotf	HDMI EOTF (Electro-Optical Transfer Function) Type: String Value: "Auto", "SMPTE-2084", "HDR", “SDR”
displaypref	HDMI Display Preference Type: String Value: "Auto", "SDI1", "SDI2", "HDMI"
forceedid	HDMI Force EDID selector Type: String

URI Argument	Description
	Value: "Yes", "No"

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values, except ‘hdmiindex’ which must be specified. For POST ‘hdmiindex’ must be specified followed by any of the below URI arguments and the associated value the user would like to set. The GET result output field names re-use the argument names and possible values.

Advanced Audio Decode Configuration Command

Table 2.247 Command Details

Command Detail	Description
Command URL	Advanced Audio On/Off Control: https://192.168.0.1/ws/v2/service_cfg/decode/advaudio_control Advance Audio Settings: HDMI: https://192.168.0.1/ws/v2/service_cfg/decode/advaudio_hdmi Advanced Audio Settings: SDI: https://192.168.0.1/ws/v2/service_cfg/decode/advaudio_sdi
Command Information	Allows set or get of decode settings for advanced audio parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	**Advanced Audio Control** GET "https://192.168.0.1/ws/v2/service_cfg/decode/ advaudio_ control" **HDMI Advanced Audio Settings** All HDMI Audio Settings: GET "https://192.168.0.1/ws/v2/service_cfg/decode/advaudio_ hdmi" HDMI Audio Setting for specified HDMI instance/pair/slot: GET "https://192.168.0.1/ws/v2/service_cfg/decode/ advaudio_ hdmi?hdmiindex=1&pair=1&slot=1&decoder=2" **SDI Advanced Audio Settings**

Command Detail	Description
	All SDI Audio Settings: GET "https://192.168.0.1/ws/v2/service_cfg/decode/advaudio_sdi" Audio Setting for specified SDI instance/group/slot: GET "https://192.168.0.1/ws/v2/service_cfg/decode/ advaudio_ sdi?sdiindex=1&group=1&slot=1"
POST Syntax Setting parameters using command line arguments is limited to maximum of 26 arguments after the ?.	Advanced Audio Control: POST "https://192.168.0.1/ws/v2/service_cfg/decode/ advaudio_ control?enable=Yes" HDMI Advanced Audio Settings: POST "https://192.168.0.1/ws/v2/service_cfg/decode/ advaudio_ hdmi?hdmiindex=1&pair=1&slot=1&decoder=2& chan=Downmix%20Right" SDI Advanced Audio Settings: POST "https://192.168.0.1/ws/v2/service_cfg/decode/ advaudio_ sdi?sdiindex=1&group=1&slot=1& input=PCM%20Pair%201&chan=Right/Left"

URI Parameters (extension to the Command URL separated by /): N/A

Note All of the URI Arguments listed for HDMI Settings apply to both GET and POST. For GET, the URI arguments do not need any values, except ‘hdmiindex’ which must be specified. For POST all keys must be specified followed by any of the below non-key URI arguments and the associated value the user would like to set. The GET result output field names re-use the argument names and possible values.

Table 2.248 Advanced Audio Control URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

Advanced Audio Control URI Argument	Description
enable	Advanced Audio Control Type: String Value: "Yes", "No"

Table 2.249 HDMI Settings URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

HDMI Settings URI Argument	Description
hdmiindex (key)	HDMI Device Instance Type: Integer Value: 1..2
pair (key)	PCM Pair Type: Integer Value: 1..4
slot (key)	PCM Pair Slot Type: String Value: "Left", "Right"
decoder	Audio Decoder Type: String Value: "1", "2", "3", "4", "Off"
chan	Audio Channel Type: String Value: "Downmix Left" or "Downmix Right" **Note** The space character must be replaced with '%20' in POST command.

Note All of the URI Arguments listed for VBI Settings apply to both GET and POST. For GET, the URI arguments do not need any values. For POST the arguments in the following table need to be set with a value. The GET result output field names re-use the argument names and possible values.

Table 2.250 SDI Settings URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

SDI Settings URI Argument	Description
sdiindex (key)	Audio SDI Device Instance Type: Integer Value: 1..2
group (key)	Audio Group

SDI Settings URI Argument	Description
	Type: Integer Value: 1..4
slot (key)	Group Slot Type: String Value: "1/2" or "3/4"
input	Audio Input Type: String Value: "PCM Pair 1", "PCM Pair 2", "PCM Pair 3", "PCM Pair 4", "Compressed Audio 1", "Compressed Audio 2", "Compressed Audio 3", "Compressed Audio 4", "Off" **Note** The space character must be replaced with '%20' in POST command.
chan	Audio Channel Order Type: String Value: "Left/Right" or "Right/Left"

VBI Configuration Command

Table 2.251 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/decode/vbi
Command Information	Allows set or get of decode settings for VBI parameters.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All VBI Settings: GET "https://192.168.0.1/ws/v2/service_cfg/decode/hdmi?hdmiindex=1" Single VBI Setting: GET "https://192.168.0.1/ws/v2/service_cfg/decode/vbi? vitspal17"
POST Syntax Setting parameters using	POST "https://192.168.0.1/ws/v2/service_cfg/decode/vbi?wstlnstart=Standard&vitspal17=Disable"

Command Detail	Description
command line arguments is limited to maximum of 26 arguments after the ?.	

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.252 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
wstinstart	WST Line Start Type: String Value: "Standard", "Early"
vitspal17	Enable VITS PAL Line 17 Type: String Value: "Enable", "Disable"
vitspal18	Enable VITS PAL Line 18 Type: String Value: "Enable", "Disable"
vitspal330	Enable VITS PAL Line 330 Type: String Value: "Enable", "Disable"
vitspal331	Enable VITS PAL Line 331 Type: String Value: "Enable", "Disable"
vitcmode	VITC Mode Type: String Value: "Passthrough", "Suppress", ”Auto:Modify”, “Auto:Create”
vitctmcode	VITC Time Code Type: String Value: "LTC", "VITC", "BOTH"

URI Argument	Description
vitcdropfr	Enable VITC Drop Frame Type: String Value: "Disable", "Enable"

POST Examples:

Example 1: Set various VBI Setting Arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example enables the VITC Color Mode and the VITC Drop Frame.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/decode/vbi?vitcdropfr=Enable&vitcmode=Suppress"
```

If successful, the following is an example of the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Retrieve the full VBI Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/vbi"
```

If successful, the following is an example of the return body:

```
"decode": {
        "vbi": {
            "wstinstart": "Standard",
            "vitspal17": "Disable",
            "vitspal18": "Disable",
            "vitspal330": "Disable",
            "vitspal331": "Disable",
            "vitcmode": "Passthrough",
            "vitctmcode": "LTC",
            "vitcdropfr": "Enable"
        }
    }
```

Example 3: Retrieve specific VBI Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example will retrieve the VITC Color Mode and the VITC Drop Frame.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/decode/vbi?vitcdropfr"
```

If successful, the following is an example of the return body:

```
"decode": {
        "vbi": {
            "vitcdropfr": "Enable"
        }
    }
```

CA (Conditonal Access) Configuration Commands

This section lists the conditional access configuration commands.

BISS CA Configuration Command

Table 2.253 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/ca/biss
Command Information	Allows set or get of BISS Conditional Access configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All BISS Settings: GET "https://192.168.0.1/ws/v2/service_cfg/ca/biss" Single BISS Setting:

Command Detail	Description
	GET "https://192.168.0.1/ws/v2/service_cfg/ca/biss?bissmode"
POST Syntax Setting parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST “https://192.168.0.1/ws/v2/service_cfg/ca/biss? bissmode=Mode%201"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.254 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
port (Key) Applies to multi-stream units only.	Input port to be configured. Type: Integer Values: 1 .. 16 (NFE), 1 .. 2 (ASI), 2 (MOIP)
stream (Key) (optional) Applies to multi-stream units only.	Multi Input Stream ID. If not specified, default value of 1 is used. Type: Integer Values: 1..6 (RF), 2 (ASI), 1..32 (MOIP)
bissmode	BISS Mode of operation Type: String Value: "Mode 1", "Mode E"
mod1seswrd (Write Only)	Mode-1 Session Word Type: String Value: String with 12 characters
modeseswrd (Write Only)	Mode-E Encrypted Session Word Type: String Value: String with 16 characters
modeinjid (Write Only)	Mode-e Injected ID Type: String Value: String with 14 characters

Note For POST, the value for the argument seen below must be specified, but for GET there are no values for arguments. Also, the write only commands that can be used for POST are noted below.

POST Examples:

Example 1: Set various BISS Setting Arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example set the bissmode to Mode 1.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_cfg/ca/biss?&bissmode=Mode%201"
```

If successful, the following is an example of the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 2: Retrieve the PowerVu CA Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/ca/biss"
```

If successful, the following is an example of the return body:

```
  "ca": {
        "biss": {
            "bissmode": "Mode 1",
            "mod1seswrd": "************",
            "modeseswrd": "****************",
            "modeinjid": "**************"
        }
  }
```

Note The mod1seswrd, modeseswrd, and modeinjid are write-only, so their values will not be shown when trying to perform GET. Hence, only the bissmode will provide useful information from a BISS GET.

CI (Common Interface) Configuration Command

Table 2.255 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/ca/ci
Command Information	Allows set or get of Common Interface (CI) Conditional Access configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All CI CAM Settings: GET "https://192.168.0.1/ws/v2/service_cfg/ca/ci" Single CI CAM Setting: GET "https://192.168.0.1/ws/v2/service_cfg/ca/ci?sengdquery"
POST Syntax Setting parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_cfg/ca/ci?sendquery=Enable"

Table 2.256 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
sendquery	Send Query to CAM before enabling descrambling Type: String Value: "Enable", "Disable"
autoreset	Auto Reset CAM on failure Type: String Value: "Enable", "Disable"
listmgmt	CA PMT List management mechanism

URI Argument	Description
	Type: String Value: AddDel, UpdateAll
tsonetchk	Transport ID/ Original Network ID check Type: String Value: "Enable", "Disable"
tsid	Transport ID Type: Integer Value: 0..65535
onetid	Original Network Identifier Type: Integer Value: 0..65535
tsrouting	Transport Routing Type: String Value: "EntireTS", "ServicesOnly"

Note The arguments above are intended to be used with POST. When this command is used as a GET command these arguments generally serve as output field descriptions.

POST Examples:

Example 1: Set various CI Setting Arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example disables autoreset and sets tsid to 100.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_cfg/ca/ci?autoreset=Disable&tsid=100"
```

GET Examples:

Example 2: Retrieve the CI Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/ca/ci"
```

If successful, the following is an example of the return body:

```
{
    "ca": {
        "ci": {
            "sendquery": "Disable",
            "autoreset": "Disable",
            "listmgmt": "AddDel",
            "tsonetchk": "Disable",
            "tsid": "0",
            "onetid": "0",
            "tsrouting": "ServicesOnly"
        }
    }
}
```

PE CA Configuration Command

Table 2.257 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/ca/pe
Command Information	Allows set or get of Conditional Access Program Entry (PE) configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All PE CA Settings: GET "https://192.168.0.1/ws/v2/service_cfg/ca/pe" Single PE CA Setting: GET "https://192.168.0.1/ws/v2/service_cfg/ca/pe?peid=1&cidecrypt
POST Syntax Setting parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_cfg/pe?peid=1&cidecrypt=ON"

URI Parameters (extension to the Command URL separated by /): N/A

Note The arguments below are intended to be used with POST. When this command is used as a GET command these arguments generally serve as output field descriptions.

Table 2.258 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
peid	Program Entry ID Type: Integer Value: 1..32
cidecrypt	Program CAM Decrypt Mode Type: String Value: "OFF", "ON", "Comp"
cipe1slot	CAM Slot for Program Entry 1 Type: String Value: "TOP", "BOTTOM", "AUTO"
cislot	CAM Slot for Program Entries other the PE1 Type: String Value: "TOP", "BOTTOM"

POST Examples:

Example 1: Set various PE Setting Arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example sets Program Entry ID 1 to ON.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_cfg/ca/pe?peid=1&cidecrypt=ON"
```

GET Examples:

Example 2: Retrieve the PE Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/ca/pe"
```

If successful, the following is an example of the return body:

```
{
    "ca": {
        "pe": [
            {
                "peid": "1",
                "cidecrypt": "ON",
                "cislot": "TOP",
                "cipe1slot": "TOP"
            },
             …
            {
                "peid": "16",
                "cidecrypt": "ON",
                "cislot": "TOP",
                "cipe1slot": "AUTO"
            }
        ]
    }
}
```

CICOMP CA Configuration Command

Table 2.259 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/ca/cicomp
Command Information	Allows set or get of Conditional Access CI Component level configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All CICOMP CA Settings: GET "https://192.168.0.1/ws/v2/service_cfg/ca/cicomp" Single CICOMP CA Setting: GET "https://192.168.0.1/ws/v2/service_cfg/ca/cicomp?peid=1&cidecrypt"
POST Syntax Setting parameters using command	POST "https://192.168.0.1/ws/v2/service_cfg/ca/cicomp?idx=5&peid=1&cidecrypt=ON&oper=Add"

Command Detail	Description
line arguments is limited to maximum of 26 arguments after the ?.	

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.260 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
idx (Key) Mandatory for POST command	Index of Row in UIC Table Type: Integer Value: 1..64
peid	Program Entry ID Type: Integer Value: 1..32
mode	CI Component Mode Type: String Value: "PID", "STREAM"
pid	CI Component PID Type: Integer Value: 0..8192
strmcat	Stream Category Type: String Value: "VID", "AUD", "SUBT", "TTX", "USER"
strmval	User Stream Type Value Type: Integer Value: 0..255
strminst	User Stream Instance Type: Integer Value: 1..64
oper	Component operation Type: String

URI Argument	Description
	Value: "Add", "Delete", "Edit"

Note The arguments above are intended to be used with POST. When this command is used as a GET command these arguments generally serve as output field descriptions.

GET Examples:

Example 1: Retrieve all the CICOMP Settings

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/ca/cicomp"
```

If successful, the following is an example of the return body:

```
{
    "ca": {
        "cicomp": [
            {
                "idx": "1",
                "peid": "1",
                "mode": "PID",
                "pid": "8192",
                "strmcat": "USER",
                "strmval": "0",
                "strminst": "1",
                "oper": "Inactive"
            },
            …
            {
                "idx": "64",
                "peid": "1",
                "mode": "PID",
                "pid": "8192",
                "strmcat": "USER",
                "strmval": "0",
                "strminst": "1",
                "oper": "Inactive"
            }
        ]
    }
}
```

POST Examples:

Note POST requires previous knowledge of target row (idx – key) from GET command.

