



MicroClimate Management System

REST API Developer Guide

MCMS 6.2

323-1961-306 - Standard - Revision A

March 2026

Copyright© 2024 - 2026 Ciena® Corporation. All rights reserved.

Contents

LEGAL NOTICES	vii
CHAPTER 1	
Overview	13
CHAPTER 2	
REST API	15
API overview	15
Architecture	16
Endpoint design	18
API programming model	20
API versioning	20
Document versioning	20
Security	21
Documentation	22
Examples	23
CHAPTER 3	
API references	25
Request Sequence	25
Request Format	25
Response format	26
HTTP methods	30
HTTP status codes	30
URL parameters	31
PON controllers	32
PON controller states	32
PON controller configurations	33
PON controller authentication states	33
PON controller engine state	34
PON controller alarm configurations	34
PON controller alarm histories	35
PON automation configurations	35
PON automation alarm configurations	36
PON automation alarm histories	37
PON automation states	37
PON controller statistics	38
PON controller logs	38
Switches	39
Switch configurations	39
PON automation configurations for switches	40
PON automation states for switches	40
PON automation alarm configurations for switches	41
PON automation alarm histories for switches	41
OLTs	42

OLT states	42
OLT configurations	43
OLT alarm configurations	43
OLT alarm histories	44
OLT PON automation configurations	45
OLT PON automation alarm configurations	45
OLT PON automation alarm histories	46
OLT PON automation states	46
OLT statistics	47
OLT logs	47
OLT debug	48
OLT actions	48
ONUs	49
ONU states	50
ONU configurations	50
ONU alarm configurations	51
OLT alarm histories	52
ONU PON automation configurations	52
ONU PON automation alarm configurations	53
ONU PON automation alarm histories	53
ONU PON automation states	54
ONU statistics	54
ONU logs	55
ONU CPEs	55
MIB reset states	56
MIB current states	56
ONU actions	57
Files	57
OLT firmware	58
ONU firmware	58
Pictures	59
SLAs	60
Downstream QoS maps	61
ONU service configurations	62
CPEs	62
PON automation	63
PON automation states	63
PON automation configurations	64
PON automation alarm configurations	64
PON automation alarm configurations for switches	65
PON automation alarm histories	66
PON automation alarm histories for switches	66
PON automation statistics	67
PON automation logs	67
PON automation tasks	68
PON automation tasks states	68
PON automation tasks configurations	68
PON automation tasks detailed states	69
Databases	70

Connection Configuration	70
Connection status	71
Database selection	72
Users	72
PON manager	73
CHAPTER 4	
Use cases and examples	75
Use case examples	75
Curl	75
Postman	76
Python	77
Programming model	78
List of procedures	78
Procedure 1 Determining the status of an ONU using CNTL-STATE	80
Procedure 2 Determining the registration status of an ONU using OLT-STATE	82
Procedure 3 Provisioning service for a subscriber on an ONU	84
Procedure 4 Pre-provisioning service for an ONU	87
Procedure 5 Pre-provisioning an OLT device	89
Procedure 6 Disabling service for an ONU	91
Procedure 7 Upgrading the firmware on an ONU	94
Procedure 8 Upgrading the firmware on many ONUs	96
Procedure 9 Resetting an ONU	99
Procedure 10 Associating a subscriber configuration to a device	100
Procedure 11 Disassociating a subscriber configuration from a device	101
Procedure 12 Creating an unassociated subscriber	102
CHAPTER 5	
Debugging and troubleshooting	103
Logs	103
PON manager logs	103
Traceback Logs	103
Apache logs	104
Errors	104

LEGAL NOTICES

THIS DOCUMENT CONTAINS CONFIDENTIAL AND TRADE SECRET INFORMATION OF CIENA CORPORATION AND ITS RECEIPT OR POSSESSION DOES NOT CONVEY ANY RIGHTS TO REPRODUCE OR DISCLOSE ITS CONTENTS, OR TO MANUFACTURE, USE, OR SELL ANYTHING THAT IT MAY DESCRIBE. REPRODUCTION, DISCLOSURE, OR USE IN WHOLE OR IN PART WITHOUT THE SPECIFIC WRITTEN AUTHORIZATION OF CIENA CORPORATION IS STRICTLY FORBIDDEN. EVERY EFFORT HAS BEEN MADE TO ENSURE THAT THE INFORMATION IN THIS DOCUMENT IS COMPLETE AND ACCURATE AT THE TIME OF PUBLISHING; HOWEVER, THE INFORMATION CONTAINED IN THIS DOCUMENT IS SUBJECT TO CHANGE.

While the information in this document is believed to be accurate and reliable, except as otherwise expressly agreed to in writing CIENA PROVIDES THIS DOCUMENT “AS IS” WITHOUT WARRANTY OR CONDITION OF ANY KIND, EITHER EXPRESS OR IMPLIED. The information and/or products described in this document are subject to change without notice. For the most up-to-date technical publications, visit www.ciena.com.

GAI Use Restriction. Unless expressly authorized in writing by Ciena on a case-by-case basis, you may not enter or use any Ciena Documentation in any Generative AI system, or include such information in the data for training such a system.

Copyright© 2024-2026 Ciena® Corporation All Rights Reserved

Use or disclosure of data contained in this document is subject to the Legal Notices and restrictions in this section and, unless governed by a valid license agreement signed between you and Ciena, the Licensing Agreement that follows.

The material contained in this document is also protected by copyright laws of the United States of America and other countries. It may not be reproduced or distributed in any form by any means, altered in any fashion, or stored in a data base or retrieval system, without express written permission of the Ciena Corporation.

Security

Ciena® cannot be responsible for unauthorized use of equipment and will not make allowance or credit for unauthorized use or access.

Contacting Ciena

Corporate Headquarters	410-694-5700 or 800-921-1144	www.ciena.com
Customer Technical Support/Warranty		www.ciena.com/support/

Sales and General Information	North America: 1-800-207-3714 International: +44 20 7012 5555	E-mail: sales@ciena.com
In North America	410-694-5700 or 800-207-3714	E-mail: sales@ciena.com
In Europe	+44-207-012-5500 (UK)	E-mail: sales@ciena.com
In Asia	+81-3-3248-4680 (Japan)	E-mail: sales@ciena.com
In India	+91-22-42419600	E-mail: sales@ciena.com
In Latin America	011-5255-1719-0220 (Mexico City)	E-mail: sales@ciena.com
Training		E-mail: learning@ciena.com

For additional office locations and phone numbers, please visit the Ciena web site at www.ciena.com.

READ THIS LICENSE AGREEMENT (“LICENSE”) CAREFULLY BEFORE INSTALLING OR USING CIENA SOFTWARE OR DOCUMENTATION. THIS LICENSE IS AN AGREEMENT BETWEEN YOU AND CIENA COMMUNICATIONS, INC. (OR, AS APPLICABLE, SUCH OTHER CIENA CORPORATION AFFILIATE LICENSOR) (“CIENA”) GOVERNING YOUR RIGHTS TO USE THE SOFTWARE. BY INSTALLING OR USING THE SOFTWARE, YOU ACKNOWLEDGE THAT YOU HAVE READ THIS LICENSE AND AGREE TO BE BOUND BY IT.

1. License Grant. Ciena may provide “Software” to you either (1) embedded within or running on a hardware product (together with Software, “Product”) or (2) as a standalone application, and Software includes upgrades acquired by you from Ciena or a Ciena authorized reseller. The terms of this License apply to your use of the Software and associated documentation whether such has been provided by Ciena, an affiliate of Ciena, or by means of an authorized reseller or distributor. Subject to these terms, and payment of all applicable License fees including any usage-based fees, Ciena grants you, as end user, a non-exclusive, non-transferable, personal License to use the Software only in object code form, subject to any applicable authorized use, activation requirements, usage levels, scope of functionality and release level of the Software, as set forth in the applicable quote accepted by Buyer upon Buyer’s issuance of an acceptable purchase order (“Order”), and in accordance with the detailed ordering information in the Ciena’s generally available, applicable, Product documentation as of the date of such Order. Unless the context does not permit, Software also includes associated documentation. Where an Order is for a (a) perpetual license, you may use the Software and associated documentation for as long as you use the Product for internal business use, or a (b) subscription license, you may only use the Software and associated documentation during the subscription term. A subscription license includes Software upgrades and/or technical support Services during the subscription term (that are not included in a perpetual license), in accordance with the Order and as further described in the applicable Ciena’s service description as of the date of the applicable Order. Prior to the expiration of each subscription term, Ciena will send you a quote for the annual renewal fee(s). To renew the subscription Software license(s) for additional subscription terms, you issue an Order in advance of the then-current expiration date of such subscription term.

2. Open Source and Third Party Licenses. If any Software is subject to an open-source license that provides the end user with rights that are broader than this License, then such rights shall take precedence. Ciena warrants that using Software in accordance with its documentation will

not subject you to any obligation to disclose, distribute or license your own software that interacts with Software.

3. Title. You are granted no title or ownership rights in or to the Software. Unless specifically authorized by Ciena in writing, you are not authorized to create any derivative works based upon the Software. Title to the Software, including any copies or derivative works based thereon, and to all copyrights, patents, trade secrets and other intellectual property rights in or to the Software, are and shall remain the property of Ciena and/or its licensors. Ciena's licensors are third party beneficiaries of this License. Ciena reserves to itself and its licensors all rights in the Software not expressly granted to you.

4. Confidentiality. The Software contains trade secrets of Ciena. Such trade secrets include, without limitation, the design, structure and logic of individual Software programs, their interactions with other portions of the Software, internal and external interfaces, and the programming techniques employed. The Software and related technical and commercial information, and other information received in connection with the purchase and use of the Software that a reasonable person would recognize as being confidential, are all confidential information of Ciena ("Confidential Information").

5. Obligations. You shall:

- i) Hold the Software and Confidential Information in strict confidence for the benefit of Ciena using your best efforts to protect the Software and Confidential Information from unauthorized disclosure or use, and treat the Software and Confidential Information with the same degree of care as you do your own similar information, but no less than reasonable care;
- ii) Keep a current record of the location of each copy of the Software you make;
- iii) Use the Software only in accordance with the authorized usage level;
- iv) Preserve intact any copyright, trademark, logo, legend or other notice of ownership on any original or copies of the Software, and affix to each copy of the Software you make, in the same form and location, a reproduction of the copyright notices, trademarks, and all other proprietary legends and/or logos appearing on the original copy of the Software delivered to you; and
- v) Issue instructions to your authorized personnel to whom Software is disclosed, advising them of the confidential nature of the Software and provide them with a summary of the requirements of this License.

6. Restrictions. You shall not:

- i) Use the Software or Confidential Information a) for any purpose other than your own internal business purposes; and b) other than as expressly permitted by this License;
- ii) Allow anyone other than your authorized personnel who need to use the Software in connection with your rights or obligations under this License to have access to the Software;
- iii) Make any copies of the Software except such limited number of copies, in machine readable form only, as may be reasonably necessary for execution in accordance with the authorized usage level or for archival purposes only;
- iv) Make any modifications, enhancements, adaptations, derivative works, or translations to or of the Software;
- v) Reverse engineer, disassemble, reverse translate, decompile, or in any other manner decode the Software;

vi) Make full or partial copies of the associated documentation or other printed or machine-readable matter provided with the Software unless it was supplied by Ciena in a form intended for reproduction;

vii) Export or re-export the Software and/or the associated documentation from the country in which it was received from Ciena or its authorized reseller unless authorized by Ciena in writing; or

viii) Publish the results of any benchmark tests run on the Software.

7. Audit: Upon Ciena's reasonable request you shall permit Ciena to audit the use of the Software to ensure compliance with this License.

8. U.S. Government Use. The Software is provided to the Government only with restricted rights and limited rights. Use, duplication, or disclosure by the Government is subject to restrictions set forth in FAR Sections 52-227-14 and 52-227-19 or DFARS Section 52.227-7013(C)(1)(ii), as applicable. The Software and any accompanying technical data (collectively "Materials") are commercial within the meaning of applicable Federal acquisition regulations. The Materials were developed fully at private expense. U.S. Government use of the Materials is restricted by this License, and all other U.S. Government use is prohibited. In accordance with FAR 12.212 and DFAR Supplement 227.7202, the Software is commercial computer software and the use of the Software is further restricted by this License.

9. Term of License. This License is effective until the applicable subscription term expires or the License is terminated. You may terminate this License by giving written notice to Ciena. This License will terminate immediately if (i) you breach any term or condition of this License or (ii) you become insolvent, cease to carry on business in the ordinary course, have a receiver appointed, enter into liquidation or bankruptcy, or any analogous process in your home country. Termination shall be without prejudice to any other rights or remedies Ciena may have. Upon any termination of this License, you shall destroy and erase all copies of the Software in your possession or control, and forward written certification to Ciena that all such copies of Software have been destroyed or erased. Your obligations to hold the Confidential Information in confidence, as provided in this License, shall survive the termination of this License.

10. Compliance with laws. You agree to comply with all laws related to your installation and use of the Software. Software is subject to U.S. export control laws and may be subject to export or import regulations in other countries. If Ciena authorizes you to import or export the Software in writing, you shall obtain all necessary licenses or permits and comply with all applicable laws.

11. Limitation of Liability. ANY LIABILITY OF CIENA SHALL BE LIMITED IN THE AGGREGATE TO THE AMOUNTS PAID BY YOU TO CIENA OR ITS AUTHORIZED RESELLER FOR THE SOFTWARE. THIS LIMITATION APPLIES TO ALL CAUSES OF ACTION, INCLUDING WITHOUT LIMITATION BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS. THE LIMITATIONS OF LIABILITY DESCRIBED IN THIS SECTION ALSO APPLY TO ANY LICENSOR OF CIENA. NEITHER CIENA NOR ANY OF ITS LICENSORS SHALL BE LIABLE FOR ANY INJURY, LOSS OR DAMAGE, WHETHER INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL INCLUDING WITHOUT LIMITATION ANY LOST PROFITS, CONTRACTS, DATA OR PROGRAMS, AND THE COST OF RECOVERING SUCH DATA OR PROGRAMS, EVEN IF INFORMED OF THE POSSIBILITY OF SUCH DAMAGES IN ADVANCE.

12. General. Ciena may assign this License to an affiliate or to a purchaser of the intellectual property rights in the Software. You shall not assign or transfer this License or any rights hereunder, and any attempt to do so will be void. This License shall be governed by the laws

of the State of New York without regard to conflict of laws provisions. The U.N. Convention on Contracts for the International Sale of Goods shall not apply hereto. This License constitutes the complete and exclusive agreement between the parties relating to the license for the Software and supersedes all proposals, communications, purchase orders, and prior agreements, verbal or written, between the parties. If any portion hereof is found to be void or unenforceable, the remaining provisions shall remain in full force and effect.

CHAPTER 1

Overview

This document describes the architecture and design of the MCMS RESTful Application Programming Interface (REST API), the structure of the REST endpoints, the request and response formats, and the reference and usage information for the REST API.

To build applications using the REST API effectively, developers need to understand the feature set and data model used to manage the PON network.

Although the open source MongoDB is shown as part of the MCMS architecture, MongoDB is not provided as part of the MCMS REST API package. MongoDB is a dependency of the REST API.

The following table lists the chapters in this document.

Table 1 Chapters in MCMS REST API developer guide

Chapter	Description
REST API	Describes the REST API interface and the various endpoints it manages.
API references	Describes listings and descriptions of API endpoints, message formats, applicable HTTP methods and response codes, and query parameters supported by the API.
Use cases and examples	Describes common use cases and examples using the MCMS REST API.
Debug and troubleshooting	Describes the location of log files and common errors.

Audience

This document is intended for developers building applications that use the REST API component to manage PON networks and devices.

Trademark and copyright acknowledgments

- Tibit™ MicroPlug™ OLT, and Tibit™ MicroClimate Management System® (MCMS) are registered trademarks of Ciena Corporation.

14 Overview

- Ciena® is a trademark of Ciena Corporation.

CHAPTER 2

REST API

The REST API is a component of the PON Manager that provides an application programming interface over HTTPS for managing PON devices.

Customers can build device provisioning, service configuration, performance monitoring, log collection, and other applications on top of the API for managing the PON network. In addition to customer applications, the PON Manager Web App utilizes the API's PON and user management interfaces for its operation. The API implements a JSON interface that aligns directly with the PON Controller management data model and interfaces with the MongoDB datastore for accessing configuration, state, statistics, logging, and file collections.

