Vertiv™ Liebert® IntelliSlot™ RDU120 API/CLI Specification

User Guide

The information contained in this document is subject to change without notice and may not be suitable for all applications. While every precaution has been taken to ensure the accuracy and completeness of this document, Vertiv assumes no responsibility and disclaims all liability for damages result from use of this information or for any errors or omissions.

Refer to local regulations and building codes relating to the application, installation, and operation of this product. The consulting engineer, installer, and/or end user is responsible for compliance with all applicable laws and regulations relating to the application, installation, and operation of this product.

The products covered by this instruction manual are manufactured and/or sold by Vertiv. This document is the property of Vertiv and contains confidential and proprietary information owned by Vertiv. Any copying, use, or disclosure of it without the written permission of Vertiv is strictly prohibited.

Names of companies and products are trademarks or registered trademarks of the respective companies. Any questions regarding usage of trademark names should be directed to the original manufacturer.

Technical Support Site

If you encounter any installation or operational issues with your product, check the pertinent section of this manual to see if the issue can be resolved by following outlined procedures.

Visit https://www.vertiv.com/en-us/support/ for additional assistance.

TABLE OF CONTENTS

This page intentionally left blank

1 Overview

1.1 Notice to Users

Vertiv reserves the right to make changes to this document without notice to any user or reseller of this product. Vertiv also reserves the right to substitute or terminate distribution of this document, with no obligation to notify any person or party of such substitutions or terminations.

1.2 Copyrights

1.3 Trademarks

All Trademarks contained herein are registered to Vertiv Group Corp.

1.4 Use and Disclosure Restrictions

The software and documentation contained in this publication are copyrighted materials.

1.5 Document Usage

All reasonable efforts have been made to provide the accuracy of this document from any technical or typographical errors or missions. Vertiv and its affiliates disclaim responsibility for any labor, materials, or costs incurred as a result of usage of this document. No shall Vertiv and its affiliates be liable for any damages, inclusive of loss of profits or data, arising from the use of or in connection with this document.

1.6 Reporting Document Errors

Should you discover any error or identify a deficiency in this document, please take time to contact us at the following email address: Vertiv-Documents@vertiv.com. Please be sure to provide us with the document name, part number, and page number(s). Also, please provide us with description of the error or the deficiency for the document. If you would like for us to contact you, please provide us with your name and contact information. Thank you for your time. We appreciate any comments and feedback you can provide.

1.7 Document Revision

Table 1.1 Document Revision

Revision	Notes
1.0.0	Initial document release.
1.1.0	• Added user lockout policy. • Added user password expiration. • Added sample passwordRules SET command to Authentication Configuration: /api/auth/conf on page 1. • Updated range limits for Authentication: /api/auth on page 34, and Configuration: /api/conf on page 1. • Updated System: /api/sys on page 213 table to show current keys and added sample GET responses. • Removed dhcpEnabled from Network/Interface/LAN on page 1. • Added file transfer via Special File Transfers: /Transfer, SCP on page 292. • Added notes for configuration Backup Download on page 293 and Backup Upload on page 293. • Added note for a valid "network address" on SNMP v1/v2c Trusted Access on page 163. • Extended Filtering on page 19 section to include filtering based on a dynamic object's ID. • Added source highlighting to JSON and YAML-formatted entries. • Added /api/alarm documentation. • Added /api/dev documentation for sensors. • Extended /equipment to include multi-module info. • Modified all IDs in Equipment on page 303 API to Camel case. • Add /transfer/equipment/firmware and /equipment/firmware points.
1.2.0	• Added /api/conf/systemResource documentation. • Limit the configuration to one static IPv4 address and one static IPv6 address, and restrict DNS settings to a maximum of two IPv4 and two IPv6 DNS servers. • Added /api/conf/datalogging documentation. • Added documentation for new data points related to email event consolidation and SMTP authentication. • Updated SSH enabled default from 'true' to 'false'. • Updated assetTag02Ext3 upper range from 23 to 32.
1.3.0	• Added /api/conf/ydn23/server documentation. • Rewrote Remote Syslog on page 146 or multiple syslog targets (up to two), per-target enabled, server, port, and transport (udp/tls), including CLI examples. • Updated configuration summary rows for remote syslog under Configuration: /api/conf on page 55. • Added PLD Import on page 297. • Added primary HTTP method examples (GET, POST, PUT, DELETE) and crossreferences to conventional usage alongside existing explicit cmd examples across API documentation. • Normalized indentation to two spaces in JSON and YAML source blocks for consistency. • Added Remote Syslog mTLS Certificate Upload on page 298 (Syslog client PEM (Mutual TLS) on page 298, Remote syslog target PEM (Server authentication) on page 300). • Updated Equipment Firmware Update on page 297 to describe polling Firmware on page 320 for verification and update status after upload. • Updated Firmware on page 320 with an invalid firmware response example and clarified that image verification errors can appear in data while the GET returns retCode 0. • Corrected Action email format example under Action on page 234 to match the updated email format.

1.8 API Version

Table 1.2 API Version

Version	Product	Notes
1.0.0	v1.4.1	Initial release.
1.1.0	v1.5.0	• Added Lockout Policy Rules on page 51, added locked under User on page 36. • Added Expiry Policy on page 53, added passwordExpiry under User on page 36. • All dynamic objects will check for duplicate IDs when performing an ADD command. • Removed dhcpEnabled from Network/Interface/LAN on page 124. • Modified all IDs in Equipment on page 303 API to Camel case. • Added System Resource on page 186.
1.2.0	v1.6.0	• Added Data Logging on page 81. • Added enableEventConsolidation, consolidationEventLimit, consolidationTimeLimit, and smtpAuthentication under Email on page 94. • Updated SSH enabled default from 'true' to 'false' in SSH on page 182.
1.3.0	v1.7.0	• Added YDN23 Server on page 211. • Remote syslog configuration uses syslogRemote Syslog Target on page 147 entries with UDP or TLS transport; replaces the prior single-address model. • Added PLD Import on page 297. • Added Remote Syslog mTLS Certificate Upload on page 298 (mTLS certificate upload). • Updated Equipment Firmware Update on page 297 documentation to describe polling Firmware on page 320 are for verification and update status after upload. • Updated Firmware on page 320 with an invalid firmware response example and clarified that image verification errors can appear in data while the GET returns retCode 0. • Corrected Action email format example under Action on page 234 to match the updated email format.

1.9 Glossary

Table 1.3 Glossary/Compatibility

Version	Note
v1	RDU120 running firmware 1.3.0 or newer

This page intentionally left blank

2 API

2.1 API Usage

The RDU120 Web API is designed to provide developers and integrators an easy to use method to communicate with the device. All of the device's actions can be accessed through this API over HTTP or HTTPS using JSON data structures, with **GET**, **POST**, **PUT**, and **DELETE** on /api/ paths as the primary calling style, and an explicit cmd field in the JSON body or query string as a fully supported alternative (or the only option for a few operations, as described under Requests below).

2.1.1 Paths

All client requests are performed on a path starting with /api/ (an example usage would be http://<ip_address>/api/conf/network) each node in the path represents a key in the JSON hierarchy. Thus, /api/ is the top of the tree, and a request on that level will affect the entire API tree. /api/ contains the keys "dev", "alarm", "conf", "sys", and "auth", so to call commands on "conf" directly, the HTTP request will be directed at the path /api/conf/. /api/conf/ has keys like "network", "system", "contact", etc., so to access "network", use the path /api/conf/network/, and so on. Paths can be narrowed down all the way to the leaves of the tree, for example: /api/conf/network/dns/0/address.

2.1.2 Requests

The usual way to call the API is to send an HTTP request to a path under /api/ and let the **HTTP method** select the operation when cmd is omitted from both the JSON body and the query string: **GET** performs a **get**, **POST** performs a set, **PUT** performs an **add**, and **DELETE** performs a **delete**. For POST, PUT, and other methods that accept a body, parameters are passed in a top-level data object (same shape as in the examples below). Reads that do not need a body may be issued as **GET** with an empty body.

Authentication can be sent as Authorization: Bearer <token> (recommended for integrations that already hold a session token from logins, see User on page 36), or using token, username, and password fields in the JSON body or query string as described later in this section.

Alternative (cmd syntax): You may instead include an explicit cmd field in the JSON body (or query string) to name the operation. That form is required for commands that are not selected by the HTTP method mapping alone: **merge**, **control**, **ack**, **login**, **logout**, **reset**, **sendTest**, and **reboot**. You may also use explicit cmd for **get**, **set**, **add**, or **delete** when you prefer a single POST-style client.

The following illustrates a get using the primary HTTP style (no cmd in the body):

```
GET http://<ip_address>/api/conf/email/ HTTP/1.1
Authorization: Bearer a4b3c2d1
```

The same logical **get**, using the alternative body that names cmd explicitly (here including a filter, placeholder, See Filtering on page 19).

```
{
    "token": "a4b3c2d1", ①
    "cmd": "get",        ②
    "data": {},          ③
    "filter": {}         ④
}
```

① A session ID issued by the server on logins, see User on page 36. If the token is not supplied, then the blank token ("") is used and the command will be attempted as the guest user. See Remote Authentication on page 36.

② When cmd is present, it selects the operation directly. When cmd is omitted from the JSON object and the query string, the operation is inferred from the HTTP method as described above. Supported cmd values are: "get", "set", "merge", "add", "control", "delete", "ack", "login", "logout", "reset", "sendTest", and "reboot".

③ Parameters for the command are passed in the "data" object. These are only needed for "set", "merge", "add", "control", "login", and "reset". This will generally be a JSON object, but when setting on a leaf it will be any other datatype.

④ Filter to be applied to the command. Only applicable to the "get" command. Will be a JSON object. See Filtering on page 19.

Alternatively, a username and password can be submitted on each command en lieu of logging in and keeping track of the session token:

```
{
    "username": "admin",
    "password": "password",
    "cmd": "get"
}
```

If a token is supplied alongside a username and password in a request, the "Invalid credential combination" error is returned.

In addition to sending these parameters as JSON fields, the "username", "password", "token", and "cmd" fields can instead be sent in as the query string, so the alternative form can be conveyed as an HTTP GET (or other method) to a path like /api/conf?username=admin&password=password&cmd=set, making the HTTP body optional in any command that doesn't require a "data" field. If the same field is supplied in both the query string and the JSON object, the JSON object contents will take precedence.

Unless a subsection explicitly labels a request as an HTTP method (GET, POST, PUT, DELETE), the Request examples under Commands use the alternative cmd style for a compact, uniform JSON shape; they remain fully supported and are equivalent to the HTTP-mapped style where both apply. See Commands on the facing page.

2.1.3 Responses

Responses to client requests come back with an JSON-format HTTP body as follows:

```
{
    "retCode": 0,        ①
    "retMsg": "Success", ①
    "data": {}           ②
}
```

① Non-zero return codes represent failures. See Error Codes on page 22 for more on retCodes and their corresponding retMsg strings.

② Returned data. This will generally be a JSON object, but when getting a leaf it will be any other datatype. The object will come back as empty for any command other than "get" and "login".

2.1.4 Commands

Logical API operations are listed below. For get, set, add, and delete, the preferred integration style is to use the HTTP method on the URL path and omit cmd, as described in Requests on page 5. The explicit cmd field is still supported for those operations (see the Request examples in each subsection) and is required for merge, control, ack, login, logout, reset, sendTest, and reboot.

The following API commands exist: get, set, add, merge, control, delete, ack, login, logout, reset, sendTest, and reboot.

Get

Get operations are only valid on objects that exist. Most operations will affect only the node at the path specified, but two — "get" and "set" — are also recursively applied to every child under the specified node. Thus, a get on /api/conf/email/ will return a JSON object starting at the depth indicated by the path, and traversing down to the leaf objects, like so:

API: *GET /api/conf/email/*

```
GET http://<ip_address>/api/conf/email/ HTTP/1.1
Authorization: Bearer <token>
```

API (cmd): */api/conf/email/target : ge*t

```
{
    "token": "",
    "cmd": "get",
    "data": {}
}
```

Response


```
{
    "data": {
        "port": 25,
        "subjectPrefix": "",
        "username": "",
        "server": "",
        "password": null,
        "sender": "",
        "passwordSet": false,
        "enableEventConsolidation": false,
        "consolidationEventLimit": 20,
        "consolidationTimeLimit": 60,
        "smtpAuthentication": false,
        "target": []
    },
    "retCode": 0,
    "retMsg": "Success"
}
```


Walking further down the tree, the responses get smaller and smaller:

API: *GET /api/conf/email/target*

```
GET http://<ip_address>/api/conf/email/target HTTP/1.1
Authorization: Bearer <token>
```

API (cmd): */api/conf/email/target : get*

```
{
    "token": "",
    "cmd": "get",
    "data": {}
}
```

Response


```
{
    "data": [
        {
            "name": "john.z.smith@myorg.com",
            "id": "0"
        }
    ],
    "retCode": 0,
    "retMsg": "Success"
}
```

API: *GET /api/conf/email/target/0*

```
GET http://<ip_address>/api/conf/email/target/0 HTTP/1.1
Authorization: Bearer <token>
```

API (cmd): */api/conf/email/target/0 : get*

```
{
    "token": "",
    "cmd": "get",
    "data": {}
}
```

Response

```
{
    "data": {
        "name": "john.z.smith@myorg.com"
    },
    "retCode": 0,
    "retMsg": "Success"
}
```

And finally at a leaf:

API: *GET /api/conf/email/target/0/name*

```
GET http://<ip_address>/api/conf/email/target/0/name HTTP/1.1
Authorization: Bearer <token>
```

API (cmd): */api/conf/email/target/0/name : get*

```
{
    "token": "",
    "cmd": "get",
    "data": {}
}
```

Response

```
{
    "data": "john.z.smith@myorg.com",
    "retCode": 0,
    "retMsg": "Success"
}
```

Set

Set operations are only valid on objects that exist. Like the Get operation, set operations are also recursively applied to every child under the specified node. Thus, a set on /api/conf/email/ will traverse down to the leaf objects, unincluded fields are left unchanged:

API: *POST /api (set)*

```
POST http://<ip_address>/api HTTP/1.1
Authorization: Bearer <token>
Content-Type: application/json

{
    "conf": {
        "system": {
            "label": "my-ups"
    },
    "email": {
        "username": "john.w.smith@myorg.com"
        }
    }
}
```

API (cmd): */api : set*

```
{
    "username": "admin",
    "password": "password",
    "cmd": "set",
    "data": {
        "conf": {
            "system": {
                "label": "my-ups"
            },
            "email": {
                "username": "john.w.smith@myorg.com"
            }
        }
    }
}
```

Response

```
{
    "data": {},
    "retCode": 0,
    "retMsg": "Success"
}
```

Or just:

API: *POST /api/conf/email/target/0/name (set)*

```
POST http://<ip_address>/api/conf/email/target/0/name HTTP/1.1
Authorization: Bearer <token>
Content-Type: application/json

{
    "name": "mr.user1@email.com"
}
```

API (cmd): */api/conf/email/target/0/name : set*

```
{
    "username": "admin",
    "password": "password",
    "cmd": "set",
    "data": "mr.user1@email.com"
}
```

Response

```
{
    "retCode": 0,
    "retMsg": "Success",
    "data": {}
}
```

Add

An Add operation is used when creating a new key. Attempting to create a key that exists will result in a "Cannot add..." error. The following is an example of successfully adding a new user.

API: *PUT /api/auth/user (add)*

```
PUT http://<ip_address>/api/auth/user HTTP/1.1
Authorization: Bearer <token>
Content-Type: application/json

{
    "id": "merger1",
    "password": "Rdu120main123$",
    "source": "local",
    "scope": null,
    "enabled": true,
    "publicKey": [],
    "privilege": "administrator"
}
```

API (cmd): */api/auth/user : add*

```
{
    "username": "rdu120-main",
    "password": "Rdu120main123$",
    "cmd": "add",
    "data": {
        "id": "merger1",
        "password": "Rdu120main123$",
        "source": "local",
        "scope": null,
        "enabled": true,
        "publicKey": [],
        "privilege": "administrator"
    }
}
```

Response

```
{
    "data": {
        "id": "merger1"
    },
    "retCode": 0,
    "retMsg": "Success"
}
```

The following is an example of trying to add a user that already exists.

API: *PUT /api/auth/user (add)*

```
PUT http://<ip_address>/api/auth/user HTTP/1.1
Authorization: Bearer <token>
Content-Type: application/json

{
    "id": "merger1",
    "password": "Rdu120main123$",
    "source": "local",
    "scope": null,
    "enabled": true,
    "publicKey": [],
    "privilege": "administrator"
}
```

API (cmd): */api/auth/user : add*

```
{
    "username": "rdu120-main",
    "password": "Rdu120main123$",
    "cmd": "add",
    "data": {
        "id": "merger1",
        "password": "Rdu120main123$",
        "source": "local",
        "scope": null,
        "enabled": true,
        "publicKey": [],
        "privilege": "administrator"
    }
}
```

Response

```
{
    "data": {},
    "retCode": 1001,
    "retMsg": "Cannot add user: merger1"
}
```

Merge

The merge operation does not currently work on Vertiv™ Liebert® IntelliSlot™ RDU120.

MERGE is only expressed with an explicit cmd of merge in the JSON body or query string; there is no separate HTTP method that selects merge.

A MERGE operation is identical to a SET operation, with the following exceptions:

- MERGE operations will attempt Add on Set below behavior. If the MERGE command encounters a key which doesn't exist, it will attempt to ADD it. If this key does not support ADD operations, it will skip that key and suppress the "key not found" error.
- The MERGE operation will skip any read-only fields it encounters, suppressing the generated "read only" error.

If no part of the MERGE command is successful, the command will return an error. The MERGE command will not suppress errors having to do with field content restrictions or access restrictions, only those relating to field presence.

Add on Set

When a MERGE operation is called on a part of the tree that does not yet exist, an ADD operation will be attempted. This may be used, for example, to create several objects at once instead of calling individual ADD operations. This behavior only affects nodes for which both ADD and SET are allowed operations.

For example, if the following command were executed by a user with admin permissions, admin privileges, if present, would be removed from "extantUser". When "newUser" is not found in the API, instead of refusing to complete the command, "newUser" will be added as a new control-level user with the specified settings.

▼ API: *api/auth/user: merge (Admin)*

api/auth/user: merge *(Admin)*

```
{
    "username": "rdu120-main",
    "password": "Rdu120main123$",
    "cmd": "merge",
    "data": {
        "id": "merger",
        "password": "Merger123$",
        "source": "local",
        "scope": null,
        "enabled": true,
        "publicKey": [],
        "privilege": "administrator"
    }
}
```

▼ CLI: merge auth

admin> merge auth = ARGS

```
merge auth = {"extantUser": {"admin": false}, "newUser": {"password":
"passwordString", "enabled": true, "control": true, "admin": false}}
```

Control

This feature is not yet implemented nor required. Look for this in future versions of the API.

CONTROL will require an explicit cmd when implemented; it is not selected by HTTP method alone.

Delete

A Delete operation is used to remove an existing object using the "id" of the object. After the operation the object will no longer exist. The following example will remove user 'myuser' using the 'rdu120-main' administrator account.

API: *DELETE /api/auth/user/myuser*

```
DELETE http://<ip_address>/api/auth/user/myuser HTTP/1.1
Authorization: Bearer <token>
```

API (cmd): */api/auth/user/myuser : delete*

```
{
    "username": "rdu120-main",
    "password": "Rdu120main123$",
    "cmd": "delete"
}
```

Response

```
{
    "data": {},
    "retCode": 0,
    "retMsg": "Success"
}
```

Ack

This feature is not yet implemented nor required. Look for this in future versions of the API.

ACK will require an explicit cmd when implemented; it is not selected by HTTP method alone.

Login

The Login command is used to authenticate an API user using username and password. A successful authentication results in an API token being returned along with other attributes concerning the authenticated user. The API token can be used in subsequent API transactions that require an authenticated user.

Login is only available using an explicit cmd of login in the JSON body or query string (HTTP method alone does not select it).

The system will keep track of the last 4 unique tokens per user. Each token will expire after 6 hours of inactivity or until the user logs out. Repeated authentication failures on a given user will trigger a lockout for that user. After 2 authentication failures, any subsequent failure will trigger a 60 minute lockout period. During the period, any attempts to log in will result in the same authentication error returned when the supplied password was incorrect. A successful login attempt will clear the count of authentication failures for the user. Attempting to log in as a lockedout user will restart the lockout period.

API (cmd): */api/auth/user/rdu120-main : login*

```
{
    "cmd": "login",
    "username": "rdu120-main",
    "password": "Rdu120main123!"
}
```

Response

```
{
    "data": {
        "token": "ENo1eGki-htxg9yr66CjV_HYGMqNXM5p",
        "privilege": "administrator",
        "source": "local",
        "scope": null
    },
    "retCode": 0,
    "retMsg": "Success"
}
```

Attribute	Description
Token	The API token used in subsequent authenticated API calls.
Privilege	The user's assigned privilege.
Source	The authentication source.
Scope	The authenticated API scope available for this user. **null** means unlimited scope.

Logout

The Logout command is used to remove authenticated privileges for an API user.

Logout is only available using an explicit cmd of logout in the JSON body or query string.

API (cmd): */api/auth/user/user4 : logout*

```
{
    "cmd": "logout",
    "token": "19pdhGq7o8ihMA6AWyr8o6ANHdFm-J6Z"
}
```

Response

```
{
    "data": {},
    "retCode": 0,
    "retMsg": "Success"
}
```

Reset

The Reset command is used to reset the card according to the target value.

Reset is only available using an explicit cmd of reset and a data object (HTTP method alone does not select it).

API (cmd): */api/sys : reset*

```
{
    "token": "r5ldO45TAe3Yc_tvXFU-c5XE3Yz4oIGG",
    "cmd": "reset",
    "data": {
        "target": "fullDefaults"
    }
}
```

Response

```
{
    "data": {},
    "retCode": 0,
    "retMsg": "Success"
}
```

SendTest

The SendTest command is used to send test emails or SNMP traps in order to test configured emails or SNMP trap targets. The following example shows a SendTest command being used to send a test email to the configured email target with id of 1.

SendTest is only available using an explicit cmd of sendTest in the JSON body or query string.

API (cmd): */api/conf/email/target/1 : sendTest*

```
{
    "cmd": "sendTest",
    "token": "19pdhGq7o8ihMA6AWyr8o6ANHdFm-J6Z"
}
```

Response

```
{
    "data": {
        "id": "1"
    },
    "retCode": 0,
    "retMsg": "Success"
}
```

Reboot

The Reboot command is used to reboot the system.

Reboot is only available using an explicit cmd of reboot in the JSON body or query string.

API (cmd): */api/sys : reboot*

```
{
    "cmd": "reboot",
    "token": "x5BE4BUdQ71rb6L8m0IK2rFR__AuHgD4"
}
```

Response

```
{
    "data": {},
    "retCode": 0,
    "retMsg": "Success"
}
```

2.1.5 Filtering

For filtered reads, use GET on the path you would otherwise read and send the filter tree in the JSON request body. The GET method selects the get operation, so cmd is not required. Authentication can use Authorization: Bearer and/or a token field in the body, like other requests.

The examples below show the primary GET style first, then the same request using explicit cmd get (see Requests on page 5).

The filter value is a JSON object. It can be used to build custom queries that return a response containing specific keys. Each key in the filter tree represents a desired key to be returned, with 2 special cases:

- **%/regex:** JSON keys beginning with "%/" will be interpreted as POSIX-extended regular expressions. Any keys at this level in the API which do not match this expression will be filtered out of the response.
- **{} or null:** JSON values that are empty objects or null return every descendent of their key, unfiltered.

API: *GET /api (filtered get)*

```
GET http://<ip_address>/api HTTP/1.1
Authorization: Bearer <token>
Content-Type: application/json

{
    "token": "psbr3BgfsSfmlm1RzzTpRWJ5JnHzc-Tr",
    "filter": {
    "conf": {
        "%/email": {
            "%/target": {},
            "%/server": {}
            }
        }
    }
}
```

API (cmd): */api : get (filter)*

```
{
    "token": "psbr3BgfsSfmlm1RzzTpRWJ5JnHzc-Tr",
    "cmd": "get",
    "filter": {
        "conf": {
            "%/email": {
                "%/target": {},
                "%/server": {}
            }
        }
    }
}
```

Response (selected portions)

```
{
    "data": {
        "conf": {
            "email": {
                "server": "amr-smtp.int.vertivco.com ",
                "target": [
                    {
                        "name": "john.z.smith@myorg.com",
                        "id": "0"
                    },
                    {
                        "name": "sara.t.jones@myorg.com",
                        "id": "1"
                    }
                ]
            }
        }
    },
    "retCode": 0,
    "retMsg": "Success"
}
```

Filter based on where target/id value is "0".