Example 2: Set various CICOMP Setting Arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example sets the mode for program entry 1 to STREAM.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_cfg/ca/cicomp?idx=5&peid=1&mode=STREAM"
```

Expected response if CA card is **not installed** in NCI slot:

```
{
    "response": {
        "code": "10",
        "result": "failure",
        "message": "CI Validation Failed - NCI Required"
    }
}
```

Expected response if CA card is **installed** in NCI slot:

```
{
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
}
```

Example 3: Retrieve the CICOMP Settings for a specific UIC Table row (idx)

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

Here is an example for row idx=5:

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/ca/cicomp?idx=5"
```

If successful, the return body will look like:

```
{
    "ca": {
        "cicomp": {
            "idx": "5",
            "peid": "1",
            "mode": "STREAM",
            "pid": "8192",
            "strmcat": "USER",
            "strmval": "0",
            "strminst": "1",
            "oper": "Inactive"
        }
    }
}
```

CIMMI CA Configuration Command

Table 2.261 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/ca/cimmi
Command Information	Allows commands to be sent to Conditional Access CI Man-Machine Interface (MMI).
HTTP Methods	POST
Access Type	Write
Access Level	User, Admin
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/ca/cimmi? cislot=TOP&cmd=reset"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.262 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
cislot (Key) Mandatory for POST command	CI Slot Type: String Value: "TOP", "BOTTOM"
cmd	Command to be sent Type: String Value: "reset"

POST Examples:

Example 1: Reset the top CI slot

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example sets the mode for program entry 1 to STREAM.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID:$token" -k
"https://192.168.0.1/ws/v2/service_cfg/ca/cimmi?cislot=TOP&cmd=reset"
```

Expected response:

```
{
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
}
```

Transcoding Configuration Commands

This section lists the transcoding configuration commands.

HD Transcode Configuration Command

Table 2.263 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/transcode/hd
Command Information	Allows set or get of HD transcode configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All HD transcode Settings: GET "https://192.168.0.1/ws/v2/service_cfg/transcode/hd" Single HD transcode Setting: GET "https://192.168.0.1/ws/v2/service_cfg/transcode/hd?vres"
POST Syntax Setting parameters using command	POST "https://192.168.0.1/ws/v2/service_cfg/transcode/hd?vres=Auto"

Command Detail	Description
line arguments is limited to maximum of 26 arguments after the ?.	

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.264 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
index (Key)	Transcoder to be configured Type: Integer Values: 1..N (number of transcoders depends on model and installed options)
vres	Select the video format of the transcoded output Type: String Values: "MPEG2 Auto", "MPEG2 HD Output", "MPEG2 SD Output", "H.264 Auto", "H.264 HD Output", "H.264 SD Output", "No Xcode"
hres	Select the output horizontal resolution for HD video output Type: String Values: "Full", "3/4"
bitratemode	Select whether the output bit rate is constant or variable for HD video output Type: String Values: "CBR", "VBR"
bitrate	Select the output bit rate for HD output video Type: Float Values: 10.0..25.0
pulldown32	Select whether 3:2 pulldown is enabled or not Type: String Values: "Disabled", "Enabled"
gop	Select whether to manually select the Group of Pictures settings Type: String Values: "I Frame Sync", "Fixed", "Dynamic"
fixedgopm	If the HD GOP Control is under Fixed GOP, select the M values Type: Integer

URI Argument	Description
	Values: 1…8 (inclusive in increments of 1)
fixedgopn	If the HD GOP Control is under Fixed GOP, select the N values Type: Integer Values: 0…60 (inclusive in increments of 1)

Note All of the URI Arguments in the table below apply to both GET and POST. For GET, the URI arguments do not need any values, except the index which must be specified. For POST, the index must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

POST Examples:

Example 1: Changing one HD Transcoding Parameter

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example changes the bitratemode of transcoder 5 to CBR.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/transcode/hd?index=5&bitratemode=CBR"
```

If successful, the following is an example of the return body:

```
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 2: Changing multiple HD Transcoding Parameters

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example changes several HD transcoder 5 parameters. It will set the bitratemode to CBR, bitrate to 18, and pulldown32 to Disabled.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/transcode?/hd?index=5&bitratemode=CBR&bitrate=18&pulldown32=Disabled"
```

If successful, the following is an example of the return body:

```
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 3: GET the full HD transcoder values for a specific transcoder

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves all of the HD transcoder settings for transcoder 5.

Note The user must specify the index in the URI.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/transcode/hd?index=5"
```

If successful, the following is an example of the return body:

```
  "transcode": {
        "hd": {
            "index": "5",
            "vres": "HD Output",
            "hres": "Full",
            "bitratemode": "CBR",
            "bitrate": "16.0",
            "gop": "Fixed",
            "fixedgopn": "15",
            "fixedgopm": "2",
            "pulldown32": "Disabled
        }
    }
```

Example 4: GET a specific HD transcode value for a specific transcoder

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves one specific parameter value from the ASI tuning settings for ASI port 1.

Note The user must specify the port number in the URI followed by the URI argument the user wishes to retrieve. This example retrieves the bitrate from transcoder 5.

Note In GET URIs, only the index argument must contain a value for the transcoder, all other arguments do not contain values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/transcode/hd?index=5&bitrate"
```

If successful, the following example is the return body:

```
"transcode": {
        "hd": {
            "index": "5",
            "bitrate": "16.0"
        }
    }
```

Example 5: GET multiple HD transcoder values for a specific transcoder

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves multiple parameter values from the HD transcoder settings for transcoder 5.

Note The user must specify the port number in the URI followed by the URI arguments the user wishes to retrieve. This example retrieves the bitratemode, bitrate and pulldown32 values from transcoder 5.

Note In GET URIs, only the index argument must contain a value for the transcoder, all other arguments do not contain values.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/transcode/hd?index=5&bitratemode&bitrate&pulldown32"
```

If successful, the following is an example of the return body:

```
"transcode": {
        "hd": {
            "index": "5",
            "bitrate": "16.0",
            "bitratemode": "CBR",
            "pulldown32": "Disabled"
        }
    }
```

SD Transcode Configuration Command

Table 2.265 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/transcode/sd
Command Information	Allows set or get of SD transcode configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All SD transcode Settings: GET "https://192.168.0.1/ws/v2/service_cfg/transcode/sd" Single SD transcode Setting: GET "https://192.168.0.1/ws/v2/service_cfg/transcode/sd?vres"
POST Syntax Setting parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_cfg/transcode/sd? vres=Auto"

URI Parameters (extension to the Command URL separated by /): N/A

Note All of the URI Arguments in the table below apply to both GET and POST. For GET, the URI arguments do not need any values, except the index which must be specified. For POST, the index must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen in the following table.

Table 2.266 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
index (Key)	Transcoder to be configured Type: Integer Values: 1..N (number of transcoders depends on model and installed options)

URI Argument	Description
vres	Select the video format of the transcoded output Type: String Values: "Auto", "HD Output", "SD Output"
hres	Select the output horizontal resolution for HD video output Type: String Values: "Full", "3/4"
bitratemode	Select whether the output bit rate is constant or variable for HD video output Type: String Values: "CBR", "VBR"
bitrate	Select the output bit rate for HD output video Type: Float Values: 10.0..25.0
pulldown32	Select whether 3:2 pulldown is enabled or not Type: String Values: "Disabled", "Enabled"
aspectratio	Select the aspect ratio of the SD video output Type: String Values: "4:3", "16:9"
aspectconv	Select the desired aspect ratio conversion method for SD video output Type: String Values: "None", "Auto", "Auto AFD", "16:9 L/B", "4:3 P/B", "14:9", "4:3 CCO", "16:9 SCALE"
ccpkt1	Select which content to place in the first Closed Captioning Packet for SD video output Type: String Values: "None", "CEA 708", "SCTE-20"
ccpkt2	Select which content to place in the second Closed Captioning Packet for SD video output Type: String Values: "None", "CEA 708", "SCTE-20"
gop	Select whether to manually select the Group of Pictures settings

URI Argument	Description
	Type: String Values: "I Frame Sync", "Fixed", "Dynamic"
fixedgopm	If the HD GOP Control is under Fixed GOP, select the M values Type: Integer Values: 1…8 (inclusive in increments of 1)
fixedgopn	If the HD GOP Control is under Fixed GOP, select the N values Type: Integer Values: 0…60 (inclusive in increments of 1)

POST Examples:

Example 1: Changing one SD Transcoding Parameter

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes the bitratemode of transcoder 5 to CBR.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/sd/transcode?index=5&bitratemode=CBR"
```

If successful, the following is an example of the return body will be:

```
 "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 2: Changing multiple SD Transcoding Parameters

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example changes several HD transcoder 5 parameters. It will set the bitratemode to CBR, bitrate to 14 and, pulldown32 to Disabled.

```
curl -X POST -i -H "Accept: application/json" -H
"X-SESSION-ID: $token" -k "https://192.168.0.1/ws/v2/
service_
cfg/transcode/sd?index=5&bitratemode=CBR&bitrate=14&pulldown32=Disabled"
```

If successful, the following is an example of the return body:

```
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 3: GET the full SD transcoder values for a specific transcoder

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves all of the SD transcoder settings for transcoder 5.

Note The user must specify the index in the URI.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/transcode/sd?index=5"
```

If successful, the following is an example of the return body:

```
"transcode": {
        "sd": {
            "index": "5",
            "vres": "SD Output",
            "hres": "720",
            "bitratemode": "CBR",
            "bitrate": "4.0",
            "gop": "Fixed",
            "fixedgopn": "15",
            "fixedgopm": "2",
            "pulldown32": "Disabled",
            "aspectratio": "16:9",
            "aspectconv": "None",
            "ccpkt1": "CEA 708",
            "ccpkt2": "None"
        }
    }
```

Example 4: GET a specific SD transcode value for a specific transcoder

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is

192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves one specific parameter value from the ASI tuning settings for ASI port 1.

Note The user must specify the port number in the URI followed by the URI argument the user wishes to retrieve. This example retrieves the bitrate from transcoder 5.

Note In GET URIs, only the index argument must contain a value for the transcoder, all other arguments do not contain values.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/transcode/sd?index=5&bitrate"
```

If successful, the following is an example of the return body:

```
"transcode": {
        "sd": {
            "index": "5",
            "bitrate": "14.0"
        }
    }
```

Example 5: GET multiple SD transcoder values for a specific transcoder

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves multiple parameter values from the HD transcoder settings for transcoder 5.

Note The user must specify the index number in the URI followed by the URI arguments the user wishes to retrieve. The following example retrieves the bitratemode, bitrate and pulldown32 values from transcoder 5.

Note In GET URIs, only the index argument must contain a value for the transcoder, all other arguments do not contain values.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/transcode/sd?index=5&bitratemode&bitrate&pulldown32"
```

If successful, the following is an example of the return body:

```
"transcode": {
        "sd": {
            "index": "5",
            "bitrate": "14.0",
            "bitratemode": "CBR",
            "pulldown32": "Disabled"
        }
    }
```

SUBT (Subtitle) Transcode Configuration Command

Table 2.267 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/transcode/subt
Command Information	Allows set or get of subtitles transcode configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All SUBT transcode Settings GET "https://192.168.0.1/ws/v2/service_cfg/transcode/subt" Single SUBT transcode Setting GET "https://192.168.0.1/ws/v2/service_cfg/transcode/subt?opmode"
POST Syntax Setting parameters using command line arguments is limited to maximum of 26 arguments after the ?.	POST "https://192.168.0.1/ws/v2/service_cfg/transcode/subt?index=1&opmode=Off"

URI Parameters (extension to the Command URL separated by /): N/A

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values, except the index which must be specified. For POST, the index must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

Table 2.268 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
index (Key)	Transcoder to be configured Type: Integer Values: 1..N (number of transcoders depends on model and installed options)
opmode	Operational Mode of Subtitle Type: String Value: "Off", "On", "Imitext", "DVB"
langmenu	Language Menu Options Type: String Value: "Language List", "Language Entry", "PMT Order"
pmtorder	PMT Order of Subtitle PID Type: String Values: "First", "Second", "Third", "Fourth", "Fifth", "Sixth", "Seventh", "Eighth"
langlist	Language List Type: String Value: 3 character language string
langentry	Language Entry Type: String Value: 3 character language string
imitext	imitext Position Type: String Values: "Standard", "Extended"
foregndcol	Foreground Color Type: String Values: "Auto", "Yellow", "White"
backgndcol	Foreground Color Type: String Values: "None", "Auto", "Shadow", "Opaque", "Semi"

POST Examples:

Example 1: Changing one SUBT Transcoding Parameter

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example changes the opmode of transcoder 5 to Imitext.

```
curl -X POST -i -H "Accept: application/json" -H
"X-SESSION-ID: $token" -k "https://192.168.0.1/ws/v2/service_cfg/transcode/
subt?index=5&opmode=Imitext"
```

If successful, the following is an example of the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Example 2: Changing multiple SUBT Transcoding Parameters

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example changes several SUBT transcoder 5 parameters. It will set the opmode to "Imitext", langmenu to "PMT Order" and pmtorder to "Second".

```
curl -X POST -i -H "Accept: application/json" -H
"X-SESSION-ID: $token" -k "https://192.168.0.1/ws/v2/service_cfg/transcode/
subt?index=5&opmode=Imitext&langmenu=PMT%20Order&pmtorder=Second"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples:

Example 3: GET the full SUBT transcoder values for a specific transcoder

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves all of the SUBT transcoder settings for transcoder 5.

Note The user must specify the index in the URI.

```
curl -X GET -i -H "Accept: application/json" -H
"X-SESSION-ID: $token" -k "https://192.168.0.1/ws/v2/service_cfg/transcode/
subt?index=5"
```

If successful, the following is an example of the return body:

```
{
    "transcode": {
        "subt": {
            "index": "5",
            "opmode": "Imitext",
            "langmenu": "PMT Order",
            "pmtorder": "Second",
            "langlist": "eng",
            "langentry": "eng",
            "imitext": "Standard",
            "foregndcol": "Auto",
            "backgndcol": "Auto"
        }
    }
}
```

Example 4: GET a specific SUBT transcode value for a specific transcoder

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves one specific parameter value from the ASI tuning settings for ASI port 1.

Note The user must always specify the port number in the URI followed by the URI argument the user wishes to retrieve. This example retrieves the langmenu from transcoder 5.

Note In GET URIs only, the index argument must contain a value for the transcoder, all other arguments so not contain values.

```
curl -X GET -i -H "Accept: application/json" -H
"X-SESSION-ID: $token" -k "https://192.168.0.1/ws/v2/service_cfg/transcode/
subt?index=5&langmenu"
```

If successful, the following is an example of the return body:

```
{
"transcode": {
        "subt": {
            "index": "5",
            "langmenu": "PMT Order"
        }
       }
}
```

Example 5: GET multiple SUBT transcoder values for a specific transcoder

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves multiple parameter values from the HD transcoder settings for transcoder 5.

Note The user must always specify the port number in the URI followed by the URI arguments the user wishes to retrieve. This example retrieves the langmenu, langentry, and foregndcol values from transcoder 5.

Note In GET URIs only, the index argument must contain a value for the transcoder, all other arguments do not contain values.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/transcode/subt?index=5&langmenu&langentry&foregndcol"
```

If successful, the return body will be:

```
{
    "transcode": {
        "subt": {
            "index": "5",
            "foregndcol": "Auto",
            "langentry": "eng",
            "langmenu": "PMT Order"
        }
    }
}
```

User Configuration Command

Table 2.269 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/user
Command Information	Allows set or get of user configuration.

Command Detail	Description
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All user configuration settings: GET https://192.168.0.1/ws/v2/service_cfg/user Single user configuration setting: GET "https://192.168.0.1/ws/v2/service_cfg/user?lossofinput"
POST Syntax	POST "https://192.168.0.1/ws/v2/service_cfg/user? lossofinput=No%20Output"

URI Parameters: (extension to the Command URL separated by /): N/A

Table 2.270 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
lossofinput	Select whether outputs should transmit black or not transmit any data when there is a loss of input. **Note** The spaces in string values in request payloads must be URL encoded. Type: String Value: "Black Output", "No Output"
outonalarm	Select the action to take on video output, in case of an alarm condition affect that output. **Note** The spaces in string values in request payloads must be URL encoded. Type: String Value: "No Action ", "Mute Outputs", "Freeze Frame"

Note All of the URI Arguments below apply to both GET and POST. In-depth CURL examples can be seen below following the table.

POST Examples:

Example 1: Changing one user parameter

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example changes the lossofinput parameter to "No Output".

```
curl -k -X POST -H "X-SESSION-ID:$token"
"https://192.168.0.1/ws/v2/service_cfg/user?lossofinput=No%20Output"
```

If successful, the following is an example of the return body:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<response><code>10</code><result>Successfully updated item loiaction
with value No Output</result><message></message></response>
```

The following example changes the outonalarm parameter to "Mute Outputs":

```
curl -k -X POST -H "X-SESSION-ID:$token"
"https://192.168.0.1/ws/v2/service_cfg/user?outonalarm=Mute%20Outputs"
```

If successful, the following is an example of the return body:

```
<?xml version="1.0" encoding="ISO-8859-1" ?><response><code>10</code><result>
Successfully updated item OutOnAlarm with value Mute Outputs</result>
<message></message></response>
```

GET Examples:

Example 2: GET the full set of user parameters

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves all of the user settings.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/user"
```

If successful, the following is an example of the return body:

```
"user": {
        "lossofinput": "No Output"
        "outonalarm" : "Mute Outputs"
    }
```

Example 3: GET a specific user parameter

The following example assumes that the user has successfully logged onto the unit, received the session id and set it to the variable token. In addition, it is assumed that the IP of the unit is

192.168.0.1. You must change the IP to the specific unit IP in use. The following example retrieves one specific parameter value from the user settings.

```
curl -X GET -i -H "Accept: application/json" -H
"X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/user?lossofinput"
```

If successful, the following is an example of the return body:

```
"user": {
        "lossofinput": "No Output"
    }
```

Disaster Recovery (D/R) Configuration Commands

This section describes the disaster recovery commands.

Disaster Recovery (D/R) Global Configuration Command

Table 2.271 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/dr/global
Command Information	Allows to set or get of D/R global configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All D/R global Settings GET “https://192.168.0.1/ws/v2/service_cfg/dr/global” Single D/R global Setting, for example, GET “https://192.168.0.1/ws/v2/service_cfg/dr/global?enable”
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	For example, POST “https://192.168.0.1/ws/v2/service_cfg/dr/global?enable=No”

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.272 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
enable	Enable or Disable D/R Feature

URI Argument	Description
	Type: String Value: “Yes”, “No”
siglock	Amount of time to wait before declaring signal is locked Type: Integer Value: 5 … 255
siglost	Amount of time to wait before declaring signal is lost Type: Integer Value: 5 … 2160000
verify	Amount of time allowed for verification to complete Type: Integer Value: 10 … 255
cfgsource	D/R configuration source Type: String Value: “Local”

Note For POST, the value for the argument seen below must be specified, but for GET, there are no values for arguments.