REST API implements endpoints for managing the following:

- device configuration and status for PON Automation, PON Controllers, OLTs, ONUs, and switches
- service configurations, including ONU Service Configuration (SRV-CFG) files, VLANs, Service Level Agreements (SLAs), 802.1X Authentication, DHCP Relay, and PPPoE
- performance management statistics
- device alarms and logging
- file management, including OLT firmware, ONU firmware, and device pictures

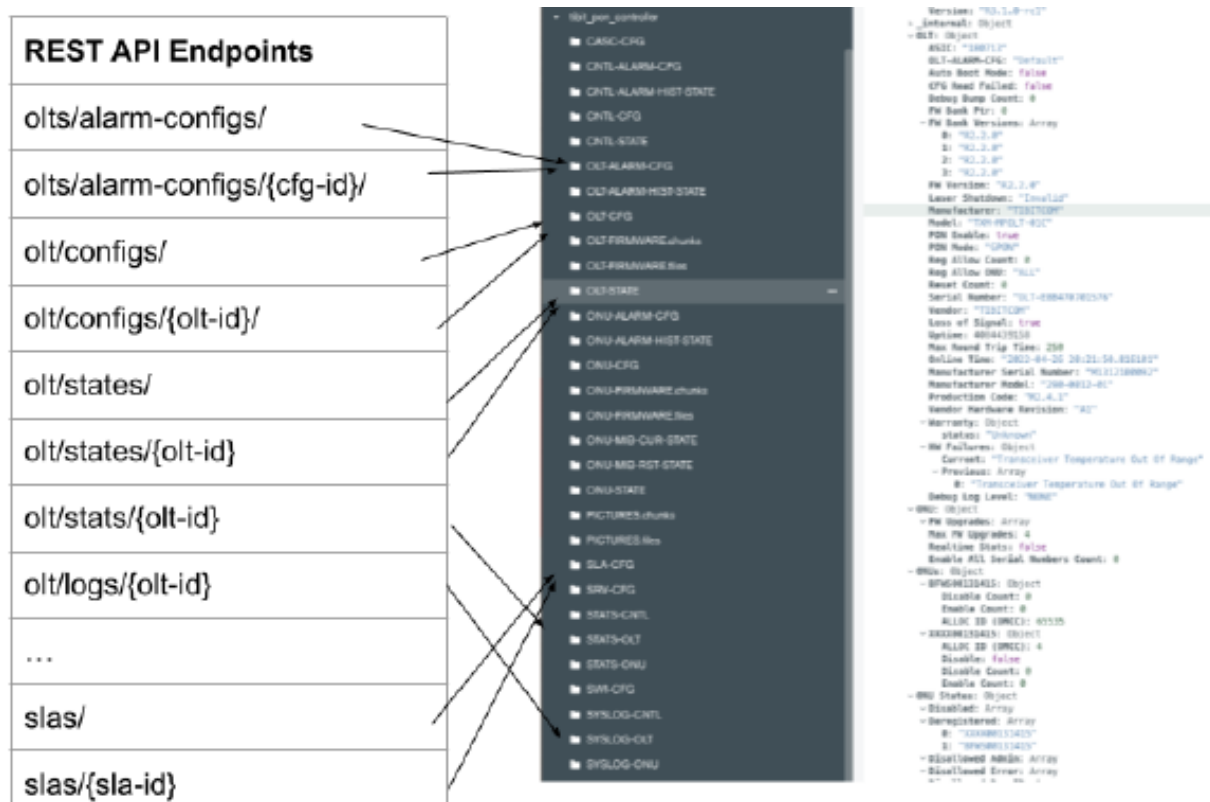
API overview

REST API is designed around best practices for a RESTful interface over a secure HTTPS transport and provides API endpoints for provisioning and monitoring OLT devices, as well as the subtended ONU devices and third-party ONUs compliant with the XGS-PON and 10G-EPON standards.

The API is aligned with the MCMS PON Controller data model in MongoDB, where most API endpoints and data map directly to collections and documents in MongoDB.

The following figure describes the REST API MongoDB collections and documents.

Figure 1 REST API MongoDB collections and documents

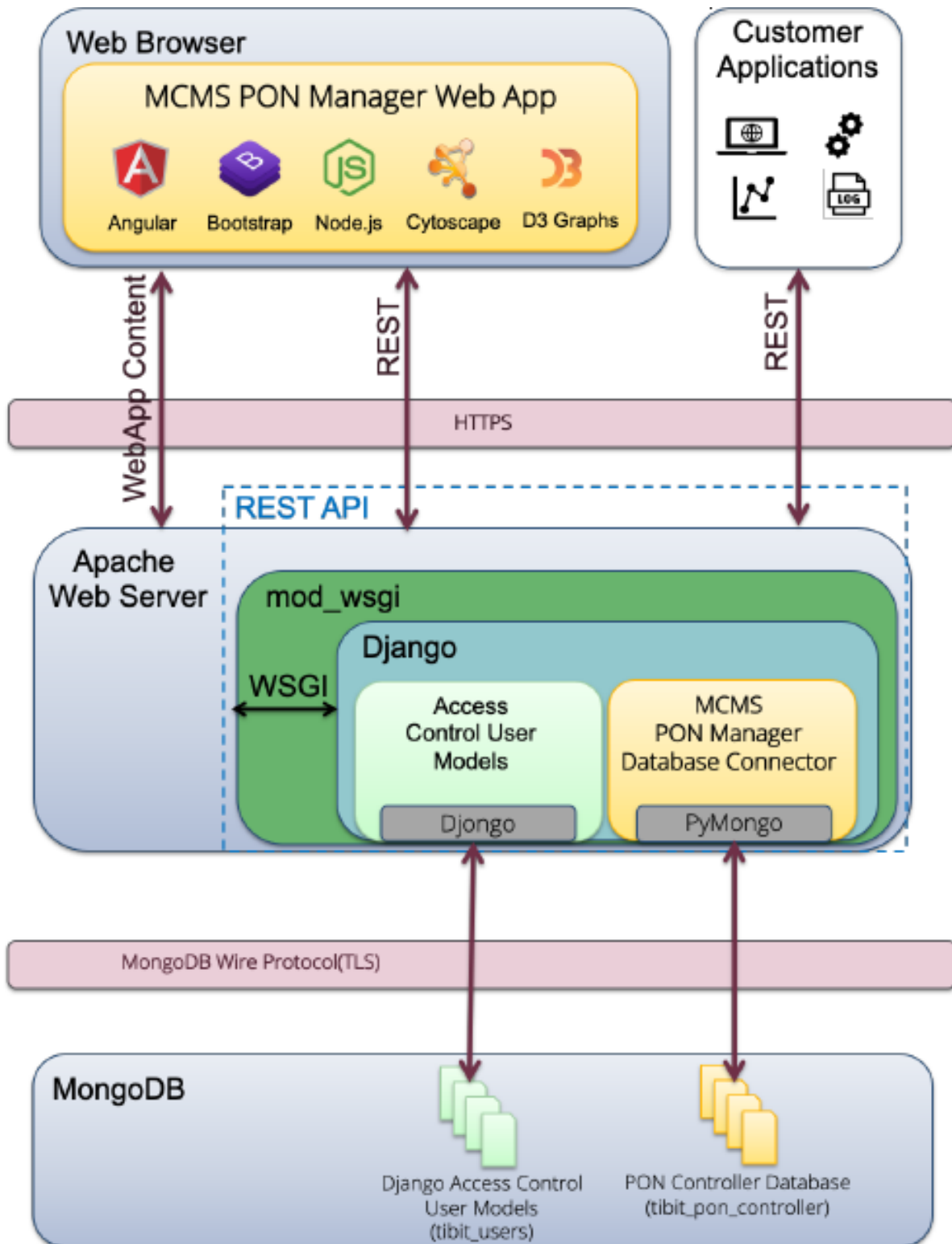


Architecture

The PON Manager software is composed of a graphical user front-end web application (Web App) and a REST API interface that provides access to the MongoDB datastore.

These software components integrate with the Apache2 web server and Django REST framework.

The following figure describes the MCMS PON manager architecture.

Figure 2 MCMS PON manager architecture

Web application

The core of the web app is built on the Angular framework and Bootstrap web front-end toolkit, along with libraries that support specific elements of the user interface.

The Cytoscape library provides network visualization utilities for graphical tools such as the Polyglot OMCI editor. Device statistics charts and graphs are built using the D3 Graphs package and ngx-charts library. The Font Awesome library provides icons and fonts for the user interface.

Web server

The web server is built on the Apache HTTP server, which hosts the web application graphical user interface, as well as the REST API for the PON Manager.

The `mod_wsgi` module provides an interface between the Apache server and Django REST framework through the web server gateway interface (WSGI) specification for Python. PON Manager supports the use of HTTPS for the REST API only. Non-secure HTTP is not supported.

Django REST framework

The REST API is built on the open source Django REST framework and toolkit for Python.

By default, Django does not support integration with non-relational databases. The Django library is used in parallel with a custom module to communicate with the MongoDB. As HTTP requests are received, they are handled by Django accordingly, but are mapped to the custom module for database operations instead of Django's default object relational model (ORM). API endpoints are implemented using the PyMongo library for interfacing with MongoDB.

The PON Manager uses the user, group, permissions, and session management features of the Django REST framework to provide secure access and authorization to the web interface.

MongoDB

The Mongo database provides the datastore for the MicroClimate™ Management System and is used to store PON device provisioning and monitoring information collected by the PON Controller.

MongoDB is an open source, secure database which employs a NoSQL architecture.

In addition to the PON device configuration and monitoring information, MongoDB is also used to store users, permissions, and session information used by the Django REST framework. The PON Controller defines the data model used to manage the PON network.

Endpoint design

The MCMS REST API URL format follows best practices for a “nouns, not verbs” REST API design, with endpoints designed around resources.

An endpoint refers to a URL available for use within the REST API and the HTTP method associated with the request. A resource is defined by the data collection and ID, where specific resources are identified by an ID in the URL after the collection mapping.

The API is designed to map to the PON Controller database model as closely as possible. Most URL endpoints map directly to a MongoDB collection. In the MCMS API, resources are:

- PON Controllers

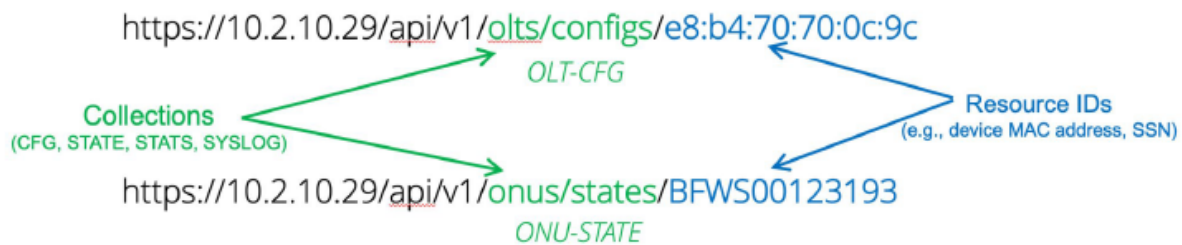
- OLTs
- ONU
- Switches
- SLAs
- files

Resource IDs are typically:

- MAC Addresses
- XGS-PON Serial Numbers
- file names

The following figure describes examples of URLs identifying a specific OLT configuration and ONU state.

Figure 3 URL identifying OLT configurations



When a resource ID contains a forward slash character, the ID must be double utf-8 encoded.

For example: `'/api/v1/slas/gold1g/1g' -> '/api/v1/slas/gold1g%252F1g'`

Message formats

MCMS REST API request and response messages only support JSON formatted payloads. POST and PUT requests use a common request body format with a data field containing the MongoDB document or other payload data required for the API.

In addition to the HTTP response status code, API responses contain status, data, and details fields providing detailed response information.

Request Format	Success Response Format	Error Response Format
{	{	{
- "data": {	- "status": - "success"	- "status": - "fail"
<MongoDB doc>	- "data": {...}	- "details": {...}
- }	- }	- }
- }		

Private APIs

The REST API implements several private endpoints only used by the PON Manager Web App.

Private APIs are not published in customer documentation and should not be used in customer applications. Private endpoints are not considered stable APIs and are subject to change from release to release.

API programming model

The REST API does not support PATCH or partial updating of documents.

To update a document, upload the entire document. The normal sequence of events is to retrieve a document, modify the document, and then PUT or POST the document.

API versioning

The MCMS REST API maintains its own version scheme.

The API version is identified at the beginning of the URL after the host information for every request. The format of the version string is v1 where the 1 is replaced by the required version number.

A new version of the API occurs:

- at any endpoint request message format changes
- at any endpoint response message format changes
- when required query parameters are added or removed

Endpoints are versioned independently of others. If a change is made to one endpoint, after v1 is released, it will now be accessible at v1 and v2. The v1 functionality is unchanged from its prior implementation. If accessed with v1 in the URL, the original unchanged action is performed. If accessed with v2, the new version is used. All endpoints are accessible via v1 or v2.

Note: MongoDB documents contained in API payload requests are versioned separately from the API version.

Document versioning

Each document in the database has a version that corresponds to the PON Controller version.

It defines the format or schema version for a specific document. This version determines the format of the document, and the set of fields in the document. Device configurations use [CNTL] [CFG Version] to determine the correct schema version. The document version is written when a PON Controller creates new documents. Upgrade a document to the latest schema version with the database upgrade feature.

MongoDB documents are versioned separately from the API. For example, in the following PUT request, the API versioning covers changes to the parts of the message highlighted in blue. This is independent of the format of the contents of the data field in the API message highlighted in red. The format of the contents of the data field is determined by the MongoDB document version as defined by the CNTL.CFG Version field.

The following figure describes the MongoDB document versions as defined by the CNTL.CFG version field.

Figure 4 MongoDB document version as defined by the CNTL.CFG version field

```

PUT: /v2/onus/configs/<ONU ID>/
{
  "data": {
    " id": "BFWS00123193",
    "CNTL": {
      "CFG Version": "R4.0.0"
    },
    ...
  }
}

```

Schema versioning

Schema for database documents are saved in the /opt/tibit/ponmgr/api/schema_files directory.

The API schema files are versioned based on the release. The schema version is based on the PON Controller configuration version of the document.

The following table describes the schema for validating input in API requests.

Table 2 Schema request types

Request type	Description
GET	The schema is used to validate query parameters, including the query filter and projection paths. The API returns a 400 error for an invalid query path.
PUT and POST	The schema is used to validate the format, data types, and ranges in documents provided with the requests. The API returns a warning if a required field is missing. The API returns a 400 error if an incorrect value is detected.

Security

The MCMS REST API uses the same secure HTTPS transport, user management, session-based authentication, and role-based access controls used by the PON Manager Web Application.

The following figure shows all users.

Figure 5 All users

All Users

Filter

Email ↑	Last Name	First Name	Assigned Roles	Date Created	Last Login ↑
anotheruser@tbitcom.com	user	another	None	9/17/2021, 9:33:00 AM	Never
api.user@tbitcom.com	User	API	API	11/16/2021, 1:59:08 PM	Never
All roles: API					
read.only@tbitcom.com	Only	Read	Read Only	2/3/2021, 2:33:52 PM	2/3/2021, 4:00:47 PM
tbit@tbitcom.com	Communications	Tibit	Administrators	10/5/2020, 9:58:24 AM	11/16/2021, 1:54:41 PM

Items per page: 10 1 - 4 of 4

EDIT

CREATE

Users can create and configure users, roles, and permissions to allow access through the REST interface. For example, configure one or more API-specific users to restrict access through the REST interface for managing ONUs and subscriber services, without providing access to more system administration functions.

The following figure shows all user roles and how to edit role API permissions.

Figure 6 Edit role API permissions

Users

Roles

All User Roles

Filter

Name ↑	User Count	Permission Count
API	1	30
Administrators	1	52
Read Only	1	13

Items per page: 10 1 - 3 of 3

CREATE

Edit role API permissions

Permission Type	Read	Update	Create	Delete	All
Databases	☐	☐	☐	☐	☐
Files	☒	☒	☒	☒	☒
Services	☐	☐	☐	☐	☐
Sites	☒	☒	☒	☒	☒
Network					
Controllers	☒	☒	☒	☒	☒
Ota	☒	☒	☒	☒	☒
Onus	☒	☒	☒	☒	☒
Switches	☒	☒	☒	☒	☒

Documentation

API documentation is provided with the PON Manager package.

An OpenAPI schema is provided in both YAML and JSON formats, which describes request and response formats, as well as response status codes and query parameters for a given endpoint. Parsable JSON schema files are provided for the PON management data structures defined

by the PON Controller database. These JSON schemas define the format for the request and response bodies transmitted in API HTTP messages.

The following table describes the location and descriptions for the JSON schemas.

Table 3 JSON file locations and descriptions

File/Directory	Description
/opt/tibit/ponmgr/	Root directory of the PON Manager application.
doc/R<version>/	Directory containing the JSON schema definition files for various document types defined for the MCMS PON Controller database.
doc/openapi.json	OpenAPI schema file describing the PON Manager REST API in a JSON file format.
doc/openapi.yaml	OpenAPI schema file describing the PON Manager REST API in a YAML file format.

Examples

API usage examples are provided with the PON Manager package. These examples include Python scripts, instructions on how to access the API through cURL commands (libcurl), and a Postman environment and collection.

The Python scripts show how to provision subscriber services and gather monitoring information for OLTs and ONUs on the PON network. A markdown file provides examples of cURL commands for use cases similar to the Python examples, such as provisioning a service on an ONU. Import postman files for use in the graphical interface to visualize requests and responses. All examples are found in the examples directory in the PON Manager installation directory.

The following table describes the location and description of example files.