API: *GET /api (filtered get)*

```
GET http://<ip_address>/api HTTP/1.1
Authorization: Bearer <token>
Content-Type: application/json

{
    "token": "0eoEC-sd6WGO_XN4lpImwyfHRkrjIzqw",
    "filter": {
        "conf": {
            "%/email": {
                "%/server": {},
                "%/target": {
                    "%/0": {}
                }
            }
        }
    }
}
```

API (cmd): */api : get (filter)*

```
{
    "token": "0eoEC-sd6WGO_XN4lpImwyfHRkrjIzqw",
    "cmd": "get",
    "filter": {
        "conf": {
            "%/email": {
            "%/server": {},
            "%/target": {
                "%/0": {}
                }
            }
        }
    }
}
```

Response (selected portions)

```
{
    "data": {
        "conf": {
            "email": {
                "server": "amr-smtp.int.vertivco.com",
                "target": [
                    {
                        "name": "john.z.smith@myorg.com",
                        "id": "0"
                    }
                ]
            }
        }
    },
    "retCode": 0,
    "retMsg": "Success"
}
```

2.1.6 Error Codes

Table 2.1 Success

Code	Name	Notes
0	Success	Operation has succeeded.

Table 2.2 Authentication Errors (1000-1999)

Code	Name	Notes
1000	No Admin user configured	At least one Admin user must be configured on the system.
1001	Not Authorized	The current user is not authorized.
1002	Not Authorized: Session expired	The token used is no longer valid.
1003	Not Authorized: Not enough permissions	The current user does not have enough permissions to perform the operation.
1006	Invalid credential combination	Both username/password and token were provided or only one of username or password was provided.
1008	Must have at least one admin user	At least one Admin user must be configured on the system.

Table 2.3 JSON Format Errors (2000-2999)

Code	Name	Notes
2000	Malformed JSON	Received JSON is not valid or corrupt.
2001	Missing field	An expected field was not found in the JSON structure.
2002	Duplicate fields	The same field was set multiple times, e.g. in the HTTP body and query string.

Table 2.4 Path Errors (3000-3999)

Code	Name	Notes
3000	Invalid path	Supplied path does not fulfill system requirements.
3001	Path not found	Supplied path was not found.
3002	Identifier not found	One of the fields in the received JSON structure does not exist.
3003	Field not applicable	A field in the JSON structure exists but should not have been sent.

Table 2.5 Data Validation Errors (4000-4999)

Code	Name	Notes
4000	Invalid input	An input field is invalid but does not fit in other data validation categories.
4001	Input too long	An input field exceeds the maximum allowed length.
4002	Invalid characters	An input field contains invalid characters for the field.
4003	Invalid serial	An input field is an invalid serial number.
4004	Invalid boolean	An input field is an invalid boolean value.

Table 2.5 Data Validation Errors (4000-4999) (continued)

Code	Name	Notes
4005	Out of range	An input field falls outside the valid range for the field.
4006	Invalid integer	An input field is not an integer when one is expected.
4007	Invalid number	An input field is not a number when one is expected.
4008	Invalid URL	An input field is not a valid URL when one is expected.
4009	Invalid IP	An input field is not a valid IP address when one is expected.
4010	Paths not allowed	An input field contains a path when one is not expected.
4011	Invalid username	An input field is an unsupported user name.
4012	Invalid email address	An input field is not a valid email address when one is expected.
4013	Invalid option	An input field contains an invalid option selection.
4014	Invalid datetime	An input field is not a valid date or time when one is expected.
4015	Out of bounds	An input field is out of the allowed bounds for the field.
4016	Invalid week	An input field represents an invalid days of the week selection.
4017	Duplicate entry	An input field would create a duplicate when one is not allowed.
4018	Invalid Route	A network route was misconfigured.
4019	Invalid password	An input field is not a valid password when one is expected
4020	Invalid network address	An input field is not a valid network address when one is expected

Table 2.6 Other Errors (5000-5999)

Code	Name	Notes
5000	Unknown error	A system error occurred for which no other error code applies.
5001	Command not allowed	The received command is not allowed at the specified path.
5002	System busy	The action attempted cannot be currently executed and should be retried.

Table 2.7 Data Consistency Errors (6000-6999)

Code	Name	Notes
6000	Inconsistent state	The command will leave the system in an inconsistent state so it is rejected.
6001	Syslog enabled requires target	Enabling remote syslog requires a target host be specified.
6002	NTP mode requires servers	Enabling Network Time Protocol (NTP) requires servers to query.
6003	Start time must come before end time	Time was received for which the end came before the start.
6004	Invalid SNMPv3 auth/priv combination	SNMPv3 privacy cannot be used without authentication.
6005	Port not available	There was an attempt to set a port number to one already in use.
6006	OneView missing credentials	Enabling OneView requires that a OneView username and password be set.
6007	Time not settable	Setting datetime requires manual time mode.

Table 2.8 Upload Errors (7000-7999)

Code	Name	Notes
7000	Invalid firmware package	The package is formatted incorrectly or corrupt.
7001	Invalid file key	The package specifies a wrong OEM key and cannot be used with this unit.
7002	Invalid version	The version is too old or otherwise unsupported.
7003	Invalid product	The package is meant for a different hardware architecture.
7004	Invalid certificate file	The SSL certificate provided could not be parsed.
7005	Invalid certificate password	The password did not work with the SSL certificate provided.
7007	Firmware update already in progress	A firmware update is already in progress and cannot be interrupted
7008	Invalid CA certificate file	The 802.1x Certificate Authority (CA) certificate provided is not valid
7009	Invalid client certificate file	The 802.1x client certificate provided is not valid
7010	Invalid private key password	The 802.1x private key password provided is not valid
7011	Invalid CN in certificate file	A warning after a successful upload indicating that the Common Name (CN) in the certificate does not match the hostname
7012	Invalid .zip file	The 802.1x .zip file provided is invalid, corrupt, or missing required certificate files
7013	Invalid PLD package	The PLD package file provided is invalid, incompatible package version or missing required fields
7014	Invalid syslog TLS CA certificate file	The syslog TLS Certificate Authority (CA) certificate provided is not valid
7015	Invalid syslog TLS client certificate file	The syslog TLS client certificate provided is not valid

Table 2.9 Apache Custom Error Code

Code	Apache	Name	Notes
1001	401, 403	Not authorized	The current user is not authorized
1001	407	Proxy Authentication Required	Proxy requires authentication before proceeding
1001	511	Network Authentication Required	Authentication is required to access the network
4000	400	Invalid input	An input field is invalid but does not fit in other data validation categories
5000	500	Unknown error	A system error occurred for which no other error code applies
5002	429	System busy	The action attempted cannot be currently executed and should be retried
5003	404, 408, 502, 503, 504	System Unreachable	Unable to establish connection with system

2.1.7 Error Precedence

API command errors are processed and returned in a specific order. Once an error is encountered, the command is terminated and the error code returned immediately. If multiple errors exist, only the one with the highest precedence is returned. Within a given precedence category, the order in which errors are returned is command and implementation dependent.

Errors are checked in three stages:

1. Request integrity: Verifies that the JSON object, path, and command are valid. These conditions are checked in the following order:

Precedence	Description	Error Codes
1	JSON is valid, path is valid and can be resolved.	2000, 3000, 3001
2	Top-level fields outside of "data" are valid.	1006, 2002
3	Command exists for the given path.	5001

2. Authentication and token: Verifies that the token provided is valid and that the user has enough permissions to execute the command. The existence of an admin user is also verified at this stage. These conditions are checked in the following order:

Precedence	Description	Error Codes
1	An admin user exists OR the request is on a permanently available path OR the request is the addition of an admin user when one does not exist.	1001
2	Token is valid, recognized, and unexpired.	1002
3	User is authorized to execute command.	1003, 1000

3. Command execution: Verifies data received in relation to the command. These conditions are not checked in any particular order and the error returned depends on the command and data present.

Description	Error Codes
User operations	1008
JSON object key validity, presence or absence	3002, 3003
Data validation	4000, 4001, 4002, 4003, 4004, 4005, 4006, 4007, 4008, 4009, 4010, 4011, 4012, 4013, 4014, 4015, 4016, 4017, 4018, 4019, 4020
Data consistency	2001, 6000, 6001, 6002, 6003, 6004, 6005
Firmware upload	7000, 7001, 7002, 7003
Any other error	5000, 5002

2.1.8 CLI

As an alternative to the API described above, a command-line interface is also available over SSH or a serial port when available, using the same tree structure, but a modified syntax. This section describes the modified syntax. The CLI can be used with either this modified syntax or with the original API syntax. The CLI also supports Filtering on page 19, including the use of regex.

Logging in to the CLI will work as follows:

- **SSH:** Connect over secure shell using the desired username and password combination.
- **Serial Port:** Pressing the "enter" key will display a username prompt. Enter the user name followed by the "enter" key and a password prompt will appear.

Any API commands will be authenticated as the user that logged in. To log in as guest, use the username "guest" and the password "guest". The CLI has an idle timeout of 10 minutes, after which the active user will be logged out.

Every message sent to the CLI will have the following syntax:

```
COMMAND PATH [= ARGUMENT]
```

COMMAND is the only required part. It is a single word from the following list: get, set, logout, add, delete, control, ack, sendTest, reset, reboot, diagnosticApp, and help.

The "guest" user is limited to adding an administrator user when and administrator doesn't exist, along with the get, logout and help commands.

PATH is the location in the tree that you want to apply the COMMAND. Starting at the root (/api), each child key is separated by spaces. The path "/api/conf/network" in the API is represented as "conf network" in the CLI. Available trees in the CLI include sys, alarm, auth, conf, and dev. The api tree is considered the default tree for the CLI, and can be accessed by omitting the tree name, as in "conf network", or by including the tree name, as in "api conf network".

ARGUMENT, if present, is always preceded by an equal sign and is in JSON or YAML format. It is only used with the COMMANDs set, merge, add, control, and reset. Strings in ARGUMENT should be enclosed in quotes if they contain special characters like space, tilde, colon, or comma or certain keywords, like null. A null value in ARGUMENT may be represented with either the word "null", without quotes, or with a tilde (~). As such, the strings "null" and "~" must be enclosed in quotes, or they will be interpreted as null.

Get

The Get CLI command is used to get elements from the API tree. The result is returned using YAML. To see what the YAML output looks like, login as the "guest user" and type the following command:

Request

```
get auth
```

Response

```
user:
    -     scope: ~
        source: local
        privilege: view
        password: ~
        enabled: true
        passwordSet: false
        id: guest
conf:
    passwordRules:
        allowRepeatCharacters: false
        allowUsernameInclusion: false
        minDigits: 0
        minUppercase: 0
        minSymbols: 0
        minLength: 8
    scope: ~
```

To see what the entire tree looks like, login as an administrator user and type the following command:

Request

```
get
```

Response

```
---
dev: ~
conf:
    usb:
        dev: ~
        enabled: true
    serial:
        portParameters: 8N1
        enabled: true
        baudRate: 115200
    velocity:
        ethernet:
            ipAddress: ""
```

```
            port: 47808
            networkNumber: 1000
            enabled: false
        serial:
            nodeId: 1
            networkNumber: 1001
            maxAddress: 127
            nodeIdAssignmentMethod: automatic
            baudRate: 38400
        internal:
            networkNumber: 1010
        managedDevice:
            psi: 0
            connectStatus: unsupported
            lanType: velocity-serial
    system:
        factoryAccessEnabled: false
        location: Jim's Cube
        description: GXT5 Desktop
        contact: Jim Taylor
        label: my-ups
...
```

The entire data tree is returned in YAML format. You'll see some top-level keys like dev, alarm, sys, conf, etc. To see just the system configuration, you could type:

And, paying attention to the keys returned, go any number of levels deeper:

```
get conf system location
```

Set

The Set CLI command is used to modify configuration and control.

To set the system location:

Request

```
set conf system location = Delaware Ohio
```

Response

```
--- ①
```

① CLI Success response

"set" commands are special like "get" commands in that they can be applied at any level:

Request

```
set conf system = {location: Delaware Ohio, contact: Fred Flintstone}
```

Response

```
--- ①
```

The result of the above two set operations is show below:

Request

```
get conf system
```

Response

```
---
factoryAccessEnabled: false
location: Delaware Ohio
description: GXT5 Desktop
contact: Fred Flintstone
label: my-ups
```

Using the Web API syntax in the CLI

The CLI can process API requests in a similar format to the web API. This format requires "cmd" and "path" fields for get operations. This format requires "cmd", "path", and "data" fields for set operations. The "cmd" field can be any valid API command. The "path" field is any valid API endpoint. The "data" field is any valid leaf endpoint.

Using the Web Get API syntax in the CLI

To get the system information using API syntax:

Request

```
{"cmd": "get", "path": "api/conf/system/"}
```

Response

```
---
factoryAccessEnabled: false
location: Delaware Ohio
description: GXT5 Desktop
contact: Fred Flintstone
label: my-ups
```

To get the system contact information using API syntax:

Request

```
{"cmd": "get", "path": "api/conf/system/contact"}
```

Response

```
---
contact: Fred Flintstone
```

Using the Web Set API syntax in the CLI

To set the system information using API syntax:

Request

```
{ "cmd": "set", "path": "api/conf/system", "data": {"location": "Westerville Ohio",
"contact": "Barney Rubble"} }
```

Response

```
---
```

And the result is: .

Request

```
{"cmd": "get", "path": "api/conf/system/"}
```

Response

```
---
factoryAccessEnabled: false
location: Westerville Ohio
description: GXT5 Desktop
contact: Barney Rubble
label: my-ups
```

Using Filters in the CLI

The CLI supports filtering using API syntax. To specify a filter using the CLI and API syntax, use the **";filter:"** command as the last argument of the command. The filter supports he same regex filtering as described in Filtering on page 19.

For example, to get all email targets, the CLI specifies the email configuration key and the filter is programmed to return just the email targets:

Request

```
get conf email ;filter:{"%/target":{}}
```

Response

```
    ---
target:
    - name: john.z.smith@myorg.com
        id: 0
    - name: sara.t.jones@myorg.com
        id: 1
```

2.1.9 SSH Commands

Commands can be also passed from the command line of an external machine using SSH. Commands executed this way will set the exit code to 1 if there is an error and print the error message to STDERR. The syntax for SSH commands is the same as that of CLI commands.

Authentication should be passed in as if using regular API. The guest user can be used for this, if it is enabled, in which case the username/password combination is "guest"/"guest". SSH will request a password unless a valid token file is passed in, as with the **-i** flag.

SSH command example

```
ssh user@host get conf network ethernet ①②
ssh user@host set conf contact location = "Building 04"
```

① "user" should be replaced with the appropriate username.

② "host" should be replaced with the IP address of the host unit.

2.1.10 Data Formats

Table 2.10 Data Formats

Data Format	Details
API Path	An absolute path into the API JSON hierarchy.
Array	A sequence of objects, all of the same type. Array capacity is specified for each one. When doing a set on an array, the provided contents will replace the existing values.
Boolean	Binary condition. Can only be true or false.
Date/Time	RFC 2822 date format with whitespace separator and 4 digit timezone with sign. "YYYY-MM-DD HH:MM:SS +/-0000".
Days	A 7 character representation of the days of the week starting with Monday in the form "MTWTFSS". Used to select which days are to be used. Selected days are shown with the first letter of their name present. Unselected days are shown with a '-'. An example showing only Monday through Friday selected would be "MTWTF--".
Dev Path	A path into the API JSON hierarchy relative to /api/dev
Email Address	String from 1 to 254 characters in length.
Float	Number with fractional components. Unless otherwise noted, Float Min is -10000000.0 and Float Max is 10000000.0.
Host	String from 1 to 255 characters in length. Can contain an IP Address.
Hostname	1 to 63 characters complying with the rules for label in RFC 1035 and extended in RFC 1123. As per the RFC: "They must start with a letter or digit, end with a letter or digit, and have as interior characters only letters, digits, and hyphen."
ID	An identifier for an object of a particular type. When setting, must be a valid reference to an existing object.
Integer	Number without fractional components. Unless otherwise noted, Integer Min is -10000000 and Integer Max is 10000000
IP Address	An IP address fitting either the IPv4 Address or IPv6 Address formats.
IPv4 Address	Dotted decimal notation of an IPv4 address. Represented as 4 decimal numbers separated by periods. For example 192.168.123.123
IPv6 Address	RFC 5952 recommended IPv6 address text representation.
JSON Object	Object enclosed in {} brackets, which maps string-value pairs. For example, {"name":"ExampleName","credentials": {"username":"exampleuser", "password":"examplepassword", "ID": 1234567890}} is a valid JSON object.
Language Code	Two letter language code. Available codes are: "en", "ja".
Password	A String of at least 1 character in length.
String	Sequence of characters. Unless otherwise noted, String Max is 300. All ASCII and Unicode characters with the exception of control characters (ASCII 0x00-0x1F and 0x7F), less-than '<' and greater-than '>' signs are allowed.
Time	4 integer representation of a time of day. Uses 2 integers for hours followed by a ':' followed by two integers for minutes. Hours range from 00 to 24 and minutes from 00 to 59. 24:00 is equal to 00:00.
UNIX Timestamp	Number of seconds elapsed since January 1, 1970 00:00:00 in UTC.
URL	String from 1 to 255 characters in length. Can contain an IP Address.
Username	A special type of String. Must be 1 to 32 characters in length. The first character must be 'a'-'z' or 'A'-'Z'. Subsequent characters must be 'a'-'z', 'A'- 'Z', '0'-'9', '-' or '_' <<<

2.2 Authentication: /api/auth

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**set, add, delete on these objects require admin privilege**
user, see Remote Authentication on page 36.	Array	1 to 8 objects	"guest"	Commands: add, delete, login, logout
id	String	maxLength 32		read-only
enabled	Boolean	true, false	true	
privilege	String	"administrator", "control", "view"	"view"	
active	Boolean	true, false	false	read-only
password	Password	maxLength 30		Can also be set by user, get returns null
passwordSet	Boolean	true, false		read-only
scope	String	0 to String Max	null	
locked	Boolean	true, false	false	read-only
source	String	"local", "remoteAuth", "factory", "unknown"	local	read-only
passwordEx piry	Integer	Integer Max	0	read-only
user/publicKey, see SSH Public Key Authentication on page 42.	Array	0 to 3 objects		User can also add, delete
id	String	maxLength 32		read-only
label	String	maxLength 120	"key"	Can also be set by user
key	String	maxLength 800		Can also be set by user
conf/password Rules, see Password Rules on page 45.				
allowRepeatCharacters	Boolean	true, false	true	
allowUsernameInclusion	Boolean	true, false	true	
minDigits	Integer	0 to 30	0	
minLength	Integer	8 to 30	8	
minSymbols	Integer	0 to 8	0	
minUppercase	Integer	0 to 30	0	
conf/scope, see Scope on page 47.	Array	0 to 10 objects		Commands: add, delete
id	String	0 to String Max		read-only
name	String	0 to String Max		read-only
filter	JSON Object		{}	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**set, add, delete on these objects require admin privilege**
label	String	0 to String Max		
remoteAttribute	String	0 to String Max		
conf/lockoutPo licy, see Lockout Policy Rules on page 51.				Commands: get, set
enabled	Boolean	true, false	true	
maxTries	Integer	3 to 50	10	
neverLockA dmin	Boolean	true, false	false	
timePeriod	Integer	3 to 120	60	minutes
conf/expiryPol icy, see Expiry Policy on page 53.				
adminNever Expires	Boolean	true, false	false	
expiryDays	Integer	1 to 999	90	days
expiryEnabl ed	Boolean	true, false	false	

The API: GET /api/auth/user on the next page object lists all local system users, remote users with active sessions, and provides a means to authenticate local or remote users.

2.2.1 Security and Special Rules

For security related reasons, special rules apply to API: GET /api/auth/user on the next page.

An authenticated user can set their own "publicKey" and "password", but "privilege" and "scope" can only be set by admin privileged users. Restrictions on password security may be configured in Password Rules on page 45. A password cannot be set to null; if a set would set a password to null, that portion of the command will be ignored.

Whereas most parts of the system allow unrestricted get access to all users, only admin privileged users can see API: GET /api/auth/user on the next page in its entirety. Non-admin users making a get request will only see their own configuration plus that of the "guest" user.

Error precedence differs from what is described in the Error Codes on page 22 section. A get, set, delete, or logout operation on a nonexistent /api/auth/user/ID path by an unauthorized (non-admin) user will return a "1003 - Not Authorized: Not enough permissions" error, as shown in Table 2.2 on page 22 as if the requested path exists and the user simply does not have enough permissions. A login operation on a nonexistent /api/auth/user/ID path will return a "1001 - Not Authorized" error, as shown in Table 2.2 on page 22 identical to a login operation on a valid /api/auth/user/ID path with an incorrect "password".

2.2.2 Privileges

Each user has three privilege levels: view, control, and administrator. The user needs to be enabled in order to login. With view access, the user has the basic ability to view all parts of the system (with the exception of other users, as explained in the Security and Special Rules section above). Control gives the user access to basic maintenance functionality, such as the non-get commands specified in Device: /api/dev on page 240. This is in addition to view level privileges. Admin gives view and control privileges, along with access to system level functionality, such as the non-get commands in Configuration: /api/conf on page 55, Device: /api/dev on page 240, System: /api/sys on page 213, and API: GET /api/auth/user on the next page.

The system requires at least one local, non-guest user with admin privilege to be created before it can be used. When the system is in such a state (that is, after a factory reset), the allowed commands to the system are

- **get** on System: /api/sys on page 213.
- **add** an admin user on API: GET /api/auth/user below.
- **set** on /api, where an admin user is added as part of the set operation.

All commands other than these will return a "1000 - No admin user configured" error, as shown in **Table 2.2** on page 22.

Attempts to delete or remove admin privileges from the last non-guest admin will return a "1008-Must have at least one admin user" error, as shown in **Table 2.2** on page 22.

2.2.3 Guest User

The user "guest" is a special user that defines the behavior for users that aren't logged into the system. It is persistent: it can neither be added nor deleted, and it is the only user that exists after a reset to factory defaults. As it only defines logged-out user settings, it can never be logged into or logged out of, and thus it never has a password set. The persistent session token "" (blank string) is the only token that can be used for guest. The "source" field is always "local". If guest is disabled, unauthenticated get commands on all API branches except System: /api/sys on page 213 will return a "1003 - Not Authorized: Not enough permissions" error, as shown in **Table 2.2** on page 22. Before any admin users exist, the guest has the one-time ability to add an admin user.

2.2.4 Remote Authentication

Users can also be authenticated via LDAP, TACACS+, or RADIUS. Only authenticated remote users with currently active sessions will be shown in the /api/auth/user/ID list; remote users will be removed from /api/auth/ when they are logged out. Privileges of remote users are controlled via groups and/or attributes as discussed in the ▼ API:GET /api/conf/remoteAuth/ldap on page 137, ▼ API (cmd): api/conf/remoteAuth/tacacs on page 144, and ▼ API: GET /api/conf/remoteAuth/radius on page 141 configuration sections.

2.2.5 User

API: *GET /api/auth/user*

```
GET http://<device-ip>/api/auth/user HTTP/1.1

[
    {
        "id": "admin",          ①
        "enabled": true,        ②
        "privilege": "view",    ③
        "active": false,        ④
        "password": null,       ⑤
        "passwordSet": false,   ⑥
        "source": "local",      ⑦
        "publicKey": [],        ⑧
        "scope": "1"            ⑨
        "locked": false         ⑩
        "passwordExpiry": 0     ⑪
    }
]
```

① The username provided when adding a user is used as the object ID.

② If false, attempts to log in as this user will return a "1001 - Not Authorized" error, as shown in **Table 2.2** on page 22. If true, the user has basic "get" permissions.

③ Will return "view", "control," or "administrator." Users with "view" privilege have no maintenance or administration permissions. Users with "control" privilege have basic maintenance privileges. Users with "administrator" privilege have system administration permissions.

④ Indicates whether the user is logged in.

⑤ "set" operations use "Password on page 33" strings, "get" operations return only null.

⑥ Indicates whether the password field has been set.

⑦ Will return "local" for local users, "remoteAuth" for remotely authenticated users, "factory" for factory users, or "unknown" for none of the above.

⑧ api/auth/user/publicKey. See SSH Public Key Authentication on page 42.

⑨ If null, user has full API access, up to their permissions level. If equal to a key in Scope on page 47, restricts user access according to that scope.

⑩ Indicates whether the user is locked.

⑪ The number of days remaining before password expires.

▼ **API (cmd):** *api/auth/user: get*

api/auth/user: get

```
[
    {
        "id": "admin",          ①
        "enabled": true,        ②
        "privilege": "view",    ③
        "active": false,        ④
        "password": null,       ⑤
        "passwordSet": false,   ⑥
        "source": "local",      ⑦
        "publicKey": [],        ⑧
        "scope": "1"            ⑨
        "locked": false         ⑩
        "passwordExpiry": 0     ⑪
    }
]
```

① The username provided when adding a user is used as the object ID.