POST Examples

Example 1: Set various D/R global Setting Arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example set the commode to Std.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/global?&siglock=40"
```

If successful, the return body will be:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples

Example 2: Retrieve the full D/R global Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
Curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/global"
```

If successful, the return body will be:

```
"dr": {
        "global": {
            "enable": "Yes",
            "siglock": "40",
            "siglost": "120",
            "verify": "60",
            "cfgsource": "Local"
        }
    }
```

Note The “profile” parameter is only for display, configuring the parameter is not currently supported.

Disaster Recovery (D/R) Backups (Profiles) Global Configuration Command

Table 2.273 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/dr/backupsglb
Command Information	Allows to set or get D/R Profiles global configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All D/R backup tuning settings GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupsglb" Single D/R backup tuning sseetings, for example, GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupsglb?recid=1"

Command Detail	Description
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	For example, POST "https://192.168.0.1/ws/v2/service_cfg/dr/backupsglb?recid=1&inputname=MOIP"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.274 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
recid (Key)	Record index of D/R profile configuration Type: Integer Values: 1 … 96
name	D/R Profile name Type: String
inputtype	D/R Profile input type Type: String Values: “NONE”, “RF”, “MOIP”, “ZIXI”, “ABR”, “ASI”
csiid	D/R Profile ID in the corresponding D/R Profile Tuning storage Type: Integer Values: range depends on input type
rowstatus	D/R Profile record status Type: String Values: "Active", "Inactive"

POST Examples:

Example 1: Set various D/R global profile arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example set the commode to Std.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/dr/backupsglb?&recid=11&inputtype=MOIP"
```

If successful, the following is the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples

Example 2: Retrieve D/R profile 11 status

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/backupsglb?recid=11"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupsglb": {
            "recid": "11",
            "name": "",
            "inputtype": "MOIP",
            "csiid": "0",
            "rowstatus": "Inactive"
        }
    }
}
```

Disaster Recovery (D/R) RF Backup Configuration Command

Note This is supported on single-stream units only.

Table 2.275 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/dr/backupsrf
Command Information	Allows to set or get D/R RF backup (profile) configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All D/R RF backups tuning settings

Command Detail	Description
	GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupsrf" Single D/R RF backup tuning ssetting, for example, GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupsrf?csiid=1"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	For example, POST "https://192.168.0.1/ws/v2/service_cfg/dr/backupsrf?csiid=1&pol=Vertical"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.276 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R RF backup ID Type: Integer Values: 1 … 16
csiidx (Key)	D/R RF backup IDX Type: Integer Values: 0 … 16 **Note** This is not supported in the current release.
inputname	D/R RF backup input name Type: String Values: "RF1", "RF2", "RF3", "RF4"
netid	D/R RF backup network id Type: Integer Values: 0 ~ 65535
streamid	D/R RF backup Stream ID Type: Integer Values: 1..6
mod	D/R RF backup modulation system

URI Argument	Description
	Type: String Values: for example, "QPSK”, DVB-S"
dnlkfreq	D/R RF backup frequency, in GHz Type: Float Values: 0.0 … 15.0
symrate	D/R RF backup symbol rate Type: Float Values: 1.0 ... 45.0
fec	D/R RF backup FEC mode Type: String Values: "Auto", "1/2", "2/3", "3/4", "5/6", "7/8", "8/9"
orbpos	D/R RF backup Orbital Position, in deg Type: Float Values: 0.0 ... 360.0
ewflag	D/R RF backup West/East Flag Type: String Values: "N/A", "East", "West"
pol	D/R RF backup Orbital Polarization Type: String Values: "Horizontal", "Vertical", "Circ_l", "Circ_r", "Auto"
rolloff	D/R RF backup roll-off Type: String Values: ".20", ".25", ".35", "Auto"
rowstatus	D/R RF backup record status Type: String Values: "Yes", "No"

POST Examples

Example 1: Set various D/R backup transport Arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example set the commode to Std.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/dr/tuning?&csirec=1&rowstatus=Active&input=RF2
&dnlkfreq=3.53&symrate=30"
```

If successful, the following is the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples

Example 2: Retrieve the full D/R backup Settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/tuning"
```

If successful, the following is the return body:

```
"dr": {
        "tuning": [
            {
                "csirec": "1",
                "fec": "Auto",
                "dnlkfreq": "3.53",
                "input": "RF2",
                "mod": "DVB-S",
                "netid": "1",
                "symrate": "30.0",
                "rolloff": ".35",
                "rowstatus": "Active"
            },
            {
                "csirec": "2",
                "fec": "Auto",
                "dnlkfreq": "3.449",
                "input": "RF1",
                "mod": "DVB-S",
                "netid": "1",
                "symrate": "28.3465",
                "rolloff": ".35",
                "rowstatus": "Inactive"
            },
            {
                "csirec": "3",
                "fec": "Auto",
                "dnlkfreq": "3.449",
                "input": "RF1",
                "mod": "DVB-S",
                "netid": "1",
                "symrate": "28.3465",
                "rolloff": ".35",
                "rowstatus": "Inactive"
            }
        ]
    }
```

Disaster Recovery (D/R) MOIP Backup Configuration Command

Note This is supported on multi-stream units only.

Table 2.277 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/dr/backupsmoip
Command Information	Allows to set or get D/R MOIP backup (profile) settings.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All D/R MOIP backups tuning settings GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupsmoip" Single D/R MOIP backup tuning setting, for example, GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupsmoip?csiid=1"
POST Syntax Setting Parameters using command line arguments is limited to maximum	For example, POST "https://192.168.0.1/ws/v2/service_cfg/dr/backupsmoip?csiid=1&dejlat=120"

Command Detail	Description
of 26 arguments after the ?.	

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.278 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R MOIP backup ID Type: Integer Values: 1 … 32
interface	D/R MOIP backup ETH interface Type: String Values: "Data1", "Data2", "Data3" (multi-stream units only), "Data4" (multi-stream units only)
usemcast	D/R MOIP backup ‘Is Multicast Destination’ flag Type: String Values: "Yes", "No”
mcastaddr	D/R MOIP backup destination IP address Type: IPv4 address in standard notation
fecmode	D/R MOIP backup FEC Mode Type: String Values: "None", "1D", "2D"
tsdestport	D/R MOIP backup TS destination UDP port Type: Integer Values: 1..65535 (some ports are reserved and cannot be used)
feccolport	D/R MOIP backup FEC Columns destination UDP port Type: Integer Values: 1..65535 (some ports are reserved and cannot be used)
fecrowport	D/R MOIP backup FEC Rows destination UDP port Type: Integer Values: 1..65535 (some ports are reserved and cannot be used)
dejalg	D/R MOIP backup de-jittering algorithm

URI Argument	Description
	Type: String Values: “VBR”, “CBR”
dejlat	D/R MOIP de-jittering buffer latency, in ms Type: Integer Values: 40..150
rowstatus	D/R MOIP backup record status Type: String Values: "Active", "Inactive"

POST Examples

Example 1: Set various D/R MOIP backup arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example set the commode to Std.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/backupsmoip?&csiid=1&dejlat=120"
```

If successful, the following is the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples

Example 2: Retrieve D/R MOIP backup 1 settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/sservice_cfg/dr/backupsmoip?csiid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupsmoip": {
            "csiid": "1",
            "interface": "Data1",
            "usemcast": "Yes",
            "mcastaddr": "225.1.1.1",
            "fecmode": "None",
            "tsdestport": "49152",
            "feccolport": "49154",
            "fecrowport": "49156",
            "dejalg": "VBR",
            "dejlat": "120",
            "rowstatus": "Inactive"
        }
    }
}
```

Disaster Recovery (D/R) Zixi Backup Configuration Command

Note This is supported on multi-stream units only.

Table 2.279 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/dr/backupszixi
Command Information	Allows to set or get D/R Zixi backup (profile) settings.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All D/R Zixi backups tuning ssettings GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupszixi" Single D/R Zixi backup tuning setting, for example, GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupszixi?csiid=1"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	For example, POST "https://192.168.0.1/ws/v2/service_cfg/dr/backupszixi?csiid=1&interface=Data1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.280 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R Zixi backup ID Type: Integer Values: 1 … 4
netid	D/R Zixi backup network id Type: Integer Values: 0 ~ 65535
type	D/R Zixi backup type Type: String Values: the only supported option is “Pull”
interface	D/R MOIP backup ETH interface Type: String Values: “Mgmt”, "Data1", "Data2", "Data3" (multi-stream unit only), "Data4" (multi-stream unit only)
source	D/R Zixi backup source Type: String Values: up to 200 characters
sid	D/R Zixi backup Stream ID (name) Type: String Values: up to 30 characters
port	D/R Zixi backup UDP port Type: Integer Values: 1..65535 (some ports are reserved and cannot be used)
pw	D/R Zixi stream password Type: String Values: up to 127 characters
latmode	D/R Zixi backup latency mode Type: String Values: “Static”, “Increasing”, “Dynamic”
latency	D/R Zixi backup latency, in ms

URI Argument	Description
	Type: Integer Values: 0..10000
fecmaxjitr	D/R Zixi backup FEC maximum jitter, in ms Type: Integer Values: 0 .. 10000
fecstufnul	D/R Zixi backup “FEC Stuffing NULL Packets Enabled” flag Type: String Values: “Yes”, “No”
decrkey	D/R Zixi backup Decryption Key Type: String Range: up to 64 characters
fecmode	D/R Zixi backup FEC mode Type: String Values: “Off”, “On”, “Adaptive”
fecovhd	D/R Zixi backup FEC Overhead Type: Integer Values: 0 to 10000
fecblkms	D/R Zixi backup FEC block size, in ms Type: Integer Values: 0 to 10000
fecconawar	D/R Zixi backup FEC Content Aware Type: String Values: “Yes”, “No”
rowstatus	D/R Zixi backup record status Type: String Values: "Active", "Inactive"

POST Examples:

Example 1: Set various D/R Zixi backup arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example set the commode to Std.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/dr/backupszixi?&csiid=1&interface=Data1"
```

If successful, the following is the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples

Example 2: Retrieve D/R Zixi backup 1 settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/backupszixi?csiid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupszixi": {
            "csiid": "1",
            "netid": "1",
            "type": "Pull",
            "interface": "Data1",
            "source": "",
            "sid": "",
            "port": "0",
            "latmode": "Static",
            "latency": "3000",
            "fecmaxjitr": "0",
            "fecstufnul": "No",
            "decrkey": "",
            "fecmode": "On",
            "fecovhd": "10",
            "fecblkms": "50",
            "fecconawar": "No",
            "rowstatus": "Inactive"
        }
    }
}
```

Disaster Recovery (D/R) ABR Backup Configuration Command

Note This is supported on multi-stream units only.

Table 2.281 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/dr/backupsabr
Command Information	Allows to set or get D/R ABR backup (profile) settings.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All D/R ABR backups tuning settings GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupsabr" Single D/R ABR backup tuning setting, for example, GET "https://192.168.0.1/ws/v2/service_ cfg/dr/backupsabr?csiid=1"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	For example, POST "https://192.168.0.1/ws/v2/service_ cfg/dr/backupsabr?csiid=1&interface=Data1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.282 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R ABR backup ID Type: Integer Values: 1 … 4

URI Argument	Description
interface	D/R ABR backup ETH interface Type: String Values: “Mngmt”, "Data1", "Data2", "Data3" (multi-stream units only), "Data4" (multi-stream units only)
url	D/R ABR stream URL Type: String Values: up to 200 characters
latency	D/R ABR backup Target Latency, in ms Type: Integer Values: 0..4294967295
propdelay	D/R ABR backup Propagation Delay, in ms Type: Integer Values: 0..4294967295
vod	D/R ABR Live VOD Type: String Values: "Yes”, “No"
rowstatus	D/R ABR backup record status Type: String Values: "Active", "Inactive"

POST Examples:

Example 1: Set various D/R ABR backup arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example set the commode to Std.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/dr/backupsabr?&csiid=1&interface=Data1"
```

If successful, the following is the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples

Example 1: Retrieve D/R ABR backup 1 status

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/backupsabr?csiid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupsabr": {
            "csiid": "1",
            "interface": "Data1",
            "url": "",
            "latency": "6000",
            "propdelay": "0",
            "vod": "No",
            "rowstatus": "Inactive"
        }
    }
}
```

Disaster Recovery (D/R) ASI Backup Configuration Command

Note This is supported on multi-stream units only.

Table 2.283 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/dr/backupsasi
Command Information	Allows to set or get D/R ASI backup (profile) settings.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin

Command Detail	Description
GET Syntax	All D/R ASI backups tuning settings GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupsasi" Single D/R ASI backup tuning settings, for example, GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupsasi?csiid=1"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	For example, POST "https://192.168.0.1/ws/v2/service_cfg/dr/backupsasi?csiid=1&port=1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.284 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R ASI backup ID Type: Integer Values: 1 (single-stream units only), 1..2 (multi-stream units only)
port	D/R ASI port ID Type: Integer Value: 1 (single-stream units only), 1..2 (multi-stream units only)
rowstatus	D/R ASI backup record status Type: String Values: "Active", "Inactive"

POST Examples:

Example 1: Set various D/R ASI backup arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example set the commode to Std.

```
curl -X POST -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/backupsasi?&csiid=1&port=1"
```

If successful, the following is the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples

Example 2: Retrieve D/R ASI backup 1 settings

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/backupsasi?csiid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupsasi": {
            "csiid": "1",
            "port": "1",
            "rowstatus": "Active"
        }
    }
}
```

Disaster Recovery (D/R) SRT Backup Configuration Command

Note This is supported on multi-stream units only.

Table 2.285 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/dr/backupssrt
Command Information	Allows to set or get D/R SRT backup (profile) settings.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All D/R SRT backups tuning settings GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupssrt"

Command Detail	Description
	Single D/R SRT backup tuning setting, for example, GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupssrt?csiid=1"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	For example, POST "https://192.168.0.1/ws/v2/service_cfg/dr/backupssrt?csiid=1&interface=Data1"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.286 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
csiid (Key)	D/R SRT backup ID Type: Integer Values: 1 … 4
netid	D/R SRT backup network ID Type: Integer Values: 0 ~ 65535
type	D/R SRT Connection Type Type : String Value : "Listener", "Caller", "Rendezvous"
interface	D/R MOIP backup ETH interface Type: String Values: "Mgmt", "Data1", "Data2", "Data3" (D9800 MS only), "Data4" (multi-stream units only)
remoteip	D/R SRT input source ip address Type : String Value : Up to 200 characters
remoteport	D/R SRT backup remote UDP port Type: Integer Values: 0..65535 (some ports are reserved and can’t be used)

URI Argument	Description
listenport	D/R SRT backup listen UDP port Type: Integer Values: 0..65535 (some ports are reserved and can’t be used)
rowstatus	D/R SRT backup record status Type: String Values: "Active", "Inactive"

POST Examples:

Example 1: Set various D/R SRT backup arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example set the commode to Std.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/dr/backupssrt?&csiid=1&interface=Data1"
```

If successful, the return body will look like:

```
"response": {
       "code": "10",
       "result": "success",
       "message": ""
}
```

GET Examples

Example 2: Retrieve D/R Zixi backup 1 settings

The following example assumes that the user has successfully logged onto the unit, received the session ID and set it to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/backupssrt?csiid=1"
```

If successful, the return body will look:

```
{
       "dr": {
               "backupssrt": {
               "csiid": "1",
               "netid": "1",
               "type": "Caller",
               "interface": "Mgmt",
               "remoteip": "",
               "remoteport": "0",
               "listenport": "0",
               "rowstatus": "Inactive"
       }
   }
}
```

Disaster Recovery (D/R) Backup Nodes (Search Paths) Configuration Command

Table 2.287 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/dr/backupnodes
Command Information	Allows to set or get D/R backup nodes settings.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All D/R backup nodes settings GET "https://192.168.0.1/ws/v2/service_cfg/dr/backupnodes" Single D/R backup nodes settings, for example, GET https://192.168.0.1/ws/v2/service_ cfg/dr/backupnodes?peid=PE1&recid=1"
POST Syntax Setting Parameters using command line arguments is limited to maximum of 26 arguments after the ?.	For example, POST "https://192.168.0.1/ws/v2/service_ cfg/dr/backupnodes?peid=PE1&recid=1&inputtype=MOIP"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.288 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
peid (Key)	D/R backup node Program Entry ID Type: String Values: "PE1" … "PE16" (single-stream units only), "PE1" … "PE32" (multi-stream units)
recid (Key)	D/R backup node index for specific PE Type: Integer Values: 1 … 3
csiid	D/R backup node profile ID in the input-dependent D/R backup table Type: Integer Values: range is input-dependent
name	D/R backup node profile name Type: String Values: up to 64 characters
inputtype	D/R backup input type Type: String Values: “NONE”, “RF”, “MOIP”, “ZIXI”, “ABR”, “ASI”, "SRT"
bkpchan	D/R backup node channel number Type: Integer Values: 0 … 65535
bkpchandisp	D/R backup node channel display name Type: String Values: up to 8 characters
rowstatus	On GET, returns D/R backup node status (Active/Inactive). A deleted row for a row will show “Inactive” in a GET response. A successfully added row for a node will show “Active” in a GET response. On POST, Add or Delete an instance of backup node (Add/Delete) Type: String Values: "Add" (R/W), "Delete" (R/W), "Active" (RO), "Inactive" (RO)

POST Examples:

Example 1: Add a profile and set various D/R backup node arguments

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use. This example set the commode to Std.