Table 4 Example file locations and descriptions

File/Directory	Description
/opt/tibit/ponmgr/	Location of the root directory of the PON Manager application.
examples/python/	Directory containing Python example code for configuring and provisioning OLT and ONU devices and subscriber services.
examples/curl/	Directory containing a markdown file with cURL examples for accessing the REST API.
examples/postman/	Directory containing a Postman application environment and collection files. Import these files into Postman to access the REST API through the tool (https://www.postman.com/).

CHAPTER 3

API references

Developers use API references to build applications.

Request Sequence

Applications using the API follow the same general sequence described in this section.

Before sending requests to retrieve or modify data, applications log into and establish a secure authenticated session. All API requests include the authentication Session ID and CSRF cookies as well as the CSRF Token. When the application is done with the session, the application must logout and terminate the session.

API sessions follow this sequence:

- 1 Log in to authenticate the user with the web server.
POST -/v1/users/authenticate/
- 2 Select the database for this session. (This step is optional) If skipping this step, the default database is used.
PUT -/v1/databases/selection/
- 3 Execute one or more requests to fetch data and configure the database. For example:
GET -/v1/onus/configs/BFWS00123193
or
PUT -/v1/onus/configs/BFWS00123193
- 4 Log out of the web server.
GET -/v1/users/logout/

Request Format

The REST API requires specific HTTP headers and request body formats for all API requests it receives.

All API requests contain the following HTTP headers:

- Cookie: <Session ID Cookie>: Is used by the API to identify the user's session to verify the user is logged in and has permissions to access the requested endpoint.
- Cookie: <CSRF Token Cookie>: Is used by the API to identify the user's session to verify the user is logged in and has permissions to access the requested endpoint.

- X-CSRFToken: <CSRF Token>: Is used by the API to identify the user's session to verify the user is logged in and has permissions to access the requested endpoint.
- Referer: <API Host Address>: Lets the API know which client is making the request, for example, the PON Manager web UI.

Additionally, all PUT and POST requests contain the following content type header:

- Content-Type: application/json: Is required when the request contains a body. This tells the API that the content is JSON data and some other format.

All PATCH requests must contain the following content type header:

- Content-Type: application/json-patch+json

The cookies are returned after a successful login attempt. The CSRF Token cookie is the same value used for the X-CSRFToken header.

The REST API also expects specific formats of any bodies included in a request. PUT and POST requests follow a consistent request body format with an outer data tag containing the MongoDB document or other JSON payload as defined by the endpoint. The request body format is as follows:

```
{
  -"data": {
    <MongoDB document>
  }
}
```

API request bodies are specific to each endpoint. Refer to the documentation below for details on the specific request body format for a specific endpoint.

An example of a PUT request to select the active database:

```
PUT @ -/databases/selection/
Body: {
  -"data": -"DatabaseId"
}
```

PATCH requests contain an [RFC 6902] request body as follows:

```
Body: [
<JSON PATCH Operation>,
...
]
```

Response format

API responses have a consistent response body format.

The body contains a mix of the following:

- Requested data
- Status indicator
- Warning messages
- Error messages

All MCMS REST API responses use the following formats.

Success

Data validation warning

Data validation error

Server error

The API returns a success response payload for requests that the PON Manager can process and complete successfully. The response payload contains a status field that reports if the request was accepted and processed or if it encounters an error. GET responses also include a data field containing the requested data. The following is an example of a successful response after retrieving a resource:

```
GET @ -/onus/states/<ONU ID>/
{
  -"status": -"success",
  -"data": {
    <ONU-STATE MongoDB document>
  }
}
```

A successful PUT, POST, or PATCH request may have the following body with an HTTP status code of 200 or 201:

```
{
  -"status": -"success"
}
```

PUT, POST, and PATCH requests may return a status field containing validation warning. This means that the provided data in the request body does not match the defined schema. However, it is not a breaking difference and will not affect the MCMS functionality, so the request is processed as a success. The API updates the database and returns an HTTP status code 200. Validation warning responses include an additional details field that describes the warning in detail including the offending attribute name, the value, the schema path to the attribute, and a description of why validation failed.

The following is an example warning stating that the CNTL.CFG Version is a required field that is missing from the document provided in the PUT request:

PUT @ -/olts/<OLT ID>/

```
{
  -"status": -"validation warning",
  -"details": {
    -"level": -"warning",
    -"message": -"JSON validation failed: -'CFG Version' is a required -
property",
    -"collection": -"OLT-CFG",
    -"id": -"<OLT ID>",
    -"path": -"['CNTL']",
    -"bad value": {},
    -"bad value type": -"object",
    -"validator": -"required",
    -"schema": {
      -"type": -"object",
      -"properties": {
        -"CFG Version": {
          -"type": -"string",
          -"description": -"Capability of configuration file format."
        }
      }
    },
    -"required": ["CFG Version"]
  }
}
```

A validation error is similar to the validation warning described above. PUT, POST, or PATCH requests may return a status field containing fail with a JSON validation failure message. This error indicates that data validation failed due to the data provided in the request body not matching the schema defined for this document. Moreover, the request was NOT successful and no update was made to the database. The API returns this response with a 400 HTTP status code, indicating that the data provided is invalid and not usable.

The following is an example request setting an integer value of 0 for the auth_db fields that requires a string value:

```
PUT @ -/databases/Default/
{
  -"status": -"fail",
  -"details": {
    -"level": -"error",
    -"message": -"JSON validation failed: 0 is not of type -'string'",
    -"collection": -"DATABASES",
    -"id": null,
    -"path": -"['auth_db']",
    -"bad value": 0,
    -"bad value type": -"integer",
    -"validator": -"type",
    -"schema": {
      -"type": -"string",
      -"pattern": -"^([\^\\/\.\ -';$*<>:|?]+)$",
      -"maxlength": 63
    }
  }
}
```

Server errors returned with 400 or 500 level status codes include a status field with a value of fail in the response body. The Data Validation Error described above is a subset of the possible server error responses from the API. There are many ways an error response may occur, such as:

- A malformed request body
- Missing or invalid query parameters
- Missing permissions
- Internal server error

The following is an example that shows the 400 status code response body for a missing query parameter that is required:

```
GET @ -/olts/stats/<OLT ID>/
{
  -"status": -"fail",
  -"details": {
    -"message": -"Parameter -'start-time' is required"
  }
}
```

HTTP methods

There are four basic HTTP operations available for use in the MCMS REST API.

Not all HTTP methods are available on all endpoints. Each method maps to a CRUD operation. The equivalent MongoDB operations are also listed below. See the MongoDB documentation for more details on these operations.

The following table describes the different HTTP methods.

Table 5 HTTP methods

HTTP method	CRUD	MongoDB	Description
GET	Read	find, findOne	Used to retrieve data such as device configuration, state, statistics, and logs.
POST	Create	insertOne	Used to create a new resource on the server or database.
PUT	Update	replaceOne	Used to replace an existing or create a new resource configuration. This operation completely replaces the existing document with the new document from the request body.
PATCH	Update	updateOne	Used to update a portion of an existing document resource.
DELETE	Delete	deleteOne	Used to delete the configuration or state for the specified resource.

HTTP status codes

Listed below are the most common HTTP response codes returned by the MCMS REST API and their meaning.

Other status codes may also occur, but do not appear often.

The following table describes HTTP status codes.

Table 6 HTTP status codes

Status	Description
200 - OK	The request completed successfully with no errors.
201 - Created	The request completed successfully and created a new resource.
204 - No Content	The request completed successfully with no data returned.

Status	Description
400 - Bad Request	The request format or request data contained an error. This is often caused by a request data validation error.
401 - Unauthorized	The user was not authenticated. Returned after a failed login attempt.
403 - Forbidden	The request is not complete because the user is either not authenticated or does not have the required permissions.
404 - Not Found	The requested resource is not found. The error can occur if the URL does not exist or a file or database entry does not exist.
409 - Conflict	The requested resource is not created because the ID is already in use. This error can occur when uploading files or creating databases.
415 - Unsupported Media Type	Indicates that the Content-Type in the request is not supported.
500 - Server Error	A generic server error response that contains information about the failure in the response body.

URL parameters

There are various URL query parameters available within the MCMS REST API.

However, every endpoint supports its own subset of options. Query parameters available for each endpoint are found in the description of the endpoint.

Note: Note: some tools, such as cURL, require any space characters in the query parameters to be escaped as “%20”.

The following table describes URL query parameters.

Table 7 URL query parameters

Name	Description	Example
start-time	Only consider documents newer than the UTC timestamp.	2021-01-01 00:00:00
end-time	Only consider documents older than the UTC timestamp.	2021-01-01 00:00:00
query	Indicates a custom query on the MongoDB database. Takes a comma separated list of “<key>=<value>” strings, or a MongoDB query document. Note: the query document only supports comparison and logical operators.	CNTL.Version=R2.3.0 or CNTL.Version: R2.3.0

Name	Description	Example
projection	Limits the database attributes returned in the response. Use 1 to include, 0 to exclude. Takes a comma separated list of "<key>=0 1" strings.	CNTL.Version=1
sort	Sorts the response data on the given attribute in ascending or descending order.	_id=1
limit	Limits the number of items included in the response data. All endpoints that use the limit parameter use a default value of 1000.	50
skip	Indicates the number of items excluded from the beginning of the results data list.	10
next	A document _id that excludes all items from the results list that have this ID or come before it.	70:b3:d5:52:34:1e
distinct	A database attribute to retrieve all unique values for the endpoint's collection.	CNTL.Version

PON controllers

Developers use reference APIs for PON Controllers to add PON controller functionality to applications.

PON controller states

The /controllers/states/ resource and endpoints are used to monitor PON Controller devices. A controller resource is identified by the device MAC address and maps to a CNTL-STATE document in MongoDB.

The following table describes state HTTP methods and endpoints.

Table 8 State HTTP methods and endpoints

HTTP method	Endpoint
GET	GET /controllers/states/ Get a list of CNTL-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /controllers/states/<CNTL ID>/ Get a CNTL-STATE document identified by <CNTL ID> MAC address.
DELETE	DELETE /controllers/states/<CNTL ID>/ Delete a CNTL-STATE document identified by <CNTL ID> MAC address.

PON controller configurations

The /controllers/configs/ resource and endpoints are used to configure PON Controller devices.

A controller resource is identified by the device MAC address and maps to a CNTL-CFG document in MongoDB.

The following table describes the configuration HTTP methods and endpoints

Table 9 Configuration HTTP methods and endpoints

HTTP method	Endpoint
GET	GET /controllers/configs/ Get a list of CNTL-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /controllers/configs/ Create a CNTL-CFG document.
GET	GET /controllers/configs/<CNTL ID>/ Get a CNTL-CFG document identified by <CNTL ID> MAC address.
PUT	PUT /controllers/configs/<CNTL ID>/ Create or replace a CNTL-CFG document identified by <CNTL ID> MAC address.
PATCH	PATCH /controllers/configs/<CNTL ID>/ Modify a CNTL-CFG document identified by <CNTL ID> MAC address.
DELETE	DELETE /controllers/configs/<CNTL ID>/ Delete a CNTL-CFG document identified by <CNTL ID> MAC address.

PON controller authentication states

The /controllers/auth/states/ resource and endpoints monitor the MCMS Authenticator, which is packaged with the PON Controller.

An authenticator state resource is identified by the device MAC address and maps to a CNTL-AUTH-STATE document in MongoDB.

The following table describes authentication state HTTP methods and endpoints.

Table 10 Authentication state HTTP methods and endpoints

HTTP method	Endpoint
GET	GET /controllers/auth/states/ Get a list of CNTL-AUTH-STATE documents.

HTTP method	Endpoint
GET	GET /controllers/auth/states/<CNTL ID>/ Get a CNTL-AUTH-STATE document identified by <CNTL ID>.
DELETE	DELETE /controllers/auth/states/<CNTL ID>/ Delete a CNTL-AUTH-STATE document identified by <CNTL ID>.

PON controller engine state

The /controllers/engine/states/ resource and endpoints monitor the MCMS UMT Relay process, which is packaged with the PON Controller.

An engine state resource is identified by the device MAC Address and maps to a CNTL-ENGINE-STATE document in MongoDB.

The following table describes the engine state HTTP methods and endpoints.

Table 11 Engine state HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /controllers/engine/states/ Get a list of CNTL-ENGINE-STATE documents.
GET	GET /controllers/engine/states/<CNTL ID>/ Get a CNTL-ENGINE-STATE document identified by <CNTL ID>.
DELETE	DELETE /controllers/engine/states/<CNTL ID>/ Delete a CNTL-ENGINE-STATE document identified by <CNTL ID>.

PON controller alarm configurations

The /controllers/alarm-configs/ resource and endpoints configure PON Controller alarm profiles.

An alarm configuration resource is identified by the configuration ID and maps to a CNTL-ALARM-CFG document in MongoDB.

The following table describes the alarm configuration HTTP methods and endpoints.

Table 12 Alarm configuration HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /controllers/alarm-configs/ Get a list of CNTL-ALARM-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /controllers/alarm-configs/

HTTP methods	Endpoints
	Create a CNTL-ALARM-CFG document.
GET	GET /controllers/alarm-configs/<CFG ID>/ Get a CNTL-ALARM-CFG document identified by <CFG ID>.
PUT	PUT /controllers/alarm-configs/<CFG ID>/ Create or replace a CNTL-ALARM-CFG document identified by <CFG ID>.
DELETE	DELETE /controllers/alarm-configs/<CFG ID>/ Delete a CNTL-ALARM-CFG document identified by <CFG ID>.

PON controller alarm histories

The /controllers/alarm-histories/ resource and endpoints get PON controller alarm histories.

An alarm histories resource is identified by the device MAC Address and maps to a CNTL-ALARM-HIST-STATE document in MongoDB.

The following table describes the PON controller alarm histories HTTP methods and endpoints.

Table 13 PON controller alarm histories HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /controllers/alarm-histories/ Get a list of CNTL-ALARM-HIST-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /controllers/alarm-histories/<CNTL ID>/ Get a CNTL-ALARM-HIST-STATE document identified by <CNTL ID>.
DELETE	DELETE /controllers/alarm-histories/<CNTL ID>/ Delete a CNTL-ALARM-HIST-STATE document identified by <CNTL ID>.

PON automation configurations

The /controllers/automation/configs/ resource and endpoints configure PON Automation for PON Controllers.

A PON Controller automation resource is identified by the device MAC Address or “Global” and maps to a CNTL-AUTO-CFG document in MongoDB.

The following table describes the PON controller automation configuration HTTP methods and endpoints.

Table 14 PON controller automation configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /controllers/automation/configs/ Get a list of CNTL-AUTO-CFG documents.
POST	POST /controllers/automation/configs/ Create a CNTL-AUTO-CFG document.
GET	GET /controllers/automation/configs/<CFG ID>/ Get a CNTL-AUTO-CFG document identified by <CFG ID>.
PUT	PUT /controllers/automation/configs/<CFG ID>/ Create or replace a CNTL-AUTO-CFG document identified by <CFG ID>.
DELETE	DELETE /controllers/automation/configs/<CFG ID>/ Delete a CNTL-AUTO-CFG document identified by <CFG ID>.

PON automation alarm configurations

The /controllers/automation/alarm-configs/ resource and endpoints configure PON automation PON controller alarm profiles.

An alarm configuration resource is identified by the configuration ID and maps to a CNTL-AUTO-ALARM-CFG document in MongoDB.

The following table describes the PON automation alarm configurations HTTP methods and endpoints.

Table 15 PON automation alarm configurations HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /controllers/automation/alarm-configs/ Get a list of CNTL-AUTO-ALARM-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /controllers/automation/alarm-configs/ Create a CNTL-AUTO-ALARM-CFG document.
GET	GET /controllers/automation/alarm-configs/<CFG ID>/ Get a CNTL-AUTO-ALARM-CFG document identified by <CFG ID>.
PUT	PUT /controllers/automation/alarm-configs/<CFG ID>/

HTTP method	Endpoints
	Create or replace a CNTL-AUTO-ALARM-CFG document identified by <CFG ID>.
DELETE	DELETE /controllers/automation/alarm-configs/<CFG ID>/ Delete a CNTL-AUTO-ALARM-CFG document identified by <CFG ID>.

PON automation alarm histories

The /controllers/automation/alarm-histories/ resource and endpoints get PON Automation PON Controller alarm histories.

An alarm histories resource is identified by the device MAC Address and maps to a CNTL-AUTO-ALARM-HIST-STATE document in MongoDB.

The following table describes the PON automation alarm histories HTTP methods and endpoints.

Table 16 PON automation alarm histories HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /controllers/automation/alarm-histories/ Get a list of CNTL-AUTO-ALARM-HIST-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /controllers/automation/alarm-histories/<CNTL ID>/ Get a CNTL-AUTO-ALARM-HIST-STATE document identified by <CNTL ID>.
DELETE	DELETE /controllers/automation/alarm-histories/<CNTL ID>/ Delete a CNTL-AUTO-ALARM-HIST-STATE document identified by <CNTL ID>.

PON automation states

The /controllers/automation/states/ resource and endpoints monitor PON Automation for PON Controllers.