② If false, attempts to log in as this user will return a "1001 - Not Authorized" error, as shown in **Table 2.2** on page 22. If true, the user has basic "get" permissions.

③ Will return "view", "control," or "administrator." Users with "view" privilege have no maintenance or administration permissions. Users with "control" privilege have basic maintenance privileges. Users with "administrator" privilege have system administration permissions.

④ Indicates whether the user is logged in.

⑤ "set" operations use "Password on page 33" strings, "get" operations return only null.

⑥ Indicates whether the password field has been set.

⑦ Will return "local" for local users, "remoteAuth" for remotely authenticated users, "factory" for factory users, or "unknown" for none of the above.

⑧ api/auth/user/publicKey. See SSH Public Key Authentication on page 42.

⑨ If null, user has full API access, up to their permissions level. If equal to a key in Scope on page 47, restricts user access according to that scope.

⑩ Indicates whether the user is locked.

⑪ The number of days remaining before password expires.

▼ **CLI:** get auth user

user> get auth user

```
-     id: admin                 ①
    enabled: true               ②
    privilege: administrator    ③
    active: false               ④
    password: ~                 ⑤
    passwordSet: false          ⑥
    source: local               ⑦
    publicKey:                  ⑧
    ...
    scope: 1                    ⑨
    locked: false               ⑩
    passwordExpiry: 0           ⑪
```

① The username provided when adding a user is used as the object ID.

② If false, attempts to log in as this user will return a "1001 - Not Authorized" error, as shown in **Table 2.2** on page 22. If true, the user has basic "get" permissions.

③ Will return "view", "control," or "administrator." Users with "view" privileges have no maintenance or administration permissions. Users with "control" privileges have basic maintenance permissions. Users with "administrator" privileges have system administration permissions.

④ Indicates whether the user is logged in.

⑤ "set" operations use "Password on page 33" strings, "get" operations return only null.

⑥ Indicates whether the password field has been set.

⑦ Will return "local" for local users, "remoteAuth" for remotely authenticated users, "factory" for factory users, or "unknown" for none of the above.

⑧ api/auth/user/publicKey. See SSH Public Key Authentication on page 42.

⑨ If null, user has full API access, up to their permissions level. If equal to a key in Scope on page 47, restricts user access according to that scope.

⑩ Indicates whether the user is locked.

⑪ The number of days remaining before password expires.

Command: add

API: *PUT /api/auth/user*

```
PUT http://<device-ip>/api/auth/user HTTP/1.1
Content-Type: application/json

{
    "id": "admin",                ①
    "password": "abcd1234",       ②
    "enabled": true,              ③
    "privilege": "administrator", ④
    "scope": null                 ⑤
}
```

① Required field. Must comply with "Username on page 33" rules.

② Required field. Must comply with "Password on page 33" rules.

③ If false, attempts to log in as this user will return a "1001 - Not Authorized" error, as shown in Table 2.2 on page 22. If true, the user has basic "get" permissions.

④ Optional field. If not specified, defaults to "view." Users with "view" privileges have no maintenance or administration permissions. Users with "control" privileges have basic maintenance permissions. Users with "administrator" privileges have system administration permissions.

⑤ Optional field. If not specified, defaults to null.

▼ API: *api/auth/user : add*

api/auth/user:

```
{
    "cmd": "add",
    "data":{
    "id": "admin",                ①
    "password": "abcd1234",       ②
    "enabled": true,              ③
    "privilege": "administrator", ④
    "scope": null                 ⑤
    }
}
```

① Required field. Must comply with "Username on page 33" rules.

② Required field. Must comply with "Password on page 33" rules.

③ If false, attempts to log in as this user will return a "1001 - Not Authorized" error, as shown in Table 2.2 on page 22. If true, the user has basic "get" permissions.

④ Optional field. If not specified, defaults to "view." Users with "view" privileges have no maintenance or administration permissions. Users with "control" privileges have basic maintenance permissions. Users with "administrator" privileges have system administration permissions.

⑤ Optional field. If not specified, defaults to null.

▼ **CLI:** add auth user

admin> add auth user = ARGS

```
add auth user = {"id": "adminCL2", "password": "abcd1234", "enabled": true,
"privilege": "administrator", "scope": null} ① ② ③ ④ ⑤ ⑥
```

① Required fields: id, password.

② ID must comply with "Username on page 33" rules.

③ Password must comply with "Password on page 33" rules.

④ If enabled is false, attempts to log in as this user will return a "1001 - Not Authorized" error, as shown in **Table 2.2** on page 22. If true, the user has basic "get" permissions.

⑤ Privilege is an optional field. If not specified, defaults to "view." Users with "view" privileges have no maintenance or administration permissions. Users with "control" privileges have basic maintenance permissions. Users with "administrator" privileges have system administration permissions.

⑥ Scope is optional, and will default to null if not specified.

Command: delete

▼ **API:** *DELETE /api/auth/user/<id>*

```
DELETE http://<device-ip>/api/auth/user/<id> HTTP/1.1
```

▼ **API (cmd):** *api/auth/user/<id>: delete*

api/auth/user/<id>:

```
{
    "cmd": "delete"
}   ①
```

① Command to delete user entry.

▼ **CLI:** *delete auth user <id>*

admin> delete auth user <id>

```
delete auth user localuser ①
```

① Command to delete user entry.

Command: login

Log in as the user specified in the path. Each login command will return a unique token. The system will keep track of up to 8 unique tokens per user. Each token will expire after 6 hours of inactivity or until the user logs out.

Repeated authentication failures on a given user will trigger a lockout for that user. After 3 failed authentication attempts, a 30 minute lockout period will be placed on the user. During this period, any attempts to log in will result in the same authentication error returned when the supplied password is incorrect. A successful login attempt will clear the count of authentication failures for the user. Attempting to log in as a locked-out user will restart the lockout period.

▼ API (cmd): *api/auth/user/ldapadminuser : login*

api/auth/user/ldapadminuser: login

```
{
    "username": "ldapadminuser",
    "password": "abc123"
}
```

login return data

```
{
    "token": "f3e2d1c0",           ①
    "privilege": "administrator",
    "source": "remoteAuth",        ②
    "scope": null
}
```

① A randomly generated string to keep track of open sessions. The token expires after 6 hours of inactivity or when the user explicitly logs out.

② Specifies the origin of the user.

Command: logout

Log out of user specified in the path. Logging out will invalidate all tokens for that user.

▼ API (cmd): *api/auth/user/admin : logout*

api/auth/user/admin: logout (Same user, in this case "admin")

```
{
    "token": "f3e2d1c0"
}
```

logout return data

```
{}
```

2.2.6 SSH Public Key Authentication

Except for guest users, any user can add a public key to the device in order to bypass password authentication when connecting via the SSH Protocol. Each user may add at most 3 public keys. Users with admin privilege can execute add, set, delete, get operations on all existing users. Any view user can execute add, set, delete, get operations only on their own public key set. A validation error will occur if the public key exceeds 4096-bit length, if the "key" field exceeds 800 characters, or if the public key fails to comply with the format "<Key algorithm> <key blob> <Optional comment>" defined in section 6.6 of RFC 4253.

Additionally, provided keys must meet the following standards: * RSA keys must meet or exceed 3072-bit length. * EC keys must be NIST P-384.

Command: get

▼ API:*GET /api/auth/user/admin/publicKey*

```
GET http://<device-ip>/api/auth/user/admin/publicKey HTTP/1.1

[
    {
        "id": "0"       ①
        "key": ""       ②
        "label": "key", ③
    }
]
```

① The ID of the public key.

② The string representing the user's SSH public key.

③ The label to identify the user's SSH public key.

▼ API (cmd): *api/auth/user/admin/publicKey : get (admin)*

api/auth/user/admin/publicKey: get *(admin)*

```
[
  {
    "id": "0"       ①
    "key": ""       ②
    "label": "key", ③
  }
]
```

① The ID of the public key.

② The string representing the user's SSH public key.

③ The label to identify the user's SSH public key.

▼ **CLI:** get auth user admin publicKey

user> get auth user admin publicKey

```
-  id: 0      ①
   key:       ②
   label: key ③
```

① The ID of the public key.

② The string representing the user's SSH public key.

③ The label to identify the user's SSH public key.

Command: add

▼ **API:** *PUT /api/auth/user/admin/publicKey*

```
PUT http://<device-ip>/api/auth/user/admin/publicKey HTTP/1.1
Content-Type: application/json

{
    "id": "0",     ①
    "key": "",     ①
    "label": "key"
}
```

① Required field.

▼ **API (cmd):** *api/auth/user/admin/publicKey : add*

api/auth/user/admin/publicKey:

```
{
    "cmd": "add",
    "data": {
      "id": "0",     ①
      "key": "",     ①
      "label": "key"
  }
}
```

① Required field.

▼ **CLI:** add auth user admin publicKey

user> add auth user admin publicKey = ARGS

```
add auth user admin publicKey = {id: id, key: KEY, label: key}①
```

① Required fields: key.

Command: delete

▼ API: *DELETE /api/auth/user/admin/publicKey/<id>*

```
DELETE http://<device-ip>/api/auth/user/admin/publicKey/<id> HTTP/1.1
```

▼ API: *api/auth/user/admin/publicKey/ID : delete*

api/auth/user/admin/publicKey/ID:

```
{
  "cmd": "delete"
} ①
```

① Command to delete public key entry.

▼ CLI: *delete auth user admin publicKey <id>*

admin> delete auth user admin publicKey <id>

```
delete auth user admin publicKey id ①
```

① Command to delete public key entry.

2.3 Authentication Configuration: /api/auth/conf

The Vertiv™ Liebert® IntelliSlot™ RDU120 Authentication Configuration API exists to allow password rules to be configured for the system.

2.3.1 Password Rules

Strong password restrictions configuration. Includes settings to place restriction on user passwords.

▼ API: *GET /api/auth/conf/passwordRules*

```
GET http://<device-ip>/api/auth/conf/passwordRules HTTP/1.1

{
    "allowRepeatCharacters": true,   ①
    "allowUsernameInclusion": false, ②
    "minDigits": 3,                  ③
    "minLength": 8,                  ④
    "minSymbols": 1,                 ⑤
    "minUppercase": 1                ⑥
}
```

① If false, passwords may not include more than 2 of the same letter in a contiguous group.

② If false, passwords may not include associated username.

③ Sets minimum number of digits required.

④ Sets minimum length required. Minimum 6.

⑤ Sets minimum symbols required. Symbols are defined here as printable ASCII characters not including a-z, A-Z, and 0-9.

⑥ Sets minimum uppercase letters required.

▼ API (cmd): *api/auth/conf/passwordRules: get*

api/auth/conf/passwordRules: get

```
{
        "allowRepeatCharacters": true,   ①
        "allowUsernameInclusion": false, ②
        "minDigits": 3,                  ③
        "minLength": 8,                  ④
        "minSymbols": 1,                 ⑤
        "minUppercase": 1                ⑥
}
```

① If false, passwords may not include more than 2 of the same letter in a contiguous group.

② If false, passwords may not include associated username.

③ Sets minimum number of digits required.

④ Sets minimum length required. Minimum 6.

⑤ Sets minimum symbols required. Symbols are defined here as printable ASCII characters not including a-z, A-Z, and 0-9.

⑥ Sets minimum uppercase letters required.

▼ **CLI:** get auth conf passwordRules

user> get auth conf passwordRules

```
allowRepeatCharacters: true   ①
allowUsernameInclusion: false ②
minDigits: 3                  ③
minLength: 8                  ④
minSymbols: 1                 ⑤
minUppercase: 1               ⑥
```

① If false, passwords may not include more than 2 of the same letter in a contiguous group.

② If false, passwords may not include associated username.

③ Sets minimum number of digits required.

④ Sets minimum length required. Minimum 6.

⑤ Sets minimum symbols required. Symbols are defined here as printable ASCII characters not including a-z, A-Z, and 0-9.

⑥ Sets minimum uppercase letters required.

The password rules fields are settable.

Example using "minLength" field:

▼ **API:** *POST /api/auth/conf/passwordRules*

```
POST http://<device-ip>/api/auth/conf/passwordRules HTTP/1.1
Content-Type: application/json

{
    "minLength" : 8
}    ①
```

① Command to set the minimum password length.

▼ **API (cmd):** *api/auth/conf/passwordRules : set*

api/auth/conf/passwordRules:

```
{
    "cmd": "set",
    "data": {
        "minLength" : 8
    }
}    ①
```

① Command to set the minimum password length.

▼ **CLI:** set auth conf passwordRules

admin> set auth conf passwordRules = ARGS

```
set auth conf passwordRules minLength = 8 ①
```

① Command to set the minimum password length.

2.3.2 Scope

Access level information associated with the device. Allows the admin to create scopes to limit user access.

The filter specifies the parts of the API to which the user has access. An empty object or a value of null will match the entire tree below the current key. The root of the filter is expected to be /api. Currently scope filters only support outlet filtering. The example filter allows access to outlets 1-3 of the device with ID ABC1234567890DEF and to all outlets on the device with ID G0987654321HIJKL.

Scopes can be applied to Remote Authentication on page 135. "remoteAttribute" must contain the user groups the scope applies to. LDAP and TACACS users should use groups for this purpose. Radius users should use Management-Policy-Id. The list of user groups returned from the authentication server is searched for values contained within "remoteAttribute". The first scope whose "remoteAttribute" matches the user groups is the one that is used.

A user's scope defines what that user can access and modify within the user's permission level. A scoped user will only have access to API: GET /api/auth/user on page 36, System: /api/sys on page 213, and the outlets specified. The guest user may not be scoped. Users with admin permissions may not be scoped.

IMPORTANT! Scope currently only works for API users.

▼ **API:** *GET /api/auth/conf/scope*

```
GET http://<device-ip>/api/auth/conf/scope HTTP/1.1

[
    {
        "id": "0",
        "filter": {
            "dev":{
                "ABC1234567890DEF": {
                    "outlet": {
                        "1":null,
                        "2":null,
                        "3":null
                    },
                },
                "G0987654321HIJKL":null
            }
        },
        "name": "Scope 0",
        "label": "Scope 0",
        "remoteAttribute": ""
    }
]
```

▼ **API (cmd):** *api/auth/conf/scope: get*

api/auth/conf/scope: get

```
[
    {
        "id": "0",
        "filter": {
            "dev": {
                "ABC1234567890DEF": {
                    "outlet":{
                        "1":null,
                        "2":null,
                        "3":null
                    },
                },
                "G0987654321HIJKL":null
            }
        },
        "name": "Scope 0",
        "label": "Scope 0",
        "remoteAttribute": ""
    }
]
```

▼ **CLI:** get auth conf scope

user> get auth conf scope

```
-   id: 0
    filter:
        dev:
          ABC1234567890DEF:
            outlet:
              1: ~
              2: ~
              3: ~
        G0987654321HIJKL: ~
    name: Scope 0
    label: Scope 0
    remoteAttribute: ""
```

Command: add

Creates a new scope.

▼ API: *PUT /api/auth/conf/scope*

```
PUT http://<device-ip>/api/auth/conf/scope HTTP/1.1
Content-Type: application/json

{
    "filter": {                              ①
        "dev": {
            "ABC1234567890DEF": {
                "outlet": {
                    "1":null,
                    "2":null,
                    "3":null
                }
            },
            "G0987654321HIJKL":null
        }
    },
    "label": "Scope 0",                      ②
    "remoteAttribute": ""                    ③
}
```

① Filter to be applied. Must be a valid JSON object. An empty object or null value will match all descendants.

② Label for the scope. Defaults to the same value as "name", but is settable.

③ Specifies the policy to which the scope applies when used in conjunction with Remote Authentication on page 135.

▼ API: *api/auth/conf/scope : add*

api/auth/conf/scope:

```
{
    "cmd": "add",
    "data": {
      "filter": {                            ①
        "dev": {
            "ABC1234567890DEF": {
                "outlet":{
                 "1":null,
                 "2":null,
                 "3":null
              }
            },
            "G0987654321HIJKL":null
        }
    },
    "label": "Scope 0",                      ②
    "remoteAttribute": ""                    ③
  }
}
```

① "filter" is the filter to be applied to the scope. It must be a valid JSON object. An empty object or null value will match all descendants.

② Label for the scope. Defaults to the same value as "name", but is settable.

③ Specifies the policy to which the scope applies when used in conjunction with Remote Authentication on page 135.

IMPORTANT! CLI add is currently not working for scope.

▼ **CLI:** add auth conf scope (Admin)

admin> add auth conf scope = ARGS

```
add auth conf scope =
{"filter":{"dev":{"ABC1234567890DEF":{"outlet":{"1":null,"2":null,"3":null}},"G09876
54321HIJKL":null}},"label":"Scope 0","remoteAttribute":""}
① ② ③
```

① "filter" is the filter to be applied to the scope. It must be a valid JSON object. An empty object or null value will match all descendants.

② "label" is the label for the scope. Defaults to the same value as "name".

③ "remoteAttribute" specifies the policy to which the scope applies when used in conjunction with Remote Authentication on page 135.

Command: delete

▼ **API:** *api/auth/conf/scope/<id>*

```
DELETE http://<device-ip>/api/auth/conf/scope/<id> HTTP/1.1
```

▼ **API:** *api/auth/conf/scope/<id> : delete*

api/auth/conf/scope/<id>:

```
{
"cmd": "delete"
} ①
```

① Command to delete a scope entry.

▼ **CLI:** delete auth conf scope <id>

admin> delete auth conf scope <id>

```
delete auth conf scope 0 ①
```

① Command to delete a scope entry.

2.3.3 Lockout Policy Rules

The lockout policy allows the setting of the maximum number of attempts, whether the system can lockout admin users, and set the time interval between consecutive authentication failures before account lockout.

▼ API: *GET /api/auth/conf/lockoutPolicy*

```
GET http://<device-ip>/api/auth/conf/lockoutPolicy HTTP/1.1

{
    "enabled": true,         ①
    "maxTries": 5,           ②
    "neverLockAdmin": false, ③
    "timePeriod": 60         ④
}
```

① Indicates if the lockout policy is enabled.

② The maximum number of attempts before locking the user account.

③ Indicates if an admin user account can be locked out by the system.

④ Time interval in minutes in which consecutive authentication failures must happen for the user account to lock out.

▼ API (cmd): *api/auth/conf/lockoutPolicy : get*

api/auth/conf/lockoutPolicy: get

```
{
    "enabled": true,         ①
    "maxTries": 5,           ②
    "neverLockAdmin": false, ③
    "timePeriod": 60         ④
}
```

① Indicates if the lockout policy is enabled.

② The maximum number of attempts before locking the user account.

③ Indicates if an admin user account can be locked out by the system.

④ Time interval in minutes in which consecutive authentication failures must happen for the user account to lock out.

▼ CLI: get auth conf lockoutPolicy

user> get auth conf lockoutPolicy

```
enabled: true         ①
maxTries: 10          ②
neverLockAdmin: false ③
timePeriod: 60        ④
```

① Indicates if the lockout policy is enabled.

② The maximum number of attempts before locking the user account.

③ Indicates if an admin user account can be locked out by the system.

④ Time interval in minutes in which consecutive authentication failures must happen for the user account to lock out.

The lockout policy fields are settable.

Example using the "maxTries" field:

▼ API: *POST /api/auth/conf/lockoutPolicy*

```
POST http://<device-ip>/api/auth/conf/lockoutPolicy HTTP/1.1
Content-Type: application/json

{
    "maxTries": 5
} ①
```

① Command to set lockout policy maxTries.

▼ API (cmd): *api/auth/conf/lockoutPolicy : set*

api/auth/conf/lockoutPolicy:

```
{
    "cmd": "set",
    "data": {
        "maxTries": 5
    }
}    ①
```

① Command to set lockout policy maxTries.

▼ CLI: set auth conf lockoutPolicy

admin> set auth conf lockoutPolicy maxTries = ARGS

```
set auth conf lockoutPolicy maxTries = 5 ①
```

① Command to set lockout policy maxTries.

2.3.4 Expiry Policy

Manages the password expiration settings for all users.

▼ API: *GET /api/auth/conf/expiryPolicy*

```
GET http://<device-ip>/api/auth/conf/expiryPolicy HTTP/1.1

{
    "adminNeverExpires": false, ①
    "expiryDays": 90,           ②
    "expiryEnabled": false      ③
}
```

① Indicates if the password for users with administrator privileges expires.

② The number of days after which user passwords expire.

③ Indicates if expiry policy has been enabled.

▼ API (cmd): *api/auth/conf/expiryPolicy : get*

api/auth/conf/expiryPolicy: get

```
{
    "adminNeverExpires": false, ①
    "expiryDays": 90,           ②
    "expiryEnabled": false      ③
}
```

① Indicates if the password for users with administrator privileges expires.

② The number of days after which user passwords expire.

③ Indicates if expiry policy has been enabled.

▼ CLI: get auth conf expiryPolicy

user> get auth conf expiryPolicy

```
adminNeverExpires: false   ①
expiryEnabled: false       ②
expiryDays: 90             ③
```

① Indicates if the password for users with administrator privileges expires.

② The number of days after which user passwords expire.

③ Indicates if expiry policy has been enabled.

The expiry policy fields are settable.

Example using the "expiryDays" field:

▼ **API:** *POST /api/auth/conf/expiryPolicy*

```
POST http://<device-ip>/api/auth/conf/expiryPolicy HTTP/1.1
Content-Type: application/json

{
    "expiryDays": 90
}     ①
```

① Command to set the number of days after which user passwords expire.

▼ **API (cmd):** *api/auth/conf/expiryPolicy : set*

api/auth/conf/expiryPolicy:

```
{
    "cmd": "set",
    "data": {
    "expiryDays": 90
    }
}    ①
```

① Command to set the number of days after which user passwords expire.

▼ **CLI:** set auth conf expiryPolicy

admin> set auth conf expiryPolicy expiryDays = ARGS

```
set auth conf expiryPolicy expiryDays = 90 ①
```

① Command to set the number of days after which user passwords expire.

2.4 Configuration: /api/conf

The Vertiv™ Liebert® IntelliSlot™ RDU120 Configuration API exists to allow configuration to be read and modified. The configuration API is a protected API whereby the API user must be authenticated before being able to read or modify the configuration. The following table provides a detailed description of all configuration settings. Following the table, API and CLI examples are provided for each configuration API category.