```
curl -X POST -i -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_
cfg/dr/backupnodes?&peid=PE1&recid=1&name=MyDrBackupPlan1&bkpchan=123&csiid=1
&rowstatus=Add”
```

If successful, the following is the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

GET Examples

Example 2: Retrieve the status of D/R backup node 1 on PE1

The following example assumes that the user has successfully logged onto the unit, received the session ID, and set the unit to the variable token. In addition, it is assumed that the IP of the unit is 192.168.0.1. You must change the IP to the specific unit IP in use.

```
curl -X GET -i  -H "Accept: application/json" -H "X-SESSION-ID: $token" -k
"https://192.168.0.1/ws/v2/service_cfg/dr/backupnodes?peid=PE1&recid=1"
```

If successful, the following is the return body:

```
{
    "dr": {
        "backupnodes": {
            "peid": "PE1",
            "recid": "1",
            "csiid": "1",
            "name": "MyDrBackupPlan1",
            "inputtype": "RF",
            "bkpchan": "123",
            "bkpchandisp": "123",
            "rowstatus": "Active"
        }
    }
}
```

Alarms and Warnings Configuration Commands

This section lists all the commands for alarm and warning settings.

Fault Settings Top Level Configuration Command

Table 2.289 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/faults
Command Information	Allows set or get of fault settings (Alarms and Warning) configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	Query all fault configuration settings GET "https://192.168.0.1/ws/v2/service_cfg/faults?js", or GET "https://192.168.0.1/ws/v2/service_cfg/faults/settings?js", or Query all alarm configuration settings only GET "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms?js" or Query all warning configuration settings only GET "https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings?js" or Single Alarm or Warning entry (by key): GET "https://192.168.0.1/ws/v2/service_cfg/faults? textid="10%20 1?js", or GET "https://192.168.0.1/ws/v2/service_cfg/faults/settings? textid="10%20 1?js" , or "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms? textid="10%201?js", or "https://192.168.0.1/ws/v2/service_ cfg/faults/settings/warnings? textid="910%201?js"
POST Syntax	Enable specific fault instance (with trap and relay as well): POST "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms? js&textid=10%201&enable=Yes&trap=Yes&relay=Yes" Disable specific fault instance (with trap and relay as well): POST "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms? js&textid=10%201&enable=No&trap=No&relay=No"

URI Parameters: (extension to the Command URL separated by /): N/A

Table 2.290 Command Control URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js (optional)	Format output using JSON standard. Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML.
session (optional)	Alternate method of setting Login Token session ID. Use either one of this method or -H "X-SESSION-ID: $token" (where $token represents value of session id from Login command response); do not use both methods simultaneously. Type: String.

Table 2.291 Output Field Descriptions and URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

Output Field	Description
textid (key)	Unique internal text ID of the fault. Note that this must be previously known and specified in the API call when targeting individual Alarm or Warning settings instance rows. When the fault classification can only have one instance, this will be a string representation of the enumerated integer id (FAULT_CLASS) for that classification. When the fault classification can have multiple instances, then the format of this field will be a string representation of two integers seperated by a space character (%20 in URL syntax). The first integer in the textid will be the enumerated id value for the fault classification (FAULT_CLASS) whereas the second integer is the instance value (INDX). The combined FAULT_CLASS and INDX (if any) value is used to generate the unique part of the fault name field value that distinguishes it from other faults. Type: String Format: Regex pattern is: ^(\d+\s){0,1}\d+$

Output Field	Description
name (read-only)	Fault name. Unique name of the fault (includes instance id if applicable). This string is padded with spaces to always fill the indicated maxLength. Type: String minLength: 23 (not including NULL terminator) maxLength: 23 (not including NULL terminator)
enable	Fault enable. Select whether this fault is enabled or not. Type: String Value: "Yes" or "No" or "No*"
relay	Relay enable. If this fault is enabled, select whether it will trigger the relay or not. Type: String Value: "Yes" or "No" or "No*"
trap	Trap enable. If this fault is enabled, select whether it will send SNMP trap messages or not. Type: String Value: "Yes" or "No" or "No*" In Release 4.75 or later, the following applies: - If the trap setting for this fault is set to "Yes" and is an Alarm type, and the device_ctl protocols (see Protocols Device Control Command, on page 486) setting for "faultalarm" is set to "Yes", then syslogs will be sent out at "crit" severity with the details of that trap and fault when it occurs. - If the trap setting for this fault is set to "Yes" and is a Warning type, and the device_ctl protocols (see Protocols Device Control Command, on page 486) setting for "faultwarn" is set to "Yes", then syslogs will be sent out at "err" severity with the details of that trap and fault when it occurs. - The SNMP trap details (including the OID values) are sent in one syslog and the fault status details will be sent out as another syslog (both at either syslog crit or err severity depending on the fault type, Alarm or Warning respectively).
delayctrl	Select whether to enable fault debouncing or not. Type: String Value: "Default", "Config", or "Global"
delay	Fault delay. Set the number of milliseconds a fault state must be present before setting the fault. Type: Integer

Output Field	Description
	Minimum: 0 Maximum: 600000
resetdelay	Fault reset delay. Set the number of milliseconds a fault state must be absent before clearing the fault. Type: Integer Minimum: 0 Maximum: 600000
actwarns	Number of active warnings Type: Integer Minimum: 0 Maximum: 4294967295
clrfaults	Number of cleared faults Type: Integer Minimum: 0 Maximum: 4294967295
mibpriority	SNMP MIB Priority of the fault. Lower number is higher priority Type: Integer Minimum: 0 Maximum: 1024
priority ("key" when used with "status" section only)	Priority of the fault. Lower number is higher priority Type: Integer Minimum: 0 Maximum: 1024

Note All indicated QueryString arguments are optional and read-only when used with the ws/v2/status/faults APIs. Indicating one or more Query Arguments without a value will filter output to only record items matching each argument. Indicating one or more Query Arguments with a value will filter output to only the record(s) with a match for the argument=value request. Arguments marked as "key" will always be returned in the indicated sections of the output schema.

XML Examples:

Input: (Read back all faults settings (Alarms and Warnings) in default XML format):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_
cfg/faults"
```

Or

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_
cfg/faults/settings"
```

Expected Output is the same for both cases (values for example purposes only):

```
<?xml version="1.0" encoding="UTF-8"?>
<faults>
  <settings>
    <warnings>
      <textid>105 1</textid>
      <name>PSI/SI Stream Err TS 1</name>
      <enable>Yes</enable>
      <relay>No</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>274</mibpriority>
      <priority>274</priority>
    </warnings>
    <warnings>
      <textid>115 1</textid>
      <name>NetworkID Mismatch TS 1</name>
      <enable>Yes</enable>
      <relay>No</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>315</mibpriority>
      <priority>315</priority>
    </warnings>
    <warnings>
      <textid>117 1</textid>
      <name>D/R No Search Path     </name>
      <enable>Yes</enable>
      <relay>No</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>348</mibpriority>
      <priority>348</priority>
    </warnings>
    <warnings>
      <textid>118 1</textid>
      <name>D/R Malformed DRT      </name>
      <enable>Yes</enable>
      <relay>No</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>349</mibpriority>
      <priority>349</priority>
    </warnings>
    <alarms>
// several data records removed to save room in documentation
    </alarms>
    <alarms>
      <textid>147 1</textid>
      <name>Audio 3&amp;4 Not Licensed </name>
      <enable>Yes</enable>
      <relay>Yes</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>415</mibpriority>
      <priority>415</priority>
    </alarms>
    <alarms>
// several data records removed to save room in documentation
    </alarms>
    <alarms>
      <textid>92 1</textid>
      <name>Xcode License Fault    </name>
      <enable>Yes</enable>
      <relay>Yes</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>261</mibpriority>
      <priority>261</priority>
    </alarms>
  </settings>
</faults>
```

Converting XML Escaped Character for use in QueryString Filters:

Note Some of the values in name and details fields in output XML may contain XML escape sequences for characters that are normally not allowed in isolation in XML data. When using these strings as inputs for QueryParameter filtering, in addition to spaces requiring replacement with %20 (i.e. Hexadecimal 20), any XML escape sequence character needs to be replaced with a URL format % code for that character. For example, an ampersand (&) character (as per the "Audio 3&amp;4 Not Licensed" value in the example output above) would be replaced with %26 (for example, Hexadecimal 36).

Examples using QueryString filters:

Input: (Read back faults settings for only "Xcode License Fault" in default XML format):

```
curl -k "https://192.168.0.1/ws/v2/status/faults/settings?
session=$token&name=Xcode%20License%20Fault%20%20%20%20"
```

Expected result:

```
<?xml version="1.0" encoding="ISO-8859-1" ?><faults><settings><textid>92
1</textid><name>Xcode License Fault    </name></settings></faults>
```

Input: (Read back faults settings for only "Audio 3 & Audio 4 Alarm" in default XML format):

```
curl -k
"https://192.168.0.1/ws/v2/status/faults/settings?session=$token&name=Audio%2
03%264%20Not%20Licensed%20"
```

Expected result:

```
<?xml version="1.0" encoding="ISO-8859-1" ?><faults><settings>
<textid>147 1</textid><name>Audio 3&amp;4 Not Licensed
</name></settings></faults>
```

Input: (Read back all faults settings (Alarms and Warnings) in default XML format):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_
cfg/faults"
```

Or

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_
cfg/faults/settings"
```

JSON Examples:

Input: (Read back all faults settings (Alarms and Warnings) in default XML format):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_
cfg/faults?js"
```

Or

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/service_
cfg/faults/settings?js"
```

Expected Output is the same for both cases (values for example purposes only):

```
{
    "faults": {
        "settings": {
            "warnings": [
                {
                    "textid": "105 1",
                    "name": "PSI/SI Stream Err TS 1",
                    "enable": "Yes",
                    "relay": "No",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "274",
                    "priority": "274"
                },
                {
                    "textid": "115 1",
                    "name": "NetworkID Mismatch TS 1",
                    "enable": "Yes",
                    "relay": "No",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "315",
                    "priority": "315"
                },
                {
                    "textid": "117 1",
                    "name": "D/R No Search Path     ",
                    "enable": "Yes",
                    "relay": "No",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "348",
                    "priority": "348"
                },
                {
                    "textid": "118 1",
                    "name": "D/R Malformed DRT      ",
                    "enable": "Yes",
                    "relay": "No",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "349",
                    "priority": "349"
                },
                {
// several data records removed to save room in documentation
                }
            ],
            "alarms": [
                {
// several data records removed to save room in documentation
                },
                {
                    "textid": "147 1",
                    "name": "Audio 3&4 Not Licensed ",
                    "enable": "Yes",
                    "relay": "Yes",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "415",
                    "priority": "415"
                },
                {
// several data records removed to save room in documentation
                },
                {
                    "textid": "92 1",
                    "name": "Xcode License Fault    ",
                    "enable": "Yes",
                    "relay": "Yes",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "261",
                    "priority": "261"
                }
            ]
        }
```

```
    }
}
```

Converting Non-URL Characters from JSON String Values for use in QueryString Filters:

Note Some of the values in name and details fields in output JSON may be for characters that are normally not allowed in isolation in HTML data. When using these strings as inputs for QueryParameter filtering, in addition to spaces requiring replacement with %20 (i.e. Hexadecimal 20), these characters need to be replaced with a URL format % code for that character. For example, an ampersand (&) character (as per the "Audio 3&4 Not Licensed" value in the example output above) would be replaced with %26 (for example, Hexadecimal 36).

Examples using QueryString filters:

Input: (Read back faults settings for only "Audio 3 & Audio 4 Alarm" in default XML format):

```
curl -k
"https://192.168.0.1/ws/v2/status/faults/settings?session=$token&name=Xcode%2
0License%20Fault%20%20%20%20&js"
```

Expected result:

```
{
    "faults": {
        "settings": {
            "textid": "92 1",
            "name": "Xcode License Fault    "
        }
    }
}
```

Input: (Read back faults settings for only "Audio 3 & Audio 4 Alarm" in default XML format):

```
curl -k
"https://192.168.0.1/ws/v2/status/faults/settings?session=$token&name=Audio%2
03%264%20Not%20Licensed%20&js"
```

Expected result:

```
{
    "faults": {
        "settings": {
            "textid": "147 1",
            "name": "Audio 3&4 Not Licensed "
        }
    }
}
```

Alarm Settings Configuration Command

Table 2.292 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms
Command Information	Allows set or get of alarm settings configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	Query all alarm configuration settings only GET "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms?js" Single Alarm entry (by key): GET "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms? textid="10%201?js"
POST Syntax	Enable specific alarm instance (with trap and relay as well): POST "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms?js&textid=10%201&enable=Yes&trap=Yes&relay=Yes" Disable specific alarm instance (with trap and relay as well): POST "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms?js&textid=10%201&enable=No&trap=No&relay=No"

For Output Field Descriptions and URI Query/Set Arguments, refer to Fault Settings Top Level Configuration Command, on page 456.

XML Examples:

Input: (Read back all Alarms settings in default XML format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms"
```

Expected Output (values for example purpose only):

```
<?xml version="1.0" encoding="UTF-8"?>
<faults>
  <settings>
    <alarms>
// several data records removed to save room in documentation
    </alarms>
    <alarms>
      <textid>147 1</textid>
      <name>Audio 3&amp;4 Not Licensed </name>
      <enable>Yes</enable>
      <relay>Yes</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>415</mibpriority>
      <priority>415</priority>
    </alarms>
    <alarms>
// several data records removed to save room in documentation
    </alarms>
    <alarms>
      <textid>92 1</textid>
      <name>Xcode License Fault    </name>
      <enable>Yes</enable>
      <relay>Yes</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>261</mibpriority>
      <priority>261</priority>
    </alarms>
  </settings>
</faults>
```

Changing a Setting:

Input:

Enable specific fault instance (with trap and relay as well):

```
curl -k "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms?
js&textid=10%201&enable=Yes&trap=Yes&relay=Yes"
```

Or

Disable specific fault instance (with trap and relay as well):

```
curl -k "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms?
js&textid=10%201&enable=No&trap=No&relay=No"
```

Expected Response:

```
<response><code>10</code><result>success</result><message>Successful</messag
e>
</response>
```

JSON Examples:

Input: (Read back all Alarms settings in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings"
```

Expected Output (values for example purpose only):

```
{
    "faults": {
        "settings": {
            "alarms": [
                {
// several data records removed to save room in documentation
                },
                {
                    "textid": "147 1",
                    "name": "Audio 3&4 Not Licensed ",
                    "enable": "Yes",
                    "relay": "Yes",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "415",
                    "priority": "415"
                },
                {
// several data records removed to save room in documentation
                },
                {
                    "textid": "92 1",
                    "name": "Xcode License Fault    ",
                    "enable": "Yes",
                    "relay": "Yes",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "261",
                    "priority": "261"
                }
            ]
        }
    }
}
```

Changing a Setting:

Input:

Enable specific fault instance (with trap and relay as well):

```
curl -k "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms?
js&textid=10%201&enable=Yes&trap=Yes&relay=Yes&js"
```

Or

Disable specific fault instance (with trap and relay as well):

```
curl -k "https://192.168.0.1/ws/v2/service_cfg/faults/settings/alarms?
js&textid=10%201&enable=No&trap=No&relay=No&js"
```

Expected Response:

```
{
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
}
```

Warning Settings Configuration Command

Table 2.293 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings
Command Information	Allows set or get of warning settings configuration.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	Query all warning configuration settings only.

Command Detail	Description
	GET "https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings?js" Single Alarm entry (by key): GET "https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings?textid="91%201?js"
POST Syntax	Enable specific warning instance (with trap and relay as well): POST "https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings?js&textid=91%201&enable=Yes&trap=Yes&relay=Yes" Disable specific warning instance (with trap and relay as well): POST "https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings?js&textid=91%201&enable=No&trap=No&relay=No"

For Output Field Descriptions and URI Query/Set Arguments, refer to Fault Settings Top Level Configuration Command, on page 456.

XML Examples:

Input: (Read back all Warnings settings in default XML format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings"
```