A PON Controller automation resource is identified by the device MAC address and maps to a CNTL-AUTO-STATE document in MongoDB.

The following table describes the PON automation state HTTP methods and endpoints.

Table 17 PON automation state HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /controllers/automation/states/

HTTP methods	Endpoints
	Get a list of CNTL-AUTO-STATE documents.
GET	GET /controllers/automation/states/<CNTL ID>/ Get a CNTL-AUTO-STATE document identified by <CNTL ID>.
DELETE	DELETE /controllers/automation/states/<CNTL ID>/ Delete a CNTL-AUTO-STATE document identified by <CNTL ID>.

PON controller statistics

The /controllers/stats/ resource and endpoints monitor statistics for PON Controllers.

A PON Controller resource is identified by the device MAC address and maps to the STATS-CNTL-<CNTL ID> collection in versions R3.0.0 and earlier and STATS-CNTL in versions R3.1.0 and later in MongoDB.

The following table describes the statistics for HTTP methods and endpoints.

Table 18 Statistics for HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /controllers/stats/<CNTL ID>/ Get a list of STATS-CNTL documents. URL Parameters: start-time (required), end-time (optional)
DELETE	DELETE /controllers/stats/<CNTL ID>/ Delete the STATS-CNTL collection.

PON controller logs

The /controllers/logs/ resource and endpoints monitor logs for PON Controllers.

A PON Controller resource is identified by the device MAC address and maps to the SYSLOG-CNTL-<CNTL ID> collection in versions R3.0.0 and earlier and SYSLOG-CNTL in versions R3.1.0 and later in MongoDB.

The following table describes the logs for HTTP methods and endpoints.

Table 19 Logs for HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /controllers/logs/<CNTL ID>/ Get a list of SYSLOG-CNTL documents. URL Parameters: start-time (required), end-time (optional)
DELETE	DELETE /controllers/logs/<CNTL ID>/

HTTP methods	Endpoints
	Delete the SYSLOG-CNTL collection.

Switches

Developers use reference APIs for switches to add switch functionality to applications.

Switch configurations

The /switches/configs/ resource and endpoints configure switches.

A switch resource is identified by the device MAC address and maps to a SWI-CFG document in MongoDB.

The following table describes the switch configuration HTTP methods and endpoints.

Table 20 Switch configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /switches/configs/ Get a list of SWI-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /switches/configs/ Create a SWI-CFG document.
GET	GET /switches/configs/<SWI ID>/ Get a SWI-CFG document identified by <SWI ID> MAC address.
PUT	PUT /switches/configs/<SWI ID>/ Create or replace a SWI-CFG document identified by <SWI ID> MAC address.
PATCH	PATCH /switches/configs/<SWI ID>/ Modify a SWI-CFG document identified by <SWI ID> MAC address.
DELETE	DELETE /switches/configs/<SWI ID>/ Delete a SWI-CFG document identified by <SWI ID> MAC address.

Note: The Netconf.Name field is used by the MCMS Netconf interface to allow user specific naming to reference device configuration. This field indexed in MongoDB and must be unique. The API will not allow a duplicate Netconf.Name to be created. By default, this can be the same as _id for the device.

PON automation configurations for switches

The /switches/automation/configs/ resource and endpoints configure PON Automation for Switches.

A Switch automation resource is identified by the device MAC address or Global and maps to a SWI-AUTO-CFG document in MongoDB.

The following table describes the PON automation configurations for switches HTTP methods and endpoints.

Table 21 PON automation configurations for switches HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /switches/automation/configs/ Get a list of SWI-AUTO-CFG documents.
POST	POST /switches/automation/configs/ Create a SWI-AUTO-CFG document.
GET	GET /switches/automation/configs/<CFG ID>/ Get a SWI-AUTO-CFG document identified by <CFG ID>.
PUT	PUT /switches/automation/configs/<CFG ID>/ Create or replace a SWI-AUTO-CFG document identified by <CFG ID>.
DELETE	DELETE /switches/automation/configs/<CFG ID>/ Delete a SWI-AUTO-CFG document identified by <CFG ID>.

PON automation states for switches

The /switches/automation/states/ resource and endpoints monitor PON Automation for switches.

A switch automation resource is identified by the device MAC address and maps to a SWI-AUTO-STATE document in MongoDB.

The following table describes the PON automation states for switch HTTP methods and endpoints.

Table 22 PON automation states for switch HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /switches/automation/states/ Get a list of SWI-AUTO-STATE documents.
GET	GET /switches/automation/states/<SWI ID>/ Get a SWI-AUTO-STATE document identified by <SWI ID>.
DELETE	DELETE /switches/automation/states/<SWI ID>/

HTTP method	Endpoints
	Delete a SWI-AUTO-STATE document identified by <SWI ID>.

PON automation alarm configurations for switches

The /switches/automation/alarm-configs/ resource and endpoints configure PON automation switch alarm profiles.

An alarm configuration resource is identified by the configuration ID and maps to a SWI-AUTO-ALARM-CFG document in MongoDB.

The following table describes the PON automation switch alarm profiles HTTP methods and endpoints.

Table 23 PON automation switch alarm profiles HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /switches/automation/alarm-configs/ Get a list of SWI-AUTO-ALARM-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /switches/automation/alarm-configs/ Create a SWI-AUTO-ALARM-CFG document.
GET	GET /switches/automation/alarm-configs/<CFG ID>/ Get a SWI-AUTO-ALARM-CFG document identified by <CFG ID>.
PUT	PUT /switches/automation/alarm-configs/<CFG ID>/ Create or replace a SWI-AUTO-ALARM-CFG document identified by <CFG ID>.
DELETE	DELETE /switches/automation/alarm-configs/<CFG ID>/ Delete a SWI-AUTO-ALARM-CFG document identified by <CFG ID>.

PON automation alarm histories for switches

The /switches/automation/alarm-histories/ resource and endpoints get PON automation switch alarm histories.

An alarm histories resource is identified by the device MAC Address and maps to a SWI-AUTO-ALARM-HIST-STATE document in MongoDB.

The following table describes the PON automation switch alarm histories HTTP methods and endpoints.

Table 24 PON automation switch alarm histories HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /switches/automation/alarm-histories/ Get a list of SWI-AUTO-ALARM-HIST-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /switches/automation/alarm-histories/<SWI ID>/ Get a SWI-AUTO-ALARM-HIST-STATE document identified by <SWI ID>.
DELETE	DELETE /switches/automation/alarm-histories/<SWI ID>/ Delete a SWI-AUTO-ALARM-HIST-STATE document identified by <SWI ID>.

OLTs

Developers use reference APIs for OLTs to add OLT functionality to applications.

OLT states

The /olts/states/ resource and endpoints monitor OLT devices.

An OLT resource is identified by the device MAC address and maps to an OLT-STATE document in MongoDB.

The following table describes the OLT states for HTTP methods and endpoints.

Table 25 OLT states for HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/states/ Get a list of OLT-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /olts/states/<OLT ID>/ Get an OLT-STATE document identified by <OLT ID> MAC address.
DELETE	DELETE /olts/states/<OLT ID>/ Delete an OLT-STATE document identified by <OLT ID> MAC address.

OLT configurations

The /olts/configs/ resource and endpoints monitor OLT devices.

An OLT resource is identified by the device MAC address and maps to an OLT-CFG document in MongoDB.

The following table describes the OLT configuration HTTP methods and endpoints.

Table 26 OLT configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/configs/ Get a list of OLT-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /olts/configs/ Create an OLT-CFG document.
GET	GET /olts/configs/<OLT ID>/ Get an OLT-CFG document identified by <OLT ID> MAC address.
PUT	PUT /olts/configs/<OLT ID>/ Create or replace an OLT-CFG document identified by <OLT ID> MAC address.
PATCH	PATCH /olts/configs/<OLT ID>/ Modify an OLT-CFG document identified by <OLT ID> MAC address.
DELETE	DELETE /olts/configs/<OLT ID>/ Delete an OLT-CFG document identified by <OLT ID> MAC address.

Note: The Netconf.Name field is used by the MCMS Netconf interface to allow user specific naming to reference device configuration. This field indexed in MongoDB and must be unique. The API will not allow a duplicate Netconf.Name to be created. By default, this can be the same as _id for the device.

OLT alarm configurations

The /olts/alarm-configs/ resource and endpoints configure OLT alarm profiles.

An alarm configuration resource is identified by the configuration ID and maps to an OLT-ALARM-CFG document in MongoDB.

The following table describes the OLT alarm configuration HTTP methods and endpoints.

Table 27 OLT alarm configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/alarm-configs/ Get a list of OLT-ALARM-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /olts/alarm-configs/ Create an OLT-ALARM-CFG document.
GET	GET /olts/alarm-configs/<CFG ID>/ Get an OLT-ALARM-CFG document identified by <CFG ID>.
PUT	PUT /olts/alarm-configs/<CFG ID>/ Create or replace an OLT-ALARM-CFG document identified by <CFG ID>.
DELETE	DELETE /olts/alarm-configs/<CFG ID>/ Delete an OLT-ALARM-CFG document identified by <CFG ID>.

OLT alarm histories

The /olts/alarm-histories/ resource and endpoints get OLT alarm histories.

An alarm histories resource is identified by the device MAC Address and maps to an OLT-ALARM-HIST-STATE document in MongoDB.

The following table describes the OLT alarm histories HTTP methods and endpoints.

Table 28 OLT alarm histories HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/alarm-histories/ Get a list of OLT-ALARM-HIST-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /olts/alarm-histories/<OLT ID>/ Get an OLT-ALARM-HIST-STATE document identified by <OLT ID>.
DELETE	DELETE /olts/alarm-histories/<OLT ID>/ Delete an OLT-ALARM-HIST-STATE document identified by <OLT ID>.

OLT PON automation configurations

The /olts/automation/configs/ resource and endpoints configure PON Automation for OLTs.

An OLT automation resource is identified by the device MAC address or Global and maps to an OLT-AUTO-CFG document in MongoDB.

The following table describes the OLT PON automation configuration HTTP methods and endpoints.

Table 29 OLT PON automation configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/automation/configs/ Get a list of OLT-AUTO-CFG documents.
POST	POST /olts/automation/configs/ Create an OLT-AUTO-CFG document.
GET	GET /olts/automation/configs/<CFG ID>/ Get an OLT-AUTO-CFG document identified by <CFG ID>.
PUT	PUT /olts/automation/configs/<CFG ID>/ Create or replace an OLT-AUTO-CFG document identified by <CFG ID>.
DELETE	DELETE /olts/automation/configs/<CFG ID>/ Delete an OLT-AUTO-CFG document identified by <CFG ID>.

OLT PON automation alarm configurations

The /olts/automation/alarm-configs/ resource and endpoints configure PON automation OLT alarm profiles.

An alarm configuration resource is identified by the configuration ID and maps to an OLT-AUTO-ALARM-CFG document in MongoDB.

The following table describes the PON automation OLT alarm profiles HTTP methods and endpoints.

Table 30 PON automation OLT alarm profiles HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/automation/alarm-configs/ Get a list of OLT-AUTO-ALARM-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /olts/automation/alarm-configs/

HTTP method	Endpoints
	Create an OLT-AUTO-ALARM-CFG document.
GET	GET /olts/automation/alarm-configs/<CFG ID>/ Get an OLT-AUTO-ALARM-CFG document identified by <CFG ID>.
PUT	PUT /olts/automation/alarm-configs/<CFG ID>/ Create or replace an OLT-AUTO-ALARM-CFG document identified by <CFG ID>.
DELETE	DELETE /olts/automation/alarm-configs/<CFG ID>/ Delete an OLT-AUTO-ALARM-CFG document identified by <CFG ID>.

OLT PON automation alarm histories

The /olts/automation/alarm-histories/ resource and endpoints get PON automation OLT alarm histories.

An alarm histories resource is identified by the device MAC Address and maps to an OLT-AUTO-ALARM-HIST-STATE document in MongoDB.

The following table describes the PON automation OLT alarm histories HTTP methods and endpoints.

Table 31 PON automation OLT alarm histories HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /olts/automation/alarm-histories/ Get a list of OLT-AUTO-ALARM-HIST-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /olts/automation/alarm-histories/<OLT ID>/ Get an OLT-AUTO-ALARM-HIST-STATE document identified by <OLT ID>.
DELETE	DELETE /olts/automation/alarm-histories/<OLT ID>/ Delete an OLT-AUTO-ALARM-HIST-STATE document identified by <OLT ID>.

OLT PON automation states

The /olts/automation/states/ resource and endpoints monitor PON automation for OLTs.

An OLT automation resource is identified by the device MAC address and maps to an OLT-AUTO-STATE document in MongoDB.

The following table describes the OLT PON automation state HTTP methods and endpoints.

Table 32 OLT PON automation state HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/automation/states/ Get a list of OLT-AUTO-STATE documents.
GET	GET /olts/automation/states/<OLT ID>/ Get an OLT-AUTO-STATE document identified by <OLT ID>.
DELETE	DELETE /olts/automation/states/<OLT ID>/ Delete an OLT-AUTO-STATE document identified by <OLT ID>.

OLT statistics

The /olts/stats/ resource and endpoints monitor statistics for OLTs.

An OLT resource is identified by the device MAC address and maps to the STATS-OLT-<OLT ID> collection in versions R3.0.0 and earlier and STATS-OLT in versions R3.1.0 and later in MongoDB.

The following table describes the OLT statistic HTTP methods and endpoints.

Table 33 OLT statistic HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/stats/<OLT ID>/ Get a list of STATS-OLT documents. URL Parameters: start-time (required), end-time (optional)
DELETE	DELETE /olts/stats/<OLT ID>/ Delete the STATS-OLT collection.

OLT logs

The /olts/logs/ resource and endpoints monitor logs for OLTs.

An OLT resource is identified by the device MAC address and maps to the SYSLOG-OLT-<OLT ID> collection in versions R3.0.0 and earlier and SYSLOG-OLT in versions R3.1.0 and later in MongoDB.

The following table describes the OLT log HTTP methods and endpoints.

Table 34 OLT log HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/logs/<OLT ID>/ Get a list of SYSLOG-OLT documents. URL Parameters: start-time (required), end-time (optional)
DELETE	DELETE /olts/logs/<OLT ID>/ Delete the SYSLOG-OLT collection.

OLT debug

The /olts/debug/ resource and endpoints monitor debug information for OLTs.

An OLT resource is identified by the device MAC address and maps to the DEBUG-OLT collection in versions R3.2.0 and later in MongoDB.

The following table describes the OLT debug HTTP methods and endpoints.

Table 35 OLT debug HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/debug/<OLT ID>/ Get a list of DEBUG-OLT documents. URL Parameters: start-time (optional), end-time (optional), limit (optional), skip (optional)
DELETE	DELETE /olts/debug/<OLT ID>/ Delete the documents for the given OLT from the DEBUG-OLT collection.

OLT actions

The /olts/<OLT ID>/ endpoints provide a way to execute common actions on an OLT device.

An OLT resource is identified by the device MAC address and maps to the OLT-CFG collection in MongoDB.

All OLT action GET requests return a document of the following data: { "Pending": False|True }

The following table describes the OLT action HTTP methods and endpoints.

Table 36 OLT action HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /olts/<OLT ID>/disable-onu/<ONU ID>/

HTTP method	Endpoints
	Gets the pending status of the Disable ONU action identified by <OLT ID> and <ONU ID>.
PUT	PUT /olts/<OLT ID>/disable-onu/<ONU ID>/ Send the Disable ONU signal for the ONU identified by <ONU ID> managed by the OLT <OLT ID>. Request Data: { "disable": True False }
GET	GET /olts/<OLT ID>/broadcast-enable-onus/ Gets the pending status of the Broadcast Enable ONUs action for the OLT identified by <OLT ID>
PUT	PUT /olts/<OLT ID>/broadcast-enable-onus/ Send the Broadcast Enable ONU signal for the OLT identified by <OLT ID>.
GET	GET /olts/<OLT ID>/allow-registration/ Gets the pending status of the Allow ONU Registration action for the OLT identified by <OLT ID>.
PUT	PUT /olts/<OLT ID>/allow-registration/ Send the Allow ONU Registration signal for the OLT identified by <OLT ID>. Request Data: { "onu-id": <ONU MAC Address/SSN> "ALL" }
GET	GET /olts/<OLT ID>/reset/ Gets the pending status of the Reset action for the OLT identified by <OLT ID>.
PUT	PUT /olts/<OLT ID>/reset/ Send the Reset signal for the OLT identified by <OLT ID>.
GET	GET /olts/<OLT ID>/protection-sync-olts/ Gets the peer OLT in a Type-B protection group for the for the OLT identified by <OLT ID>.
PUT	PUT /olts/<OLT ID>/protection-sync-olts/ Synchronizes certain OLT configuration parameters from the OLT identified by <OLT ID> the peer OLT in a Type-B protection group.

ONUs

Developers use reference APIs for ONUs to add ONU functionality to applications.

ONU states

The /onus/states/ resource and endpoints monitor ONU devices.

An ONU resource is identified by the device serial number (when using XGS-PON) or MAC address (when using 10G EPON) and maps to an ONU-STATE document in MongoDB.