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
bacnet/server, see BACnet Server on page 68.				set
access	String	"readOnly", "readWrite"	"readOnly "	
apduTimeout	Integer	1 to 65535	3000	
apduRetries	Integer	0 to 8	3	
deviceName	String	maxlength 50	Device1130000	
deviceInstance	Integer	0 to 4194302	1130000	
mode	String	"disabled", "ip", "mstp"	"disabled"	
bacnet/server/ip, see BACnet IP on page 71.				set
bbmdAddress	IPv4 Address	maxLength 15	""	
fdrEnabled	Boolean	true or false	false	
fdrTtl	Integer	10 to 43200	1800	
maxCovSubscriptions	Integer	0 to 65535	5000	
port	Integer	1024 to 49151	47808	
bacnet/server/ip/trustedAccess, see BACnet IP Trusted Access on page 73.	Array	0 to 5 trustedAccess entries		add, delete, set
id	String	maxLength 32		read-only
address	IP Address	maxLength 45		
prefix	Integer	0 to 128	0	
bacnet/server/mstp, see BACnet MSTP on page 77.				set
baudRate	String	"9600", "19200", "38400", "57600", "76800", "115200"	"38400"	
maxCovSubscriptions	Integer	0 to 65535	1000	
maxInfoFrames	Integer	1 to 8	8	
maxNodeId	Integer	3,7,15,31,63,127	127	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
nodeId	Integer	0 to 127	1	
portParameters	String	"8N2", "8E1", "8O1", "8N1", "8E2", "8O2"	"8N2"	
cli, see CLI on page 79.				set
sessionIdleTimeout	Integer	1 to 600	5	minutes
cloud/remoteService, see Cloud on page 85.				set
connectionTestResult	String	"SUCCESS", "FAILURE"		read-only
connectionRetryTime	Integer	30 to 600	30	seconds
serviceCenterCountry	String	Service Center Country on page 340	""	
deviceInstanceId	String	maxlength 32	""	
siteId	String	maxlength 32	""	
siteEquipmentTagNumber	String	maxlength 32	""	
serialNumberOverrideEnabled	Boolean	true, false	false	
configuredSerialNumber	String	maxlength 32		
cloudURL	String	maxlength 48	mq- service.vertiv.cloud	
deviceDataSamplingEnabled	Boolean	true, false	false	
enabled	Boolean	true, false	false	
cloud/remoteService/diag, see Diagnostic on page 89.				read-only
cloudMessage	String	0 to 124		
manDeviceRegisterConfirmed	Boolean	true or false	false	
gatewayRegisterConfirmed	Boolean	true or false	false	
gatewayRegistered	Boolean	true or false	false	
manDeviceRegistered	Boolean	true or false	false	
gatewayRegisterConfirmed	Boolean	true or false	false	
dmpCommStatus	String	"Online", "Offlne", "Not Supported"		
manDeviceStatus	String	"Online", "Offline"		
deviceRuleFileInfo	String	maxLength 62		
lastError	String	maxLength 62	""	
lastSuccess	String	maxLength 96		

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
remoteServiceOPStatus	String	"Normal Operation", "Abnormal Operation", "Reevaluating operational state...", "Pending operational state re-evaluation ", "Shutting down", "Starting"		
errorCount	Integer	0 to Integer Max	0	
status	String	"Connected", "Not Connected ", "Not Monitoring"		
cloud/remoteService/proxy, see Proxy on page 92.				set
username	String	maxlength 50	""	
authEnabled	Boolean	true or false	false	
port	Integer	1 to 65535	80	
address	String	maxlength 64	""	
passwordSet	Boolean	true, false	false	read-only
password	String	maxLength 30	""	
enabled	Boolean	true, false	false	
datalogging, see Data Logging on page 81.				set
enabled	Boolean	true, false	true	
intervalSec	Integer	60 to 86400	600	seconds
datalogging/data Point, see Data Point Configuration on page 84.	Array	read-only collection		read-only
id	String	maxLength 32		read-only
gddId	Integer	1 to 65535		read-only
moduleIndex	Integer	1 to 64		read-only
enabled	Boolean	true, false	true	read-only
alias	String	maxLength 100		read-only
description	String	maxLength 200		read-only
email, see Email on page 94.				set
passwordSet	Boolean	true, false	false	read-only
port	Integer	1 to 65535	25	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
sender	String	maxLength 120		
server	IP Address	maxLength 64		
password	String	maxLength 64		
subjectPrefix	String	maxLength 125		
username	String	maxLength 64		
enableEventCon solidation	Boolean	true, false	false	
consolidationEv entLimit	Integer	1 to 50	20	
consolidationTi meLimit	Integer	10 to 120	60	minutes
smtpAuthenticat ion	Boolean	true, false	false	
email/target, see Email Target on page 98.	Array	1 to 3 targets		add, delete, set, sendTest
id	String	0 to String Max		read-only
name	String	maxLength 120		
lldp, see LLDP on page 101.				set
enabled	Boolean	true, false	false	
password	String	maxLength 32		
passwordSet	Boolean	true, false	false	read-only
systemDescription	String	maxLength 32		
txHoldMultiplier	Integer	2 to 10	3	
txInterval	Integer	5 to 32768	60	seconds
username	String	maxLength 32		
locale, see Locale on page 103.				
systemLanguage	Language Code	"en", "ja"	"en"	set
measurementSy stem	String	"us", "metric"	"us"	set
modbus/server, see Modbus Server on page 105.				set
access	String	"readOnly", "readWrite"	"readOnly "	
mode	String	"disabled", "tcp", "rtu"	"disabled"	
modbus/server/rtu, see Modbus RTU on page 107.				set
baudRate	String	"9600", "19200", "38400"	"9600"	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
nodeId	Integer	1 to 247	1	
portParameters	String	"8N2", "8E1", "8O1", "8N1", "8E2", "8O2"	"8N2"	
modbus/server/tcp, see Modbus TCP on page 109.				set
maxClients	Integer	0 to 5	4	
port	Integer	1 to 32767	502	
modbus/server/tcp/trustedAccess, see Modbus TCP Trusted Access on page 111.	Array	0 to 5 trustedAccess entries		add, delete, set
id	String	maxLength 32		read-only
address	IP Address	maxLength 45		
prefix	Integer	0 to 128	24	
network, see Network on page 114.				set
hostname	String	maxLength 63	"rdu120-" + macAddr	
dnsSearchList	String	maxLength 100		
defaultIpEnabled	Boolean	true, false	true	
ip4Enabled	Boolean	true, false	true	
ip6Enabled	Boolean	true, false	true	
ipV4GW	IPv4 Address	maxLength 15		
ipV6GW	IPv6 Address	maxLength 45		
ipV4BootMode	String	"STATIC", "DHCP", "BOOTP"	"DHCP"	
ipV6BootMode	String	"STATIC", "DHCP"	"DHCP"	
ipv4DnsSource	String	"none", "automatic", "configured"	"automatic"	read-only
ipv6DnsSource	String	"none", "automatic", "configured"	"automatic "	read-only
dhcpNtpOpt42	Boolean	true, false	true	
network/dns, see Network/DNS on page 118.	Array	add up to 2 IPv4 and 2 IPv6 DNS addresses		add, delete
id	String	0 to String Max		read-only
address	IP Address	maxLength 45		
mutable	Boolean	true or false	true	read-only

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
network/route, see Network/Route on page 121.	Array	Up to 8 routes		add, delete
id	String	0 to String Max		read-only
destination	IP Address	maxLength 45	""	
gateway	IP Address	maxLength 45	""	
interface	String	0 to String Max		
mutable	Boolean	true or false		read-only
prefix	Integer	0 to 128		
network/interface/lan, see Network/Interface/LAN on page 124.				set
enabled	Boolean	true, false	false	
removable	Boolean	true, false	false	read-only
name	String	0 to String Max		read-only
label	String	maxLength 300	"eth0"	
macAddr	String	maxLength 18		read-only
network/interface/lan/address, see Network/Interface/LAN/Address on page 127.	Array	Up to 1 IPv4 and 1 IPv6 static address		add, delete
id	String	0 to String Max		read-only
address	IP Address	maxLength 45		
prefix	Integer	0 to 128		
mutable	Boolean	true or false		read-only
network/interface/lan/link, see Network/Interface/LAN/Link on page 130.				set
state	String	"up", "down", "missing"		read-only
uptime	Integer	0 to Integer Max		read-only
speed	String	"10Mbps", "100Mbps", "1Gbps"		read-only
duplex	String	"half", "full"		read-only
supportedModes	String	"10baseT/Half"," 10baseT/Full", "100baseT/Half", "100baseT/Full", "1000baseT/Full"		read-only

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
configSpeedDuplex	String	"Auto", "10Mbps/half", "10Mbps/full", "100Mbps/half", "100Mbps/full", "1Gbps/full"		
network/interface/lan/link/stat, see Network/Interface/LAN/Link on page 130.				
rxCount	Integer	0 to Integer Max		read-only
rxDropped	Integer	0 to Integer Max		read-only
txCount	Integer	0 to Integer Max		read-only
txDropped	Integer	0 to Integer Max		read-only
network/interface/lan/dot1x, see Network/Interface/LAN/Dot1x on page 133.				set
enabled	Boolean	true, false	false	
identity	String	maxLength 32		Cannot be empty if enabled is true
privKeyPassword	String	0 to String Max		
remoteAuth, see Remote Authentication on page 135.				set
mode	String	"none", "ldap", "tacacs", "radius"	"none"	
remoteAuth/ldap, see LDAP on page 137.				set
adminGroup	String	maxLength 120	"admin"	
baseDn	String	maxLength 120	""	
bindDn	String	maxLength 120	""	
controlGroup	String	maxLength 120	"control"	
enabledGroup	String	maxLength 120	"enabled"	
groupFilter	String	maxLength 125	"(objectClass=posixGroup)"	
groupIdNum	String	maxLength 120	"gidNumber"	
groupMemberUid	String	maxLength 120	"memberUid"	
host	String	maxLength 120	""	
mode	String	"openLdap", "activeDirectory"	"openLdap "	
password	String	maxLength 120	""	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
passwordSet	Boolean	true or false	false	read-only
port	Integer	1 to 65535	389	
securityType	String	"ssl", "startTls"	"ssl"	
userFilter	String	maxLength 120	"(objectClass=posixAccount)"	
userId	String	maxLength 120	"uid"	
userIdNum	String	maxLength 120	"uidNumber"	
remoteAuth/radius, see Radius on page 141.				set
adminGroup	String	maxLength 120	"admin"	
authenticationServer1	String	maxLength 120	""	
authenticationServer2	String	maxLength 120	""	
controlGroup	String	maxLength 120	"control"	
enabledGroup	String	maxLength 120	"enabled"	
groupAttribute	String	"filter-id", "management-privilege-level"	"filter-id"	
sharedSecret	String	maxLength 120	""	
sharedSecretSet	Boolean	true or false	false	read-only
remoteAuth/tacacs, see TACACS on page 144.				set
accountingServer1	String	maxLength 120	""	
accountingServer2	String	maxLength 120	""	
adminAttribute	String	maxlength 120	"admin=1"	
authenticationServer1	String	maxLength 120	""	
authenticationServer2	String	maxLength 120	""	
controlAttribute	String	maxLength 120	"control=1 "	
enabledAttribute	String	maxLength 120	"enabled= 1"	
service	String	"ppp", "raccess"	"ppp"	
sharedSecret	String	maxLength 120	""	
sharedSecretSet	Boolean	true or false	false	read-only
remoteSyslog, see Remote Syslog on page 146.				

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
remoteSyslog/target, see Remote Syslog Target on page 147.	Array	0 to 2 syslog targets		add, delete, set
id	String			read-only
enabled	Boolean	true or false	false	
server	String	maxLength 120	""	
port	Integer	1 to 65535	514	
transport	String	"udp", "tls"	"udp"	
serial, see Serial on page 151.				set
baudRate	String	"9600", "19200", "38400", "57600", "76800", "115200"	"115200"	
enabled	Boolean	true or false	true	
portParameters	String	"8N2", "8E1", "8O1", "8N1", "8E2", "8O2"	"8N1"	read-only
snmp, see SNMP on page 153.				set
heartBeatTrapInterval	Integer	0, 5, 30, 60, 240, 480, 720, 1440	1440	0 means disabled, minutes
Port	Integer	1 to 65535	161	
authTrapsEnabled	Boolean	true or false	false	
snmp/mibSupport, see SNMP MIB Support on page 155.				set
rfc1628MIBEnabled	Boolean	true or false	true	
rfc1628MIBTrapEnabled	Boolean	true or false	true	
lgpMIBEnabled	Boolean	true or false	true	
lgpMIBTrapEnabled	Boolean	true or false	true	
lgpMIBSystemNotifyTrapEnabled	Boolean	true or false	true	
extendedVBEnabled	Boolean	true or false	false	
snmp/v1v2c, see SNMP v1/v2c on page 157.				set
enabled	Boolean	true or false	false	
snmp/v1v2c/access, see SNMP v1/v2c Access on page 159.	Array	0 to 8 entries		add, delete, set
id	String	maxLength 32		read-only
access	String	"read", "write"	"read"	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
community	String	maxLength 32	""	
snmp/v1v2c/access/trustedAccess, see SNMP v1/v2c Trusted Access on page 163.	Array	0 to 1 entries		add, delete, set
id	String	maxLength 32		read-only
address	IP Address	maxLength 45		
prefix	Integer	0 to 128	24	
snmp/v1v2c/trap, see SNMP v1/v2c Trap on page 166.	Array	0 to 8 entries		add, delete, set, sendTest
id	String	maxLength 32		read-only
community	String	maxLength 32	""	
port	Integer	1 to 65535	162	
target	String	maxLength 63	""	
snmp/v1v2c/generateTestTraps, see SNMP v1/v2c Test Traps on page 170				sendTest
snmp/v3, see SNMP v3 on page 170.				set
enabled	Boolean	true or false	false	
engineId	String	maxLength 64	""	read-only
engineIdText	String	maxLength 27	""	An empty engine ID will cause the engineID to be automatically generated using the MAC Address
engineIdType	String	"Text", "MAC Address"	"MAC Address"	
snmp/v3/user, see SNMP v3 User on page 173.	Array	0 to 8 users		add, delete, set
id	String	maxLength 32		read-only
access	String	"read", "write", "trap"	"read"	
authType	String	"none", "md5", "sha1"	"none"	
authPassword	String	8 to 64	""	
authPasswordSet	Boolean	true or false	false	read-only
enabled	Boolean	true or false	false	
privPassword	String	8 to 64	""	
privPasswordset	Boolean	true or false	false	read-only
privType	String	"none", "des", "aes"	"none"	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
username	String	maxLength 64	""	
snmp/v3/user/target, see SNMP v3 User Trap Target on page 178.	Array	0 to 2 entries		add, delete, set, sendTest
id	String	maxLength 32		read-only
port	Integer	1 to 65535	162	
target	String	maxLength 63	""	
snmp/v3/generate TestTraps, see SNMP v3 Test Traps on page 181.				sendTest
ssh, see SSH on page 182.	target	String	"key"	set, reset - generate a new host ssh key
enabled	Boolean	true or false	false	
port	Integer	1 to 65535	22	
system, see System on page 184.				set
contact	String	maxLength 120	""	
description	String	maxLength 120	""	
label	String	maxLength 120	"rdu120"	Populates /api/sys/name
location	String	maxLength 120	""	
factoryAccessEnabled	Boolean	true or false	false	Enables factory access functions.
systemResource, see System Resource on page 186.				
timeRangeCollection	String	"5m", "30m", "1h", "2h", "4h", "8h", "12h", "24h"	"5m"	read-only
time, see Time on page 187.				set
source	String	"ntp", "modbus", "bacnet", "lsc", "api", "internal", "not set"	"ntp"	read-only
mode	String	"automatic", "manual"	"automatic "	
datetime	String	YYYY-MM-DD HH:MI:SS	""	
zone	String	Time Zone on page 325	"UTC"	
ntpServer1	String	maxLength 64	"0.pool.ntp.org"	
ntpServer2	String	maxLength 64	"1.pool.ntp.org"	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
syncManagedDeviceEnabled	Boolean	true or false	false	
usb, see USB on page 189.				set
enabled	Boolean	true or false	false	Enables/disables the front-panel USB
usb/dev, see USB Devices on page 191.	Array			
id	String	maxLength 64		read-only
vendorId	String	maxLength 64		read-only
productId	String	maxLength 64		read-only
bcdDevice	String	maxLength 64		read-only
manufacturer	String	maxLength 64		read-only
product	String	maxLength 64		read-only
serial	String	maxLength 64		read-only
usb/dev/conf, see USB Device Configuration on page 193.				
maxPower	String	maxLength 20		read-only
selfPowered	Boolean	true or false		read-only
remoteWakeup	Boolean	true or false		read-only
velocity/managedDevice, see Velocity Managed Device on page 195.				set
lanType	String	"velocity-serial", "esp2-internal", "igm", "ip", "autodetect-internal", "mb-internal"	"autodetect-internal"	
connectStatus	String	"connected", "not-connected", "unsupported"	"unsupported"	read-only
psi	String	maxLength 30	"0.0"	read-only
velocity/ethernet, see Velocity Managed Device Ethernet on page 197.				set
enabled	Boolean	false		
ip Address	IP Address	maxLength 45		
port	Integer	1 to 65535	47808	
networkNumber	Integer	1 to 65534	1000	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
velocity/serial, see Velocity Managed Device Serial on page 199.				set
baudRate	String	"38400", "19200", "9600"	"38400"	
maxAddress	Integer	3, 7, 15, 31, 63, 127	127	
networkNumber	Integer	1 to 65534	1001	
nodeId	Integer	0 to 127	127	
nodeAssignmentMethod	String	"automatic", "manual"	"automatic "	
webserver, see Web Server on page 201.				set
redirectEnabled	Boolean	true or false	false	
httpsEnabled	Boolean	true or false	true	
httpsPort	Integer	1 to 49151	443	
httpEnabled	Boolean	true or false	false	
httpPort	Integer	1 to 49151	80	
sessionIdleTimeout	Integer	1 to 600	5	minutes
webserver/genCertificate, see Generate Self-Signed Certificate on page 204.				set, reset - generate self-signed certificate based upon configuration
organization	String	maxLength 32	""	
organizationUnit	String	maxLength 32	""	
emailAddress	String	maxLength 120	""	
countryCode	String	maxLength 2	""	
stateProvince	String	maxLength 32	""	
cityLocality	String	maxLength 32	""	
commonName	String	maxLength 63	""	read-only
ssMode	String	"Use Default Values", "Use Configured Settings"	"Use Default Values"	
webserver/trustedAccess, see Trusted Access on page 206.	Array			add, set, delete
id	String	maxLength 32		read-only
address	IP Address	maxLength 45		
prefix	Integer	0 to 128	0	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
webserver-reset, see Web Server Self-Signed Certificate Generation on page 210.				reset - generate self-signed certificate
ydn23/server, see YDN23 Server on page 211.				set
enabled	Boolean	true, false	false	
access	String	"readOnly", "readWrite"	"readOnly "	
address	Integer	1 to 254	1	
dataRate	String	"1200", "2400", "4800", "9600", "19200", "38400"	"9600"	

2.4.1 BACnet Server

The BACnet server configuration settings allow the user to enable or disable the BACNET server and configure the access parameters.

▼ API: *GET /api/conf/bacnet/server*

```
GET http://<device-ip>/api/conf/bacnet/server HTTP/1.1

{
    "access": "readOnly",          ①
    "apduTimeout": 3000,           ②
    "apduRetries": 3,              ③
    "deviceName": "Device1130000", ④
    "deviceInstance": 1130000,     ⑤
    "mode": "disabled",            ⑥
    "ip": {},                      ⑦
    "mstp": {}                     ⑧
}
```

① The allowed access for a BACnet client.

② The APDU Timeout; the amount of time a client waits for a response from a BACnet device.

③ The APDU retries; the number of times a client will retry sending an APDU.

④ The device name; the name of the BACnet device.

⑤ The device instance; the instance of the BACnet device.

⑥ The BACnet server mode; disabled, IP, or MSTP.

⑦ BACnet IP on page 71.

⑧ BACnet MSTP on page 77.

▼ **API (cmd):** *api/conf/bacnet/server*

api/conf/bacnet/server: get

```
{
    "access": "readOnly",          ①
    "apduTimeout": 3000,           ②
    "apduRetries": 3,              ③
    "deviceName": "Device1130000", ④
    "deviceInstance": 1130000,     ⑤
    "mode": "disabled",            ⑥
    "ip": {},                      ⑦
    "mstp": {}                     ⑧
}
```

① The allowed access for a BACnet client.

② The APDU Timeout; the amount of time a client waits for a response from a BACnet device.

③ The APDU retries; the number of times a client will retry sending an APDU.

④ The device name; the name of the BACnet device.

⑤ The device instance; the instance of the BACnet device.

⑥ The BACnet server mode; disabled, IP, or MSTP.

⑦ BACnet IP on page 71.

⑧ BACnet MSTP on page 77.

▼ **CLI:** get conf bacnet server

user> get conf bacnet server

```
access: readOnly              ①
apduTimeout: 3000             ②
apduRetries: 3                ③
deviceName: Device1130000     ④
deviceInstance: 1130000       ⑤
mode: disabled                ⑥
ip:                           ⑦
    ...
mstp:                         ⑧
    ...
```

① The allowed access for a BACnet client.

② The APDU Timeout; the amount of time a client waits for a response from a BACnet device.

③ The APDU retries; the number of times a client will retry sending an APDU.

④ The device name; the name of the BACnet device.

⑤ The device instance; the instance of the BACnet device.

⑥ The BACnet server mode; disabled, IP, or MSTP.

⑦ BACnet IP on the facing page.

⑧ BACnet MSTP on page 77.

The BACnet server fields are settable.

Example using "deviceName" field:

▼ **API:** *POST /api/conf/bacnet/server*

```
POST http://<device-ip>/api/conf/bacnet/server HTTP/1.1
Content-Type: application/json

{
    "deviceName" : "Vertiv_Device"
}     ①
```

① Command to set the BACnet deviceName.

▼ **API (cmd):** *api/conf/bacnet/server : set*

api/conf/bacnet/server:

```
{
    "cmd": "set",
    "data": {
        "deviceName" : "Vertiv_Device"
    }
} ①
```

① Command to set the BACnet deviceName.

▼ **CLI:** *set conf bacnet server deviceName*

admin> set conf bacnet server deviceName = ARGS

```
set conf bacnet server deviceName = "Vertiv_Device" ①
```

① Command to set the BACnet deviceName.

BACnet IP

The BACnet IP configuration settings.

▼ API: *GET /api/conf/bacnet/server/ip*

```
GET http://<device-ip>/api/conf/bacnet/server/ip HTTP/1.1

{
    "bbmdAddress": "",           ①
    "fdrEnabled": false,         ②
    "fdrTtl": 1800,              ③
    "maxCovSubscriptions": 5000, ④
    "port": 47808,               ⑤
    "trustedAccess": []          ⑥
}
```

① The IP address of the BACnet BBMD.

② Enable Foreign Device Registration.

③ The time to live of the BACnet BBMD device.

④ The maximum number of BACnet COV subscriptions.

⑤ The port number of the BACnet server.

⑥ BACnet IP Trusted Access on page 73.

▼ API (cmd): *api/conf/bacnet/server/ip*

api/conf/bacnet/server/ip: get

```
{
    "bbmdAddress": "",           ①
    "fdrEnabled": false,         ②
    "fdrTtl": 1800,              ③
    "maxCovSubscriptions": 5000, ④
    "port": 47808,               ⑤
    "trustedAccess": []          ⑥
}
```

① The IP address of the BACnet BBMD.

② Enable Foreign Device Registration.

③ The time to live of the BACnet BBMD device.

④ The maximum number of BACnet COV subscriptions.

⑤ The port number of the BACnet server.

⑥ BACnet IP Trusted Access on page 73.

▼ **CLI:** get conf bacnet server ip

user> get conf bacnet server ip

```
bbmdAddress: ""           ①
fdrEnabled: false         ②
fdrTtl: 1800              ③
maxCovSubscriptions: 5000 ④
port: 47808               ⑤
trustedAccess:            ⑥
    ...
```

① The IP address of the BACnet BBMD.

② Enable Foreign Device Registration.

③ The time to live of the BACnet BBMD device.

④ The maximum number of BACnet COV subscriptions.

⑤ The port number of the BACnet server.

⑥ BACnet IP Trusted Access on the facing page.

The BACnet server IP fields are settable.

Example using "fdrTtl" field:

▼ **API:** *POST /api/conf/bacnet/server/ip*

```
POST http://<device-ip>/api/conf/bacnet/server/ip HTTP/1.1
Content-Type: application/json

{
    "fdrTtl" : 1800
}     ①
```

① Command to set the time of BACnet Foreign Device Time-to-Live.

▼ **API (cmd):** *api/conf/bacnet/server/ip : set*

api/conf/bacnet/server/ip:

```
{
    "cmd": "set",
    "data": {
        "fdrTtl" : 1800
    }
}     ①
```

① Command to set the time of BACnet Foreign Device Time-to-Live.

▼ **CLI:** set conf bacnet server ip fdrTtl

admin> set conf bacnet server ip fdrTtl = ARGS

```
set conf bacnet server ip fdrTtl = 1800 ①
```

① Command to set the time of BACnet Foreign Device Time-to-Live.

BACnet IP Trusted Access

The BACnet IP Trusted Access configuration settings. When empty, the BACnet server is open to all clients. When configured, the BACnet server will be open to all clients that meet the Trusted Access configuration.

▼ **API:** *GET /api/conf/bacnet/server/ip/trustedAccess*

```
GET http://<device-ip>/api/conf/bacnet/server/ip/trustedAccess HTTP/1.1

[
    {
        "prefix": 24,             ①
        "address": "192.168.0.1", ②
        "id": "id0"               ③
    }
]
```

① The prefix for the IP address/network that is trusted.

② The IP address/network of the client that is trusted.

③ The ID of the trusted access entry.

▼ **API (cmd):** *api/conf/bacnet/server/ip/trustedAccess*

api/conf/bacnet/server/ip/trustedAccess: get

```
[
  {
    "prefix": 24,             ①
    "address": "192.168.0.1", ②
    "id": "id0"               ③
  }
]
```

① The prefix for the IP address/network that is trusted.

② The IP address/network of the client that is trusted.

③ The ID of the trusted access entry.

▼ **CLI:** get conf bacnet server ip trustedAccess

user> get conf bacnet server ip trustedAccess

```
-  prefix: 24          ①
   address: 192.168.0.1 ②
   id: id0              ③
```

① The prefix for the IP address/network that is trusted.

② The IP address/network of the client that is trusted.

③ The ID of the trusted access entry.

▼ **API:** *GET /api/conf/bacnet/server/ip/trustedAccess/<id>*

```
GET http://<device-ip>/api/conf/bacnet/server/ip/trustedAccess/<id> HTTP/1.1

{
    "prefix": 24,            ①
    "address": "192.168.0.1" ②
}
```

① The prefix for the IP address/network that is trusted.

② The IP address/network of the client that is trusted.

▼ **API (cmd):** *api/conf/bacnet/server/ip/trustedAccess/<id> : get*

api/conf/bacnet/server/ip/trustedAccess/<id>:

```
{
"prefix": 24,            ①
"address": "192.168.0.1" ②
}
```

① The prefix for the IP address/network that is trusted.