Expected Output (values for example purpose only):

```
<?xml version="1.0" encoding="UTF-8"?>
<faults>
  <settings>
    <warnings>
      <textid>105 1</textid>
      <name>PSI/SI Stream Err TS 1</name>
      <enable>Yes</enable>
      <relay>No</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>274</mibpriority>
      <priority>274</priority>
    </warnings>
    <warnings>
      <textid>115 1</textid>
      <name>NetworkID Mismatch TS 1</name>
      <enable>Yes</enable>
      <relay>No</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>315</mibpriority>
      <priority>315</priority>
    </warnings>
    <warnings>
      <textid>117 1</textid>
      <name>D/R No Search Path     </name>
      <enable>Yes</enable>
      <relay>No</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>348</mibpriority>
      <priority>348</priority>
    </warnings>
    <warnings>
      <textid>118 1</textid>
      <name>D/R Malformed DRT      </name>
      <enable>Yes</enable>
      <relay>No</relay>
      <trap>Yes</trap>
      <delayctrl>Default</delayctrl>
      <delay>0</delay>
      <mibpriority>349</mibpriority>
      <priority>349</priority>
    </warnings>
    <warnings>
// several data records removed to save room in documentation
    </warnings>
  </settings>
</faults>
```

Changing a Setting:

Input:

Enable specific fault instance (with trap and relay as well):

```
curl -k "https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings?
js&textid=105%201&enable=Yes&trap=Yes&relay=Yes"
```

Or

Disable specific fault instance (with trap and relay as well):

```
curl -k "https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings?
js&textid=105%201&enable=No&trap=No&relay=No"
```

Expected Response:

```
<response><code>10</code><result>success</result><message>Successful</messag
e>
</response>
```

JSON Examples:

Input: (Read back all Warnings settings in JSON format):

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings?js"
```

Expected Output (values for example purpose only):

```
{
    "faults": {
        "settings": {
            "warnings": [
                {
                    "textid": "105 1",
                    "name": "PSI/SI Stream Err TS 1",
                    "enable": "Yes",
                    "relay": "No",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "274",
                    "priority": "274"
                },
                {
                    "textid": "115 1",
                    "name": "NetworkID Mismatch TS 1",
                    "enable": "Yes",
                    "relay": "No",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "315",
                    "priority": "315"
                },
                {
                    "textid": "117 1",
                    "name": "D/R No Search Path     ",
                    "enable": "Yes",
                    "relay": "No",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "348",
                    "priority": "348"
                },
                {
                    "textid": "118 1",
                    "name": "D/R Malformed DRT      ",
                    "enable": "Yes",
                    "relay": "No",
                    "trap": "Yes",
                    "delayctrl": "Default",
                    "delay": "0",
                    "mibpriority": "349",
                    "priority": "349"
                },
                {
// several data records removed to save room in documentation
                }
             ]
            ]
        }
    }
}
```

Changing a Setting:

Input:

Enable specific fault instance (with trap and relay as well):

```
curl -k "https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings?
js&textid=105%201&enable=Yes&trap=Yes&relay=Yes&js"
```

Or

Disable specific fault instance (with trap and relay as well):

```
curl -k "https://192.168.0.1/ws/v2/service_cfg/faults/settings/warnings?
js&textid=105%201&enable=No&trap=No&relay=No&js"
```

Expected Response:

```
{
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
}
```

Device Control Commands

This section lists all the device control commands.

Ethernet Device Control Command

Table 2.294 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/device_ctl/eth https://192.168.0.1/ws/v2/device_ctl/eth/<Level_1_URI_Parameter>
Command Information	Allows get of Ethernet ports information or configuration of Ethernet data or control ports.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax This command uses the ports referenced in Port Information, on page 22. However in this API, all of the port numbers have +1 added. As a result, the Management Port is port1, DATA1 is port2, and DATA2 is port3, and so on.	All port Info: GET "https://192.168.0.1/ws/v2/device_ctl/eth" Single Ethernet port info: GET "https://192.168.0.1/ws/v2/device_ctl/eth/port?portid=1" Single Ethernet port item: GET "https://192.168.0.1/ws/v2/device_ctl/eth/port?portid=1&ipv4addr"
POST Syntax Setting parameters using command line arguments is limited to maximum of 26 arguments after the ?. For Ethernet ports configuration, only HTTP data body is supported. POST from XML/JSON file is not supported.	Change management port configuration: POST "https://192.168.0.1/ws/v2/device_ctl/eth/port?portid=1&ipv4addr=[ipa.ddr.ess.one]" Where [ipa.ddr.ess.one] = valid ipv4 address on port 1 subnet If changing the subnet, then the change data port configuration: POST "https://192.168.0.1/ws/v2/device_ctl/eth/port?portid=2&ipv4addr=[ipa.ddr.ess.two]"

Command Detail	Description
	Where [ipa.ddr.ess.two] = valid ipv4 address on port 2's subnet

Table 2.295 Level_1 URI Parameters (extension to the Command URL separated by /)

URI Parameter	Description
port	Indicates that port parameters will be specified. When the command URL ishttps://192.168.0.1/ws/v2/device_ctl/eth/port, then the command line must also specify the portid URI Query/Set argument and value.

Table 2.296 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
portid (Key)	Ethernet Port Identification Type: digit Value: 1, 2, 3, 4, 5 Notes: 1 - Management Port 2 – DATA1 3 – DATA2 4 – DATA3 5 – DATA4
name (Read-Only)	Ethernet Port Name Type: string Value: "Management", "Data1", "Data2", "Data3", "Data4"
ipver	IP version selection Type: String Value: IPv4, IPv6 **Note** Only IPv4 is currently supported.
ipv4addr	IP V4 Address Type: String

URI Argument	Description
	Value: IP address, for example, 192.168.0.1
ipv4mask	IP V4 mask Type: digit Value: 8..30
ipv4defgw	IP V4 Default Gateway Type: String Value: IP address, for example, 192.131.244.254
phymode	IP Physical Port Mode Type: String Value: Auto, 10HD, 10FD, 100HD, 100FD, 1000FD, N/A
ipv4dns1	IP V4 DNS Primary Name Server Address Type: String Value: IP address Default: 0.0.0.0
ipv4dns2	IP V4 DNS Secondary Name Server Address Type: String Value: IP address Default: 0.0.0.0
ipv4dns3	IP V4 DNS Tertiary Name Server Address Type: String Value: IP address Default: 0.0.0.0
dhcpEnable	Enable or Disable DHCP Type: String Value: No, Yes

Note All of the URI Arguments below apply to both GET and POST. For GET, the URI arguments do not need any values, except the index which must be specified. For POST the index must be specified followed by any of the below URI arguments and the associated value the user would like to set. In-depth CURL examples can be seen below following the table.

GET Method Examples:

Display physical Ethernet Port information in XML or JSON. It can be used to display all ports information or specific port or port field information according to the input parameters.

Show all port Info in XML format (info for two additional ports Data4 and Data5 only available with Multi-Stream units not shown in the example below):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_ctl/eth"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<devicectl>
<eth>
    <port>
        <portid>1</portid>
        <name>Management</name>
        <ipver>IPv4</ipver>
        <ipv4addr>192.168.0.1</ipv4addr>
        <ipv4mask>24</ipv4mask>
        <ipv4defgw>192.168.0.254</ipv4defgw>
        <phymode>Auto</phymode>
        <ipv4dns1>161.44.124.122</ipv4dns1>
        <ipv4dns2>0.0.0.0</ipv4dns2>
        <ipv4dns3>0.0.0.0</ipv4dns3>
        <dhcpEnable>No</dhcpEnable>
    </port>
        <port>
            <portid>2</portid>
            <name>Data1</name>
            <ipver>IPv4</ipver>
            <ipv4addr>192.168.1.2</ipv4addr>
            <ipv4mask>24</ipv4mask>
            <ipv4defgw>192.168.1.254</ipv4defgw>
            <phymode>Auto</phymode>
        <ipv4dns1>0.0.0.0</ipv4dns1>
        <ipv4dns2>0.0.0.0</ipv4dns2>
        <ipv4dns3>0.0.0.0</ipv4dns3>
        <dhcpEnable>No</dhcpEnable>
    </port>
        <port>
            <portid>3</portid>
            <name>Data2</name>
            <ipver>IPv4</ipver>
            <ipv4addr>192.168.1.3</ipv4addr>
            <ipv4mask>24</ipv4mask>
            <ipv4defgw>192.168.1.254</ipv4defgw>
            <phymode>Auto</phymode>
        <ipv4dns1>0.0.0.0</ipv4dns1>
        <ipv4dns2>0.0.0.0</ipv4dns2>
        <ipv4dns3>0.0.0.0</ipv4dns3>
        <dhcpEnable>No</dhcpEnable>
    </port>
    </eth>
</devicectl>
```

Show a single port information in XML format:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/eth/port?portid=1"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<devicectl>
<eth>
    <port>
        <portid>1</portid>
        <name>Management</name>
        <ipver>IPv4</ipver>
        <ipv4addr>192.168.0.1</ipv4addr>
        <ipv4mask>24</ipv4mask>
        <ipv4defgw>192.168.0.254</ipv4defgw>
        <phymode>Auto</phymode>
        <ipv4dns1>161.44.124.122</ipv4dns1>
        <ipv4dns2>0.0.0.0</ipv4dns2>
        <ipv4dns3>0.0.0.0</ipv4dns3>
        <dhcpEnable>No</dhcpEnable>
    </port>
    </eth>
</devicectl>
```

Show a single port item in JSON format:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_ctl/eth/port?portid=2&ipv4addr&js=1"
```

Expected output (values are for example purposes only):

```
{
    "devicectl": {
        "eth": {
            "port": {
                "portid": "2",
                "ipv4addr": "192.168.1.2"
            }
        }
    }
}
```

POST Method Examples:

Configure physical Ethernet Port. When the Management Port IP is changed the connection will be immediately lost and user have to re-login the system, this will also result in the token not being returned. In order to avoid mixing the command with other commands together, IP control port configurations is only supported from HTTP POST body.

Management port:

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_
ctl/eth/port?portid=1&ipv4addr=192.168.0.0&js=1"
```

If successful, the following is an example of the return body:

```
 "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Data Port:

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_
ctl/eth/port?portid=2&ipv4addr=192.131.244.7&js=1"
```

If successful, the following is an example of the return body:

```
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Important Caveat Regarding Changing Settings for the Management Port:

When changing settings for the Management port (portid=1), the pending network reconfiguration may result in a network disconnect between your client and the server thus preventing the response message from being received. In this case, the result of the POST operation may be an HTTP timeout

(HTTP_500 return code range) (and login session would also be lost). Add the "-i" option to the curl command syntax in order to examine the HTTP response header for the HTTP return code. When network connectivity returns, we recommend that you verify the changed setting(s) via a new login, followed by a subsequent GET ws/v2/device_ctl/eth/port operation.

Version Selection Device Control Command

Table 2.297 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/device_ctl/ver https://192.168.0.1/ws/v2/device_ctl/ver/<Level_1_URI_ Parameter>
Command Information	Allows getting all runnable software version information, selecting a software version or deleting a software version.
HTTP Methods	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All version Info GET "https://192.168.0.1/ws/v2/device_ctl/ver"
POST Syntax Only HTTP data body is supported. POST from XML/JSON file is not supported.	Select a version: POST "https://192.168.0.1/ws/v2/ver/verselect? idx=[s/w version idx#]&reboot=Yes" Delete a version: POST "https://192.168.0.1/ws/v2/ver/verselect? idx=[s/w version idx#]&erase=Yes"

Table 2.298 Level_1 URI Parameters (extension to the Command URL separated by /)

URI Parameter	Description
verselect	Version to be selected URI Query/Set Arguments below are only valid when this level URI parameter is used

Note All of the URI Arguments below apply to both GET and POST except where indicated. For GET, the URI arguments do not need any values, except the idx which must be specified. For POST, the idx must be specified followed by any of the below URI arguments and the associated value the user would like to set. Various CURL examples can be seen below.

Table 2.299 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
idx (Key)	Software version index Type: Digit Value: 0 ~ 8
name	Software version Name Type: String Value: software version naming format, for example, T0.10B7(T0.10B5) **Note** For POST command syntax, this option is not used. It is valid as exist (no value) filter for GET command, also in the output data.
reboot	To run the software version to be selected Type: String Value: Yes, No
erase	To delete the software version to be selected. Type: String Value: Yes, No
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML

GET Method Examples:

Display runnable software versions information in XML or JSON.

Show all version info in XML:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_ctl/ver"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<devicectl><ver><verselect><idx>0</idx><name>1.10</name><reboot>No</reboot><e
rase>No</erase>

</verselect><verselect><idx>1</idx><name>T2.25C1</name><reboot>No</reboot><er
ase>No</erase>

</verselect><verselect><idx>2</idx><name>T2.25B7</name><reboot>No</reboot><er
ase>No</erase>

</verselect><verselect><idx>3</idx><name></name><reboot>No</reboot><erase>No<
/erase></verselect>

<verselect><idx>4</idx><name></name><reboot>No</reboot><erase>No</erase></ver
select><verselect>

<idx>5</idx><name></name><reboot>No</reboot><erase>No</erase></verselect><ver
select><idx>6</idx>

<name></name><reboot>No</reboot><erase>No</erase></verselect><verselect><idx>
7</idx><name>

</name><reboot>No</reboot><erase>No</erase></verselect><verselect><idx>8</idx
><name></name>
<reboot>No</reboot><erase>No</erase></verselect></ver></devicectl>
```

Show all version Info in JSON:

```
{
    "devicectl": {
        "ver": {
            "verselect": [
                {
                    "idx": "0",
                    "name": "1.10",
                    "reboot": "No",
                    "erase": "No"
                },
                {
                    "idx": "1",
                    "name": "T2.25C1",
                    "reboot": "No",
                    "erase": "No"
                },
                {
                    "idx": "2",
                    "name": "T2.25B7",
                    "reboot": "No",
                    "erase": "No"
                },
                {
                    "idx": "3",
                    "name": "",
                    "reboot": "No",
                    "erase": "No"
                },
                {
                    "idx": "4",
                    "name": "",
                    "reboot": "No",
                    "erase": "No"
                },
                {
                    "idx": "5",
                    "name": "",
                    "reboot": "No",
                    "erase": "No"
                },
                {
                    "idx": "6",
                    "name": "",
                    "reboot": "No",
                    "erase": "No"
                },
                {
                    "idx": "7",
                    "name": "",
                    "reboot": "No",
                    "erase": "No"
                },
                {
                    "idx": "8",
                    "name": "",
                    "reboot": "No",
                    "erase": "No"
                }
            ]
        }
    }
}
```

POST Method Examples:

Select a software version to run or delete the software version.

Select a version:

```
curl -k -X POST -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/ver/verselect?idx=1&reboot=Yes&js=1"
```

If successful, the following is an example of the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

And system will reboot successfully to the new version.

Delete a version:

```
curl -k -X POST -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/ver/verselect?idx=2&erase=Yes&js=1"
```

If successful, the following is an example of the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Reset Device Control Command

Table 2.300 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/device_ctl/reset/<Level_1_URI_Parameter>
Command Information	Allows performing a device reset or factory reset (restore factory default settings). A device reset reboots the target unit. A factory reset does not trigger any reboot for the PowerVu Professional Receiver.
HTTP Method	POST
Access Type	Write
Access Level	User, Admin
GET Syntax	N/A
POST Syntax	Factory Reset:

Command Detail	Description
Only HTTP data body is supported. POST from XML/JSON file is not supported.	POST "https://192.168.0.1/ws/v2/device_ctl/reset/factory" Reboot: POST "https://192.168.0.1/ws/v2/device_ctl/reset/reboot"

Table 2.301 Level_1 URI Parameters (extension to the Command URL separated by /)

URI Parameter	Description
factory	Do a factory reset.
reboot	Reboot the unit.

Table 2.302 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
js	Format output using JSON standard Type: exist Values: any value or empty (any of js=1, js, js=0, or js=9999 are valid) Omitting this argument formats the output by default in XML.

POST Method Examples:

Factory Reset or Reboot the unit.

Factory reset:

```
curl -k -X POST -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/reset/factory?js=1"
```

If successful, the following is an example of the return body:

```
"response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

Reboot:

```
curl -k -X POST -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/reset/reboot?js=1"
```

If successful, the following is an example of the return body:

```
    "response": {
        "code": "10",
        "result": "success",
        "message": ""
    }
```

The system will reboot successfully to the same version (after about 1 minute).