The following table describes the ONU state HTTP methods and endpoints.

Table 37 ONU state HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/states/ Get a list of ONU-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /onus/states/<ONU ID>/ Get an ONU-STATE document identified by <ONU ID> MAC address.
DELETE	DELETE /onus/states/<ONU ID>/ Delete an ONU-STATE document identified by <ONU ID> MAC address.

ONU configurations

The /onus/configs/ resource and endpoints monitor ONU devices.

An ONU resource is identified by the device Serial Number (when using XGS-PON) or MAC address (when using 10G EPON) and maps to an ONU-CFG document in MongoDB.

The following table describes the ONU configuration HTTP methods and endpoints.

Table 38 ONU configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/configs/ Get a list of ONU-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /onus/configs/ Create an ONU-CFG document.
GET	GET /onus/configs/<ONU ID>/ Get an ONU-CFG document identified by <ONU ID> MAC address/Serial Number.
PUT	PUT /onus/configs/<ONU ID>/

HTTP method	Endpoints
	Create or replace an ONU-CFG document identified by <ONU ID> MAC address/Serial Number.
PATCH	PATCH /onus/configs/<ONU ID>/ Modify an ONU-CFG document identified by <ONU ID> MAC address/Serial Number.
DELETE	DELETE /onus/configs/<ONU ID>/ Delete an ONU-CFG document identified by <ONU ID> MAC address/Serial Number.

Note: The Netconf.Name field is used by the MCMS Netconf interface to allow user specific naming to reference device configuration. This field indexed in MongoDB and must be unique. The API will not allow a duplicate Netconf.Name to be created. By default, this can be the same as _id for the device.

ONU alarm configurations

The /onus/alarm-configs/ resource and endpoints configure ONU alarm profiles.

An alarm configuration resource is identified by the configuration ID and maps to an ONU-ALARM-CFG document in MongoDB.

The following table describes the ONU alarm configuration HTTP methods and endpoints.

Table 39 ONU alarm configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/alarm-configs/ Get a list of ONU-ALARM-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /onus/alarm-configs/ Create an ONU-ALARM-CFG document.
GET	GET /onus/alarm-configs/<CFG ID>/ Get an ONU-ALARM-CFG document identified by <CFG ID>.
PUT	PUT /onus/alarm-configs/<CFG ID>/ Create or replace an ONU-ALARM-CFG document identified by <CFG ID>.
DELETE	DELETE /onus/alarm-configs/<CFG ID>/ Delete an ONU-ALARM-CFG document identified by <CFG ID>.

OLT alarm histories

The /onus/alarm-histories/ resource and endpoints get ONU alarm histories.

An alarm histories resource is identified by the device MAC Address and maps to an ONU-ALARM-HIST-STATE document in MongoDB.

The following table describes the ONU alarm histories HTTP methods and endpoints.

Table 40 ONU alarm histories HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/alarm-histories/ Get a list of ONU-ALARM-HIST-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /onus/alarm-histories/<ONU ID>/ Get an ONU-ALARM-HIST-STATE document identified by <ONU ID>.
DELETE	DELETE /onus/alarm-histories/<ONU ID>/ Delete an ONU-ALARM-HIST-STATE document identified by <ONU ID>.

ONU PON automation configurations

The /onus/automation/configs/ resource and endpoints configure PON Automation for ONUs.

An ONU automation resource is identified by the device serial number (when using XGS-PON) or MAC address (when using 10G EPON) or Global and maps to an ONU-AUTO-CFG document in MongoDB.

The following table describes the ONU PON automation configuration HTTP methods and endpoints.

Table 41 ONU PON automation configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/automation/configs/ Get a list of ONU-AUTO-CFG documents.
POST	POST /onus/automation/configs/ Create an ONU-AUTO-CFG document.
GET	GET /onus/automation/configs/<CFG ID>/ Get an ONU-AUTO-CFG document identified by <CFG ID>.
PUT	PUT /onus/automation/configs/<CFG ID>/ Create or replace an ONU-AUTO-CFG document identified by <CFG ID>.

HTTP method	Endpoints
PATCH	PATCH /onus/automation/configs/<CFG ID>/ Modify an ONU-AUTO-CFG document identified by <CFG ID>.
DELETE	DELETE /onus/automation/configs/<CFG ID>/ Delete an ONU-AUTO-CFG document identified by <CFG ID>.

ONU PON automation alarm configurations

The /onus/automation/alarm-configs/ resource and endpoints configure PON automation ONU alarm profiles.

An alarm configuration resource is identified by the configuration ID and maps to an ONU-AUTO-ALARM-CFG document in MongoDB.

The following table describes the PON automation ONU alarm profiles HTTP methods and endpoints.

Table 42 PON automation ONU alarm profiles HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/automation/alarm-configs/ Get a list of ONU-AUTO-ALARM-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /onus/automation/alarm-configs/ Create an ONU-AUTO-ALARM-CFG document.
GET	GET /onus/automation/alarm-configs/<CFG ID>/ Get an ONU-AUTO-ALARM-CFG document identified by <CFG ID>.
PUT	PUT /onus/automation/alarm-configs/<CFG ID>/ Create or replace an ONU-AUTO-ALARM-CFG document identified by <CFG ID>.
DELETE	DELETE /onus/automation/alarm-configs/<CFG ID>/ Delete an ONU-AUTO-ALARM-CFG document identified by <CFG ID>.

ONU PON automation alarm histories

The /onus/automation/alarm-histories/ resource and endpoints get PON automation ONU alarm histories.

An alarm histories resource is identified by the device MAC Address and maps to an ONU-AUTO-ALARM-HIST-STATE document in MongoDB.

The following table describes the PON automation ONU alarm histories HTTP methods and endpoints.

Table 43 PON automation ONU alarm histories HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /onus/automation/alarm-histories/ Get a list of ONU-AUTO-ALARM-HIST-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /onus/automation/alarm-histories/<ONU ID>/ Get an ONU-AUTO-ALARM-HIST-STATE document identified by <ONU ID>.
DELETE	DELETE /onus/automation/alarm-histories/<ONU ID>/ Delete an ONU-AUTO-ALARM-HIST-STATE document identified by <ONU ID>.

ONU PON automation states

The /onus/automation/states/ resource and endpoints monitor PON Automation for ONUs.

An ONU automation resource is identified by the device serial number (when using XGS-PON) or MAC address (when using 10G EPON) and maps to an ONU-AUTO-STATE document in MongoDB.

The following table describes the ONU PON automation state HTTP methods and endpoints.

Table 44 ONU PON automation state HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/automation/states/ Get a list of ONU-AUTO-STATE documents.
GET	GET /onus/automation/states/<ONU ID>/ Get an ONU-AUTO-STATE document identified by <ONU ID>.
DELETE	DELETE /onus/automation/states/<ONU ID>/ Delete an ONU-AUTO-STATE document identified by <ONU ID>.

ONU statistics

The /onus/stats/ resource and endpoints monitor statistics for ONUs.

An ONU resource is identified by the device serial number (when using XGS-PON) or MAC address (when using 10G EPON) and maps to the STATS-ONU-<ONU ID> collection in versions R3.0.0 and earlier and STATS-ONU in versions R3.1.0 and later in MongoDB.

The following table describes the ONU statistic HTTP methods and endpoints.

Table 45 ONU statistic HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/stats/<ONU ID>/ Get a list of STATS-ONU documents. URL Parameters: start-time (required), end-time (optional)
DELETE	DELETE /onus/stats/<ONU ID>/ Delete the STATS-ONU collection.

ONU logs

The /onus/logs/ resource and endpoints monitor logs for ONUs.

An ONU resource is identified by the device serial number (when using XGS-PON) or MAC Address (when using 10G EPON) and maps to the SYSLOG-ONU-<ONU ID> collection in versions R3.0.0 and earlier and SYSLOG-ONU in versions R3.1.0 and later in MongoDB.

The following table describes the ONU log HTTP methods and endpoints.

Table 46 ONU log HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/logs/<ONU ID>/ Get a list of SYSLOG-ONU documents. URL Parameters: start-time (required), end-time (optional)
DELETE	DELETE /onus/logs/<ONU ID>/ Delete the SYSLOG-ONU collection.

ONU CPEs

The /onus/cpe-states/ resource and endpoints monitor CPEs for ONUs.

An ONU resource is identified by the device serial number (when using XGS-PON) or MAC address (when using 10G EPON) and maps to the ONU-CPE-STATE collection in MongoDB.

The following table describes the ONU CPE HTTP methods and endpoints.

Table 47 ONU CPE HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/cpe-states/ Get a list of ONU-CPE-STATE documents.

HTTP method	Endpoints
	URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /onus/cpe-states/<ONU ID>/ Get an ONU-CPE-STATE document identified by <ONU ID> MAC address/Serial Number.
DELETE	DELETE /onus/cpe-states/<ONU ID>/ Delete the ONU-CPE-STATE document identified by <ONU ID> MAC address/Serial Number.

MIB reset states

The /onus/mib-rst/ resource and endpoints monitor MIB reset states for ONUs.

An ONU resource is identified by the device serial number (when using XGS-PON) or MAC address (when using 10G EPON) and maps to the ONU-MIB-RST-STATE collection in MongoDB.

The following table describes the MIB reset state HTTP methods and endpoints.

Table 48 MIB reset state HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/mib-rst/ Get a list of ONU-MIB-RST-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /onus/mib-rst/<ONU ID>/ Get an ONU-MIB-RST-STATE document identified by <ONU ID> MAC address/Serial Number.
DELETE	DELETE /onus/mib-rst/<ONU ID>/ Delete the ONU-MIB-RST-STATE document identified by <ONU ID> MAC address/Serial Number.

MIB current states

The /onus/mib-cur/ resource and endpoints monitor MIB current states for ONUs.

An ONU resource is identified by the device serial number (when using XGS-PON) or MAC address (when using 10G EPON) and maps to the ONU-MIB-CUR-STATE collection in MongoDB.

The following table describes the MIB current state HTTP methods and endpoints.

Table 49 MIB current state HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/mib-cur/ Get a list of ONU-MIB-CUR-STATEdocuments. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /onus/mib-cur/<ONU ID>/ Get an ONU-MIB-CUR-STATEdocument identified by <ONU ID> MAC address/Serial Number.
DELETE	DELETE /onus/mib-cur/<ONU ID>/ Delete the ONU-MIB-CUR-STATE document identified by <ONU ID> MAC address/Serial Number.

ONU actions

The /onus/<ONU ID>/ endpoints provide a way to execute common actions on an ONU device.

An ONU resource is identified by the device serial number (when using XGS-PON) or MAC address (when using 10G EPON) and maps to the ONU-CFG collection in MongoDB.

All ONU action GET requests return a document of the following data: { "Pending": False|True }

The following table describes the ONU action HTTP methods and endpoints.

Table 50 ONU action HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /onus/<ONU ID>/reset/ Gets the pending status of the Reset action for the ONU identified by <ONU ID>.
PUT	PUT /onus/<ONU ID>/reset/ Send the Reset signal for the ONU identified by <ONU ID>.

Files

Developers use reference APIs for files to add firmware and pictures to applications.

OLT firmware

The `/files/olt-firmware/` resource and endpoints are used to manage OLT firmware.

An OLT firmware resource is identified by the filename and maps to the OLT-FIRMWARE bucket in MongoDB.

The following table describes the OLT firmware HTTP methods and endpoints.

Table 51 OLT firmware HTTP methods and endpoints

HTTP method	Endpoints
GET	GET <code>/files/olt-firmware/</code> Get a list of OLT Firmware.
GET	GET <code>/files/olt-firmware/<FILENAME>/</code> Get an OLT Firmware's metadata identified by <code><FILENAME></code> .
PUT	PUT <code>/files/olt-firmware/<FILENAME>/</code> Replace an OLT Firmware's metadata identified by <code><FILENAME></code> . See OLT Firmware PUT Request Data OLT Firmware POST Request Data , for the request body example.
POST	POST <code>/files/olt-firmware/<FILENAME>/</code> Upload an OLT Firmware identified by <code><FILENAME></code> . See OLT Firmware PUT Request Data OLT Firmware POST Request Data for the request body example.
DELETE	DELETE <code>/files/olt-firmware/<FILENAME>/</code> Delete an OLT Firmware identified by <code><FILENAME></code> .

OLT Firmware PUT Request Data

OLT Firmware POST Request Data

```
{ -"metadata": { -"Compatible Manufacturer": -"TIBITCOM",
  -"Compatible Model": [ -"180713" -],
  -"Version": -"R3.2.0" -} -}
{ -"file": <base64 file data>, -"metadata": {
  -"Compatible Manufacturer": -"TIBITCOM",
  -"Compatible Model": [ -"180713" -],
  -"Version": -"R3.2.0" -} -}
```

ONU firmware

The `/files/onu-firmware/` resource and endpoints are used to manage ONU firmware.

An ONU firmware resource is identified by the filename and maps to the ONU-FIRMWARE bucket in MongoDB.

The following table describes the ONU firmware HTTP methods and endpoints.

Table 52 ONU firmware HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /files/onu-firmware/ Get a list of ONU Firmware.
GET	GET /files/onu-firmware/<FILENAME>/ Get an ONU Firmware's metadata identified by <FILENAME>.
PUT	PUT /files/onu-firmware/<FILENAME>/ Replace an ONU Firmware's metadata identified by <FILENAME>. See ONU Firmware PUT Request Data ONU Firmware POST Request Data for the request body example.
POST	POST /files/onu-firmware/<FILENAME>/ Upload an ONU Firmware identified by <FILENAME>. See ONU Firmware PUT Request Data ONU Firmware POST Request Data for the request body example.
DELETE	DELETE /files/onu-firmware/<FILENAME>/ Delete an ONU Firmware identified by <FILENAME>.

ONU Firmware PUT Request Data

ONU Firmware POST Request Data

```
{ -"metadata": { -"Compatible Manufacturer": -"TIBITCOM",
  -"Compatible Model": [ -"MicroPlug ONU" -],
  -"Version": -"R3.2.0" -} -}
{ -"file": <base64 file data>, -"metadata": {
  -"Compatible Manufacturer": -"TIBITCOM",
  -"Compatible Model": [ -"MicroPlug ONU" -],
  -"Version": -"R3.2.0" -} -}
```

Pictures

The /files/pictures/ resource and endpoints are used to manage device pictures.

A picture resource is identified by the filename and maps to the PICTURES bucket in MongoDB.

The following table describes the picture HTTP methods and endpoints.

Table 53 Picture HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /files/pictures/

HTTP method	Endpoints
	Get a list of device pictures.
GET	GET /files/pictures/<FILENAME>/ Get a device picture identified by <FILENAME>.
PUT	PUT /files/pictures/<FILENAME>/ Replace a device picture's metadata identified by <FILENAME>. See Picture PUT Request Data Picture POST Request Data for request body example.
POST	POST /files/pictures/<FILENAME>/ Upload a device picture identified by <FILENAME>. See Picture PUT Request Data Picture POST Request Data for request body example.
DELETE	DELETE /files/pictures/<FILENAME>/ Delete a device picture identified by <FILENAME>.

Picture PUT Request Data

Picture POST Request Data

```
{ -"metadata": { -"Device Type": -"ONU" -} -}
{ -"file": <base64 file data>,
  -"metadata": { -"Device Type": -"ONU" -} -}
```

SLAs

The /slas/ resource and endpoints manage service level (SLA) agreements.

An SLA resource is identified by the configuration ID and maps to an SLA-CFG document in MongoDB.

Note: Any endpoint that utilizes custom names/IDs must double utf-8 encode forward slash characters. Example: 'MySla/10g' -> 'MySla%252F10g'

The following table describes the SLA HTTP methods and endpoints.

Table 54 SLA HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /slas/ Get a list of SLA-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /slas/ Create an SLA-CFG document.

HTTP method	Endpoints
GET	GET /slas/<CFG ID>/ Get an SLA-CFG document identified by <CFG ID>.
PUT	PUT /slas/<CFG ID>/ Create or replace an SLA-CFG document identified by <CFG ID>.
PATCH	PATCH /slas/<CFG ID>/ Modify an SLA-CFG document identified by <CFG ID>.
DELETE	DELETE /slas/<CFG ID>/ Delete an SLA-CFG document identified by <CFG ID>.

Downstream QoS maps

The /downstream-maps/ resource and endpoints manage downstream QoS mapping configuration profiles.

A downstream QoS map resource is identified by the configuration ID and maps to a DS-MAP-CFG document in MongoDB.

The following table describes the downstream QoS map HTTP methods and endpoints.

Table 55 Downstream QoS map HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /downstream-maps/ Get a list of DS-MAP-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /downstream-maps/ Create a DS-MAP-CFG document.
GET	GET /downstream-maps/<CFG ID>/ Get a DS-MAP-CFG document identified by <CFG ID>.
PUT	PUT /downstream-maps/<CFG ID>/ Create or replace a DS-MAP-CFG document identified by <CFG ID>.
PATCH	PATCH /downstream-maps/<CFG ID>/ Modify a DS-MAP-CFG document identified by <CFG ID>.
DELETE	DELETE /downstream-maps/<CFG ID>/ Delete an DS-MAP-CFG document identified by <CFG ID>.