② The IP address/network of the client that is trusted.

▼ **CLI:** get conf bacnet server ip trustedAccess <id>

user> get conf bacnet server ip trustedAccess id0

```
prefix: 24          ①
address: 192.168.0.1 ②
```

① The prefix for the IP address/network that is trusted.

② The IP address/network of the client that is trusted.

▼ API: *PUT /api/conf/bacnet/server/ip/trustedAccess*

```
PUT http://<device-ip>/api/conf/bacnet/server/ip/trustedAccess HTTP/1.1
Content-Type: application/json

{
    "prefix" : 24,
    "address" : "192.168.0.1",
    "id": "id0" ①
}
```

① Optional "id" field is auto-generated if not set.

▼ API (cmd): *api/conf/bacnet/server/ip/trustedAccess : add*

api/conf/bacnet/server/ip/trustedAccess:

```
{
    "cmd": "add",
    "data": {
        "prefix" : 24,
        "address" : "192.168.0.1",
        "id": "id0" ①
    }
}
```

① Optional "id" field is auto-generated if not set.

▼ CLI: add conf bacnet server ip trustedAccess

user> add conf bacnet server ip trustedAccess = ARGS (Admin)

```
add conf bacnet server ip trustedAccess = {"prefix": 24, "address": "192.168.0.1",
"id": "id0"} ①
```

① Optional "id" field is auto-generated if not set.

▼ API: *DELETE /api/conf/bacnet/server/ip/trustedAccess/<id>*

```
DELETE http://<device-ip>/api/conf/bacnet/server/ip/trustedAccess/<id> HTTP/1.1
```

▼ **API (cmd):** *api/conf/bacnet/server/ip/trustedAccess/<id> : delete*

api/conf/bacnet/server/ip/trustedAccess/<id>:

```
{
    "cmd": "delete"
} ①
```

① Command to delete a BACnet IP trustedAccess entry.

▼ **CLI:** *delete conf bacnet server ip trustedAccess <id>*

admin> delete conf bacnet server ip trustedAccess <id>

```
delete conf bacnet trustedAccess id0 ①
```

① Command to delete a BACnet IP trustedAccess entry.

The BACnet IP trustedAccess fields are settable.

▼ **API:** *POST /api/conf/bacnet/server/ip/trustedAccess/<id>*

```
POST http://<device-ip>/api/conf/bacnet/server/ip/trustedAccess/<id> HTTP/1.1
Content-Type: application/json

{
    "address": "192.168.0.2"
}     ①
```

① Command to set the trustedAccess address field.

▼ **API (cmd):** *api/conf/bacnet/server/ip/trustedAccess/<id> : set*

api/conf/bacnet/server/ip/trustedAccess/<id>:

```
{
        "cmd": "set",
        "data": {
        "address": "192.168.0.2"
    }
} ①
```

① Command to set the trustedAccess address field.

▼ **CLI:** *set conf bacnet server ip trustedAccess <id>*

admin> set conf bacnet server ip trustedAccess <id> = ARGS

```
set conf bacnet server ip trustedAccess id0 = {"address": 192.168.0.2} ①
```

① Command to set the trustedAccess address field.

BACnet MSTP

The BACnet MSTP configuration settings.

▼ API: *GET /api/conf/bacnet/server/mstp*

```
GET http://<device-ip>/api/conf/bacnet/server/mstp HTTP/1.1

{
    "baudRate": "38400",          ①
    "maxCovSubscriptions": 1000, ②
    "maxInfoFrames": 8,           ③
    "maxNodeId": 127,             ④
    "nodeId": 1,                  ⑤
    "portParameters": "8N2"       ⑥
}
```

① The baud rate of the BACnet MSTP serial link.

② The maximum number of BACnet COV subscriptions.

③ The maximum number of BACnet INFO frames that can be sent while having the token.

④ The maximum node ID.

⑤ The node ID.

⑥ The port parameters.

▼ API (cmd): *api/conf/bacnet/server/mstp*

api/conf/bacnet/server/mstp: get

```
{
    "baudRate": "38400",          ①
    "maxCovSubscriptions": 1000, ②
    "maxInfoFrames": 8,           ③
    "maxNodeId": 127,             ④
    "nodeId": 1,                  ⑤
    "portParameters": "8N2"       ⑥
}
```

① The baud rate of the BACnet MSTP serial link.

② The maximum number of BACnet COV subscriptions.

③ The maximum number of BACnet INFO frames that can be sent while having the token.

④ The maximum node ID.

⑤ The node ID.

⑥ The port parameters.

▼ **CLI:** get conf bacnet server mstp

user> get conf bacnet server mstp

```
baudRate: 38400           ①
maxCovSubscriptions: 1000 ②
maxInfoFrames: 8          ③
maxNodeId: 127            ④
nodeId: 1                 ⑤
portParameters: 8N2       ⑥
```

① The baud rate of the BACnet MSTP serial link.

② The maximum number of BACnet COV subscriptions.

③ The maximum number of BACnet INFO frames that can be sent while having the token.

④ The maximum node ID.

⑤ The node ID.

⑥ The port parameters.

The BACnet server MSTP fields are settable.

Example using "baudRate" field:

▼ **API:** *POST /api/conf/bacnet/server/mstp*

```
POST http://<device-ip>/api/conf/bacnet/server/mstp HTTP/1.1
Content-Type: application/json

{
    "baudRate": "38400"
}     ①
```

① Command to set the BACnet MSTP baudrate.

▼ **API (cmd):** *api/conf/bacnet/server/mstp : set*

api/conf/bacnet/server/mstp:

```
{
    "cmd": "set",
    "data": {
        "baudRate": "38400"
    }
}    ①
```

① Command to set the BACnet MSTP baudrate.

▼ **CLI:** set conf bacnet server mstp baudRate

admin> set conf bacnet server mstp baudRate = ARGS

```
set conf bacnet server mstp baudRate = 38400 ①
```

① Command to set the BACnet MSTP baudrate.

2.4.2 CLI

The CLI configuration settings allow the user to configure the session idle timeout.

▼ **API:** *GET /api/conf/cli*

```
GET http://<device-ip>/api/conf/cli HTTP/1.1

{
    "sessionIdleTimeout": 60 ①
}
```

① The CLI session idle timeout in minutes.

▼ **API (cmd):** *api/conf/cli*

api/conf/cli: get

```
{
    "sessionIdleTimeout": 60 ①
}
```

① The CLI session idle timeout in minutes.

▼ **CLI:** get conf cli

user> get conf cli

```
sessionIdleTimeout: 60 ①
```

① The CLI session idle timeout in minutes.

▼ **API:** *POST /api/conf/cli*

```
POST http://<device-ip>/api/conf/cli HTTP/1.1
Content-Type: application/json

{
    "sessionIdleTimeout": 5
}     ①
```

① Command to set the CLI session idle timeout.

▼ **API (cmd):** *api/conf/cli : set*

api/conf/cli:

```
{
    "cmd": "set",
    "data": {
        "sessionIdleTimeout": 5
    }
} ①
```

① Command to set the CLI session idle timeout.

▼ **CLI:** set conf cli sesssionIdleTimeout

admin> set conf cli sessionIdleTimeout = ARGS

```
set conf cli sessionIdleTimeout = 5 ①
```

① Command to set the CLI session idle timeout.

2.4.3 Data Logging

The Data Logging configuration settings for periodic data collection and storage.

▼ API: *GET /api/conf/datalogging*

```
GET http://<device-ip>/api/conf/datalogging HTTP/1.1

{
    "enabled": true,     ①
    "intervalSec": 600,  ②
    "dataPoint": []      ③
}
```

① Enable or disable data logging.

② Data collection interval in seconds.

③ Data Point Configuration on page 84.

▼ API (cmd): *api/conf/datalogging*

api/conf/datalogging: get

```
{
    "enabled": true,     ①
    "intervalSec": 600,  ②
    "dataPoint": []      ③
}
```

① Enable or disable data logging.

② Data collection interval in seconds.

③ Data Point Configuration on page 84.

▼ CLI: get conf datalogging

user> get conf datalogging

```
enabled: true     ①
intervalSec: 600  ②
dataPoint:        ③
...
```

① Enable or disable data logging.

② Data collection interval in seconds.

③ Data Point Configuration on page 84.

The enabled and intervalSec fields are settable. The dataPoint collection is read-only.

▼ API: *POST /api/conf/datalogging*

```
POST http://<device-ip>/api/conf/datalogging HTTP/1.1
Content-Type: application/json

{
    "enabled": true
}     ①
```

① Command to enable/disable data logging.

▼ API (cmd): *api/conf/datalogging : set*

api/conf/datalogging:

```
{
    "cmd": "set",
    "data": {
    "enabled": true
    }
}    ①
```

① Command to enable/disable data logging.

▼ CLI: set conf datalogging enabled

admin> set conf datalogging enabled = ARGS

```
set conf datalogging enabled = true ①
```

① Command to enable/disable data logging.

▼ API: *POST /api/conf/datalogging*

```
POST http://<device-ip>/api/conf/datalogging HTTP/1.1
Content-Type: application/json

{
    "intervalSec": 300
}     ①
```

① Set data collection interval in seconds.

▼ API (cmd): *api/conf/datalogging : set*

api/conf/datalogging:

```
{
```

```
    "cmd": "set",
    "data": {
    "intervalSec": 300
    }
}    ①
```

① Set data collection interval in seconds.

▼ **CLI:** set conf datalogging intervalSec

admin> set conf datalogging intervalSec = ARGS

```
set conf datalogging intervalSec = 300 ①
```

① Set data collection interval in seconds.

Data Point Configuration

Individual data point settings within the data logging collection. Each data point represents a specific measurement to be periodically logged. The collection supports up to 20 data points (read-only collection).

▼ API: *GET /api/conf/datalogging/dataPoint*

```
GET http://<device-ip>/api/conf/datalogging/dataPoint HTTP/1.1

{
    "id": "1",                                          ①
    "gddId": 4204,                                      ②
    "moduleIndex": 1,                                   ③
    "enabled": true,                                    ④
    "alias": "System Output RMS Current L1",            ⑤
    "description": "System output RMS current phase L1" ⑥
}
```

① Unique identifier for the data point.

② Global Data Dictionary (GDD) identifier (1-65535).

③ Module index for the data point (1-64).

④ Enable or disable this specific data point.

⑤ Human-readable alias for the data point.

⑥ Detailed description of the data point.

▼ API (cmd): *api/conf/datalogging/dataPoint*

api/conf/datalogging/dataPoint: get

```
{
    "id": "1",                                          ①
    "gddId": 4204,                                      ②
    "moduleIndex": 1,                                   ③
    "enabled": true,                                    ④
    "alias": "System Output RMS Current L1",            ⑤
    "description": "System output RMS current phase L1" ⑥
}
```

① Unique identifier for the data point.

② Global Data Dictionary (GDD) identifier (1-65535).

③ Module index for the data point (1-64).

④ Enable or disable this specific data point.

⑤ Human-readable alias for the data point.

⑥ Detailed description of the data point.

▼ **CLI:** get conf datalogging dataPoint

user> get conf datalogging dataPoint

```
id: "1"                                                ①
gddId: 4204                                            ②
moduleIndex: 1                                         ③
enabled: true                                          ④
alias: "System Output RMS Current L1"                  ⑤
description: "System output RMS current phase L1" ⑥
```

① Unique identifier for the data point.

② Global Data Dictionary (GDD) identifier (1-65535).

③ Module index for the data point (1-64).

④ Enable or disable this specific data point.

⑤ Human-readable alias for the data point.

⑥ Detailed description of the data point.

2.4.4 Cloud

Remote Service

Each field serves a specific purpose in the configuration of the various aspects of the remote service settings, including diagnostic details, proxy settings, connection retries, and cloud service integration.

▼ **API**: *GET /api/conf/cloud/remoteService*

```
GET http://<device-ip>/api/conf/cloud/remoteService HTTP/1.1

{
    "diag": {},                                        ①
    "connectionTestResult": "SUCCESS",                 ②
    "proxy": {},                                       ③
    "connectionRetryTime": 30,                         ④
    "serviceCenterCountry": "United States",           ⑤
    "deviceInstanceId": "ABC123XYZ",                   ⑥
    "siteId": "site-001-North",                        ⑦
    "siteEquipmentTagNumber": "100001",                ⑧
    "serialNumberOverrideEnabled": true,               ⑨
    "configuredSerialNumber": "2127910562AD03C",       ⑩
    "cloudURL": "mq-service.vertiv.cloud",             ⑪
    "deviceDataSamplingEnabled": true,                 ⑫
    "enabled": true                                    ⑬
}
```

① Diagnostic. See Diagnostic on page 89.

② Used to store the result of a connection test for the remote service.

③ Proxy. See Proxy on page 92.

④ The time interval (in seconds) between retry attempts for establishing a connection.

⑤ Remote service center country.

⑥ Remote service device instance id.

⑦ Remote service site id.

⑧ Remote service site equipment tag number.

⑨ Indicates if remote service serial number override is enabled - if enabled the device uses the entered remote service serial number instead of the serial number obtained from the managed device.

⑩ Remote service assigned serial number.

⑪ Remote service cloud URL.

⑫ Indicates if device data sampling is enabled.

⑬ Indicates whether the remote service is enabled.

▼ API (cmd): *api/conf/cloud/remoteService*

api/conf/cloud/remoteService: get

```
{
   "diag": {},                                    ①
   "connectionTestResult": "SUCCESS",             ②
   "proxy": {},                                   ③
   "connectionRetryTime": 30,                     ④
   "serviceCenterCountry": "United States",       ⑤
   "deviceInstanceId": "ABC123XYZ",               ⑥
   "siteId": "site-001-North",                    ⑦
   "siteEquipmentTagNumber": "100001",            ⑧
   "serialNumberOverrideEnabled": true,           ⑨
   "configuredSerialNumber": "2127910562AD03C",   ⑩
   "cloudURL": "mq-service.vertiv.cloud",         ⑪
   "deviceDataSamplingEnabled": true,             ⑫
   "enabled": true                                ⑬
}
```

① Diagnostic. See Diagnostic on page 89.

② Used to store the result of a connection test for the remote service.

③ Proxy. See Proxy on page 92.

④ The time interval (in seconds) between retry attempts for establishing a connection.

⑤ Remote service center country.

⑥ Remote service device instance id.

⑦ Remote service site id.

⑧ Remote service site equipment tag number.

⑨ Indicates if remote service serial number override is enabled - if enabled the device uses the entered remote service serial number instead of the serial number obtained from the managed device.

⑩ Remote service assigned serial number.

⑪ Remote service cloud URL.

⑫ Indicates if device data sampling is enabled.

⑬ Indicates whether the remote service is enabled.

▼ **CLI:** get conf cloud remoteService

user> get conf cloud remoteService

```
diag:                                    ①
...
connectionTestResult: SUCCESS            ②
proxy:                                   ③
...
connectionRetryTime: 30                  ④
serviceCenterCountry: United States      ⑤
deviceInstanceId: ABC123XYZ              ⑥
siteId: site-001-North                   ⑦
siteEquipmentTagNumber: 100001           ⑧
serialNumberOverrideEnabled: true        ⑨
configuredSerialNumber: 2127910562AD03C  ⑩
cloudURL: mq-service.vertiv.cloud        ⑪
deviceDataSamplingEnabled: true          ⑫
enabled: true                            ⑬
```

① Diagnostic. See Diagnostic on page 89.

② Used to store the result of a connection test for the remote service.

③ Proxy. See Proxy on page 92.

④ The time interval (in seconds) between retry attempts for establishing a connection.

⑤ Remote service center country.

⑥ Remote service device instance id.

⑦ Remote service site id.

⑧ Remote service site equipment tag number.

⑨ Indicates if remote service serial number override is enabled - if enabled the device uses the entered remote service serial number instead of the serial number obtained from the managed device.

⑩ Remote service assigned serial number.

⑪ Remote service cloud URL.

⑫ Indicates if device data sampling is enabled.

⑬ Indicates whether the remote service is enabled.

The remote service fields are settable except "connectionTestResult" which is read-only.

Example using "siteEquipmentTagNumber" field:

▼ **API:** *POST /api/conf/cloud/remoteService*

```
POST http://<device-ip>/api/conf/cloud/remoteService HTTP/1.1
Content-Type: application/json

{
    "siteEquipmentTagNumber" : "100001"
}    ①
```

① Command to set the remote service site equipment tag number.

▼ **API (cmd):** *api/conf/cloud/remoteService : set*

api/conf/cloud/remoteService:

```
{
   "cmd": "set",
   "data": {
       "siteEquipmentTagNumber" : "100001"
  }
} ①
```

① Command to set the remote service site equipment tag number.

▼ **CLI:** set conf cloud remoteService siteEquipmentTagNumber

admin> set conf cloud remoteService siteEquipmentTagNumber = ARGS

```
set conf cloud remoteService siteEquipmentTagNumber = "100001" ①
```

① Command to set the site equipment tag number.

Diagnostic

The remote service diagnostic reports the current status information and any successful/error messages between the cloud and gateway.

▼ API: *GET /api/conf/cloud/remoteService/diag*

```
GET http://<device-ip>/api/conf/cloud/remoteService/diag HTTP/1.1

{
    "cloudMessage": "2024-08-19T19:07:00.0702Z - Device registration ...",    ①
    "manDeviceRegisterConfirmed": false,                                      ②
    "gatewayRegistered": true,                                                ③
    "manDeviceRegistered": false,                                             ④
    "gatewayRegisterConfirmed": true,                                         ⑤
    "dmpCommStatus": "Not Supported",                                         ⑥
    "manDeviceStatus": "Online",                                              ⑦
    "deviceRuleFileInfo": "",                                                 ⑧
    "lastError": "",                                                          ⑨
    "lastSuccess": "2024-08-19 19:29:46 UTC - Sent 1 State items --> Cloud ", ⑩
    "remoteServiceOPStatus": "Normal Operation",                              ⑪
    "errorCount": 0,                                                          ⑫
    "status": "Connected"                                                     ⑬
}
```

① Error messages and status of reponse for requests from the gateway to the remote service cloud.

② Indicates if the managed device and the remote service cloud are in agreement with respect to the device registration.

③ Indicates if the gateway is registered with a remote service's cloud platform.

④ Indicates if the managed device is registered with a remote service's cloud.

⑤ Indicates if the gateway and the remote service cloud are in agreement with respect to the gateway registration.

⑥ Indicates the status of the DMP communications with the managed device (Online | Offline | Not Supported).

⑦ Status of the managed device's communication to the card (online/offline).

⑧ Used for storing information about device rule files.

⑨ Used to store information about the last error encountered.

⑩ Used to store information about the last successful operation.

⑪ Indicates the operational status of the remote service.

⑫ Number of errors encountered by the remote service.

⑬ Indicates whether the remote service is currently monitoring the device.

▼ API (cmd): *api/conf/cloud/remoteService/diag*

api/conf/cloud/remoteService/diag: get

```
{
   "cloudMessage": "2024-08-19T19:07:00.0702Z - Device registration ...",     ①
   "manDeviceRegisterConfirmed": false,                                       ②
   "gatewayRegistered": true,                                                 ③
   "manDeviceRegistered": false,                                              ④
   "gatewayRegisterConfirmed": true,                                          ⑤
   "dmpCommStatus": "Not Supported",                                          ⑥
   "manDeviceStatus": "Online",                                               ⑦
   "deviceRuleFileInfo": "",                                                  ⑧
   "lastError": "",                                                           ⑨
   "lastSuccess": "2024-08-19 19:29:46 UTC - Sent 1 State items --> Cloud ",  ⑩
   "remoteServiceOPStatus": "Normal Operation",                               ⑪
   "errorCount": 0,                                                           ⑫
   "status": "Connected"                                                      ⑬
}
```

① Error messages and status of reponse for requests from the gateway to the remote service cloud.

② Indicates if the managed device and the remote service cloud are in agreement with respect to the device registration.

③ Indicates if the gateway is registered with a remote service's cloud platform.

④ Indicates if the managed device is registered with a remote service's cloud.

⑤ Indicates if the gateway and the remote service cloud are in agreement with respect to the gateway registration.

⑥ Indicates the status of the DMP communications with the managed device (Online | Offline | Not Supported).

⑦ Status of the managed device's communication to the card (online/offline).

⑧ Used for storing information about device rule files

⑨ Used to store information about the last error encountered.

⑩ Used to store information about the last successful operation.

⑪ Indicates the operational status of the remote service.

⑫ Number of errors encountered by the remote service.

⑬ Indicates whether the remote service is currently monitoring the device.

▼ **CLI:** get conf cloud remoteService diag

user> get conf cloud remoteService diag

```
cloudMessage: "2024-08-19T19:07:00.0702Z - Device registration ... "   ①
manDeviceRegisterConfirmed: false                                      ②
gatewayRegistered: true                                                ③
manDeviceRegistered: false                                             ④
gatewayRegisterConfirmed: true                                         ⑤
dmpCommStatus: Not Supported                                           ⑥
manDeviceStatus: Online                                                ⑦
deviceRuleFileInfo: ""                                                 ⑧
lastError: ""                                                          ⑨
lastSuccess: "2024-08-19 19:33:46 UTC - Sent 1 State items --> Cloud " ⑩
remoteServiceOPStatus: Normal Operation                                ⑪
errorCount: 0                                                          ⑫
status: Connected                                                      ⑬
```

① Error messages and status of reponse for requests from the gateway to the remote service cloud.

② Indicates if the managed device and the remote service cloud are in agreement with respect to the device registration.

③ Indicates if the gateway is registered with a remote service's cloud platform.

④ Indicates if the managed device is registered with a remote service's cloud.

⑤ Indicates if the gateway and the remote service cloud are in agreement with respect to the gateway registration.

⑥ Indicates the status of the DMP communications with the managed device (Online | Offline | Not Supported).

⑦ Status of the managed device's communication to the card (online/offline).

⑧ Used for storing information about device rule files

⑨ Used to store information about the last error encountered.

⑩ Used to store information about the last successful operation.

⑪ Indicates the operational status of the remote service.

⑫ Number of errors encountered by the remote service.

⑬ Indicates whether the remote service is currently monitoring the device.

Proxy

The Remote Service proxy settings.

▼ API: *GET /api/conf/cloud/remoteService/proxy*

```
GET http://<device-ip>/api/conf/cloud/remoteService/proxy HTTP/1.1

{
    "username": "",       ①
    "authEnabled": false, ②
    "port": 80,           ③
    "address": "",        ④
    "passwordSet": false, ⑤
    "password": "",       ⑥
    "enabled": false      ⑦
}
```

① Used for storing the proxy server username.

② Indicates if proxy server authentication is enabled.

③ Port number used for proxy communication.

④ Used for storing the proxy server address.

⑤ Indicates if proxy server password is set.

⑥ Used for storing the proxy server password.

⑦ Indicates if proxy server is enabled.

▼ API (cmd): *api/conf/cloud/remoteService/proxy*

api/conf/cloud/remoteService/proxy: get

```
{
    "username": "",       ①
    "authEnabled": false, ②
    "port": 80,           ③
    "address": "",        ④
    "passwordSet": false, ⑤
    "password": "",       ⑥
    "enabled": false      ⑦
}
```

① Used for storing the proxy server username.

② Indicates if proxy server authentication is enabled.

③ Port number used for proxy communication.

④ Used for storing the proxy server address.

⑤ Indicates if proxy server password is set.

⑥ Used for storing the proxy server password.

⑦ Indicates if proxy server is enabled.

▼ **CLI:** get conf cloud remoteService proxy

user> get conf cloud remoteService proxy

```
username: ""       ①
authEnabled: false ②
port: 80           ③
address: ""        ④
passwordSet: false ⑤
password: ""       ⑥
enabled: false     ⑦
```

① Used for storing the proxy server username.

② Indicates if proxy server authentication is enabled.

③ Port number used for proxy communication.

④ Used for storing the proxy server address.

⑤ Indicates if proxy server password is set.

⑥ Used for storing the proxy server password.

⑦ Indicates if proxy server is enabled.

The proxy fields are settable.

▼ **API:** *POST /api/conf/cloud/remoteService/proxy*

```
POST http://<device-ip>/api/conf/cloud/remoteService/proxy HTTP/1.1
Content-Type: application/json

{
    "username": "Vertiv"
}    ①
```

① Command to set the proxy username.

▼ **API (cmd):** *api/conf/cloud/remoteService/proxy : set*

api/conf/cloud/remoteService/proxy:

```
{
  "cmd": "set",
  "data": {
  "username": "Vertiv"
  }
} ①
```

① Command to set the proxy username.

▼ **CLI:** set conf cloud remoteService proxy username

admin> set conf cloud remoteService proxy username = ARGS

```
set conf cloud remoteService proxy username = "Vertiv" ①
```

① Command to set the proxy username.

2.4.5 Email

The Email configuration settings allow the user to configure email server parameters and email targets.