Protocols Device Control Command

Table 2.303 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/device_ctl/protocols
Command Information	Allows getting and setting device controls for protocols such as HTTP, SNMP, IGMP, and so on. **Note** ForSyslog related parameters settings, this API supports only the settings for the first (default) Syslog target instance. In Release 5.00 or later, if using multiple Syslog targets, refer to Monitors Profiles Device Control Command, on page 495 and Monitors Syslog Device Control Command, on page 503 for the appliable Web Services APIs.
HTTP Method(s)	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	All protocol device control info: GET "https://192.168.0.1/ws/v2/device_ctl/protocols" Single protocol device control item: GET "https://192.168.0.1/ws/v2/device_ctl/protocols?http"
POST Syntax Prior to Release 5.00, using POST syntax with the settings data coming from a XML/JSON file was not supported for this API.	Change SNMP protocol configuration: POST "https://192.168.0.1/ws/v2/device_ctl/protocols?snmp=Enable" Change IGMP protocol configuration: POST "https://192.168.0.1/ws/v2/device_ctl/protocols?igmp=Disable"

Command Detail	Description
In Release 5.00 or later, using either the standard POST with QueryString Parameters syntax or POST with data sourced from an XML or JSON file is now supported.	**Note** Changing the IGMP protocol configuration setting will still allow read back of the change in the settings values, but will have no effect on the IGMP protocol functional operation of the system.

Table 2.304 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
http	HTTP Protocol Control Type: String Value: Disable, HTTP, HTTPS
snmp	SNMP Protocol Control Type: String Value: Disable, Enable
igmp	IGMP Protocol Control Type: String Value: Disable, IGMP v3, IGMP v2 Default: IGMP v3 **Note** We recommend that you do not change this setting with Version 4.50 or later because it will have no functional effect on the IP port (s).
syslog	Syslog Mode of Operation Type: String Value: Disable, Syslog TCP, Syslog UDP
mpe	MPE Data Filter Mode Type: String Value: Fwd None, Fwd All, Fwd Filtered **Note** This setting is available for single-stream units only.
idletimeout	Protocol Idle Timeout

URI Argument	Description
	Type: Digit Value: 0, 30 ~ 1209600 inclusive
globaldns	Enable/Disable Domain Name Server Support Type: String Value: Disable, Enable Default: Disable
multimgmt	Ethernet port management control Type: String Value: Mgmt Only, All Interfaces Default: Mgmt Only
slogrowstat	Syslog Control Record Status Type: String Value: “Active”, “Inactive
slogsrvip	Syslog Server IP Address Type: String Value: IP address, for example, 0.0.0.0
slogsrvprt	Syslog Server Port Type: Digit Value: IP port, for example, 514
hwfail	Syslog Hardware Fail Log control Enables Syslog the use of "emerg" keyword severity (level 0) for logs that will indicate unrecoverable hardware failure. These logs cannot be masked or disabled. Type: String **Note** The Syslog Severity Log level control items are available only in Version 4.75 or later (applies to hwfail, assert, faultalarm, faultwarn, warning, info, debug). Value: "Yes" (cannot be changed) **Read-Only**
assert	Syslog Assert Log control

URI Argument	Description
	Enables Syslog the use of "alert" keyword severity (level 1) for logs that will indicate unexpected (should not normally occur) Software Failures. This is usually reserved for reporting crashes or low level exception conditions. Type: String Value: "Yes" (default), or "No"
faultalarm	Syslog Alarm Fault Log control Enables Syslog the use of "crit" keyword severity (level 2) for logs that will indicate status details of an Alarm Fault Event. Type: String Value: "Yes" (default), or "No"
faultwarn	Syslog Warning Fault Log control Enables Syslog the use of "crit" keyword severity (level 3) for logs that will indicate status details of a Warning Fault Event. Type: String Value: "Yes" (default), or "No"
warning	Syslog Warning Log control Enables Syslog the use of "warning" keyword severity (level 4) for logs that will indicate Software warnings. These are usually sent due to predicted error conditions in software logic or functionality that are as a result of misconfiguration, invalid input, or other unexpected scenarios. Warning Logs are common when testing Web Services APIs with invalid parameters because out of range or syntax validation violation or disallowed combinations for those parameters were detected. Type: String Value: "Yes" (default), or "No"
info	Syslog Info Log control Enables Syslog the use of "info" keyword severity (level 5) for logs that will indicate Software information intended to be useful to end users. Info Logs are useful for verifying the expected or correct operation of the system. Too many Info logs within a short period of time may be counter-productive because the log system will throttle these logs (prevent them from being sent on the network) at a preset threshold applied to the combination of logs at all severities (for example, 1000 messages per second). Type: String Value: "Yes" (default), or "No"

URI Argument	Description
debug	Syslog Debug Log control Enables Syslog the use of "info" keyword severity (level 6) for logs that will indicate Software debug (tracing). Debug Logs are useful only for Synamedia Engineering and are therefore default suppressed by the application software by default prior to this setting being applied. Type: String Value: "Yes" (default), or "No" **Note** Although this setting is default "Yes", no "debug" logs will be sent on the Syslog network unless the appropriate WS API Log Control setting for the individual process and component(s) of interest have also been set by the engineer performing the debugging. Debug logs tend to be verbose and will be throttled if the combination of all log severities are sent at a higher rate than the preset threshold (for example, 1000 messages per second). The WSAPI Log Control settings for allowing these logs to pass through should therefore only be sparingly applied (to selected component instances of interest), with knowledge of the code, by qualified Synamedia Engineering personnel.
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML

GET Method Examples:

Display protocol control information in XML or JSON.

1. Show all protocol control info in XML:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/protocols"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<devicectl><protocols><settings><http>HTTPS</http><snmp>Enable</snmp><ig
mp>Enable</igmp>
```

```
<syslog>Syslog UDP</syslog><slogsrvip>0.0.0.0</slogsrvip><slogsrvprt>514
</slogsrvprt>

<mpe>Fwd None</mpe><idletimeout>0</idletimeout><globaldns>Enable</global
dns></settings>
</protocols></devicectl>
```

2. Show all protocol control info in JSON:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/protocols?js=1"
```

Expected output (values are for example purposes only):

```
{
    "devicectl": {
        "protocols": {
            "settings": {
                "http": "HTTPS",
                "snmp": "Enable",
                "igmp": "IGMP v3",
                "syslog": "Syslog UDP",
                "slogsrvip": "0.0.0.0",
                "slogsrvprt": "514",
                "mpe": "Fwd None",
                "idletimeout": "0",
                "globaldns": "Enable"
                "multimgmt": "Mgmt Only"
            }
        }
    }
}
```

3. Show a single protocol control item in JSON:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/protocols?http&js=1"
```

Expected output (values are for example purposes only):

```
{
    "devicectl": {
        "protocols": {
            "settings": {
                "http": "HTTPS"
            }
        }
    }
}
```

POST Method Examples:

Change protocol control configurations

1. Change SNMP protocol configuration:

   ```
   curl -k -X POST -H "X-SESSION-ID: $token"
   "https://192.168.0.1/ws/v2/device_ctl/protocols?snmp=Enable&js=1"
   ```

   If successful, the return body will be:

   ```
   {
       "response": {
           "code": "10",
           "result": "success",
           "message": "Success"
       }
   }
   ```

   And SNMP protocol will be enabled.

2. Change IGMP protocol configuration:

   ```
   curl -k -X POST -H "X-SESSION-ID: $token"
   "https://192.168.0.1/ws/v2/device_ctl/protocols?igmp=Disable&js=1"
   ```

   If successful, the return body will be:

   ```
   {
       "response": {
           "code": "10",
           "result": "success",
           "message": "Success"
       }
   }
   ```

   And IGMP protocol will be disabled.

Download Device Control

For a description of the ws/v2/device_ctl/download API, refer to Determining Support for Chunky CDT Upload Command, on page 54. This API provides access to secondary settings affecting the behavior of downloading and does not transfer files itself. For details, see Legacy CDT Upload Command, on page 51.

Monitors Settings Device Control Command

Table 2.305 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/device_ctl/monitors/settings
Command Information	Allows getting and setting items that apply to the behavior of certain Monitor profiles controlled by the other APIs under device_ctl/monitors. For example, we have added the Metrics Tag Value field (metricstag), which is a tag added to all data being sent through InfluxDB from this unit. This field appears as "channel_id" in the data measurements. Each settings field applies to all instances of the applicable profile. This API provides functionality that is equivalent to the following web UI feature: System Settings > Monitoring > Monitoring Setup: Metrics Tag Configuration.
HTTP Method(s)	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	Retrieves all monitors settings information. GET "https://192.168.0.1/ws/v2/device_ctl/monitors/settings"
POST Syntax	Sets a monitors settings metricstag value. POST "https://192.168.0.1/ws/v2/device_ctl/monitors/settings?metricstag=my_custom_tag"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.306 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
metricstag	Tag to identify metrics. This field will appear as "channel_id" in the influxdb metrics data measurements. Each settings field applies to all instances of the applicable profile. Type: String Default: "" (empty string)

GET Method Examples:

Display monitors settings information in XML or JSON.

1. Show all monitors settings info in XML:

   ```
   curl -k -H "X-SESSION-ID: $token" "https://10.85.222.89/ws/v2/device_
   ctl/monitors/settings"
   ```

   Expected output (values are for example purposes only):

   ```
   <?xml version="1.0" encoding="ISO-8859-1" ?><device_
   ctl><monitors><settings><metricstag>my_custom_
   tag</metricstag></settings></monitors></device_ctl>
   ```

2. Show all monitors settings info in JSON:

   ```
   curl -k -H "X-SESSION-ID: $token" "https://10.85.222.89/ws/v2/device_
   ctl/monitors/settings?js"
   ```

   Expected output (values are for example purposes only):

   ```
   {
           "device_ctl": {
             "monitors": {
                "settings": [
                   {
                           "metricstag": "my_custom_tag",
                   }
           }
      }
   }
   ```

POST Method Examples:

Change monitors settings:

1. Add a metrics tag (XML format response expected):

   ```
   curl -k -X POST -H "X-SESSION-ID: $token"
   "https://192.168.0.1/ws/v2/device_ctl/monitors/settings?metricstag=my_
   custom_tag"
   ```

   If successful, the return body will be:

   ```
   <response><code>10</code><result>success</result><message></message>
   </response>
   ```

2. Add a metrics tag (JSON format response expected):

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_ctl/monitors/settings?metricstag=my_
custom_tag&js"
```

If successful, the return body will be:

```
{
        "response": {
            "code": "10",
            "result": "success",
            "message": ""
        }
}
```

Monitors Profiles Device Control Command

Table 2.307 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/device_ctl/monitors/profiles
Command Information	Allows getting and setting device controls for profiles (connections for protocol monitors) which supports using multiple protocol types and multiple instance configurations for each type. For example, a user or script would use this API to add an additional instance for SYSLOG_CONTROL of an alternate syslog server target in addition to the one controlling settings for the default syslog target. A common use case would be for the syslog severities associated with only Alarm and Warning Faults to be monitored on one target Syslog server ip address and port, while simultaneously monitoring a different default Syslog server ip address and port for other syslog severities. This API provides functionality that is equivalent to the following web UI feature: System Settings > Monitoring > Monitoring Setup: Connections Table +/x (add/delete and configure a row in the table).
HTTP Method(s)	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	Retrieve all monitors connections profiles info GET "https://192.168.0.1/ws/v2/device_ctl/monitors/profiles" Retrieve monitors connections profile info for a specific record instance GET "https://192.168.0.1/ws/v2/device_ctl/monitors/profiles?recidx=1 Monitors connection device control info for a specific record instance with

Command Detail	Description
	QueryParameter filter(s) GET "https://192.168.0.1/ws/v2/device_ ctl/monitors/profiles?recidx=1&profiletype&csiid"
POST Syntax	Add a monitor connection profile for additional Syslog Control instance POST "https://192.168.0.1/ws/v2/device_ctl/monitors/profiles? recidx=2&name=AltSyslog&profiletype=SYSLOG_CONTROL &csiid=2&cmdrow=Add" Delete the monitor connection profile for the second Syslog Control instance POST "https://192.168.0.1/ws/v2/device_ctl/monitors/profiles? recidx=2&name=AltSyslog&profiletype=SYSLOG_ CONTROL&csiid=2&cmdrow=Delete" **Note** The first profiletype=SYSLOG_CONTROL record instance (csiid=1) is always both available and “Active” and cannot be deleted. Add a monitor connection profile for an InfluxDB Control instance. POST "https://192.168.0.1/ws/v2/device_ctl/monitors/profiles? recidx=3&name=InDB1&profiletype=INFLUXDB&csiid=1&cmdrow=Add"

URI Parameters (extension to the Command URL separated by /): N/A

Table 2.308 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
recid (Key)	Record index of monitoring profile Type: Integer Values: 1 … 16
name	Unique Monitoring Profile name Type: String
profiletype	Monitoring Profile type Type: String Values: “NONE”, “SYSLOG_CONTROL”, “INFLUXDB”
csiid	This profile type’s instance index in the corresponding monitors syslog_ctl or influxdb settings API. For example, for SYSLOG_CONTROL profile type, this value corresponds to the value of slogindex in the GET ws/v2/monitors/syslog_ctl API response. For INFLUXDB profile type, this value corresponds to the value of idbindex in the GET ws/v2/monitors/influxdb API response. Type: Integer

URI Argument	Description
	Values: range depends on profile type
cmdrow	On GET, returns profile status (Active/Inactive). A deleted node's row will show "Inactive" in a GET response. A successfully added node's row will show "Active" in a GET response. On POST, set this value to "Add" or "Delete" an instance record of a monitoring profile (Add/Delete) Type: String Values: "Add" (R/W), "Delete" (R/W), "Active" (RO), "Inactive" (RO)
js	Format output using JSON standard Type: exist Values: any value or empty (ie any of js=1 or js or js=0 or js=9999 are valid) Omitting this argument formats the output by default in XML

GET Method Examples (for target with two instances of Monitor Setup for SYSLOG_CONTROL type followed by one instance of Monitor Setup for INFLUXDB type pre-configured):

Display profiles control information in XML or JSON.

1. Show all profiles control info in XML:

```
curl -k -H "X-SESSION-ID: $token"
"https://10.85.222.89/ws/v2/device_ctl/monitors/profiles"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><device_ctl>
<monitors><profiles><recid>1</recid><csiid>1</csiid>
<profiletype>SYSLOG_CONTROL</profiletype><name>SYSLOG_CONTROL</name>
<cmdrow>Active</cmdrow></profiles><profiles>
<recid>2</recid><csiid>2</csiid><profiletype>SYSLOG_
CONTROL</profiletype>
<name>ALT_SYSLOG_
CONTROL</name><cmdrow>Active</cmdrow></profiles><profiles>
<recid>3</recid><csiid>1</csiid><profiletype>INFLUXDB</profiletype>
<name> INDB_PROFILE_1</name><cmdrow>Active</cmdrow></profiles><profiles>
<recid>4</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>5</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>6</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>7</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>8</recid><csiid>0</csiid><profiletype>NONE</profiletype>
```

```
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>9</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>10</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>11</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>12</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>13</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>14</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>15</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles><profiles>
<recid>16</recid><csiid>0</csiid><profiletype>NONE</profiletype>
<name></name><cmdrow>Inactive</cmdrow></profiles></monitors></device_
ctl>
```

2. Show all profiles control info in JSON:

```
curl -k -H "X-SESSION-ID: $token"
"https://10.85.222.89/ws/v2/device_ctl/monitors/profiles&js"
```

Expected output (values are for example purposes only):

```
{
        "device_ctl": {
          "monitors": {
             "profiles": [
                {
                        "recid": "1",
                        "csiid": "1",
                        "profiletype": "SYSLOG_CONTROL",
                        "name": "SYSLOG_CONTROL",
                        "cmdrow": "Active"
                },
                {
                        "recid": "2"
                        "csiid": "2",
                        "profiletype": "SYSLOG_CONTROL",
                        "name": "ALT_SYSLOG_CONTROL",
                        "cmdrow": "Active"
                },
                {
                        "recid": "3",
                        "csiid": "1",
                        "profiletype": "INFLUXDB",
                        "name": "INFLUXDB_PROFILE_1",
                        "cmdrow": "Active"
                },
                {
                        "recid": "4",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "5",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "6",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "7",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "8",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "9",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "10",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "11",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "12",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "13",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "14",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "15",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                },
                {
                        "recid": "16",
                        "csiid": "0",
                        "profiletype": "NONE",
                        "name": "",
                        "cmdrow": "Inactive"
                }
            ]
        }
    }
}
```

3. Show a specific profile record in JSON:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_ctl/monitors/profiles?recid=2&js"
```

Expected output (values are for example purposes only):

```
{
        "device_ctl": {
          "monitors": {
                "profiles": {
                   "recid": "2",
                   "csiid": "2",
                   "profiletype": "SYSLOG_CONTROL",
                   "name": "ALT_SYSLOG_CONTROL",
                   "cmdrow": "Active"
                }
           }
      }
}
```

POST Method Examples:

Change profiles control configurations:

1. Add an alternate syslog control configuration monitor record configuration (XML format response expected):

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_
ctl/monitors/profiles?recidx=2&name=AltSyslog&
profiletype=SYSLOG_CONTROL&csiid=2&cmdrow=Add"
```

If successful, the return body will be:

```
<response><code>10</code><result>success</result><message></message>
</response>
```