ONU service configurations

The /service-configs/ resource and endpoints manage Service Configurations.

An SRV-CFG resource is identified by the configuration ID and maps to an SRV-CFG document in MongoDB.

Note: Any endpoint that utilizes custom names/IDs must double utf-8 encode forward slash characters. Example: 'MyService10g' -> 'MyService%252F10g'

The following table describes the ONU service configuration HTTP methods and endpoints.

Table 56 ONU service configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /service-configs/ Get a list of SRV-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /service-configs/ Create an SRV-CFG document.
GET	GET /service-configs/<CFG ID>/ Get an SRV-CFG document identified by <CFG ID>.
PUT	PUT /service-configs/<CFG ID>/ Create or replace an SRV-CFG document identified by <CFG ID>.
PATCH	PATCH /service-configs/<CFG ID>/ Modify an SRV-CFG document identified by <CFG ID>.
DELETE	DELETE /service-configs/<CFG ID>/ Delete an SRV-CFG document identified by <CFG ID>.

CPEs

The /cpes/ resource and endpoints monitor CPEs.

A CPE resource is identified by the CPE MAC address and maps to the CPE-STATE collection in MongoDB.

The following table describes the CPE HTTP methods and endpoints.

Table 57 CPE HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /cpes/states/ Get a list of CPE-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /cpes/states/<CPE ID>/ Get an CPE-STATE document identified by <CPE ID> MAC address.
DELETE	DELETE /cpes/states/<CPE ID>/ Delete the CPE-STATE document identified by <CPE ID> MAC address.

PON automation

Developers use reference APIs for PON automation to add PON automation functionality to applications.

PON automation states

The /pon_automation/states/ resource and endpoints monitor PON automation services.

A PON automation resource is identified by the PON automation service _id and maps to the AUTO-STATE collection.

The following table describes the PON automation state HTTP methods and endpoints.

Table 58 PON automation state HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /pon_automation/states/ Get a list of AUTO-STATE documents.
GET	GET /pon_automation/states/<PON AUTO ID>/ Get an AUTO-STATE document identified by <PON AUTO ID>.
DELETE	DELETE /pon_automation/states/<PON AUTO ID>/ Delete an AUTO-STATE document identified by <PON AUTO ID>.

PON automation configurations

The /pon_automation/configs/ resource and endpoints manage PON Automation services.

A PON Automation resource is identified by the PON Automation service's _id and maps to the AUTO-CFG collection.

The following table describes the PON automation configuration HTTP methods and endpoints.

Table 59 PON automation configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /pon_automation/configs/ Get a list of AUTO-CFG documents.
POST	POST /pon_automation/configs/ Create an AUTO-CFG document.
GET	GET /pon_automation/configs/<PON AUTO ID>/ Get an AUTO-CFG document identified by <PON AUTO ID>.
PUT	PUT /pon_automation/configs/<PON AUTO ID>/ Create or replace an AUTO-CFG document identified by <PON AUTO ID>.
PATCH	PATCH /pon_automation/configs/<PON AUTO ID>/ Modify an AUTO-CFG document identified by <PON AUTO ID>.
DELETE	DELETE /pon_automation/configs/<PON AUTO ID>/ Delete an AUTO-CFG document identified by <PON AUTO ID>.

PON automation alarm configurations

The /pon_automation/alarm-configs/ resource and endpoints configure PON Automation alarm profiles.

An alarm configuration resource is identified by the PON Automation service's ID and maps to an AUTO-ALARM-CFG document in MongoDB.

The following table describes the PON automation alarm configurations HTTP methods and endpoints.

Table 60 PON automation alarm configurations HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /pon_automation/alarm-configs/ Get a list of AUTO-ALARM-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)

HTTP method	Endpoints
POST	POST /pon_automation/alarm-configs/ Create an AUTO-ALARM-CFG document.
GET	GET /pon_automation/alarm-configs/<CFG ID>/ Get an AUTO-ALARM-CFG document identified by <CFG ID>.
PUT	PUT /pon_automation/alarm-configs/<CFG ID>/ Create or replace an AUTO-ALARM-CFG document identified by <CFG ID>.
DELETE	DELETE /pon_automation/alarm-configs/<CFG ID>/Delete an AUTO-A LARM-CFG document identified by <CFG ID>.

PON automation alarm configurations for switches

The /switches/automation/alarm-configs/ resource and endpoints configure PON automation switch alarm profiles.

An alarm configuration resource is identified by the configuration ID and maps to a SWI-AUTO-ALARM-CFG document in MongoDB.

The following table describes the PON automation switch alarm profiles HTTP methods and endpoints.

Table 61 PON automation switch alarm profiles HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /switches/automation/alarm-configs/ Get a list of SWI-AUTO-ALARM-CFG documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
POST	POST /switches/automation/alarm-configs/ Create a SWI-AUTO-ALARM-CFG document.
GET	GET /switches/automation/alarm-configs/<CFG ID>/ Get a SWI-AUTO-ALARM-CFG document identified by <CFG ID>.
PUT	PUT /switches/automation/alarm-configs/<CFG ID>/ Create or replace a SWI-AUTO-ALARM-CFG document identified by <CFG ID>.
DELETE	DELETE /switches/automation/alarm-configs/<CFG ID>/ Delete a SWI-AUTO-ALARM-CFG document identified by <CFG ID>.

PON automation alarm histories

The /pon_automation/alarm-histories/ resource and endpoints get PON Automation alarm histories.

An alarm histories resource is identified by the PON Automation service's ID and maps to an AUTO-ALARM-HIST-STATE document in MongoDB.

The following table describes the PON automation alarm histories HTTP methods and endpoints.

Table 62 PON automation alarm histories HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /pon_automation/alarm-histories/ Get a list of AUTO-ALARM-HIST-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /pon_automation/alarm-histories/<PON AUTO ID>/ Get an AUTO-ALARM-HIST-STATE document identified by <PON AUTO ID>.
DELETE	DELETE /pon_automation/alarm-histories/<PON AUTO ID>/ Delete an AUTO-ALARM-HIST-STATE document identified by <PON AUTO ID>.

PON automation alarm histories for switches

The /switches/automation/alarm-histories/ resource and endpoints get PON automation switch alarm histories.

An alarm histories resource is identified by the device MAC Address and maps to a SWI-AUTO-ALARM-HIST-STATE document in MongoDB.

The following table describes the PON automation switch alarm histories HTTP methods and endpoints.

Table 63 PON automation switch alarm histories HTTP methods and endpoints

HTTP methods	Endpoints
GET	GET /switches/automation/alarm-histories/ Get a list of SWI-AUTO-ALARM-HIST-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /switches/automation/alarm-histories/<SWI ID>/ Get a SWI-AUTO-ALARM-HIST-STATE document identified by <SWI ID>.

HTTP methods	Endpoints
DELETE	DELETE /switches/automation/alarm-histories/<SWI ID>/ Delete a SWI-AUTO-ALARM-HIST-STATE document identified by <SWI ID>.

PON automation statistics

The /pon_automation/stats/ resource and endpoints monitor statistics for PON automation services.

A PON automation resource is identified by the PON automation service _id and maps to the STATS-AUTO-<PON AUTO ID> collection in versions R3.0.0 and earlier and STATS-AUTO in versions R3.1.0 and later in MongoDB.

The following table describes the PON automation statistic HTTP methods and endpoints.

Table 64 PON automation statistic HTTP methods and endpoints

HTTP Method	Endpoint
GET	GET /pon_automation/stats/<PON AUTO ID>/ Get a list of STATS-AUTO documents. URL Parameters: start-time (required), end-time (optional)
DELETE	DELETE /pon_automation/stats/<PON AUTO ID>/ Delete the STATS-AUTO collection.

PON automation logs

The /pon_automation/logs/ resource and endpoints monitor logs for PON automation services.

A PON automation resource is identified by the PON automation service _id and maps to the SYSLOG-AUTO-<PON AUTO ID> collection in versions R3.0.0 and earlier and SYSLOG-AUTO in versions R3.1.0 and later in MongoDB.

The following table describes the PON automation log HTTP methods and endpoints.

Table 65 PON automation log HTTP methods and endpoints

HTTP Method	Endpoint
GET	GET /pon_automation/logs/<PON AUTO ID>/ Get a list of SYSLOG-AUTO documents. URL Parameters: start-time (required), end-time (optional)
DELETE	DELETE /pon_automation/logs/<PON AUTO ID>/ Delete the SYSLOG-AUTO collection.

PON automation tasks

PON automation task/job scheduler is used for scheduling jobs like batch upgrades of OLTs and ONUs. The /tasks/ resource and endpoints are used to schedule and monitor bulk firmware upgrade jobs for OLTs and ONUs.

PON automation tasks states

The /tasks/states/ resource and endpoints are used to monitor tasks.

A task resource is identified by the task name and maps to an AUTO-TASK-STATE document in MongoDB. See the AUTO-TASK-STATE.json schema file for a description of the parameters, data types, and ranges.

The following table describes the PON automation tasks state HTTP methods and endpoints.

Table 66 PON automation tasks state HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /tasks/states/ Get a list of AUTO-TASK-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /tasks/states/<TASK ID>/ Get an AUTO-TASK-STATE document identified by <TASK ID> task name.
DELETE	DELETE /tasks/states/<TASK ID>/ Delete an AUTO-TASK-STATE document identified by <TASK ID> task name.

PON automation tasks configurations

The /tasks/configs resource and endpoints are used to manage tasks.

A task resource is identified by the Task ID. See the AUTO-TASK-CFG.json schema file for a description of the parameters, data types, and ranges.

The following table describes the PON automation tasks configuration HTTP methods and endpoints.

Table 67 PON automation tasks configuration HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /tasks/configs/ Get a set of all task configurations.
POST	POST /tasks/configs/<TASK ID>/

HTTP method	Endpoints
	Create a task configuration.
GET	GET /tasks/configs/<TASK ID>/ Get the task configuration identified by <TASK ID>.
PUT	PUT /tasks/configs/<TASK ID>/ Update the task configuration identified by <TASK ID>.
PATCH	PATCH /tasks/configs/<TASK ID>/ Modify the task configuration identified by <TASK ID>.
DELETE	DELETE /tasks/configs/<TASK ID>/ Delete the task configuration identified by <TASK ID>.

PON automation tasks detailed states

The /tasks/detailed/states/ resource and endpoints are used to monitor tasks by device.

A task resource is identified by the task name/device ID, and maps to an AUTO-TASK-DETAILED-STATE document in MongoDB. See the AUTO-TASK-DETAILED-STATE.json schema file for a description of the parameters, data types, and ranges.

The following table describes the PON automation tasks detailed state HTTP methods and endpoints.

Table 68 PON automation tasks detailed state HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /tasks/detailed/states/ Get a list of AUTO-TASK-DETAILED-STATE documents. URL Parameters: query (optional), projection (optional), sort (optional), limit (optional), skip (optional), next (optional), distinct (optional)
GET	GET /tasks/detailed/states/<TASK ID>/<DEVICE ID>/ Get the AUTO-TASK-DETAILED-STATE identified by <TASK ID> and <DEVICE ID>.
DELETE	DELETE /tasks/detailed/states/ Delete AUTO-TASK-DETAILED-STATE documents.
DELETE	DELETE /tasks/detailed/states/<TASK ID>/<DEVICE ID>/ Delete the AUTO-TASK-DETAILED-STATE document identified by <TASK ID> and <DEVICE ID>.

Databases

Connection Configuration

The /databases/ resource and endpoints manage database connections.

A database resource is identified by the database ID.

The following table describes database HTTP methods and endpoints.

Table 69 Database HTTP methods and endpoints

HTTP Method	Endpoint
GET	GET /databases/ Get a set of all databases with their connection information.
GET	GET /databases/<DATABASE ID>/ Get the database connection information identified by <DATABASE ID>.
PUT	PUT /databases/<DATABASE ID>/ Update the database connection information identified by <DATABASE ID>.
POST	POST /databases/<DATABASE ID>/ Create the database connection identified by <DATABASE ID>.
DELETE	DELETE /databases/<DATABASE ID>/ Delete the database connection identified by <DATABASE ID>.

Database Create and Update Request Data

```
{
  -"auth_db": -"tibit_users",
  -"auth_enable": false,
  -"ca_cert_path": -"",
  -"db_uri": -"",
  -"dns_srv": false,
  -"host": -"127.0.0.1",
  -"name": -"tibit_pon_controller",
  -"password": -"pdmPass",
  -"port": -"27017",
  -"replica_set_enable": false,
  -"replica_set_hosts": [
    -"localhost",
    -"127.0.0.1"
  ],
  -"replica_set_name": -"rs0",
  -"tls_enable": false,
  -"username": -"pdmPonManager"
}
```

Connection status

The /databases/status/ resource and endpoints are used to check database connection status.

A database resource is identified by the Database ID.

The following table describes database status HTTP methods and endpoints.

Table 70 Database status HTTP methods and endpoints

HTTP Method	Endpoint
GET	GET /databases/status/ v1: Get the connection status of all database connections. v2: Get the connection status and details of all database connections. v3: Get the connection status, details, and MongoDB version of all database connections.
GET	GET /databases/status/<DATABASE ID>/ v1: Get the connection status of the database connection identified by <DATABASE ID>. v2: Get the connection status and details of the database connection identified by <DATABASE ID>. v3: Get the connection status, details, and MongoDB of the database connection identified by <DATABASE ID>.

Database selection

The /databases/selection/ and /databases/session/ resources and endpoints are used to manage database selection and session information.

A database resource is identified by the database ID.

The following table describes database selection HTTP methods and endpoints.

Table 71 Database HTTP methods and endpoints

HTTP Method	Endpoint
GET	GET /databases/selection/ Get the database connection currently selected to be used for this user.
PUT	PUT /databases/selection/ Update the database connection currently selected to be used for this user.
GET	GET /databases/session/ Get the database connection currently selected to be used for this session only.
PuT	PUT /databases/session/ Update the database connection currently selected to be used for this session only.

Users

The /users/ resource and endpoints authenticate users.

The following table describes the user HTTP methods and endpoints.

Table 72 User HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /users/logout/ Log out the user and invalidate their session.
POST	POST /users/authenticate/ Login the user identified by the POST data. See User login request data for the request body example.

User login request data

```
{
  -"email": -"api.user@gmail.com",
  -"password": -"apiUserPassword"
}
```

PON manager

The /ponmgr/ resource and endpoints access PON Manager installation information.

The following table describes the PON manager HTTP methods and endpoints.

Table 73 PON manager HTTP methods and endpoints

HTTP method	Endpoints
GET	GET /ponmgr/version/ Get a set of attributes describing the installed version of PON Manager.

CHAPTER 4

Use cases and examples

Use case examples

The PON Manager installation package contains several examples using the MCMS REST API using Curl, Postman, and Python.

The examples are located under the `/opt/tibit/ponmgr/examples` directory.

Curl

The Curl examples are located under the `/opt/tibit/ponmgr/examples/curl` directory.

To install cURL on Linux run the following commands:

```
sudo apt update && sudo apt upgrade
sudo apt install curl
```

The following is an example of a GET request which retrieves an OLT configuration document from the database. The GET request must include the cookies provided from the login response. In the following example, the response is saved to the file `OLT-CFG.json`.

```
curl --X GET https://10.1.20.102/api/v1/olts/
configs/70:b3:d5:52:35:ac/ \
-b cookies.txt --k \
> OLT-CFG.json
```

The following is an example of a PUT request which writes and updates an OLT configuration document to MongoDB. The PUT request must include the following:

- payload
- content type header
- CSRF token
- cookies
- referrer URL

76 Use cases and examples

In the following example, the request payload is contained in the file OLT-CFG.json.

```
curl --X PUT https://10.1.20.102/api/v1/olts/
configs/70:b3:d5:52:35:ac/ \
  --d @OLT-CFG.json \
  --H "Content-Type: application/json" \
  --H "X-CSRFToken:$(grep csrftoken cookies.txt|sed - 's/^.*csrftoken
\s*//')"\
  --b cookies.txt --k \
  --e https://10.1.20.102/api
```

Postman

The Postman examples are located under the /opt/tibit/ponmgr/examples/postman directory.