▼ **API:** *GET /api/conf/email*

```
GET http://<device-ip>/api/conf/email HTTP/1.1

{
    "passwordSet": false,              ①
    "port": 25,                        ②
    "sender": "",                      ③
    "server": "",                      ④
    "password": "",                    ⑤
    "subjectPrefix": "",               ⑥
    "username": "",                    ⑦
    "enableEventConsolidation": false, ⑧
    "consolidationEventLimit": 20,     ⑨
    "consolidationTimeLimit": 60,      ⑩
    "smtpAuthentication": false,       ⑪
    "target": []                       ⑫
}
```

① The indicator whether the server password has been set or not.

② The Email client sending port.

③ The Email sender address.

④ The Email server address.

⑤ The Email server password. This field is ignored when SMTP authentication is not enabled.

⑥ The Email subject prefix prepended to all email messages.

⑦ The Email username. This field is ignored when SMTP authentication is not enabled.

⑧ Indicates if event consolidation is enabled. When enabled, multiple events are combined into a single email message.

⑨ When event consolidation is enabled, specifies the number of buffered events that must accumulate before an email is sent. This trigger is independent from the time interval trigger.

⑩ When event consolidation is enabled, specifies the interval in minutes after the first buffered event is received before an email is sent. This trigger is independent from the event limit trigger.

⑪ Indicates if SMTP authentication is enabled. When enabled, the client will authenticate with the email server using the provided username and password.

⑫ Email Target. See Email Target on page 98.

▼ API (cmd): *api/conf/email*

api/conf/email: get

```
{
    "passwordSet": false,              ①
    "port": 25,                        ②
    "sender": "",                      ③
    "server": "",                      ④
    "password": "",                    ⑤
    "subjectPrefix": "",               ⑥
    "username": "",                    ⑦
    "enableEventConsolidation": false, ⑧
    "consolidationEventLimit": 20,     ⑨
    "consolidationTimeLimit": 60,      ⑩
    "smtpAuthentication": false,       ⑪
    "target": []                       ⑫
}
```

① The indicator whether the server password has been set or not.

② The Email client sending port.

③ The Email sender address.

④ The Email server address.

⑤ The Email server password. This field is ignored when SMTP authentication is not enabled.

⑥ The Email subject prefix prepended to all email messages.

⑦ The Email username. This field is ignored when SMTP authentication is not enabled.

⑧ Indicates if event consolidation is enabled. When enabled, multiple events are combined into a single email message.

⑨ When event consolidation is enabled, specifies the number of buffered events that must accumulate before an email is sent. This trigger is independent from the time interval trigger.

⑩ When event consolidation is enabled, specifies the interval in minutes after the first buffered event is received before an email is sent. This trigger is independent from the event limit trigger.

⑪ Indicates if SMTP authentication is enabled. When enabled, the client will authenticate with the email server using the provided username and password.

⑫ Email Target. See Email Target on page 98.

▼ **CLI:** get conf email

user> get conf email

```
passwordSet: false              ①
port: 25                        ②
sender: ""                      ③
server: ""                      ④
password: ""                    ⑤
subjectPrefix: ""               ⑥
username: ""                    ⑦
enableEventConsolidation: false ⑧
consolidationEventLimit: 20     ⑨
consolidationTimeLimit: 60      ⑩
smtpAuthentication: false       ⑪
target:                         ⑫
...
```

① The indicator whether the server password has been set or not.

② The Email client sending port.

③ The Email sender address.

④ The Email server address.

⑤ The Email server password. This field is ignored when SMTP authentication is not enabled.

⑥ The Email subject prefix prepended to all email messages.

⑦ The Email username. This field is ignored when SMTP authentication is not enabled.

⑧ Indicates if event consolidation is enabled. When enabled, multiple events are combined into a single email message.

⑨ When event consolidation is enabled, specifies the number of buffered events that must accumulate before an email is sent. This trigger is independent from the time interval trigger.

⑩ When event consolidation is enabled, specifies the interval in minutes after the first buffered event is received before an email is sent. This trigger is independent from the event limit trigger.

⑪ Indicates if SMTP authentication is enabled. When enabled, the client will authenticate with the email server using the provided username and password.

⑫ Email Target. See Email Target on page 98.

The Email fields are settable except "passwordSet" which is read-only.

Example using "server" field:

▼ **API:** *POST /api/conf/email*

```
POST http://<device-ip>/api/conf/email HTTP/1.1
Content-Type: application/json

{
    "server": "smtp.gmail.com"
}     ①
```

① Command to set the email SMTP server address.

▼ **API (cmd):** *api/conf/email : set*

api/conf/email:

```
{
    "cmd": "set",
    "data": {
        "server": "smtp.gmail.com"
    }
}    ①
```

① Command to set the email SMTP server address.

▼ **CLI:** set conf email server

admin> set conf email server = ARGS

```
set conf email server = "smtp.gmail.com" ①
```

① Command to set the email SMTP server address.

Email Target

The Email Target configuration settings allow the user to configure email targets.

▼ API: *GET /api/conf/email/target*

```
GET http://<device-ip>/api/conf/email/target HTTP/1.1

[
    {
        "id": "",   ①
        "name": "", ②
    }
]
```

① The Email target id.

② The Email target email address.

▼ API (cmd): *api/conf/email/target*

api/conf/email/target: get

```
[
  {
    "id": "",   ①
    "name": "", ②
  }
]
```

① The Email target id.

② The Email target email address.

▼ CLI: get conf email target

user> get conf email target

```
-   id: ""   ①
    name: "" ②
```

① The Email target id.

② The Email target email address.

▼ **API:** *PUT /api/conf/email/target*

```
PUT http://<device-ip>/api/conf/email/target HTTP/1.1
Content-Type: application/json

{
    "id": "test",
    "name": "john.z.smith@myorg.com"
}     ①
```

① Required fields: id, name (email).

▼ **CLI:** add conf email target

admin> add conf email target = ARGS

```
add conf email target = {"id": "test", "name": "john.z.smith@myorg.com"} ①
```

① Required fields: id, name (email).

▼ **API:** *DELETE /api/conf/email/target/<id>*

```
DELETE http://<device-ip>/api/conf/email/target/<id> HTTP/1.1 ①
```

① Command to delete an email target.

▼ **API (cmd):** *api/conf/email/target/<id> : delete*

api/conf/email/target/<id>:

```
{
    "cmd": "delete"
}     ①
```

① Command to delete an email target.

▼ **CLI:** *delete conf email target <id>*

admin> delete conf email target <id>

```
delete conf email target 0 ①
```

① Command to delete an email target.

▼ API: *POST /api/conf/email/target/<id>*

```
POST http://<device-ip>/api/conf/email/target/<id> HTTP/1.1
Content-Type: application/json

{
    "name": "john.z.smith@myorg.com"
}     ①
```

① Command to set the target name (email).

▼ API (cmd): *api/conf/email/target/<id>: set*

api/conf/email/target/<id>:

```
{
    "cmd": "set",
    "data": {
        "name": "john.z.smith@myorg.com"
    }
}    ①
```

① Command to set the target name (email).

▼ CLI: *set conf email target <id>*

admin> set conf email target <id> = ARGS

```
set conf email target test = {"name" : "receive@gmail.com"} ①
```

① Command to set the target name (email).

▼ API (cmd): *api/conf/email/target/<id> : sendTest (Admin)*

api/conf/email/target/<id>:

```
{
    "cmd": "sendTest"
}    ①
```

① Command to test an email target. The subject of the test message is "Test Email".

▼ CLI: *sendTest conf email target <id>*

admin> sendTest conf email target <id>

```
sendTest conf email target test ①
```

① Command to test an email target. The subject of the test message is "Test Email".

2.4.6 LLDP

The Link Layer Discovery Protocol (LLDP) configuration.

▼ API: *GET /api/conf/lldp*

```
GET http://<device-ip>/api/conf/lldp HTTP/1.1

{
    "enabled": false,        ①
    "password": "",          ②
    "passwordSet": false,    ③
    "systemDescription": "", ④
    "txHoldMultiplier": 3,   ⑤
    "txInterval": 10,        ⑥
    "username": "",          ⑦
}
```

① The enable/disable for the LLDP protocol.

② The LLDP server password.

③ The LLDP indicator that the password has been set.

④ The LLDP system description.

⑤ The LLDP transmit hold multiplier.

⑥ The LLDP transmit interval.

⑦ The LLDP server username.

▼ API (cmd): *api/conf/lldp*

api/conf/lldp: get

```
{
    "enabled": false,        ①
    "password": "",          ②
    "passwordSet": false,    ③
    "systemDescription": "", ④
    "txHoldMultiplier": 3,   ⑤
    "txInterval": 10,        ⑥
    "username": "",          ⑦
}
```

① The enable/disable for the LLDP protocol.

② The LLDP server password.

③ The LLDP indicator that the password has been set.

④ The LLDP system description.

⑤ The LLDP transmit hold multiplier.

⑥ The LLDP transmit interval.

⑦ The LLDP server username.

▼ **CLI:** get conf lldp

user> get conf lldp

```
enabled: false        ①
password: ""          ②
passwordSet: false    ③
systemDescription: "" ④
txHoldMultiplier: 3   ⑤
txInterval: 10        ⑥
username: ""          ⑦
```

① The enable/disable for the LLDP protocol.

② The LLDP server password.

③ The LLDP indicator that the password has been set.

④ The LLDP system description.

⑤ The LLDP transmit hold multiplier.

⑥ The LLDP transmit interval.

⑦ The LLDP server username.

The lldp fields are settable.

Example using "enabled" field:

▼ **API:** *POST /api/conf/lldp*

```
POST http://<device-ip>/api/conf/lldp HTTP/1.1
Content-Type: application/json

{
    "enabled": true
}     ①
```

① Command to set the LLDP enabled field.

▼ **API (cmd):** *api/conf/lldp : set (Admin)*

api/conf/lldp:

```
{
    "cmd": "set",
    "data": {
        "enabled": true
    }
}     ①
```

① Command to set the LLDP enabled field.

▼ **CLI:** set conf lldp enabled

admin> set conf lldp enabled = ARGS

```
set conf lldp enabled = true ①
```

① Command to set the LLDP enabled field.

2.4.7 Locale

The locale configuration for the card. Currently language support is English and Japanese. This only affects the card's user interface. Measurements can be US or Metric.

▼ **API:** *GET /api/conf/locale*

```
GET http://<device-ip>/api/conf/locale HTTP/1.1

{
    "systemLanguage": "en",     ①
    "measurementSystem": "us", ②
}
```

① The language displayed on the user interface.

② The measurement system is US or Metric.

▼ **API (cmd):** *api/conf/locale*

api/conf/locale: get

```
{
    "systemLanguage": "en",       ①
    "measurementSystem": "us",    ②
}
```

① The language displayed on the user interface.

② The measurement system is US or Metric.

▼ **CLI:** get conf locale

user> get conf locale

```
systemLanguage: en          ①
measurementSystem: us       ②
```

① The language displayed on the user interface.

② The measurement system is US or Metric.

The locale fields are settable.

Example using the "measurementSystem" field:

▼ **API:** *GET /api/conf/locale/measurementSystem*

```
GET http://<device-ip>/api/conf/locale/measurementSystem HTTP/1.1

us ①
```

① The measurement system is US or Metric.

▼ **API (cmd):** *api/conf/locale/measurementSystem : get*

api/conf/locale/measurementSystem:

```
us ①
```

① The measurement system is US or Metric.

▼ **CLI:** get conf locale measurementSystem

user> get conf locale measurementSystem

```
false ①
```

① The measurement system is US or Metric.

▼ **API:***POST /api/conf/locale*

```
POST http://<device-ip>/api/conf/locale HTTP/1.1
Content-Type: application/json

{
    "measurementSystem": "metric"
}     ①
```

① Command to set the locale measurement system.

▼ **API (cmd):** *api/conf/locale*

api/conf/locale:

```
{
    "cmd": "set",
    "data": {
        "measurementSystem": "metric"
    }
} ①
```

① Command to set the locale measurement system.

▼ **CLI:** set conf locale measurementSystem

admin> set conf locale measurementSystem = ARGS

```
set conf locale measurementSystem = "metric" ①
```

① Command to set the locale measurement system.

2.4.8 Modbus Server

The Modbus server configuration settings allow the user to enable/disable the Modbus server and configure the access parameters.

▼ **API:** *GET /api/conf/modbus/server*

```
GET http://<device-ip>/api/conf/modbus/server HTTP/1.1

{
    "access": "readOnly", ①
    "mode": "disabled",   ②
    "rtu": {},            ③
    "tcp": {}             ④
}
```

① The allowed access for a Modbus client.

② The Modbus server mode; disabled, IP, or RTU.

③ Modbus RTU. See Modbus RTU on page 107.

④ Modbus TCP. See Modbus TCP on page 109.

▼ **API (cmd):** *api/conf/modbus/server*

api/conf/modbus/server: get

```
{
    "access": "readOnly", ①
    "mode": "disabled",   ②
    "rtu": {},            ③
    "tcp": {}             ④
}
```

① The allowed access for a Modbus client.

② The Modbus server mode; disabled, IP, or RTU.

③ Modbus RTU. See Modbus RTU on page 107.

④ Modbus TCP. See Modbus TCP on page 109.

▼ **CLI:** *get conf modbus server*

user> get conf modbus server

```
access: readOnly ①
mode: disabled   ②
rtu: {}          ③
tcp: {}          ④
```

① The allowed access for a Modbus client.

② The Modbus server mode; disabled, IP, or RTU.

③ Modbus RTU. See Modbus RTU on the facing page.

④ Modbus TCP. See Modbus TCP on page 109.

The "access" and "mode" fields are settable.

Example using the "access" field:

▼ **API:** *POST /api/conf/modbus/server*

```
POST http://<device-ip>/api/conf/modbus/server HTTP/1.1
Content-Type: application/json

{
    "access": "readOnly"
}     ①
```

① Command to set the access field.

▼ **API (cmd):** *api/conf/modbus/server : set*

api/conf/modbus/server:

```
{
    "cmd": "set",
    "data": {
        "access": "readOnly"
    }
} ①
```

① Command to set the access field.

▼ **CLI:** set conf modbus server access

admin> set conf modbus server access = ARGS

```
set conf modbus server access = "readOnly" ①
```

① Command to set the access field.

Modbus RTU

The Modbus RTU server configuration settings allow the user to configure the RTU parameters.

▼ API: *GET /api/conf/modbus/server/rtu*

```
GET http://<device-ip>/api/conf/modbus/server/rtu HTTP/1.1

{
    "baudrate": "9600",      ①
    "portParameters": "8N2", ②
    "nodeId": 1              ③
}
```

① The Modbus RTU baud rate.

② The Modbus RTU port parameters.

③ The Modbus RTU node ID.

▼ API (cmd): *api/conf/modbus/server/rtu*

api/conf/modbus/server/rtu: get

```
{
    "baudrate": "9600",      ①
    "portParameters": "8N2", ②
    "nodeId": 1              ③
}
```

① The Modbus RTU baud rate.

② The Modbus RTU port parameters.

③ The Modbus RTU node ID.

▼ CLI: get conf modbus server rtu

user> get conf modbus server rtu

```
baudrate: 9600      ①
portParameters: 8N2 ②
nodeId: 1           ③
```

① The Modbus RTU baud rate.

② The Modbus RTU port parameters.

③ The Modbus RTU node ID.

The Modbus Server RTU fields are settable.

Example using the "baudRate" field:

▼ **API:** *POST /api/conf/modbus/server/rtu/*

```
POST http://<device-ip>/api/conf/modbus/server/rtu/ HTTP/1.1
Content-Type: application/json

{
    "baudRate": "19200"
}     ①
```

① Command to set the baudRate field.

▼ **API (cmd)** api/conf/modbus/server/rtu/ : set

api/conf/modbus/server/rtu/:

```
{
    "cmd": "set",
    "data": {
        "baudRate": "19200"
    }
} ①
```

① Command to set the baudRate field.

▼ **CLI:** set conf modbus server rtu baudRate

admin> set conf modbus server rtu baudRate = ARGS

```
set conf modbus server rtu baudRate = 19200 ①
```

① Command to set the baudRate field.

Modbus TCP

The Modbus TCP server configuration settings allow the user to configure the TCP parameters.

▼ **API:** *GET /api/conf/modbus/server/tcp*

```
GET http://<device-ip>/api/conf/modbus/server/tcp HTTP/1.1

{
    "maxClients": 4,    ①
    "port": 502         ②
    "trustedAccess": [] ③
}
```

① The maximum number of Modbus TCP clients.

② The Modbus TCP port.

③ Modbus TCP Trusted Access. See Modbus TCP Trusted Access on page 111.

▼ **API (cmd):** *api/conf/modbus/server/tcp*

api/conf/modbus/server/tcp: get

```
{
    "maxClients": 4,    ①
    "port": 502         ②
    "trustedAccess": [] ③
}
```

① The maximum number of Modbus TCP clients.

② The Modbus TCP port.

③ Modbus TCP Trusted Access. See Modbus TCP Trusted Access on page 111.

▼ **CLI:** get conf modbus server tcp

user> get conf modbus server tcp

```
maxClients: 4  ①
port: 502      ②
trustedAccess: ③
    ...
```

① The maximum number of Modbus TCP clients.

② The Modbus TCP port.

③ Modbus TCP Trusted Access. See Modbus TCP Trusted Access on page 111.

The Modbus Server TCP fields are settable.

Example using the "maxClients" field:

▼ **API:** *POST /api/conf/modbus/server/tcp*

```
POST http://<device-ip>/api/conf/modbus/server/tcp HTTP/1.1
Content-Type: application/json

{
    "maxClients": 3
}     ①
```

① Command to set the maxClients field.

▼ **API (cmd):** *api/conf/modbus/server/tcp : set*

api/conf/modbus/server/tcp:

```
{
    "cmd": "set",
    "data": {
        "maxClients": 3
    }
} ①
```

① Command to set the maxClients field.

▼ **CLI:** set conf modbus server tcp maxClients

admin> set conf modbus server tcp maxClients = ARGS

```
set conf modbus server tcp maxClients = 3 ①
```

① Command to set the maxClients field.

Modbus TCP Trusted Access

The Modbus TCP Trusted Acccess server configuration settings allow the user to configure the TCP Trusted Access parameters.

▼ API: *GET /api/conf/modbus/server/tcp/trustedAccess*

```
GET http://<device-ip>/api/conf/modbus/server/tcp/trustedAccess HTTP/1.1

[
    {
        "id": "id0",              ①
        "address": "126.4.212.1", ②
        "prefix": 24              ③
    }
]
```

① The trusted access entry id.

② The trusted access entry address.

③ The trusted access entry address prefix.

▼ API (cmd): *api/conf/modbus/server/tcp/trustedAccess*

api/conf/modbus/server/tcp/trustedAccess: get

```
[
  {
    "id": "id0",              ①
    "address": "126.4.212.1", ②
    "prefix": 24              ③
  }
]
```

① The trusted access entry id.

② The trusted access entry address.

③ The trusted access entry address prefix.

▼ CLI: get conf modbus server tcp trustedAccess

user> get conf modbus server tcp trustedAccess

```
-  id: id0             ①
   address: 126.4.212.1 ②
   prefix: 24           ③
```

① The trusted access entry id.

② The trusted access entry address.

③ The trusted access entry address prefix.

▼ API: *PUT /api/conf/modbus/server/tcp/trustedAccess*

```
PUT http://<device-ip>/api/conf/modbus/server/tcp/trustedAccess HTTP/1.1
Content-Type: application/json

{
    "id": "id0",
    "address": "126.4.212.1",
    "prefix": 24
}    ①
```

① Command to add a trustedAccess entry.

NOTE: "id" field is auto generated if not set.

▼ API (cmd): *api/conf/modbus/server/tcp/trustedAccess : add*

api/conf/modbus/server/tcp/trustedAccess:

```
{
    "cmd": "add",
    "data": {
        "id": "id0",
        "address": "126.4.212.1",
        "prefix": 24
    }
} ①
```

① Command to add a trustedAccess entry.

NOTE: "id" field is auto generated if not set.

▼ CLI: add conf modbus tcp trustedAccess

admin> add conf modbus server tcp trustedAccess = ARGS

```
add conf modbus server tcp trustedAccess = {"id": "id0", "address": "126.4.212.1",
"prefix": 24} ①
```

① Command to add a trustedAccess entry.

NOTE: "id" field is auto generated if not set.

▼ API: *DELETE /api/conf/modbus/server/tcp/trustedAccess/<id>*

```
DELETE http://<device-ip>/api/conf/modbus/server/tcp/trustedAccess/<id> HTTP/1.1
```

▼ API (cmd): *api/conf/modbus/server/tcp/trustedAccess/<id> : delete*

api/conf/modbus/server/tcp/trustedAccess/<id>:

```
{
    "cmd": "delete"
} ①
```

① Command to delete a trustedAccess object.

▼ CLI: *delete conf modbus server tcp trustedAccess <id>*

admin> delete conf modbus server tcp trustedAccess <id>

```
delete conf modbus server tcp trustedAccess id0 ①
```

① Command to delete a trustedAccess object.

The Modbus TCP trustedAccess fields are settable.

▼ API: *POST /api/conf/modbus/server/tcp/trustedAccess/<id>*

```
POST http://<device-ip>/api/conf/modbus/server/tcp/trustedAccess/<id> HTTP/1.1
Content-Type: application/json

{
    "address": "126.4.212.2"
}     ①
```

① Command to set the trustedAccess address field.

▼ API (cmd): *api/conf/modbus/server/tcp/trustedAccess/<id> : set*

api/conf/modbus/server/tcp/trustedAccess/<id>:

```
{
    "cmd": "set",
    "data": {
        "address": "126.4.212.2"
    }
} ①
```

① Command to set the trustedAccess address field.

▼ CLI: set conf modbus server tcp trustedAccess <id>

admin> set conf modbus server tcp trustedAccess <id> = ARGS

```
set conf modbus server tcp trustedAccess id0 = {"address": 126.4.212.2} ①
```

① Command to set the trustedAccess address field.

2.4.9 Network

Each field serves a specific purpose in the configuration of the network settings for the device, providing flexibility and control over network behavior and connectivity.

▼ API: *GET /api/conf/network*

```
GET http://<device-ip>/api/conf/network HTTP/1.1

{
    "hostname": "rdu120-06753e7300d3", ①
    "dnsSearchList": "",               ②
    "defaultIpEnabled": true,          ③
    "ip4Enabled": true,                ④
    "ip6Enabled": true,                ⑤
    "ipV4GW": "",                      ⑥
    "ipV6GW": "",                      ⑦
    "ipV4BootMode": "DHCP",            ⑧
    "ipV6BootMode": "DHCP",            ⑨
    "ipv4DnsSource": "automatic",      ⑩
    "ipv6DnsSource": "automatic",      ⑪
    "dhcpNtpOpt42": true,              ⑫
    "dns": [],                         ⑬
    "interface": {},                   ⑭
    "route": []                        ⑮
}
```

① The hostname of the device.

② A list of domain search suffixes for DNS resolution.

③ Indicates if the default IP configuration (192.168.123.123) is enabled.

④ Indicates if IPv4 is enabled on the device.

⑤ Indicates if IPv6 is enabled on the device.

⑥ The IPv4 address of the gateway for network traffic destined for other networks or subnets.

⑦ The IPv6 address of the gateway for network traffic destined for other networks or subnets.

⑧ Defines the method for obtaining an IPv4 address.

⑨ Defines the method for obtaining an IPv6 address.

⑩ Specifies how the DNS addresses for IPv4 are obtained. See Network/DNS on page 118 for possible states.

⑪ Specifies how the DNS addresses for IPv6 are obtained. See Network/DNS on page 118 for possible states.

⑫ Indicates if the NTP Option 42 is enabled for DHCP.

⑬ Network/DNS. See Network/DNS on page 118.

⑭ Network/Interface/LAN. See Network/Interface/LAN on page 124.

⑮ Network/Route. See Network/Route on page 121.

▼ **API (cmd):** *api/conf/network*

api/conf/network: get

```
{
    "hostname": "rdu120-06753e7300d3", ①
    "dnsSearchList": "",               ②
    "defaultIpEnabled": true,          ③
    "ip4Enabled": true,                ④
    "ip6Enabled": true,                ⑤
    "ipV4GW": "",                      ⑥
    "ipV6GW": "",                      ⑦
    "ipV4BootMode": "DHCP",            ⑧
    "ipV6BootMode": "DHCP",            ⑨
    "ipv4DnsSource": "automatic",      ⑩
    "ipv6DnsSource": "automatic",      ⑪
    "dhcpNtpOpt42": true,              ⑫
    "dns": [],                         ⑬
    "interface": {},                   ⑭
    "route": []                        ⑮
}
```

① The hostname of the device.

② A list of domain search suffixes for DNS resolution.

③ Indicates if the default IP configuration (192.168.123.123) is enabled.

④ Indicates if IPv4 is enabled on the device.

⑤ Indicates if IPv6 is enabled on the device.