2. Delete the alternate syslog control configuration monitor record configuration (XML format response expected):

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_
ctl/monitors/profiles?recidx=2&name=AltSyslog&
profiletype=SYSLOG_CONTROL&csiid=2&cmdrow=Delete"
```

   If successful, the return body will be:

```
<response><code>10</code><result>success</result><message></message>
</response>
```

3. Add an alternate syslog control configuration monitor record configuration (JSON format response expected):

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_
ctl/monitors/profiles?recidx=2&name=AltSyslog&
profiletype=SYSLOG_CONTROL&csiid=2&cmdrow=Add&js"
```

   If successful, the return body will be:

```
{
        "response": {
          "code": "10",
          "result": "success",
          "message": ""
        }
}
```

4. Add an influxdb control configuration record (JSON format response expected):

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_
ctl/monitors/profiles?recidx=3&name=InDB1&
profiletype=INFLUXDB&csiid=1&cmdrow=Add&js"
```

   If successful, the return body will be:

```
{
        "response": {
          "code": "10",
          "result": "success",
          "message": ""
        }
}
```

Monitors Syslog Device Control Command

Table 2.309 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/device_ctl/monitors/sys_ctl
Command Information	Allows getting and setting device controls for syslog when using multiple syslog target destinations. This API works in conjunction with the GET/POST ws/v2/device_ctl/monitors/profiles API. **Note** There are two records of syslog control data returned by the non-filtered GET version of this API. The state of each record is determined by the "Active" or "Inactive" value of the rowstatus parameter. After performing a factory reset (the default state of the platform), the target will have only one syslog_ctl instance "Active", which is accessed via slogindex=1. Adding or deleting a second instance may be accomplished via the web UI (System Settings > Monitoring > Monitoring Setup Connections Table) or via the POST ws/v2/device_ctl/monitors/profiles API. An individual record (instance) is accessed via slogindex=1 or slogindex=2 for this API with the corresponding use of the profiletype=”SYSLOG_CONTROL” and csiid=1 or csiid=2 for the ws/v2/device_ctl/monitors/profiles API. **Note** Adding an instance via the profiles API (or web UI) changes the rowstatus value of the corresponding record returned by this API to "Active" and deleting an instance via the profiles API (or web UI) changes the rowstatus value to "Inactive" for the corresponding record. For Syslog, only the record corresponding to slogindex=2 can be deleted (set to rowstatus value "Inactive"). You cannot delete the record corresponding to instance slogindex=1. The two syslog target destinations and independent settings control for each target instance supports customer configurations where a dedicated Syslog server target may be used to monitor alarm and warning faults, in lieu of using SNMP. This API provides functionality that is equivalent to the following web UI feature: System Settings > Monitoring > Monitoring Setup > Select a row of the Connections Table > Connection Configuration + Advanced Syslog Filtering.
HTTP Method(s)	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	Syslog protocol device control information for all instances:

Command Detail	Description
	GET "https://192.168.0.1/ws/v2/device_ctl/monitors/syslog_ctl" Syslog protocol device control information for specific instance: GET "https://192.168.0.1/ws/v2/device_ctl/monitors/syslog_ctl?slogindex=2" Syslog protocol device control item for specific instance with QueryParameter filters: GET "https://192.168.0.1/ws/v2/device_ctl/monitors/syslog_ctl?slogindex=2&faultalarm&faultwarn"
POST Syntax	Disable all but faultalarm&faultwarn (and read-only hwfail setting) severities for a specific syslog_ctl instance: POST "https://192.168.0.1/ws/v2/device_ctl/monitors/syslog_ctl?slogindex=2&faultalarm=Yes&faultwarn=Yes&assert=No&warning=No&info=No&debug=No"

Table 2.310 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
slogindex (Key)	Instance index of Syslog settings for a specific Syslog server target. Type: Integer. Value: 1 or 2 (when pre-configured via web UI). Default: For POST command, this must be specified. For GET, not specifying the slogindex will return data for all available instances.
syslog	Syslog mode of operation. Type: String. Value: Disable, Syslog TCP, or Syslog UDP.
slogrowstat	Syslog control record status. The slogrowstat for the corresponding slogindex instance is "Active" if the SYSLOG_CONTROL type instance row has been added via the web UI Monitoring Setup Connections Table or via the POST ws/v2/device_ctl/monitors/profiles API. Type: String Value: “Active” or “Inactive”.
slogsrvip	Syslog server IP address. Type: String.

URI Argument	Description
	Value: IP address, for example, 0.0.0.0.
slogsrvprt	Syslog server port. Type: Digit. Value: IP port number, for example, 514.
hwfail	Syslog hardware fail log control. Enables Syslog use of "emerg" keyword severity (level 0) for logs that indicate unrecoverable hardware failure. These logs cannot be masked or disabled. However, the chance of receiving a log at this severity is rare. Type: String. Value: "Yes" (cannot be changed). This is read-only.
assert	Syslog assert log control. Enables Syslog use of "alert" keyword severity (level 1) for logs that indicate unexpected software failures. This is normally reserved for reporting crashes or low level exception conditions. Type: String. Value: "Yes" (default) or "No".
faultalarm	Syslog alarm fault log control. Enables Syslog use of "crit" keyword severity (level 2) for logs that will indicate status details of an alarm fault event. Type: String. Value: "Yes" (default) or "No".
faultwarn	Syslog warning fault log control. Enables Syslog use of "crit" keyword severity (level 3) for logs that indicate status details of a warning fault event. Type: String. Value: "Yes" (default) or "No".
warning	Syslog warning log control. Enables Syslog use of "warning" keyword severity (level 4) for logs that indicate software warnings. These are normally sent due to predicted error conditions in software logic or functionality that are as a result of misconfiguration, invalid input, or other unexpected scenarios. Warning Logs are common when testing Web Services APIs with invalid parameters due to out of range or syntax validation violation or disallowed combinations for those parameters were detected.

URI Argument	Description
	Type: String. Value: "Yes" (default) or "No".
info	Syslog info log control. Enables Syslog use of "info" keyword severity (level 5) for logs that indicate software information intended to be useful to end-users. Info logs are useful for verifying the expected or correct operation of the system. Too many Info logs within a short period of time may be counter-productive because the log system will throttle these logs (prevent them from being sent on the network) at a preset threshold applied to the combination of logs at all severity levels (for example, 1000 messages per second). Type: String. Value: "Yes" (default) or "No".
debug	Syslog debug log control. Enables Syslog use of the "info" keyword severity (level 6) for logs that indicate software debug (tracing). Debug Logs are useful only for Synamedia Engineering and are suppressed by the application software, by default, prior to applying this setting. Type: String. Value: "Yes" (default) or "No". **Note** Although the default for this setting is "Yes", debug logs will not be sent on the Syslog network unless the appropriate WS API Log Control setting for the individual process and component(s) of interest have also been set by the engineer performing the debugging. Debug logs may be throttled if the combination of all log severity levels are sent at a higher rate than the preset threshold (for example, 1000 messages per second). The WSAPI Log Control settings for allowing these logs to pass through should only be sparingly applied (to selected component instances of interest), with knowledge of the code, by a qualified Synamedia Engineering personnel.
js	Format output using JSON standard. Type: exist. Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid). Omitting this argument formats the output, by default, in XML.

GET Method Examples (for target with two instances of Monitor Setup for SYSLOG_CONTROL type pre-configured):

Display syslog control information in XML or JSON.

1. Show all syslog control info in XML:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/monitors/syslog_ctl"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><device_
ctl><monitors><syslog_ctl><slogindex>1</slogindex><syslog>Syslog
TCP</syslog><slogsrvip>10.85.222.79</slogsrvip><slogsrvprt>514</slogsrvp
rt>
<slogrowstat>Active</slogrowstat><hwfail>Yes</hwfail><assert>Yes</asser
t>
<faultalarm>Yes</faultalarm><faultwarn>Yes</faultwarn><warn>Yes</warn>
<action>Yes</action><info>Yes</info><debug>Yes</debug></syslog_
ctl><syslog_
ctl><slogindex>2</slogindex><syslog>Disable</syslog><slogsrvip>0.0.0.0
</slogsrvip><slogsrvprt>514</slogsrvprt><slogrowstat>Active</slogrowsta
t>
<hwfail>Yes</hwfail><assert>Yes</assert><faultalarm>Yes</faultalarm>
<faultwarn>Yes</faultwarn><warn>Yes</warn><action>Yes</action>
<info>Yes</info><debug>Yes</debug></syslog_ctl></monitors></device_ctl>
```

2. Show all syslog control info in JSON:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/monitors?js"
```

```
{
     "device_ctl": {
        "monitors": {
           "syslog_ctl": [
                {
                  "slogindex": "1",
                  "syslog": "Syslog TCP",
                  "slogsrvip": "10.85.222.79",
                  "slogsrvprt": "514",
                  "slogrowstat": "Active",
                  "hwfail": "Yes",
                  "assert": "Yes",
                  "faultalarm": "Yes",
                  "faultwarn": "Yes",
                  "warn": "Yes",
                  "action": "Yes",
                  "info": "Yes",
                  "debug": "Yes"
                },
                {
                  "slogindex": "2",
                  "syslog": "Disable",
                  "slogsrvip": "10.5.132.95",
                  "slogsrvprt": "514",
                  "slogrowstat": "Active",
                  "hwfail": "Yes",
                  "assert": "Yes",
                  "faultalarm": "Yes",
                  "faultwarn": "Yes",
                  "warn": "Yes",
                  "action": "Yes",
                  "info": "Yes",
                  "debug": "Yes"
                }
            ]
        }
    }
}
```

GET Method Example filtered to return data for single instance (for target with two instances of Monitor Setup for SYSLOG_CONTROL type pre-configured):

1. Show specific instance of syslog control info in XML:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/monitors/syslog_ctl?slogindex=2"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><device_
ctl><monitors><syslog_
ctl><slogindex>2</slogindex><syslog>Enable</syslog><slogsrvip>0.0.0.0</s
logsrvip>
<slogsrvprt>514</slogsrvprt><slogrowstat>Active</slogrowstat><hwfail>Yes
</hwfail>
<assert>Yes</assert><faultalarm>Yes</faultalarm><faultwarn>Yes</faultwar
n><warn>Yes</warn>
<action>Yes</action><info>Yes</info><debug>Yes</debug></syslog_
ctl></monitors></device_ctl>
```

2. Show specific instance of syslog control info in JSON:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/monitors?js&slogindex=2"
```

Expected output (values are for example purposes only):

```
{
    "device_ctl": {
        "monitors": {
            "syslog_ctl": {
                 "slogindex": "2",
                 "syslog": "Disable",
                 "slogsrvip": "10.5.132.95",
                 "slogsrvprt": "514",
                 "slogrowstat": "Active",
                 "hwfail": "Yes",
                 "assert": "Yes",
                 "faultalarm": "Yes",
                 "faultwarn": "Yes",
                 "warn": "Yes",
                 "action": "Yes",
                 "info": "Yes",
                 "debug": "Yes"
            }
        }
    }
}
```

POST Method Examples:

Change syslog control configuration for specific instance

1. Change syslog control configuration (XML format response expected):

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_ctl/monitors/syslog_
ctl?slogindex=2&faultalarm=Yes&faultwarn=Yes&assert=No&warning=No&info=N
o&debug=No&syslog=Syslog%20TCP"
```

If successful, the return body will be:

```
<response><code>10</code><result>success</result><message></message>
</response>
```

2. Change syslog control configuration (JSON format response expected):

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_ctl/monitors/syslog_
```

```
ctl?slogindex=2&faultalarm=Yes&faultwarn=Yes&assert=No&warning=No&info=N
o&debug=No&syslog=Syslog%20TCP&js"
```

If successful, the return body will be:

```
{
   "response": {
        "code": "10",
        "result": "success",
        "message": ""
   }
}
```

Monitors InfluxDB Device Control Command

Table 2.311 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/device_ctl/monitors/inflluxdb
Command Information	Allows getting and setting of controls for up to two instances of InfuxDB target destinations. This API works with the GET/POST ws/v2/device_ ctl/monitors/profiles API. **Note** There are two records of InfluxDB control data returned by the non-filtered GET version of this API. The state of each record (instance) is determined by the "Active" or "Inactive" value of the rowstatus parameter. After performing a factory reset (the default state of the platform), both of the records will have a rowstatus value of "Inactive". Adding or deleting an instance of InfluxDB may be accomplished via the web UI (System Settings > Monitoring > Monitoring Setup Connections Table) or via the POST ws/v2/device_ctl/monitors/profiles API. An individual record (instance) is accessed via idbindex=1 or idbindex=2 for this API with the corresponding use of the profiletype=”INFLUXDB” and csiid=1 or csiid=2 for the ws/v2/device_ctl/monitors/profiles API. **Note** Adding an instance via the profiles API or web UI changes the rowstatus value of the corresponding record returned by this API to "Active" and deleting an instance via the profiles API or web UI changes the rowstatus value to "Inactive" for the corresponding record. This API provides functionality that is equivalent to the following web UI feature: System Settings > Monitoring > Monitoring Setup > Select a row of the

Command Detail	Description
	Connections Table > Connection Configuration + Advanced INFLUXDB Filtering.
HTTP Method(s)	GET, POST
Access Type	Read, Write
Access Level	User, Admin
GET Syntax	Syslog protocol device control info for all instances: GET "https://192.168.0.1/ws/v2/device_ctl/monitors/influxdb " Syslog protocol device control info for specific instance: GET "https://192.168.0.1/ws/v2/device_ctl/monitors/influxdb?idbindex=1" Syslog protocol device control item for specific instance with some QueryParameter filters: GET "https://192.168.0.1/ws/v2/device_ ctl/monitors/influxdb?idbindex=1&enable&host&port&rowstatus"
POST Syntax	Disable influxdb for a specific record (instance): POST "https://192.168.0.1/ws/v2/device_ ctl/monitors/influxdb?idbindex=1&enable=No" Set parameters for influxdb for a specific record (instance): POST "https://192.168.0.1/ws/v2/device_ctl/monitors/influxdb? idbindex=1&enable=Yes&host=010%2E083%2E212%2E079&port=8086& database=all_units&httpuser=joe&httppwd=fubar&systemgrp=Yes& transgrp=Yes" **Note** The host value in ipv4 format must satisfy two conditions: ■ Prefix each octet number with zeros so that octet size is three characters long. ■ String must be URL encoded (period character encoded as %2E) when sent in the inline Query String Parameters format, as per example above.

Table 2.312 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
idbindex (Key)	Instance index of InfluxDB settings for a specific InfluxDB server target. Type: Integer.

URI Argument	Description
	Value: 1 or 2 Default: For POST command, the value must be specified. For GET, not specifying the value will return data for all the available instances.
rowstatus	InfluxDB control record status. The rowstatus for the corresponding idbindex instance will only become “Active” if the INFLUXDB profiletype instance row has been added via the web UI Monitoring Setup Connections Table or via the POST ws/v2/device_ctl/monitors/profiles API. Type: String. Value: "Active" or "Inactive".
enable	Enable or disable sending metrics for selected InfluxDB target. Type: String. Value: "No" (default) or "Yes".
host	Host target. This refers to the IPv4 format address of the server you have set up to receive InfluxDB metrics Type: String Value: IPv4 format IP Address Default: "" (empty string) Caveats: - Each octet of the IPv4 format address must be three characters wide. If an octet does not have three characters, then you must prefix the octet with zeros in order for the field width to always be three characters wide. For example, 10.83.212.79 becomes 010.083.212.079. - The string from above must be URL encoded when used as an inline Query Parameter such that the period character must be replaced with %2E. For example, the URL encoded string value for 010.083.212.079 is 010%2E083%2E212%079.
port	Port number. This is normally set to 8086, but it can be another value, depending on your server setup. Type: Integer. Value: 0 to 65535. Default: 0.
database	Database name. The name of the database storing your metrics.

URI Argument	Description
	Type: String. Value: unique name for the database. Default: "" (empty string)
httpuser	HTTP user name The user name of account on server for HTTP authentication purposes. Type: String Default: "" (empty string)
httppwd	HTTP user password. Password for user account on server, for HTTP authentication purposes. On a GET response, for security reasons, the value of the httppwd will not be exposed and it will be returned with an asterisk representing each character of the value. Type: String. Default: "" (empty string).
systemgrp	System group Enables the sending of CPU, memory, network, and temperature data. Type: String. Value: "No" (default) or "Yes".
zixigrp	Zixi group. Enables the sending of Zixi metrics (bit rate, packet rate, and so on). Type: String. Value: "No" (default) or "Yes".
abrgrp	ABR group Enables the sending of ABR metrics. Type: String. Value: "No" (default) or "Yes".
transgrp	Transport group. Enables the sending of transport metrics Type: String. Value: "No" (default) or "Yes".

URI Argument	Description
js	Format output using JSON standard. Type: exist. Values: any value or empty (for example, any of js=1 or js or js=0 or js=9999 are valid). Omitting this argument formats the output, by default, in XML.

GET Method Examples (for target with one instance of Monitor Setup for INFLUXDB type pre-configured):

Display influxdb control information in XML or JSON.

1. Show all influxdb control info in XML:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/monitors/influxdb"
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?><device_
ctl><monitors><influxdb><idbindex>1</idbindex><enable>Yes</enable>
<hostname>010.083.212.079</hostname><port>8086</port><database>all_
units</database><httpuser></httpuser><httppwd>*****</httppwd>
<systemgrp>Yes</systemgrp><zixigrp>No</zixigrp><abrgrp>No</abrgrp>
<transgrp>Yes</transgrp><rowstatus>Inactive</rowstatus></influxdb>
<influxdb><idbindex>2</idbindex><enable>No</enable><hostname>
</hostname><port>0</port><database></database><httpuser>
</httpuser><httppwd></httppwd><systemgrp>No</systemgrp>
<transgrp>No</transgrp><rowstatus>Inactive</rowstatus></influxdb></monit
ors></device_ctl>
```