Postman examples

- 1 Import the postman_mcms_restapi_R2.3.0 folder into postman

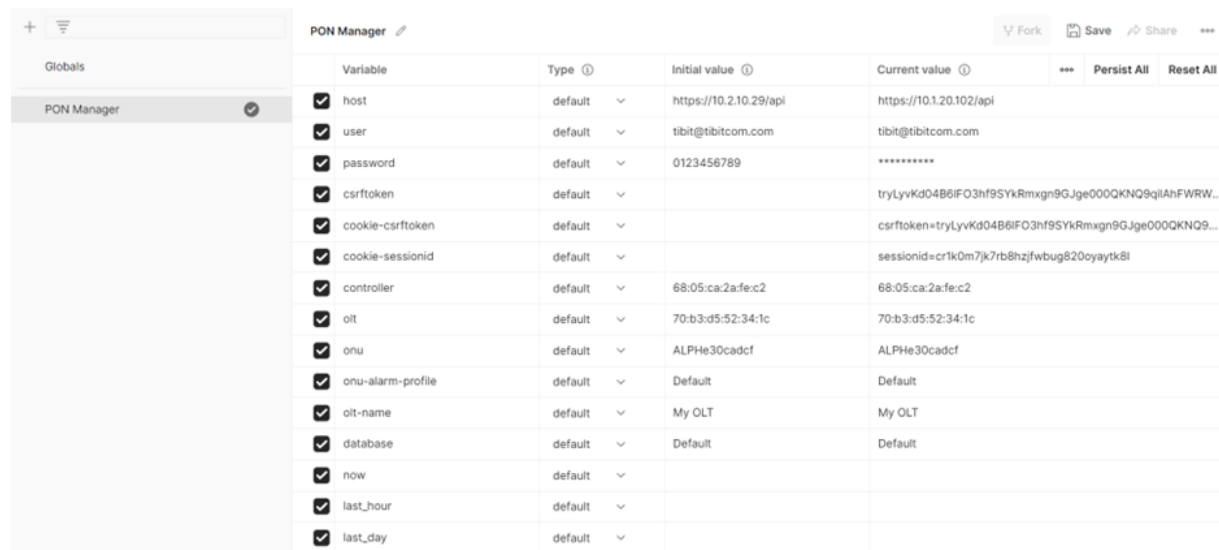
Note: Newer versions of Postman may require an account.

- 2 Select the Environments tab, and then PON Manager. Set the following. Other variables can be set if desired.

- Host
- User
- Password

Set other variables as desired.

Figure 7 PON manager



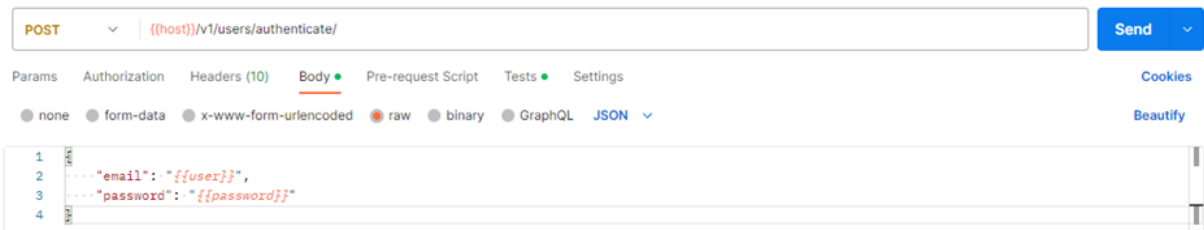
Variable	Type	Initial value	Current value	Persist All	Reset All
☒ host	default	https://10.210.29/api	https://10.1.20.102/api		
☒ user	default	tibit@tibitcom.com	tibit@tibitcom.com		
☒ password	default	0123456789	*****		
☒ csrftoken	default		tryLyvKd04B6iFO3hf9SYkRmxgn9GJge000QKNQ9qIhFWRW...		
☒ cookie-csrftoken	default		csrftoken=tryLyvKd04B6iFO3hf9SYkRmxgn9GJge000QKNQ9...		
☒ cookie-sessionid	default		sessionid=cr1k0m7jk7rb8hzjfwbug820oyaytk8l		
☒ controller	default	68:05:ca:2a:fe:c2	68:05:ca:2a:fe:c2		
☒ olt	default	70:b3:d5:52:34:1c	70:b3:d5:52:34:1c		
☒ onu	default	ALPHe30cadcf	ALPHe30cadcf		
☒ onu-alarm-profile	default	Default	Default		
☒ olt-name	default	My OLT	My OLT		
☒ database	default	Default	Default		
☒ now	default				
☒ last_hour	default				
☒ last_day	default				

- 3 Select the Collections tab and click on Login POST in the tree. The following variables use values from the Environment.

- Host
- User
- Password

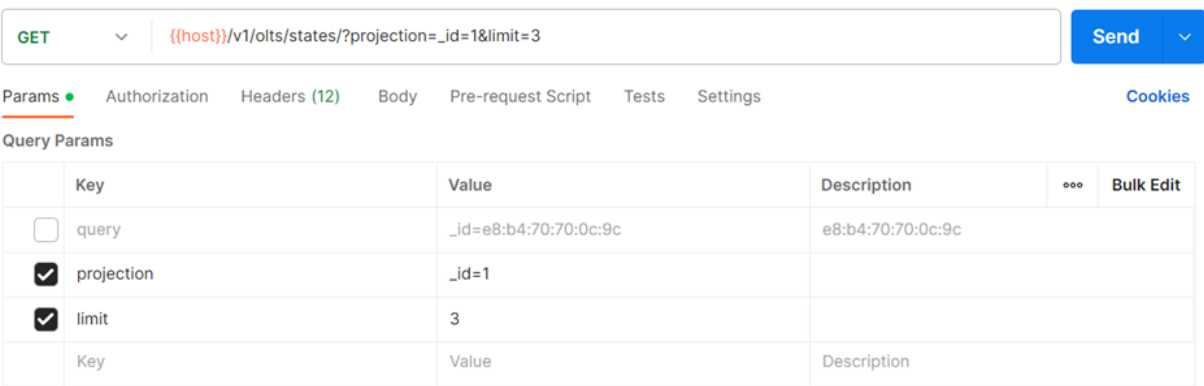
Click send to authenticate and to set the session variables.

Figure 8 Login POST



4 Send a GET request and query parameters to direct the URL the Parameters tab.

Figure 9 GET request



Python

Python examples are located under the `/opt/tibit/ponmgr/examples/python` directory.

They require Python version 3 and they request Python Package from `pypi.org`.

Use the `pip3` command to install the requests, and enter the following into the command line:
`pip3 install requests`

The following example configures service for a subscriber, where the OLT pushes/pops S-Tag 200 and the ONU pushes/pops C-Tag 25.

```
python3 config_addctag_service.py ---url https://10.2.10.29/api \
--user <email> ---password <password> \
--olt e8:b4:70:70:0c:9c ---olt_tag 200 \
--onu ALPHe30cadcf ---onu_tag 25
```

The following example queries the registration status for all ONUs configured under the specified OLT.

```
python3 get_onu_registration_status.py ---url https://10.2.10.29/ \
--user <email> ---password <password> \
--olt e8:b4:70:70:0c:9c
```

The following example resets an ONU.

```
python3 reset_onu.py ---url http://10.2.10.29/api \  
--user <email> ---password <password> \  
--onu ALPHe30cadcf
```

Programming model

This section describes the common programming model for the MCMS REST API.

This general sequence is required for each of the use cases presented.

After a logging in successfully and establishing of a secure authenticated session, all API requests must contain the following HTTP headers. Refer to [“Request Format” on page 25](#) for more information.

- Session ID Cookie
- CSRF Token Cookie

Additionally, all PUT and POST requests must contain the following HTTP headers:

- Content-Type: application/json
- X-CSRF Token
- Referrer

API sessions generally have the following sequence:

- 1 Log in to authenticate the user with the web server
POST -/v1/users/authenticate/
- 2 Select the database for this session. This is an optional step. If skipping this step, the user's default database is used for the session.
PUT -/v1/databases/selection/
- 3 Execute one or more requests to fetch data and configure the database. For example:
GET -/v1/onus/configs/BFWS00123193
or
PUT -/v1/onus/configs/BFWS00123193
- 4 Log out of the web server.
GET -/v1/users/logout/

List of procedures

REST API procedures are:

- [Procedure 1, “Determining the status of an ONU using CNTL-STATE”](#)
- [Procedure 2, “Determining the registration status of an ONU using OLT-STATE”](#)
- [Procedure 3, “Provisioning service for a subscriber on an ONU”](#)
- [Procedure 4, “Pre-provisioning service for an ONU”](#)
- [Procedure 5, “Pre-provisioning an OLT device”](#)
- [Procedure 6, “Disabling service for an ONU”](#)
- [Procedure 7, “Upgrading the firmware on an ONU”](#)
- [Procedure 8, “Upgrading the firmware on many ONUs”](#)

- Procedure 9, “Resetting an ONU”
- Procedure 10, “Associating a subscriber configuration to a device”
- Procedure 11, “Disassociating a subscriber configuration from a device”
- Procedure 12, “Creating an unassociated subscriber”

Procedure 1 Determining the status of an ONU using CNTL-STATE

Determine the status of an ONU through the PON controller as required by the network plan.

Steps

- 1 Query the ONU-STATE of the ONU by GPON Serial Number or EPON MAC Address:
GET <URL>/v1/onus/states/<ONU ID>/
- 2 Look up the PON Controller MAC Address and OLT MAC Address to which this ONU is attached:
ONU-STATE = {
 - "_id" -: -"BFWS00123193",
 - "CNTL" -: {
 - "MAC Address" -: -"68:05:ca:2a:fe:c2",
 - "Version" -: -"R4.0.0-rc99"
 - },
 - "OLT" -: {
 - "MAC Address" -: -"e8:b4:70:70:0c:9c",
 - "Reg Disallowed" -: false
 - },
 - ...
 - "Time" -: -"2023-07-19 20:40:08.730244"
}
3 Query CNTL-STATE of the PON Controller by MAC Address:
GET <URL>/v1/controllers/states/<CNTL ID>/

4 Look up the ONU Registration status from CNTL-STATE:

```
CNTL-STATE = {  
  -"_id" -: -"68:05:ca:2a:fe:c2",  
  -...  
  -"System Status" -: {  
    -"e8:b4:70:70:0c:9c" -: {  
      -"OLT State" -: -"Primary",  
      -"ONUs" -: {  
        -"ISK71e81e78" -: -"Registered",  
        -"BFWS00123193" -: -"Registered",  
        -"ALPHe30cadcf" -: -"Registered",  
        -"CIEN00099729" -: -"Registered",  
        -"TBITc84c0083" -: -"Registered"  
      }  
    }  
  },  
  -},  
  -"Time" -: -"2023-07-19 20:49:22.596464"  
}
```

Procedure 2 Determining the registration status of an ONU using OLT-STATE

Determine the status of an ONU through the OLT state use the steps outlined below.

Steps

- 1 Query ONU-STATE of the ONU by GPON Serial Number or EPON MAC Address:
GET <URL>/v1/onus/states/<ONU ID>/
- 2 Look up the OLT MAC Address to which the ONU is attached:

```
ONU-STATE = {
  -"_id" -: -"BFWS00123193",
  -"CNTL" -: {...},
  -"OLT" -: {
    -"MAC Address" -: -"e8:b4:70:70:0c:9c",
    -"Reg Disallowed" -: false
  },
  ....
  -"Time" -: -"2023-07-19 20:40:08.730244"
}
```
- 3 Query OLT-STATE of the OLT by MAC Address:
GET <URL>/v1/olts/states/<OLT ID>/
- 4 Look up the Unspecified ONUs and registered ONUs from the ONU states field in the OLT-STATE document:

```
OLT-STATE = {
  -"_id" -: -"e8:b4:70:70:0c:9c",
  ....
  -"ONU States" -: {
    -"Disabled" -: [],
    -"Deregistered" -: [],
    -"Disallowed Admin" -: [],
    -"Disallowed Error" -: [],
    -"Disallowed Reg ID" -: [],
    -"Dying Gasp" -: [],
    -"Registered" -: [
      0 -: -"BFWS00123193"
    ],
    -"Unspecified" -: [],
    -"Unprovisioned" -: []
  },
  ....
}
```

If the ONU is in either the “Registered” or “Unspecified” state, then it is online.

Procedure 3 Provisioning service for a subscriber on an ONU

Provision service for a subscriber on an ONU as required by the network plan.

Overview

This procedure configures the OLT to push or pop an outer S-Tag identifying a PON port and configures the ONU to push or pop an inner C-Tag identifying the ONU. This configuration shown in the diagram below, shows double tagged frames with the S-Tag and C-Tag on the NNI, single tagged frames with C-Tag only on the PON, and untagged or priority tagged frames on the UNI.

Figure 10 Configuration



The following table describes the parameters for provisioning service for a subscriber on an ONU.

Table 74 Parameters for provisioning service for a subscriber on an ONU

Parameter	Valid values	Description
ONU ID - ONU	string	Specifies the ID used to identify the ONU for the XGS-PON OMCC channel.
ALLOC ID	string	Specifies the TCONT ALLOC ID and XGEM port values assigned for this service.
OLT S-Tag	string	Specifies the SVLAN ID identifying the PON Port.
ONU C-Tag	string	Specifies the CVLAN ID identifying the ONU.

Provisioning service requires the following steps:

- Inventory the ONU to configure the GPON ONU ID and ALLOC IDs.
- Inventory the NNI Network to configure the VLAN Tagging.

- Set the ONU Service configuration file and related service attributes.
- Enable the OLT Service port(s) and configure SLA(s) and VLAN(s).

Steps

- 1 Get the OLT configuration to which this ONU is connected:
GET <URL>/v1/olts/configs/<OLT ID>/
- 2 Write the modified OLT configuration to MongoDB:
PUT <URL>/v1/olts/configs/<OLT ID>/
- 3 Get the ONU configuration document from MongoDB:
GET <URL>/v1/onus/configs/<ONU ID>/
- 4 Configure the ONU Service Configuration file and related attributes. Configure the following fields: ONU-CFG.ONU.SRV-CFG, ONU-CFG.ONU.CVID, and ONU-CFG.SRV-CFG Values.:

```
ONU-CFG = {
  -"_id" -: -"<ONU ID>",
  -...
  -"ONU" -: {
    -"Enable" -: true,
    -...
    -"SRV-CFG" -: -"Add CTag",
    -"CVID" -: <ONU C-Tag>,
    -...
  },
  -...
}
```

- 5 Enable and configure the OLT Service port(s) and related attributes. Configure the following fields: ONU-CFG.OLT-Service.n.Enable, ONU-CFG.OLT-Service.n.NNI Networks, ONU-CFG.OLT-Service.n.PON Networks, and ONU-CFG.OLT-Service.n.SLA.:

```
ONU-CFG = {
  -"_id" -: -"<ONU ID>",
  -...
  -"OLT-Service 0" -: {
    -"Enable" -: true,
    -...
    -"NNI Networks" -: ["s<OLT S-Tag>.c<ONU C-Tag>.c0"],
    -"PON Networks" -: ["s0.c<ONU C-Tag>.c0"],
    -...
    -"SLA-CFG" -: -"Max",
    -...
  },
  -...
}
```

- 6 Write the modified ONU configuration to MongoDB:
PUT <URL>/v1/onus/configs/<ONU ID>/

Procedure 4 Pre-provisioning service for an ONU

Pre-provision service for an ONU as required by the network plan.

Overview

Use this procedure when a known ONU is configured before it is physically connected to an OLT. It complements the procedure: Provisioning service for a subscriber on an ONU, which provides a more complete example for configuring service for a subscriber.

The following table describes the parameters for pre-provisioning service for an ONU.

Table 75 Parameters for pre-provisioning service for an ONU

Parameter	Valid values	Description
_id	string	Specifies the MAC ADDRESS or Serial Number for the new ONU.
OLT.Name	string	Specifies the freeform string nickname for the device .
NETCONF.Name	string	Specifies the unique name for the MCMS Netconf Interface. Set the same as _id.

Steps

- 1 Get the default ONU configuration:
GET <URL>/v1/onus/configs/Default/
- 2 Modify the following fields on the base configuration: _id, OLT.Name, and NETCONF.Name.
- 3 Configure the ONU Service Configuration file and related attributes. Enable and configure the OLT Service port(s) and related attributes. See procedure: Provisioning service for a subscriber on an ONU for detailed steps on provisioning service for this ONU.
- 4 Write the new ONU configuration to MongoDB:
POST <URL>/v1/onus/configs/
- 5 Get the OLT configuration that this ONU is connected to:
GET <URL>/v1/olts/configs/<OLT ID>/

- 6 Inventory the ONU. See the procedure: Provisioning service for a subscriber on an ONU for additional steps for inventorying the ONU.

```
ONUs: {  
  <new Device ID>: {  
    -"ALLOC ID (OMCC)": -"<Unique ONU ID for the OLT, -  
integer 1-128>",  
    -"Enable Count": 0,  
    -"Disable Count": 0,  
    -"Disable": false  
  },  
  -...  
}
```

- 7 Write the modified OLT configuration. Follow procedure: Provisioning service for a subscriber on an ONU for steps to configure services.

```
PUT <URL>/v1/olts/configs/<OLT ID>/
```

Procedure 5 Pre-provisioning an OLT device

Pre-provision an OLT device as required by the network plan.

Overview

Use this procedure when a known OLT is configured before it is physically connected to the management network.

The following table describes the parameters for pre-provisioning OLT device.

Table 76 Parameters for pre-provisioning an OLT device

Parameter	Valid values	Description
_id	string	Specifies the MAC ADDRESS for the new OLT.
OLT.Name	string	Specifies the freeform string nickname for the device.
NETCONF.Name	string	Specifies the unique name for the MCMS Netconf Interface. Set the same as _id.