⑥ The IPv4 address of the gateway for network traffic destined for other networks or subnets.

⑦ The IPv6 address of the gateway for network traffic destined for other networks or subnets.

⑧ Defines the method for obtaining an IPv4 address.

⑨ Defines the method for obtaining an IPv6 address.

⑩ Specifies how the DNS addresses for IPv4 are obtained. See Network/DNS on page 118 for possible states.

⑪ Specifies how the DNS addresses for IPv6 are obtained. See Network/DNS on page 118 for possible states.

⑫ Indicates if the NTP Option 42 is enabled for DHCP.

⑬ Network/DNS. SeeNetwork/DNS on page 118.

⑭ Network/Interface/LAN. See Network/Interface/LAN on page 124.

⑮ Network/Route. See Network/Route on page 121.

▼ **CLI:** get conf network

user> get conf network

```
hostname: rdu120-06753e7300d3 ①
dnsSearchList: ""             ②
defaultIpEnabled: true        ③
ip4Enabled: true              ④
ip6Enabled: true              ⑤
ipV4GW: ""                    ⑥
ipV6GW: ""                    ⑦
ipV4BootMode: DHCP            ⑧
ipV6BootMode: DHCP            ⑨
ipv4DnsSource: automatic      ⑩
ipv6DnsSource: automatic      ⑪
dhcpNtpOpt42: true            ⑫
dns:                          ⑬
    ...
interface:                    ⑭
    ...
route:                        ⑮
    ...
```

① The hostname of the device.

② A list of domain search suffixes for DNS resolution.

③ Indicates if the default IP configuration (192.168.123.123) is enabled.

④ Indicates if IPv4 is enabled on the device.

⑤ Indicates if IPv6 is enabled on the device.

⑥ The IPv4 address of the gateway for network traffic destined for other networks or subnets.

⑦ The IPv6 address of the gateway for network traffic destined for other networks or subnets.

⑧ Defines the method for obtaining an IPv4 address.

⑨ Defines the method for obtaining an IPv6 address.

⑩ Specifies how the DNS addresses for IPv4 are obtained. See Network/DNS on page 118 for possible states.

⑪ Specifies how the DNS addresses for IPv6 are obtained. See Network/DNS on page 118 for possible states.

⑫ Indicates if the NTP Option 42 is enabled for DHCP.

⑬ Network/DNS. See Network/DNS on page 118.

⑭ Network/Interface/LAN. See Network/Interface/LAN on page 124.

⑮ Network/Route. See Network/Route on page 121.

The network fields are settable except "ipv4DnsSource" and "ipv6DnsSource" which are read-only.

Example using "hostname" field:

API: *POST /api/conf/network*

```
POST http://<device-ip>/api/conf/network HTTP/1.1
Content-Type: application/json

{
    "hostname" : "rdu120-vertiv"
}     ①
```

① Command to set the hostname.

▼ **API (cmd):** *api/conf/network : set*

api/conf/network:

```
{
  "cmd": "set",
  "data": {
      "hostname" : "rdu120-vertiv"
  }
} ①
```

① Command to set the hostname.

▼ **CLI:** set conf network hostname

admin> set conf network hostname = ARGS

```
set conf network hostname = "rdu120-vertiv" ①
```

① Command to set the hostname.

Network/DNS

Supports adds and deletes for DNS addresses with a maximum of two IPv4 and two IPv6 addresses. Address field can be IPv4 or IPv6. The dynamically acquired DNS addresses will have the "mutable" field set to false, as the user cannot configure them directly, and do not count towards the maximum number of addresses that can be added. Possible DNS source states are:

"automatic"	Only dynamically acquired DNS addresses are being used.
"configured"	Only statically configured DNS addresses are being used.
"none"	DNS is disabled.

▼ API: *GET /api/conf/network/dns*

```
GET http://<device-ip>/api/conf/network/dns HTTP/1.1

[
    {
        "mutable": true,      ①
        "address": "8.8.8.8", ②
        "id": "0"             ③
    },
    {
        "mutable": true,
        "address": "8.8.4.4",
        "id": "1"
    },
    {
        "mutable": true,
        "address": "2001:4860:4860::8844",
        "id": "3"
    },
    {
        "mutable": true,
        "address": "2001:4860:4860::8888",
        "id": "2"
    }
]
```

① Indicates if the address can be modified.

② Address field can be IPv4 or IPv6.

③ Identifier for the address entry.

▼ **API (cmd):** *api/conf/network/dns*

api/conf/network/dns: get

```
[
    {
        "mutable": true,                    ①
        "address": "8.8.8.8",               ②
        "id": "0"                           ③
    },
    {
        "mutable": true,
        "address": "8.8.4.4",
        "id": "1"
    },
    {
        "mutable": true,
        "address": "2001:4860:4860::8844",
        "id": "3"
    },
    {
        "mutable": true,
        "address": "2001:4860:4860::8888",
        "id": "2"
    }
]
```

① Indicates if the address can be modified.

② Address field can be IPv4 or IPv6.

③ Identifier for the address entry.

▼ **CLI:** get conf network dns

user> get conf network dns

```
- mutable: true      ①
  address: 8.8.8.8   ②
  id: 0              ③
- mutable: true
  address: 8.8.4.4
  id: 1
- mutable: true
  address: 2001:4860:4860::8844
  id: 3
- mutable: true
  address: 2001:4860:4860::8888
  id: 2
```

① Indicates if the address can be modified.

② Address field can be IPv4 or IPv6.

③ Identifier for the address entry.

▼ **API:** *PUT /api/conf/network/dns*

```
PUT http://<device-ip>/api/conf/network/dns HTTP/1.1
Content-Type: application/json

{
    "address": "8.8.4.4"
}    ①
```

① Command to add network dns. Maximum of two IPv4 and two IPv6 DNS addresses can be added.

▼ **API (cmd):** *api/conf/network/dns : add*

api/conf/network/dns:

```
{
  "cmd": "add",
  "data": {
      "address": "8.8.4.4"
   }
} ①
```

① Command to add network dns. Maximum of two IPv4 and two IPv6 DNS addresses can be added.

▼ **CLI:** add conf network dns (Admin)

admin> add conf network dns = ARGS

```
add conf network dns = {"address": "8.8.4.4"} ①
```

① Command to add network dns. Maximum of two IPv4 and two IPv6 DNS addresses can be added.

▼ **API:** *DELETE /api/conf/network/dns/<id>*

```
DELETE http://<device-ip>/api/conf/network/dns/<id> HTTP/1.1
```

▼ **API (cmd):** *api/conf/network/dns/<id> : delete*

api/conf/network/dns/<id>:

```
{
    "cmd": "delete"
} ①
```

① Command to delete network dns.

NOTE: Entry must be mutable.

▼ CLI: *delete conf network dns <id> (Admin)*

admin> delete conf network dns <id>

```
delete conf network dns 0 ①
```

① Command to delete network dns.

NOTE: Entry must be mutable.

Network/Route

Network routes for all interfaces. "destination" and "gateway" must be both IPv4 or both IPv6.

Default routes are distinguished by a "destination" of "0.0.0.0" or "::", and will have a "prefix" of 0 and the interface "all". Only one default route can exist for IPv4 and one for IPv6.

▼ API: *GET /api/conf/network/route*

```
GET http://<device-ip>/api/conf/network/route HTTP/1.1

[
    {
        "mutable": false,          ①
        "prefix": 0,               ②
        "interface": "all",        ③
        "gateway": "126.4.212.1",  ④
        "destination": "0.0.0.0",  ⑤
        "id": "8"                  ⑥
    }
]
```

① Indicates if the address can be modified.

② Network prefix length.

③ The interface to which the route applies.

④ The gateway address for the route.

⑤ The port number of the BACnet server.

⑥ Identifier for the route entry.

▼ **API (cmd):** *api/conf/network/route*

api/conf/network/route: get

```
[
  {
    "mutable": false,          ①
    "prefix": 0,               ②
    "interface": "all",        ③
    "gateway": "126.4.212.1",  ④
    "destination": "0.0.0.0",  ⑤
    "id": "8"                  ⑥
  }
]
```

① Indicates if the address can be modified.

② Network prefix length.

③ The interface to which the route applies.

④ The gateway address for the route.

⑤ The port number of the BACnet server.

⑥ Identifier for the route entry.

▼ **CLI:** get conf network route

user> get conf network route

```
-  mutable: false              ①
   prefix: 0                   ②
   interface: all              ③
   gateway: 126.4.212.1        ④
   destination: 0.0.0.0        ⑤
   id: 8                       ⑥
```

① Indicates if the address can be modified.

② Network prefix length.

③ The interface to which the route applies.

④ The gateway address for the route.

⑤ The port number of the BACnet server.

⑥ Identifier for the route entry.

▼ **API:** *PUT /api/conf/network/route*

```
PUT http://<device-ip>/api/conf/network/route HTTP/1.1
Content-Type: application/json

{
    "destination": "0.0.0.0",
    "prefix": 0,
    "gateway": "192.168.0.1",
    "interface": "all"
}   ①
```

① Required fields: destination, prefix, gateway, interface.

▼ **API (cmd):** *api/conf/network/route : add*

api/conf/network/route:

```
{
    "cmd": "add",
    "data": {
    "destination": "0.0.0.0",
    "prefix": 0,
    "gateway": "192.168.0.1",
    "interface": "all"
    }
}   ①
```

① Required fields: destination, prefix, gateway, interface.

▼ **CLI:** add conf network route (Admin)

admin> add conf network route = ARGS

```
add conf network route = {"destination": "0.0.0.0", "prefix": 0, "gateway":
"126.4.212.1", "interface": "all"} ①
```

① Required fields: destination, prefix, gateway, interface.

▼ **API:** *DELETE /api/conf/network/route/<id>*

```
DELETE http://<device-ip>/api/conf/network/route/<id> HTTP/1.1
```

▼ API (cmd): *api/conf/network/route/<id> : delete*

api/conf/network/route/<id>:

```
{
    "cmd": "delete"
} ①
```

① Command to delete a network route.

NOTE: Entry must be mutable.

▼ CLI: *delete conf network route <id> (Admin)*

admin> delete conf network route <id>

```
delete conf network route 0 ①
```

① Command to delete a network interface.

NOTE: Entry must be mutable.

Network/Interface/LAN

Details for the LAN interface. LAN is the only current supported interface.

▼ API: *GET /api/conf/network/interface/lan*

```
GET http://<device-ip>/api/conf/network/interface/lan HTTP/1.1

{
    "enabled": true,                ①
    "removable": false,             ②
    "name": "eth0",                 ③
    "label": "eth0",                ④
    "macAddr": "00:02:99:0b:05:ac", ⑤
    "address": [],                  ⑥
    "link": {},                     ⑦
    "dot1x": {}                     ⑧
}
```

① Indicates if the interface is enabled.

② Indicates if the interface can be removed.

③ Name of the interface.

④ Label for the interface.

⑤ MAC address of the interface.

⑥ Network/Interface/LAN/Address. See Network/Interface/LAN/Address on page 127.

⑦ Network/Interface/LAN/Link. See Network/Interface/LAN/Link on page 130.

⑧ Network/Interface/LAN/Dot1x . See Network/Interface/LAN/Dot1x on page 133.

▼ **API (cmd):** *api/conf/network/interface/lan*

api/conf/network/interface/lan: get

```
{
    "enabled": true,                ①
    "removable": false,             ②
    "name": "eth0",                 ③
    "label": "eth0",                ④
    "macAddr": "00:02:99:0b:05:ac", ⑤
    "address": [],                  ⑥
    "link": {},                     ⑦
    "dot1x": {}                     ⑧
}
```

① Indicates if the interface is enabled.

② Indicates if the interface can be removed.

③ Name of the interface.

④ Label for the interface.

⑤ MAC address of the interface.

⑥ Network/Interface/LAN/Address. See Network/Interface/LAN/Address on page 127.

⑦ Network/Interface/LAN/Link. See Network/Interface/LAN/Link on page 130.

⑧ Network/Interface/LAN/Dot1x . See Network/Interface/LAN/Dot1x on page 133.

▼ **CLI:** get conf network interface lan

user> get conf network interface lan

```
enabled: true               ①
removable: false            ②
name: eth0                  ③
label: eth0                 ④
macAddr: 00:02:99:0b:05:ac  ⑤
address:                    ⑥
    ...
link:                       ⑦
    ...
dot1x:                      ⑧
    ...
```

① Indicates if the interface is enabled.

② Indicates if the interface can be removed.

③ Name of the interface.

④ Label for the interface.

⑤ MAC address of the interface.

⑥ Network/Interface/LAN/Address. See Network/Interface/LAN/Address on the facing page.

⑦ Network/Interface/LAN/Link. See Network/Interface/LAN/Link on page 130.

⑧ Network/Interface/LAN/Dot1x . See Network/Interface/LAN/Dot1x on page 133.

The interface lan fields are settable except "removable", "name", and "macAddr" which are read-only.

Example using "label" field:

▼ **API:** *POST /api/conf/network/interface/lan*

```
POST http://<device-ip>/api/conf/network/interface/lan HTTP/1.1
Content-Type: application/json

{
    "label" : "eth0"
}    ①
```

① Command to set the label.

▼ **API (cmd):** *api/conf/network/interface/lan : set*

api/conf/network/interface/lan:

```
{
  "cmd": "set",
  "data": {
      "label" : "eth0"
  }
} ①
```

① Command to set the label.

▼ **CLI:** set conf network lan

admin> set conf network interface lan label = ARGS

```
set conf network interface lan label = "eth0" ①
```

① Command to set the label.

Network/Interface/LAN/Address

IP address configuration for the LAN interface. Supports a maximum of one static IPv4 and one static IPv6 address.

▼ API: *GET /api/conf/network/interface/lan/address*

```
GET http://<device-ip>/api/conf/network/interface/lan/address HTTP/1.1

[
    {
        "mutable": false,             ①
        "prefix": 24,                 ②
        "address": "192.168.123.123", ③
        "id": "8"                     ④
    },
{
        "mutable": false,
        "prefix": 16,
        "address": "169.254.24.7",
        "id": "9"
    },
{
        "mutable": false,
        "prefix": 24,
        "address": "126.4.212.55",
        "id": "10"
    },
{
        "mutable": false,
        "prefix": 64,
        "address": "fe80::200:68ff:fe10:123a",
        "id": "11"
    }
]
```

① Indicates if the address can be modified.

② Network prefix length.

③ IP address.

④ Identifier for the address entry.

▼ API (cmd): *api/conf/network/interface/lan/address*

api/conf/network/interface/lan/address: get

```
[
    {
        "mutable": false,                     ①
        "prefix": 24,                         ②
        "address": "192.168.123.123",         ③
        "id": "8"                             ④
    },
```

```
    {
        "mutable": false,
        "prefix": 16,
        "address": "169.254.24.7",
        "id": "9"
    },
    {
        "mutable": false,
        "prefix": 24,
        "address": "126.4.212.55",
        "id": "10"
    },
    {
        "mutable": false,
        "prefix": 64,
        "address": "fe80::200:68ff:fe10:123a",
        "id": "11"
    }
]
```


① Indicates if the address can be modified.

② Network prefix length.

③ IP address.

④ Identifier for the address entry.

▼ **CLI:** get conf network interface lan address

user> get conf network interface lan address


```
- mutable: false            ①
  prefix: 24                ②
  address: 192.168.123.123  ③
  id: 8                     ④
- mutable: false
  prefix: 16
  address: 169.254.24.7
  id: 9
- mutable: false
  prefix: 24
  address: 126.4.212.55
  id: 10
- mutable: false
  prefix: 64
  address: fe80::200:68ff:fe10:123a
  id: 11
```


① Indicates if the address can be modified.

② Network prefix length.

③ IP address.

④ Identifier for the address entry.

▼ API: *PUT /api/conf/network/interface/lan/address*

```
PUT http://<device-ip>/api/conf/network/interface/lan/address HTTP/1.1
Content-Type: application/json

{
    "prefix": 24,
    "address": "192.168.0.1"
}    ①
```

① Required fields: prefix, address. Maximum of one static IPv4 and one static IPv6 address can be added.

▼ API (cmd): *api/conf/network/interface/lan/address : add*

api/conf/network/interface/lan/address:

```
{
    "cmd": "add",
    "data": {
        "prefix": 24,
        "address": "192.168.0.1"
    }
}    ①
```

① Required fields: prefix, address. Maximum of one static IPv4 and one static IPv6 address can be added.

▼ CLI: add conf network interface lan address (Admin)

admin> add conf network interface lan address = ARGS

```
add conf network interface lan address = {"prefix": 24, "address": "192.168.0.1"}
①
```

① Required fields: prefix, address. Maximum of one static IPv4 and one static IPv6 address can be added.

▼ API: *DELETE /api/conf/network/interface/lan/address/<id>*

```
DELETE http://<device-ip>/api/conf/network/interface/lan/address/<id> HTTP/1.1
```

▼ API (cmd): *api/conf/network/interface/lan/address/<id> : delete*

api/conf/network/interface/lan/address/<id>:

```
{
  "cmd": "delete"
} ①
```

① Command to delete LAN address.

NOTE: Entry must be mutable.

▼ CLI: *delete conf network interface lan address <id> (Admin)*

admin> delete conf network interface lan address <id>

```
delete conf network interface lan address 0 ①
```

① Command to delete LAN address.

NOTE: Entry must be mutable.

Network/Interface/LAN/Link

Link information and statistics for the LAN interface.

▼ API:*GET /api/conf/network/interface/lan/link*

```
GET http://<device-ip>/api/conf/network/interface/lan/link HTTP/1.1

{
    "configSpeedDuplex": "Auto", ①
    "supportedModes": [          ②
        "10baseT/Half",
        "10baseT/Full",
        "100baseT/Half",
        "100baseT/Full",
        "1000baseT/Full"
    ],
    "speed": "1Gbps",            ③
    "state": "up",               ④
    "uptime": 19102,             ⑤
    "duplex": "full",            ⑥
    "stat": {                    ⑦
        "txCount": 2750,
        "rxDropped": 0,
        "txDropped": 0,
        "rxCount": 13736
    }
}
```

① Configured speed and duplex mode.

② List of supported speed and duplex modes.

③ Current operational speed.

④ Operational state of the interface.

⑤ Uptime in seconds.

⑥ Duplex mode.

⑦ Statistics for transmitted (txCount), received (rxCount), and dropped packets (rxDropped, txDropped).

▼ **API (cmd):** *api/conf/network/interface/lan/link*

api/conf/network/interface/lan/link: get

```
{
    "configSpeedDuplex": "Auto", ①
    "supportedModes": [          ②
        "10baseT/Half",
        "10baseT/Full",
        "100baseT/Half",
        "100baseT/Full",
        "1000baseT/Full"
    ],
    "speed": "1Gbps",            ③
    "state": "up",               ④
    "uptime": 19102,             ⑤
    "duplex": "full",            ⑥
    "stat": {                    ⑦
        "txCount": 2750,
        "rxDropped": 0,
        "txDropped": 0,
        "rxCount": 13736
    }
}
```

① Configured speed and duplex mode.

② List of supported speed and duplex modes.

③ Current operational speed.

④ Operational state of the interface.

⑤ Uptime in seconds.

⑥ Duplex mode.

⑦ Statistics for transmitted (txCount), received (rxCount), and dropped packets (rxDropped, txDropped).

▼ **CLI:** get conf network interface lan link

user> get conf network interface lan link

```
configSpeedDuplex: Auto     ①
supportedModes:             ②
    - 10baseT/Half
    - 10baseT/Full
    - 100baseT/Half
    - 100baseT/Full
    - 1000baseT/Full
speed: 1Gbps                ③
state: up                   ④
uptime: 20206               ⑤
duplex: full                ⑥
stat:                       ⑦
    txCount: 3650
```

```
    rxDropped: 0
    txDropped: 0
    rxCount: 15298
```

① Configured speed and duplex mode.

② List of supported speed and duplex modes.

③ Current operational speed.

④ Operational state of the interface.

⑤ Uptime in seconds.

⑥ Duplex mode.

⑦ Statistics for transmitted (txCount), received (rxCount), and dropped packets (rxDropped, txDropped).

Only configSpeedDuplex field is settable.

▼ **API:** *POST /api/conf/network/interface/lan/link*

```
POST http://<device-ip>/api/conf/network/interface/lan/link HTTP/1.1
Content-Type: application/json

{
    "configSpeedDuplex" : "Auto"
} ①
```

① Command to set speed and duplex mode.

▼ **API (cmd):** *api/conf/network/interface/lan/link : set*

api/conf/network/interface/lan/link:

```
{
  "cmd": "set",
  "data": {
  "configSpeedDuplex" : "Auto"
 }
} ①
```

① Command to set speed and duplex mode.

▼ **CLI:** set conf network interface lan link

admin> set conf network interface lan link = ARGS

```
set conf network interface lan link configSpeedDuplex = "Auto" ①
```

① Command to set speed and duplex mode.

Network/Interface/LAN/Dot1x

802.1x authentication information.

▼ API: *GET /api/conf/network/interface/lan/dot1x*

```
GET http://<device-ip>/api/conf/network/interface/lan/dot1x HTTP/1.1

{
    "enabled": false,        ①
    "identity": "",          ②
    "privKeyPassword": null  ③
}
```

① The enable/disable for 802.1x authentication.

② The identity provided to the authentication server.

③ The password for the uploaded private key.

▼ API (cmd): *api/conf/network/interface/lan/dot1x*

api/conf/network/interface/lan/dot1x: get

```
{
    "enabled": false,       ①
    "identity": "",         ②
    "privKeyPassword": null ③
}
```

① The enable/disable for 802.1x authentication.

② The identity provided to the authentication server.

③ The password for the uploaded private key.

▼ CLI: get conf network interface lan dot1x

user> get conf network interface lan dot1x

```
enabled: false         ①
identity: ""           ②
privKeyPassword: ~     ③
```

① The enable/disable for 802.1x authentication.

② The identity provided to the authentication server.

③ The password for the uploaded private key.

The dot1x fields are settable.

▼ **API:** *POST /api/conf/network/interface/lan/dot1x*

```
POST http://<device-ip>/api/conf/network/interface/lan/dot1x HTTP/1.1
Content-Type: application/json

{
    "identity" : "Hello World"
}    ①
```

① Command to set 802.1x authentication idenitity.

▼ **API (cmd):** *api/conf/network/interface/lan/dot1x : set*

api/conf/network/interface/lan/dot1x:

```
{
    "cmd": "set",
    "data": {
        "identity" : "Hello World"
    }
} ①
```

① Command to set 802.1x authentication idenitity.

▼ **CLI:** set conf network interface lan dot1x identity

admin> set conf network interface lan dot1x identity = ARGS

```
set conf network interface lan dot1x identity = "Hello World" ①
```

① Command to set 802.1x authentication idenitity.

2.4.10 Remote Authentication

The Remote Authentication configuration settings allow the user to configure RADIUS, TACACS+, or LDAP remote authentication.

▼ API: *GET /api/conf/remoteAuth*

```
GET http://<device-ip>/api/conf/remoteAuth HTTP/1.1

{
    "mode": "none", ①
    "ldap": {},     ②
    "radius": {},   ③
    "tacacs": {}    ④
}
```

① The remote authentication service selection.

② LDAP on page 137.

③ Radius on page 141.

④ TACACS on page 144.

▼ API (cmd): *api/conf/remoteAuth*

api/conf/remoteAuth: get

```
{
    "mode": "none",  ①
    "ldap": {},      ②
    "radius": {},    ③
    "tacacs": {}     ④
}
```

① The remote authentication service selection.

② LDAP on page 137.

③ Radius on page 141.

④ TACACS on page 144.

▼ CLI: get conf remoteAuth

user> get conf remoteAuth

```
mode: none ①
ldap:      ②
    ...
radius:    ③
    ...
tacacs:    ④
    ...
```

① The remote authentication service selection.

② LDAP on the facing page.

③ Radius on page 141.

④ TACACS on page 144.

The mode field is settable.

▼ **API:** *POST /api/conf/remoteAuth*

```
POST http://<device-ip>/api/conf/remoteAuth HTTP/1.1
Content-Type: application/json

{
    "mode" : "none"
}    ①
```

① Command to set the remote authentication mode.

▼ **API: (cmd)** *api/conf/remoteAuth : set*

api/conf/remoteAuth:

```
{
  "cmd": "set",
  "data": {
  "mode" : "none"
  }
} ①
```

① Command to set the remote authentication mode.

▼ **CLI:** set conf remoteAuth mode

admin> set conf remoteAuth mode = ARGS

```
set conf remoteAuth mode = "none" ①
```

① Command to set the remote authentication mode.

LDAP

The LDAP Remote Authentication configuration settings allow the user to configure LDAP remote authentication.