2. Show all influxdb control info in JSON:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/monitors/influxdb&js"
```

Expected output (values are for example purposes only):

```
{
   "device_ctl": {
        "monitors": {
           "influxdb": [
                {
                   "idbindex": "1",
                   "enable": "Yes",
                   "hostname": "010.083.212.079",
                   "port": "8086",
                   "database": "all_units",
                   "httpuser": "joe",
                   "httppwd": "*****",
                   "systemgrp": "Yes",
                   "zixigrp": "No",
                   "abrgrp": "No",
                   "transgrp": "Yes",
                   "rowstatus": "Inactive"
                },
                {
                   "idbindex": "2",
                   "enable": "No",
                   "hostname": "",
                   "port": "0",
                   "database": "",
                   "httpuser": "",
                   "httppwd": "",
                   "systemgrp": "No",
                   "zixigrp": "No",
                   "abrgrp": "No",
                   "transgrp": "No",
                   "rowstatus": "Inactive"
                }
           ]
        }
   }
}
```

3. Show specific instance of influxdb control info in XML:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/monitors/influxdb?idbindex=1"
```

Expected output (values are for example purposes only):

```
<?xml version="1.0" encoding="ISO-8859-1" ?><device_
ctl><monitors><influxdb><idbindex>1</idbindex><enable>Yes</enable>
<hostname>010.083.212.079</hostname><port>8086</port><database>all_
units</database><httpuser>joe</httpuser><httppwd>*****</httppwd>
<systemgrp>Yes</systemgrp><zixigrp>No</zixigrp>
<abrgrp>No</abrgrp><transgrp>Yes</transgrp><rowstatus>Active</rowstatus>
</influxdb></monitors></device_ctl>
```

4. Show specific instance of influxdb control info in JSON:

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/device_
ctl/monitors/influxdb&js&idbindex=1"
```

Expected output (values are for example purposes only):

```
{
        "device_ctl": {
           "monitors": {
                "influxdb": {
                        "idbindex": "1",
                        "enable": "Yes",
                        "hostname": "010.083.212.079",
                        "port": "8086",
                        "database": "all_units",
                        "httpuser": "joe",
                        "httppwd": "*****",
                        "systemgrp": "Yes",
                        "zixigrp": "No",
                        "abrgrp": "No",
                        "transgrp": "Yes",
                        "rowstatus": "Inactive"
                   }
                }
        }
}
```

POST Method Examples:

Change influxdb control configuration for specific instance

1. Change influxdb control configuration (XML format response expected):

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_
ctl/monitors/influxdb?idbindex=1&enable=Yes&hostname=010%2E083%2E212%2E0
79&port=8086&database=all_
units&httpuser=joe&httppwd=fubar&systemgrp=Yes&transgrp=Yes"
```

   If successful, the return body will be:

```
<response><code>10</code><result>success</result><message></message>
</response>
```

2. Change influxdb control configuration (JSON format response expected):

```
curl -k -X POST -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v2/device_
```

```
ctl/monitors/influxdb?idbindex=1&enable=Yes&host=010%2E083%2E212%2E079&p
ort=8086&database=all_
units&httpuser=joe&httppwd=fubar&systemgrp=Yes&transgrp=Yes&js"
```

If successful, the return body will be:

```
{
        "response": {
            "code": "10",
            "result": "success",
            "message": ""
        }
}
```

Diagnostics API

This section contains APIs that Administrators and Users may use to diagnose specific functionality on a PowerVu Professional Receiver platform.

Note This is different from the APIs used for Diagnostics Package creation and retrieval (referred to as Export Debug File operations on the PowerVu Professional Receiver web UI).

IP Diagnostics Commands

The following is a list of the Client software requirements:

IP Diagnostics command may take up to 30 seconds, depending on status of the specified port and destination ip_address.

To avoid blocking of the PowerVu Professional Receiver integrated applications, and to implement the Progress Bar feature on the web UI, the required client software operation is a four phase approach:

1. Trigger a request for IP diagnostic.
2. Wait on status (Diagnostics Operation Status API).
3. Retrieve the output data of the command (via GET API response).
4. Format the raw data retrieved from the GET API response elements into a format (for example, text box Window on the web UI) that is useful to the user.

The APIs that are involved in Steps 1 to 3 above are described below.

IP Diagnostics Trigger Command

Table 2.313 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/ip_diags/trigger
Command Information	Trigger the ip_diags command.
HTTP Method	POST
Access Type	Write
Access Level	User, Admin
POST Syntax	POST "https://192.168.0.1/ws/v2/ip_diags?command=[command_option]& port=[ip_port_name]&ip=[target_ipv4_address]"

Table 2.314 URI Query/Set Arguments (possible fields and values preceded by ? and separated by &)

URI Argument	Description
command (mandatory)	Specify the command type. The types are: ping - performs ping command tracert - performs traceroute command
port (mandatory)	Specify the output port. **For multi-stream unit:** 0 : Management port 1: Data Port 1 2: Data Port 2 3: Data Port 3 4: Data Port 4 **For single-stream unit:** 0 : Management port 1: Data Port 1 2: Data Port 2

URI Argument	Description
ip (mandatory)	Specify the target IPv4 address. Type: String (ip dot format)

Example (issue the IP diagnostics trigger command):

Input (append &js for output in JSON format):

```
curl -k -H "X-SESSION-ID: $token" -X POST
"https://192.168.0.1/ws/v1/ip_
diags/trigger?command=ping&port=0&ip=192.168.0.72"
```

Expected output (for Success case which indicates only that IP Diag command was started):

In XML:

```
HTTP/1.1 200 OK
Date: Mon, 26 Jun 2017 11:58:09 GMT
Server: Hiawatha v9.14
Connection: keep-alive
Transfer-Encoding: chunked
Status: 200
Content-type: application/xml

<?xml version="1.0" encoding="ISO-8859-1"
?><response><code>0</code><result>success</result><message>ping data
collection is in
progress</message></response>
```

In JSON:

```
HTTP/1.1 200 OK
Date: Mon, 26 Jun 2017 12:00:15 GMT
Server: Hiawatha v9.14
Connection: keep-alive
Transfer-Encoding: chunked
Status: 200
Content-type: application/json

{
    "response": {
        "code": "0",
        "result": "success",
        "message": "ping data collection is in progress"
    }
}
```

Expected output (for Fail case which indicates that IP Diag Package Preparation was not started):

Note Additional failure reason details may be available by calling the DIAGSTAT API.

In XML:

```
HTTP/1.1 200 OK
Date: Mon, 26 Jun 2017 11:29:29 GMT
Server: Hiawatha v9.14
Connection: keep-alive
Transfer-Encoding: chunked
Status: 200
Content-type: application/xml

<?xml version="1.0" encoding="ISO-8859-1"
?><response><code>10</code><result>failure</result><message>failure to
trigger
request</message></response>
```

In JSON:

```
HTTP/1.1 200 OK
Date: Mon, 26 Jun 2017 12:00:15 GMT
Server: Hiawatha v9.14
Connection: keep-alive
Transfer-Encoding: chunked
Status: 200
Content-type: application/json

{
    "response": {
        "code": "10",
        "result": "failure",
        "message": "failure to trigger request "
    }
}
```

IP Diagnostics Operation Status Command

Table 2.315 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v1/table?t=IPDIAGSTAT
Command Information	Return information about IP Diagnostics Status.
HTTP Method	GET

Command Detail	Description
Access Type	Read
Access Level	User, Admin
GET Syntax	GET "https://192.168.0.1/ws/v1/table?t=IPDIAGSTAT"

Example (issue the IP DIAG Status command):

Input:

```
curl -k -H "X-SESSION-ID: $token"
"https://192.168.0.1/ws/v1/table?t=IPDIAGSTAT"
```

Expected output (example when diagnostics package preparation is in progress):

```
<?xml version="1.0" encoding="ISO-8859-1"
?><HDR><TABLE><RECORD><ITEM><ID>0x002D2600</ID><VALUE><![CDATA
[Inprogress]]></VALUE><NAME>
<![CDATA[OPERSTATUS]]></NAME><HELP_STR_1><![CDATA[Indicates whether the most
recent
operation passed, failed, or is in progress]]>
</HELP_STR_1><HELP_STR_2><![CDATA[This item has 3 options
available.]]></HELP_STR_2>
</ITEM><ITEM><ID>0x002D2601</ID><VALUE><![CDATA[Processing]]></VALUE><NAME><!
[CDATA[DETAILEDSTATUS]]>
</NAME><HELP_STR_1><![CDATA[State of the current Ping Traceroute collection
operation]]></HELP_STR_1>
<HELP_STR_2><![CDATA[This item has 6 options available.]]></HELP_STR_
2></ITEM><ITEM><ID>0x002D2602</ID>
<VALUE><![CDATA[23]]></VALUE><NAME><![CDATA[PERCENTCOMPLETE]]></NAME><HELP_
STR_1>
<![CDATA[Progress of the current Ping Traceroute Collection operation as a
percent]]>
</HELP_STR_1><HELP_STR_2><![CDATA[Item attribute: min(0), max(100), step size
(1), unit()]]>
</HELP_STR_2></ITEM></RECORD></TABLE></HDR>
```

Expected output (example when diagnostics package preparation is 100% completed successfully):

```
<?xml version="1.0" encoding="ISO-8859-1"
?><HDR><TABLE><RECORD><ITEM><ID>0x002D2600</ID><VALUE><![CDATA
[Pass]]></VALUE><NAME>
<![CDATA[OPERSTATUS]]></NAME><HELP_STR_1><![CDATA[Indicates whether the most
recent operation
passed, failed, or is in progress]]></HELP_STR_1><HELP_STR_2><![CDATA[This
item has 3 options
available.]]></HELP_STR_2></ITEM><ITEM><ID>0x002D2601</ID><VALUE><![CDATA
[Done]]></VALUE>
<NAME><![CDATA[DETAILEDSTATUS]]></NAME><HELP_STR_1><![CDATA[State of the
current
Ping Traceroute collection operation]]></HELP_STR_1><HELP_STR_2><![CDATA[This
item has 6 options
available.]]></HELP_STR_2></ITEM><ITEM><ID>0x002D2602</ID><VALUE><![CDATA
[100]]></VALUE>
<NAME><![CDATA[PERCENTCOMPLETE]]></NAME><HELP_STR_1><![CDATA[Progress of the
current Ping
Traceroute Collection operation as a percent]]></HELP_STR_1><HELP_STR_2>
<![CDATA[Item attribute: min(0), max(100), step size(1), unit()]]></HELP_STR_
2>
</ITEM></RECORD></TABLE></HDR>
```

Expected output (example when ip diagnostics operation did not succeed):

```
<?xml version="1.0" encoding="ISO-8859-1"
?><HDR><TABLE><RECORD><ITEM><ID>0x002D2600</ID><VALUE><![CDATA
[Fail]]></VALUE>
<NAME><![CDATA[OPERSTATUS]]></NAME><HELP_STR_1><![CDATA[Indicates whether the
most recent
operation passed, failed, or is in progress]]></HELP_STR_1><HELP_STR_2><!
[CDATA[This item
has 3 options available.]]></HELP_STR_
2></ITEM><ITEM><ID>0x002D2601</ID><VALUE><![CDATA[Failed]]>
</VALUE><NAME><![CDATA[DETAILEDSTATUS]]></NAME><HELP_STR_1><![CDATA[State of
the current Ping
Traceroute collection operation]]></HELP_STR_1><HELP_STR_2><![CDATA[This item
has 6 options
available.]]></HELP_STR_2></ITEM><ITEM><ID>0x002D2602</ID><VALUE><![CDATA
[100]]></VALUE>
<NAME><![CDATA[PERCENTCOMPLETE]]></NAME><HELP_STR_1><![CDATA[Progress of the
current
Ping Traceroute Collection operation as a percent]]></HELP_STR_1><HELP_STR_2>
<![CDATA[Item attribute: min(0), max(100), step size(1), unit()]]></HELP_STR_
2>
</ITEM></RECORD></TABLE></HDR>
```

Client software is expected to poll the status by calling this API every 5 seconds, while the Inprogress state is active, and until the OPERSTATUS is Pass and PERCENTCOMPLETE is 100 or OPERSTATUS is Fail. The first call will wait a minimum of 1 second, after the IP diagnostics trigger API, in order to allow the internal states to clear. If, at any time, the OPERSTATUS is Fail, then the client software will abort additional checking and report the last DETAILEDSTATUS.

IP Diagnostics Report Retrieval Command

Table 2.316 Command Details

Command Detail	Description
Command URL	https://192.168.0.1/ws/v2/ip_diags
Command Information	Retrieves the last IP diagnostics report (after last 100% completed operation status).
HTTP Method	GET
Access Type	Read
Access Level	User, Admin Restricted to Lock Level 0 only.
GET Syntax	GET "https://192.168.0.1/ws/v2/ip_diags"

Options: NA

Example (issue the IP diagnostics retrieval command):

Input (append &js for output in JSON format):

```
curl -k -H "X-SESSION-ID: $token" "https://192.168.0.1/ws/v2/ip_diags"
```

Expected output (example for Success case which indicates only that Diag Package Retrieval was completed):

XML format:

```
HTTP/1.1 200 OK
Date: Mon, 26 Jun 2017 13:00:58 GMT
Server: Hiawatha v9.14
Connection: keep-alive
Transfer-Encoding: chunked
Status: 200
Content-type: application/xml

<?xml version="1.0" encoding="ISO-8859-1" ?><response><code>0</code>
<result>success</result><message1>PING 10.132.24.34 (10.132.24.34): 56 data
bytes
64 bytes from 10.132.24.34: seq=0 ttl=61 time=0.819 ms
64 bytes from 10.132.24.34: seq=1 ttl=61 time=1.498 ms
64 bytes from 10.132.24.34: seq=2 ttl=61 time=0.740 ms
64 bytes from 10.132.24.34: seq=3 ttl=61 time=0.687 ms
64 bytes from 10.132.24.34: seq=4 ttl=61 time=0.677 ms

--- 10.132.24.34 ping statistics ---
```

```
5 packets transmitted, 5 packets rece</message1><message2>ved, 0 percent
packet loss
round-trip min/avg/max = 0.677/0.884/1.498 ms
</message2></response>
```

JSON format:

```
HTTP/1.1 200 OK
Date: Mon, 26 Jun 2017 13:02:55 GMT
Server: Hiawatha v9.14
Connection: keep-alive
Transfer-Encoding: chunked
Status: 200
Content-type: application/json

{
    "response": {
        "code": "0",
        "result": "success",
        "message1": "PING 10.132.24.34 (10.132.24.34):
56 data bytes\n64 bytes from 10.132.24.34: seq=0 ttl=61 time=0.819
ms\n64 bytes from 10.132.24.34: seq=1 ttl=61 time=1.498 ms\n64
bytes from 10.132.24.34: seq=2 ttl=61 time=0.740 ms\n64
bytes from 10.132.24.34: seq=3 ttl=61 time=0.687 ms\n64
bytes from 10.132.24.34: seq=4 ttl=61 time=0.677 ms\n\n--- 10.132.24.34
ping statistics ---\n5 packets transmitted, 5 packets rece",
        "message2": "ved, 0 percent packet loss\nround-trip min/avg/max =
0.677/0.884/1.498 ms\n"
    }
}
```

Expected output (example for Fail case which indicates that IP Diag report retrieval could not be completed):

Note Additional failure reason details may be available by calling the IPDIAGSTAT API.

You can put unrecognized parameter (for example, /extra) on command line to test this case:

```
curl -i -k -H "X-SESSION-ID: $token" "https://10.85.163.98/ws/v2/ip_
diags/extra"
HTTP/1.1 404 Not Found
Date: Mon, 26 Jun 2017 13:06:33 GMT
Server: Hiawatha v9.14
Connection: keep-alive
Transfer-Encoding: chunked
Status: 404
```