Steps

- 1 Get the default OLT configuration:
GET <URL>/v1/olts/configs/Default/
- 2 Modify the following fields on the base configuration: _id, OLT.Name, and NETCONF.Name.
- 3 Write the modified OLT configuration:
POST <URL>/v1/olts/configs/
- 4 Get the primary parent PON Controller configuration:
GET <URL>/v1/controllers/configs/<CNTL ID>/

- 5** Inventory the OLT on the Primary PON Controller. Append new OLT ID to the Controller Configuration's primary OLT list:
OLTs: {
 -"Primary": [
 <OLT ID>,
 -...
 -]
-}
- 6** Upload the modified Controller Configuration:
PUT <URL>/v1/controllers/configs/<CNTL ID>/
- 7** (Optional) Inventory the OLT on a Secondary PON Controller. Repeat Steps 4-6 with the secondary parent PON Controller except append the OLT ID to the "Secondary" list.
OLTs: {
 -"Secondary": [
 <OLT ID>,
 -...
 -]
-}
- 8** (Optional) Inventory the OLT on the Switch. To inventory the OLT on the Switch, get the configuration for the Switch the OLT is plugged into.
GET <URL>/v1/switches/configs/<SWI ID>/
- 9** Add the OLT and Switch Port mapping to the OLTs object in the Switch configuration.
OLTs: {
 <OLT ID>: { -"Port ID": <Switch port for the OLT, -
 e.g., -"1/0/5"> -},
 ...
}
- 10** Write the modified Switch configuration:
PUT <URL>/v1/switches/configs/<SWI ID>/

Procedure 6 Disabling service for an ONU

Disable and remove service for an ONU as required by the network plan.

Overview

Some steps in this use procedure are optional. For example, to disable the service without removing all configurations, set the ONU-CFG.OLT-Service 0.Enable to false.

Disabling service for an ONU requires the following steps:

- Disable the OLT Service port(s) and remove the VLAN configuration.
- Set the ONU Service configuration to disabled and related service attributes.
- Delete the OLT NNI Network.
- Remove the device from ONU Inventory.

Steps

- 1 Get the ONU configuration document from MongoDB:
GET <URL>/v1/onus/configs/<ONU ID>/
- 2 Set the ONU Service Configuration to Disabled and clear set all related attributes to their default values. Configure the following fields: ONU-CFG.ONU.SRV-CFG, ONU-CFG.ONU.CVID, and ONU-CFG.SRV-CFG Values.:

```
ONU-CFG = {
  -"_id" -: -"<ONU ID>",
  -...
  -"ONU" -: {
    -"Enable" -: true,
    -...
    -"SRV-CFG" -: -"Disabled",
    -"CVID" -: 0,
    -...
  },
  -...
}
```
- 3 Disable the OLT Service port(s) and clear related attributes to default values. Configure the following fields: ONU-CFG.OLT-Service.n.Enable, ONU-CFG.OLT-

Service.n.NNI Networks, ONU-CFG.OLT-Service.n.PON Networks, and ONU-CFG.OLT-Service.n.SLA:

```
ONU-CFG = {
  -"_id" -: -"<ONU ID>",
  -...
  -"OLT-Service 0" -: {
    -"Enable" -: false,
    -...
    -"NNI Networks" -: ["s<OLT S-Tag>.c<ONU C-Tag>.c0"],
    -"PON Networks" -: ["s0.c<ONU C-Tag>.c0"],
    -...
    -"SLA-CFG" -: -"Max",
    -...
  },
  -...
}
```

- 4 Write the modified ONU configuration to MongoDB:

```
PUT <URL>/v1/onus/configs/<ONU ID>/
```

- 5 Get the OLT configuration that this ONU is connected to.:

```
GET <URL>/v1/olts/configs/<OLT ID>/
```

- 6 Delete the NNI Network. Remove the NNI Network entry from the OLT configuration document: OLT-CFG.NNI Networks:

```
OLT-CFG = {
  -"_id" -: -"<OLT ID>",
  -...
  -"NNI Networks" -: [
    {
      -"TAG MATCH" -: -"s<OLT S-Tag>.c<ONU C-Tag>.c0",
      -"SLA-CFG" -: {
        -"Source" -: -"N/A"
      },
      -"Filter" -: {
        -"DHCPv4" -: -"pass",
        -"DHCPv6" -: -"pass",
        -"EAPOL" -: -"pass",
        -"PPPoE" -: -"pass"
      },
      -"Learning Limit" -: 2046
    },
    -...
  ]
  -...
}
```

- 7 Remove the ONU from the OLT's ONU Inventory. Remove the ONU entry from the OLT configuration document: OLT-CFG.ONUs.<ONU ID>.:

```
OLT-CFG = {
  -"_id" -: -"<OLT ID>",
  -...
  -"ONUs" -: {
    -"<ONU ID>" -: {
      -"ALLOC ID (OMCC)" -: <ONU ID>,
      -"Disable" -: False,
      -"Enable Count" -: 0,
      -"Disable Count" -: 0,
      -"OLT-Service 0" -: <ALLOC ID #1>,
      -"OLT-Service n" -: <ALLOC ID #n>
    }
  },
  -...
}
```

- 8 Write the modified OLT configuration to MongoDB:
PUT <URL>/v1/olts/configs/<OLT ID>/

Procedure 7 Upgrading the firmware on an ONU

Upgrade the firmware on an ONU as required by the network plan.

Overview

An ONU firmware download is initiated through the ONU configuration document in MongoDB. Modify the ONU-CFG with the desired firmware bank, files, and versions to initiate an ONU firmware download.

The following table describes the parameters for upgrading the firmware on an ONU.

Table 77 Parameters for upgrading the firmware on an ONU

Parameter	Valid values	Description
ONU-CFG.ONU.FW Bank Ptr	integer Set to either bank '0', or bank '1'	Specifies the desired bank.
ONU-CFG.ONU.FW Bank Files[0]	integer	Specifies the filename for bank 0.
ONU-CFG.ONU.FW Bank Versions[0]	integer	Specifies the firmware version string for bank 0.
ONU-CFG.ONU.FW Bank Files[1]	integer	Specifies the firmware filename for bank 1.
ONU-CFG.ONU.FW Bank Versions[1]	integer	Specifies the firmware version string for bank 1.

Steps

- 1 Get the configuration document for the ONU to be upgraded:
GET <URL>/v1/onus/configs/<ONU ID>/
- 2 Set the firmware bank versions, firmware bank files, and firmware bank pointer to the desired firmware settings. Update the following fields: ONU-CFG.ONU.FW Bank

Ptr, ONU-CFG.ONU.FW Bank Files[0], ONU-CFG.ONU.FW Bank Versions[0], ONU-CFG.ONU.FW Bank Files[1], and ONU-CFG.ONU.FW Bank Versions[1].

```
ONU-CFG = {
  -"_id" -: -"<ONU ID>",
  -...
  -"ONU" -: {
    -"FW Bank Files": [
      -"0" -: -"FW-GPON-CIEN-3802_91x-V2.3.10.bin"
      -"1" -: -"FW-GPON-CIEN-3802_91x-V2.4.3.bin"
    ],
    -"FW Bank Versions" -: [
      -"0" -: -"V2.3.1"
      -"1" -: -"V2.4.3"
    ],
    -"FW Bank Ptr": 1,
    -...
  }
}
```

- 3 Write the modified ONU configuration to MongoDB:

```
PUT <URL>/v1/onus/configs/<ONU ID>/
```

Procedure 8 Upgrading the firmware on many ONUs

Upgrade the firmware on many ONUs as required by the network plan.

Overview

An ONU firmware download for many ONUs is initiated through the task configuration document in MongoDB. Modify the AUTO-TASK-CFG with the desired devices, firmware information, and schedule times as listed below to create an ONU firmware upgrade job.

The following table describes the parameters for upgrading the firmware on many ONUs.

Table 78 Parameters for upgrading the firmware on many ONUs

Parameter	Valid values	Description
AUTO-TASK-CFG.Task .Device Type	string Set to 'ONU'	Specifies the device type for this upgrade.
AUTO-TASK-CFG.Task .Operation	string Set to 'Firmware Upgrade'	Specifies the task operation.
AUTO-TASK-CFG.Task .Scheduled Start Time	integer	Specifies the time for the job to start in UTC format.
AUTO-TASK-CFG.Task Details.ONU.FW Bank Ptr	integer Set to either bank '0', or bank '1'	Specifies the desired bank.
AUTO-TASK-CFG.Task Details.ONU.FW Bank Files[0]	integer	Specifies the firmware filename for bank 0.
AUTO-TASK-CFG.Task Details.ONU.FW Bank Versions[0]	integer	Specifies the firmware version string for bank 0.
AUTO-TASK-CFG.Task Details.ONU.FW Bank Files[1]	integer	Specifies the firmware filename for bank 1.
AUTO-TASK-CFG.Task Details.ONU.FW Bank Versions[1]	integer	Specifies the firmware version string for bank 1.

Parameter	Valid values	Description
AUTO-TASK-CFG.Task Details.Devices	string	Specifies the list of ONUs to be included in the upgrade.

Steps

- 1 Get the configuration document for the task. If creating a new job, get the Default document and modify as needed:
GET <URL>/v3/tasks/configs/<TASK ID>/
- 2 Set the configuration info to the desired settings. Update the following required fields: AUTO-TASK-CFG.Task.Device Type, AUTO-TASK-CFG.Task.Operation, AUTO-TASK-CFG.Task.Scheduled Start Time, AUTO-TASK-CFG.Task Details.ONU.FW Bank Ptr, AUTO-TASK-CFG.Task Details.ONU.FW Bank Files[0], AUTO-TASK-CFG.Task Details.ONU.FW Bank Versions[0], AUTO-TASK-CFG.Task

Details.ONU.FW Bank Files[1], AUTO-TASK-CFG.Task Details.ONU.FW Bank Versions[1], AUTO-TASK-CFG.Task Details.Devices.

```
{
  AUTO-TASK-CFG: {
    -"_id": -"<TASK ID>",
    -...
    -"Task": {
      -"Device Type": -"ONU",
      -"Operation": -"Firmware Upgrade",
      -...
      -"Scheduled Start Time": -"2025-01-14 16:05:00"
    },
    -"Task Details": {
      -"Devices": [
        -"TBITafefcfa0",
        -"TBIT4707ffd3"
      ],
      -...
    },
    -"ONU": {
      -"FW Bank Ptr": 1,
      -"FW Bank Files": {
        -"0": -"R5.0.0-ONU-FW.bin",
        -"1": -"R5.1.0-ONU-FW.bin"
      },
      -"FW Bank Versions": {
        -"0": -"R5.0.0",
        -"1": -"R5.1.0"
      }
    }
  }
}
```

- 3 Write the task configuration to MongoDB:
PUT <URL>/v3/tasks/configs/<TASK ID>/

Procedure 9 Resetting an ONU

Reset an ONU as required by the network plan. An ONU reset is implemented as an action using the MCMS REST API.

Steps

- 1 Reset the ONU:
`PUT <URL>/v1/onus/${ONU}/reset/`

Procedure 10 Associating a subscriber configuration to a device

Associate a subscriber configuration to a device as required by the network plan.

Overview

Use this procedure when an existing ONU-AUTO-CFG needs to be attached to a known ONU.

Steps

- 1 Write a patch configuration and call the PATCH endpoint for ONU automation configurations:

Body: [

```
{"op": "replace", "path": "/AUTO/Device ID", "value", <New ONU ID>}
```

]

Note: or if Registration ID is being used, follow below:

Body: [

```
{"op": "replace", "path": "/AUTO/Registration ID", "value", <New Registration ID>}
```

]

PATCH <URL>/v1/onus/automation/configs/<Subscriber ID>/

Note: Any endpoint that utilizes custom names/IDs must double utf-8 encode forward slash characters. Example: 'MySubscriber/10g' -> 'MySubscriber%252F10g'

Procedure 11 Disassociating a subscriber configuration from a device

Disassociate a subscriber configuration from a device when it is no longer required.

Overview

Use this procedure when an ONU is needs to be removed from a subscriber configuration.

Steps

- 1 Write a patch configuration to clear the Device ID and Registration ID values, then call the PATCH endpoint for ONU Automation Configurations:

Body: [

```
{ "op": "replace", "path": "/AUTO/Device ID", "value", "" },  
{ "op": "replace", "path": "/AUTO/Registration ID", "value", "" }  
]
```

PATCH <URL>/v1/onus/automation/configs/<Subscriber ID>/

Note: Any endpoint that utilizes custom names/IDs must double utf-8 encode forward slash characters. Example: 'MySubscriber/10g' -> 'MySubscriber%252F10g'

Procedure 12 Creating an unassociated subscriber

Create an unassociated subscriber as required by the network plan.

Overview

The following table describes the parameter for creating an unassociated subscriber.

Table 79 Parameter for creating an unassociated subscriber

Parameter	Valid values	Description
_id	string	Specifies the unique name for the new subscriber.

Steps

- 1 Get the Default ONU-AUTO-CFG Document:
GET <URL>/v1/onus/automation/configs/Default/
- 2 Modify the following field on the base configuration: _id.
- 3 Upload the modified ONU-AUTO-CFG Document:
POST <URL>/v1/onus/automation/configs/

CHAPTER 5

Debugging and troubleshooting

Logs

Refer to the logs for information concerning why the REST API is not working properly.

Logs are stored on the where the PON Manager is installed.

PON manager logs

PON Manager logs are located at `/var/log/tibit/ponMgr.log`.

Log messages including ERROR provide helpful information concerning why an operation was unsuccessful. It is also important to check the status code associated with a request. For example, the following error message appears in the PON Manager logs if a user attempts to get ONU configurations but is not logged in.

```
2023-06-28 18:02:15.942 ERROR 10.3.8.104 unauthenticated GET -/v2/
onus/configs/ status: 403
```

The user is not authorized to view the ONU configurations and must first use the REST API to log in as a valid user with the correct permissions.

Traceback Logs

Traceback Logs are located at `"/var/log/tibit/ponMgrTrace.log"`.

Each log message includes information such as http headers, request data, response data, and a traceback when applicable. Traceback Logging is off by default and can be enabled via PON Manager.

Enabling and filtering requests to trace log is described in the MCMS PON Manager User Guide.

Apache logs

There are two types of Apache logs: error.log and other_vhosts_access.log

error.log

other_vhosts_access.log

The Apache error log is located: /var/log/apache2/error.log, and is useful when debugging PON Manager configuration problems which can prevent the REST API from working properly.

The Apache other vhost log is located: /var/log/apache2/ other_vhosts_access.log. The vhost access logs provide more detailed information about HTTP requests between the web server and virtual host serving the REST API. These logs capture details associated with each request.

- Time
- HTTP version
- Response code
- Browser details

Errors

The following are common API errors relating to the MongoDB and syntax problems.

MongoDB Connection

Validation

Bad Request

The following is a common MongoDB connection error:

```
ERROR MongoDB server at 10.3.8.180:27017 has gone offline due to -  
Heartbeat failure: hello canceled
```

The log states that the MongoDB server has gone off-line due to heartbeat failure. One possible cause for this error is that MongoDB is no longer running on the system where the database is located. Another possibility is that the connection information is wrong. The IP address or database name could have a typo or could be pointing to a database that does not exist.

In the situation where the selected database has gone offline, the API returns a status code 500 (server error) for most requests. This is because no connection to the database can be established; and therefore, the API cannot retrieve the necessary data.

```
2023-07-28 17:29:04.402 ERROR 192.168.33.1 tibit@tibitcom.com GET -/  
v2/olts/alarm-configs/Default/ status: 500
```

```
"GET -/v2/olts/alarm-config/Default/ HTTP/1.1" 500 180
```

```
2023-07-28 17:29:04.474 ERROR 192.168.33.1 tibit@tibitcom.com GET -/  
olt/alarm/configuration/identifiers/status: 500
```

A request to the API can fail due to validation errors. The following error occurs when updating an ONU-CFG without the required CNTL property. The server rejects the configuration that does not contain all required fields, resulting in a 400 level status code warning, and an INFO level log stating that the JSON validation failed due to the missing required property.

```
WARNING 10.3.8.104 tibit@tibitcom.com PUT -/v1/onus/configs/  
CIEN00000000/ status: 400
```

When the client sends a request that the server does not understand, a 400 level status code is returned. This error occurs when a user makes a request through any means other than the browser. One example of this occurs when the URL is improperly formatted. The URL below includes a random % (percent sign) character which causes the error.

```
PUT {{host}}/v1/olts/{{olts}}%disable-onu/{{onu}}/
```

The following invalid URI path error appears in the Apache error log.

```
AH10244: invalid URI path (/api/v1/olts/Default/%disable-onu/  
CIEN00099629/)
```


MicroClimate Management System

REST API Developer Guide

Copyright© 2024 - 2026 Ciena® Corporation. All rights reserved.

MCMS 6.2

Publication: 323-1961-306

Document Status: Standard

Revision A

Document release date: March 2026

CONTACT CIENA

For additional information, office locations, and phone numbers, please visit the Ciena web site at **www.ciena.com**