▼ API:*GET /api/conf/remoteAuth/ldap*

```
GET http://<device-ip>/api/conf/remoteAuth/ldap HTTP/1.1

{
    "adminGroup": "admin",                      ①
    "baseDn": "",                               ②
    "bindDn": "",                               ③
    "controlGroup": "control",                  ④
    "enabledGroup": "enabled",                  ⑤
    "groupFilter": "(objectClass=posixGroup)",  ⑥
    "groupIdNum": "groupIdNum",                 ⑦
    "groupMemberUid": "memberOf",               ⑧
    "host": "",                                 ⑨
    "mode": "openLdap",                         ⑩
    "password": null,                           ⑪
    "passwordSet": false,                       ⑫
    "port": 389,                                ⑬
    "securityType": "ssl",                      ⑭
    "userFilter": "(objectClass=posixAccount)", ⑮
    "userId": "uid",                            ⑯
    "userIdNum": "userIdNum"                    ⑰
}
```

① The LDAP admin group.

② The LDAP base Domain Name.

③ The LDAP bind Domain Name.

④ The LDAP control group.

⑤ The LDAP enabled group.

⑥ The LDAP group filter.

⑦ The LDAP group ID number.

⑧ The LDAP group member UID.

⑨ The LDAP host.

⑩ The LDAP mode.

⑪ The LDAP password.

⑫ The LDAP password set.

⑬ The LDAP port.

⑭ The LDAP security type.

⑮ The LDAP user filter.

⑯ The LDAP user ID.

⑰ The LDAP user ID number.

▼ API (cmd): *api/conf/remoteAuth/ldap*

api/conf/remoteAuth/ldap: get

```
{
    "adminGroup": "admin",                      ①
    "baseDn": "",                               ②
    "bindDn": "",                               ③
    "controlGroup": "control",                  ④
    "enabledGroup": "enabled",                  ⑤
    "groupFilter": "(objectClass=posixGroup)", ⑥
    "groupIdNum": "groupIdNum",                 ⑦
    "groupMemberUid": "memberOf",               ⑧
    "host": "",                                 ⑨
    "mode": "openLdap",                         ⑩
    "password": null,                           ⑪
    "passwordSet": false,                       ⑫
    "port": 389,                                ⑬
    "securityType": "ssl",                      ⑭
    "userFilter": "(objectClass=posixAccount)",⑮
    "userId": "uid",                            ⑯
    "userIdNum": "userIdNum"                    ⑰
}
```

① The LDAP admin group.

② The LDAP base Domain Name.

③ The LDAP bind Domain Name.

④ The LDAP control group.

⑤ The LDAP enabled group.

⑥ The LDAP group filter.

⑦ The LDAP group ID number.

⑧ The LDAP group member UID.

⑨ The LDAP host.

⑩ The LDAP mode.

⑪ The LDAP password.

⑫ The LDAP password set.

⑬ The LDAP port.

⑭ The LDAP security type.

⑮ The LDAP user filter.

⑯ The LDAP user ID.

⑰ The LDAP user ID number.

▼ **CLI:** get conf remoteAuth ldap

user> get conf remoteAuth ldap

```
adminGroup: admin                        ①
baseDn: ""                               ②
bindDn: ""                               ③
controlGroup: control                    ④
enabledGroup: enabled                    ⑤
groupFilter: (objectClass=posixGroup)    ⑥
groupIdNum: groupIdNum                   ⑦
groupMemberUid: memberOf                 ⑧
host: ""                                 ⑨
mode: openLdap                           ⑩
password: ~                              ⑪
passwordSet: false                       ⑫
port: 389                                ⑬
securityType: ssl                        ⑭
userFilter: (objectClass=posixAccount)   ⑮
userId: uid                              ⑯
userIdNum: userIdNum                     ⑰
```

① The LDAP admin group.

② The LDAP base Domain Name.

③ The LDAP bind Domain Name.

④ The LDAP control group.

⑤ The LDAP enabled group.

⑥ The LDAP group filter.

⑦ The LDAP group ID number.

⑧ The LDAP group member UID.

⑨ The LDAP host.

⑩ The LDAP mode.

⑪ The LDAP password.

⑫ The LDAP password set.

⑬ The LDAP port.

⑭ The LDAP security type.

⑮ The LDAP user filter.

⑯ The LDAP user ID.

⑰ The LDAP user ID number.

The LDAP fields are settable.

Example using the password field:

NOTE: PasswordSet is a read-only field and is set to true when the password field is set.

▼ **API:** *POST /api/conf/remoteAuth/ldap*

```
POST http://<device-ip>/api/conf/remoteAuth/ldap HTTP/1.1
Content-Type: application/json

{
    "password": "test123"
}    ①
```

① Command to set the LDAP password.

▼ **API (cmd):** *api/conf/remoteAuth/ldap : set*

api/conf/remoteAuth/ldap:

```
{
    "cmd": "set",
    "data": {
        "password": "test123"
    }
} ①
```

① Command to set the LDAP password.

▼ **CLI:** set conf remoteAuth ldap password

admin> set conf remoteAuth ldap password = ARGS

```
set conf remoteAuth ldap password = "test123" ①
```

① Command to set the LDAP password.

Radius

The Radius Remote Authentication configuration settings allow the user to configure RADIUS remote authentication.

▼ API: *GET /api/conf/remoteAuth/radius*

```
GET http://<device-ip>/api/conf/remoteAuth/radius HTTP/1.1

{
    "adminGroup": "",               ①
    "authenticationServer1": "",    ②
    "authenticationServer2": "",    ③
    "controlGroup": "",             ④
    "enabledGroup": "",             ⑤
    "groupAttribute": "filter-id",  ⑥
    "sharedSecretSet": false,       ⑦
    "sharedSecret": null            ⑧
}
```

① The RADIUS admin group.

② The RADIUS authentication server 1.

③ The RADIUS authentication server 2.

④ The RADIUS control group.

⑤ The RADIUS enabled group.

⑥ The RADIUS group attribute.

⑦ The RADIUS shared secret set.

⑧ The RADIUS shared secret.

▼ API (cmd): *api/conf/remoteAuth/radius*

api/conf/remoteAuth/radius: get

```
{
    "adminGroup": "",               ①
    "authenticationServer1": "",    ②
    "authenticationServer2": "",    ③
    "controlGroup": "",             ④
    "enabledGroup": "",             ⑤
    "groupAttribute": "filter-id", ⑥
    "sharedSecretSet": false,       ⑦
    "sharedSecret": null            ⑧
}
```

① The RADIUS admin group.

② The RADIUS authentication server 1.

③ The RADIUS authentication server 2.

④ The RADIUS control group.

⑤ The RADIUS enabled group.

⑥ The RADIUS group attribute.

⑦ The RADIUS shared secret set.

⑧ The RADIUS shared secret.

▼ **CLI:** get conf remoteAuth radius

user> get conf remoteAuth radius

```
adminGroup: ""            ①
authenticationServer1: "" ②
authenticationServer2: "" ③
controlGroup: ""          ④
enabledGroup: ""          ⑤
groupAttribute: filter-id ⑥
sharedSecret: ~           ⑦
sharedSecretSet: false    ⑧
```

① The RADIUS admin group.

② The RADIUS authentication server 1.

③ The RADIUS authentication server 2.

④ The RADIUS control group.

⑤ The RADIUS enabled group.

⑥ The RADIUS group attribute.

⑦ The RADIUS shared secret set.

⑧ The RADIUS shared secret.

The RADIUS fields are settable.

Example using the "sharedSecret" field:

NOTE: SharedSecretSet is a read-only field and is set to true when the sharedSecret field is set.

▼ API: *POST /api/conf/remoteAuth/radius*

```
POST http://<device-ip>/api/conf/remoteAuth/radius HTTP/1.1
Content-Type: application/json

{
    "sharedSecret": "secret123"
}    ①
```

① Command to set the RADIUS password.

▼ API (cmd): *api/conf/remoteAuth/radius : set*

api/conf/remoteAuth/radius:

```
{
    "cmd": "set",
    "data": {
        "sharedSecret": "secret123"
    }
} ①
```

① Command to set the RADIUS password.

▼ CLI: set conf remoteAuth radius sharedSecret

admin> set conf remoteAuth radius sharedSecret = ARGS

```
set conf remoteAuth radius sharedSecret = "secret123" ①
```

① Command to set the RADIUS sharedSecret.

TACACS

The TACACS+ Remote Authentication configuration settings allow the user to configure TACACS+ remote authentication.

▼ API: *GET /api/conf/remoteAuth/tacacs*

```
GET http://<device-ip>/api/conf/remoteAuth/tacacs HTTP/1.1

{
    "accountingServer1": "",       ①
    "accountingServer2": "",       ②
    "adminAttribute": "",          ③
    "authenticationServer1": "",   ④
    "authenticationServer2": "",   ⑤
    "controlAttribute": "",        ⑥
    "enabledAttribute": "",        ⑦
    "service": "ppp",              ⑧
    "sharedSecret": null,          ⑨
    "sharedSecretSet": false,      ⑩
}
```

① The TACACS accounting server 1.

② The TACACS accounting server 2.

③ The TACACS admin attribute.

④ The TACACS authentication server 1.

⑤ The TACACS authentication server 2.

⑥ The TACACS control attribute.

⑦ The TACACS enabled attribute.

⑧ The TACACS service.

⑨ The TACACS shared secret.

⑩ The TACACS shared secret set.

▼ API (cmd): *api/conf/remoteAuth/tacacs*

api/conf/remoteAuth/tacacs: get

```
{
    "accountingServer1": "",     ①
    "accountingServer2": "",     ②
    "adminAttribute": "",        ③
    "authenticationServer1": "", ④
    "authenticationServer2": "", ⑤
    "controlAttribute": "",      ⑥
    "enabledAttribute": "",      ⑦
    "service": "ppp",            ⑧
    "sharedSecret": null,        ⑨
    "sharedSecretSet": false,    ⑩
}
```

① The TACACS accounting server 1.

② The TACACS accounting server 2.

③ The TACACS admin attribute.

④ The TACACS authentication server 1.

⑤ The TACACS authentication server 2.

⑥ The TACACS control attribute.

⑦ The TACACS enabled attribute.

⑧ The TACACS service.

⑨ The TACACS shared secret.

⑩ The TACACS shared secret set.

▼ CLI: *get conf remoteAuth tacacs*

user> get conf remoteAuth tacacs

```
accountingServer1: ""      ①
accountingServer2: ""      ②
adminAttribute: ""         ③
authenticationServer1: ""  ④
authenticationServer2: ""  ⑤
controlAttribute: ""       ⑥
enabledAttribute: ""       ⑦
service: ppp               ⑧
sharedSecret: ~            ⑨
sharedSecretSet: false     ⑩
```

① The TACACS accounting server 1.

② The TACACS accounting server 2.

③ The TACACS admin attribute.

④ The TACACS authentication server 1.

⑤ The TACACS authentication server 2.

⑥ The TACACS control attribute.

⑦ The TACACS enabled attribute.

⑧ The TACACS service.

⑨ The TACACS shared secret.

⑩ The TACACS shared secret set.

The TACACS fields are settable.

Example using the "sharedSecret" field:

NOTE: SharedSecretSet is a read-only field and is set to true when the sharedSecret field is set.

▼ API: *POST /api/conf/remoteAuth/tacacs*

```
POST http://<device-ip>/api/conf/remoteAuth/tacacs HTTP/1.1
Content-Type: application/json

{
    "sharedSecret": "secret123"
}   ①
```

① Command to set the TACACS password.

▼ API (cmd): *api/conf/remoteAuth/tacacs : set*

api/conf/remoteAuth/tacacs:

```
{
    "cmd": "set",
    "data": {
        "sharedSecret": "secret123"
    }
} ①
```

① Command to set the TACACS password.

▼ CLI: set conf remoteAuth tacacs sharedSecret

admin> set conf remoteAuth tacacs sharedSecret = ARGS

```
set conf remoteAuth tacacs sharedSecret = "secret123" ①
```

① Command to set the TACACS sharedSecret.

2.4.11 Remote Syslog

The Remote Syslog configuration allows up to two independent syslog server destinations. Each destination has its own enable flag, server address, port, and transport protocol (UDP or TLS). Conventionally, UDP uses port 514 and TLS uses port 6514; the server validates the port for the selected transport.

▼ API: *GET /api/conf/remoteSyslog*

```
GET http://<device-ip>/api/conf/remoteSyslog HTTP/1.1

{
    "target": []     ①
}
```

① Array of syslog targets; see Remote Syslog Target on the facing page. At most two entries.

▼ API (cmd): *api/conf/remoteSyslog*

api/conf/remoteSyslog: get

```
{
    "target": []     ①
}
```

① Array of syslog targets; see Remote Syslog Target below. At most two entries.

▼ **CLI:** get conf remoteSyslog

user> get conf remoteSyslog

```
target: []     ①
```

① List of syslog targets; see Remote Syslog Target below. At most two entries.

Remote Syslog Target

The Remote Syslog Target configuration defines each syslog server entry.

▼ **API:** *GET /api/conf/remoteSyslog/target*

```
GET http://<device-ip>/api/conf/remoteSyslog/target HTTP/1.1

[
    {
        "id": "1",          ①
        "enabled": false,   ②
        "server": "",       ③
        "port": 514,        ④
        "transport": "udp"  ⑤
    }
]
```

① Target id (read-only).

② When true, syslog is forwarded to this target according to the other fields.

③ FQDN or IPv4/IPv6 address of the syslog server; empty means unconfigured (possible only when enabled=false).

④ Network port (1 to 65535); default 514.

⑤ Transport protocol: udp or tls (default udp).

▼ **API (cmd):** *api/conf/remoteSyslog/target*

api/conf/remoteSyslog/target: get

```
[
    {
        "id": "1",          ①
        "enabled": false,   ②
        "server": "",       ③
        "port": 514,        ④
        "transport": "udp"  ⑤
    }
]
```

① Target id (read-only).

② When true, syslog is forwarded to this target according to the other fields.

③ FQDN or IPv4/IPv6 address of the syslog server; empty means unconfigured (possible only when enabled=false).

④ Network port (1 to 65535); default 514.

⑤ Transport protocol: udp or tls (default udp).

▼ **CLI:** get conf remoteSyslog target

user> get conf remoteSyslog target

```
    - id: "1"        ①
    enabled: false   ②
    server: ""       ③
    port: 514        ④
    transport: udp   ⑤
```

① Target id (read-only).

② When true, syslog is forwarded to this target according to the other fields.

③ FQDN or IPv4/IPv6 address of the syslog server; empty means unconfigured (possible only when enabled=false).

④ Network port (1 to 65535); default 514.

⑤ Transport protocol: udp or tls (default udp).

▼ **API:** *PUT /api/conf/remoteSyslog/target*

```
PUT http://<device-ip>/api/conf/remoteSyslog/target HTTP/1.1
Content-Type: application/json

{
    "id": "1",
    "enabled": false,
    "server": "192.0.2.10",
    "port": 514,
    "transport": "udp"
}   ①
```

① Required fields: id, enabled, server, port, transport.

▼ **API (cmd):** *api/conf/remoteSyslog/target : add*

api/conf/remoteSyslog/target:

```
{
    "cmd": "add",
    "data": {
        "id": "1",
        "enabled": false,
        "server": "192.0.2.10",
        "port": 514,
        "transport": "udp"
    }
}    ①
```

① Required fields: id, enabled, server, port, transport.

▼ **CLI:** add conf remoteSyslog target

admin> add conf remoteSyslog target = ARGS

```
add conf remoteSyslog target = {"id": "1", "enabled": false, "server": "192.0.2.10",
"port": 514, "transport": "udp"} ①
```

① Required fields: id, enabled, server, port, transport.

▼ **API:** *DELETE /api/conf/remoteSyslog/target/<id>*

```
DELETE http://<device-ip>/api/conf/remoteSyslog/target/<id> HTTP/1.1
```

▼ API (cmd): *api/conf/remoteSyslog/target/<id> : delete*

api/conf/remoteSyslog/target/<id>:

```
{
    "cmd": "delete"
}   ①
```

① Deletes the syslog target with the given id.

▼ CLI: *delete conf remoteSyslog target <id>*

admin> delete conf remoteSyslog target <id>

```
delete conf remoteSyslog target 1 ①
```

① Deletes the syslog target with the given id.

▼ API: *POST /api/conf/remoteSyslog/target/<id>*

```
POST http://<device-ip>/api/conf/remoteSyslog/target/<id> HTTP/1.1
Content-Type: application/json

{
    "enabled": true,
    "server": "192.0.2.10",
    "port": 6514,
    "transport": "tls"
}   ①
```

① Sets one or more of enabled, server, port, and transport on the target.

▼ API (cmd): *api/conf/remoteSyslog/target/<id> : set*

api/conf/remoteSyslog/target/<id>:

```
{
    "cmd": "set",
    "data": {
        "enabled": true,
        "server": "192.0.2.10",
        "port": 6514,
        "transport": "tls"
    }
}    ①
```

① Sets one or more of enabled, server, port, and transport on the target.

▼ CLI: *set conf remoteSyslog target <id>*

admin> set conf remoteSyslog target <id> = ARGS

```
set conf remoteSyslog target 1 = {"enabled": true, "server": "192.0.2.10", "port":
6514, "transport": "tls"} ①
```

① Sets one or more of enabled, server, port, and transport on the target.

2.4.12 Serial

The Serial configuration settings for the console port.

▼ API: *GET /api/conf/serial*

```
GET http://<device-ip>/api/conf/serial HTTP/1.1

{
    "baudRate": "115200"      ①
    "enabled": true,          ②
    "portParameters": "8N1"   ③
}
```

① Console baudrate.

② Console enable control.

③ Console serial port parameters.

▼ API (cmd): *api/conf/serial*

api/conf/serial: get

```
{
    "baudRate": "115200"    ①
    "enabled": true,        ②
    "portParameters": "8N1" ③
}
```

① Console baudrate.

② Console enable control.

③ Console serial port parameters.

▼ **CLI:** get conf serial

user> get conf serial

```
baudRate: 115200    ①
enabled: true       ②
portParameters: 8N1 ③
```

① Console baudrate.

② Console enable control.

③ Console serial port parameters.

The baudRate and enabled fields are settable.

Example using the "enabled" field:

▼ **API:** *POST /api/conf/serial*

```
POST http://<device-ip>/api/conf/serial HTTP/1.1
Content-Type: application/json

{
    "enabled": true
}    ①
```

① Command to enable/disable serial console.

▼ **API (cmd):** *api/conf/serial : set*

api/conf/serial:

```
{
    "cmd": "set",
    "data": {
        "enabled": true
    }
} ①
```

① Command to enable/disable serial console.

▼ **CLI:** set conf serial enabled

admin> set conf serial enabled = ARGS

```
set conf serial enabled = true ①
```

① Command to enable/disable serial console.

2.4.13 SNMP

The SNMP configuration settings for SNMP v1, v2c, and v3.

▼ API: *GET /api/conf/snmp*

```
GET http://<device-ip>/api/conf/snmp HTTP/1.1

{
    "v3": {},                          ①
    "v1v2c": {},                       ②
    "mibSupport": {},                  ③
    "authTrapsEnabled": false,         ④
    "heartbeatTrapInterval": 1440,     ⑤
    "port": 161                        ⑥
}
```

① snmp/v3. See SNMP v3 on page 170.

② snmp/v1v2c. See SNMP v1/v2c on page 157.

③ snmp/mibSupport. See SNMP MIB Support on page 155.

④ Enable or disable SNMP authentication traps.

⑤ The SNMP heartbeat trap interval.

⑥ The SNMP server port number.

▼ API (cmd): *api/conf/snmp*

api/conf/snmp: get

```
{
    "v3": {},                          ①
    "v1v2c": {},                       ②
    "mibSupport": {},                  ③
    "authTrapsEnabled": false,         ④
    "heartbeatTrapInterval": 1440,     ⑤
    "port": 161                        ⑥
}
```

① snmp/v3. See SNMP v3 on page 170.

② snmp/v1v2c. See SNMP v1/v2c on page 157.

③ snmp/mibSupport. See SNMP MIB Support on page 155.

④ Enable or disable SNMP authentication traps.

⑤ The SNMP heartbeat trap interval.

⑥ The SNMP server port number.

▼ **CLI:** get conf snmp

user> get conf snmp

```
v3:                          ①
    ...
v1v2c:                       ②
    ...
mibSupport:                  ③
    ...
authTrapsEnabled: false      ④
heartbeatTrapInterval: 1440  ⑤
port: 161                    ⑥
```

① snmp/v3. See SNMP v3 on page 170.

② snmp/v1v2c. See SNMP v1/v2c on page 157.

③ snmp/mibSupport. See SNMP MIB Support on the facing page.

④ Enable SNMP authentication traps.

⑤ The SNMP heartbeat trap interval.

⑥ The SNMP Server port number.

The authTrapsEnabled, heartbeatTrapInterval and port fields are settable.

Example using the "port" field:

▼ **API:** *POST /api/conf/snmp/port*

```
POST http://<device-ip>/api/conf/snmp/port HTTP/1.1
Content-Type: application/json

{
    "port": 161          ①
}
```

① The SNMP server port number.

▼ **API (cmd):** *api/conf/snmp/port: set*

api/conf/snmp/port:

```
{
    "cmd": "set",
    "data": {
        "port": 161
    }
} ①
```

① Command to set SNMP Server port number.

▼ **CLI:** set conf snmp port

admin> set conf snmp port = ARGS

```
set conf snmp port = 161 ①
```

① Command to set SNMP Server port number.

SNMP MIB Support

The SNMP MIB support configuration.

▼ **API:** *GET /api/conf/snmp/mibSupport*

```
GET http://<device-ip>/api/conf/snmp/mibSupport HTTP/1.1

{
    "extendedVBEnabled": false,            ①
    "lgpMIBSystemNotifyTrapEnabled": true, ②
    "lgpMIBTrapEnabled": true,             ③
    "lgpMIBEnabled": true,                 ④
    "rfc1628MIBTrapEnabled": true,         ⑤
    "rfc1628MIBEnabled": true              ⑥
}
```

① Add varbinds containing sysName and sysLocation to all traps.

② Enable support for Liebert Global Products (LGP) System Notification Traps. A trap containing a description of the condition will be sent each time a condition is added to or removed from the LGP MIB conditions table. lgpMIBEnabled must also be enabled.

③ Enable support for LGP MIB traps. lgpMIBEnabled must be set to true.

④ Enable support for LGP MIB objects.

⑤ Enable support for RFC-1628 MIB objects. Applies to Uninterruptible Power Supply (UPS) systems.

⑥ Enable support for RFC-1628 MIB traps. Applies to UPS systems.

▼ **API (cmd):** *api/conf/snmp/mibSupport*

api/conf/snmp/mibSupport: get

```
{
    "extendedVBEnabled": false,            ①
    "lgpMIBSystemNotifyTrapEnabled": true, ②
    "lgpMIBTrapEnabled": true,             ③
    "lgpMIBEnabled": true,                 ④
    "rfc1628MIBTrapEnabled": true,         ⑤
    "rfc1628MIBEnabled": true              ⑥
}
```

① Add varbinds containing sysName and sysLocation to all traps.

② Enable support for Liebert Global Products (LGP) System Notification Traps. A trap containing a description of the condition will be sent each time a condition is added to or removed from the LGP MIB conditions table. lgpMIBEnabled must also be enabled.

③ Enable support for LGP MIB traps. lgpMIBEnabled must be set to true.

④ Enable support for LGP MIB objects.

⑤ Enable support for RFC-1628 MIB objects. Applies to Uninterruptible Power Supply (UPS) systems.

⑥ Enable support for RFC-1628 MIB traps. Applies to UPS systems.

▼ **CLI:** get conf snmp mibSupport

user> get conf snmp mibSupport

```
extendedVBEnabled: false            ①
lgpMIBSystemNotifyTrapEnabled: true ②
lgpMIBTrapEnabled: true             ③
lgpMIBEnabled: true                 ④
rfc1628MIBTrapEnabled: true         ⑤
rfc1628MIBEnabled: true             ⑥
```

① Add varbinds containing sysName and sysLocation to all traps.

② Enable support for Liebert Global Products (LGP) System Notification Traps. A trap containing a description of the condition will be sent each time a condition is added to or removed from the LGP MIB conditions table. lgpMIBEnabled must also be enabled.

③ Enable support for LGP MIB traps. lgpMIBEnabled must be set to true.

④ Enable support for LGP MIB objects.

⑤ Enable support for RFC-1628 MIB objects. Applies to Uninterruptible Power Supply (UPS) systems.

⑥ Enable support for RFC-1628 MIB traps. Applies to UPS systems.

The mibSupport fields are settable.

Example using the "lgpMIBTrapEnabled" field.

▼ **API:** *POST /api/conf/snmp/mibSupport*

```
POST http://<device-ip>/api/conf/snmp/mibSupport HTTP/1.1
Content-Type: application/json

{
    "lgpMIBTrapEnabled": true
}     ①
```

① Command to set the lgpMIBTrapEnabled.

▼ **API (cmd):** *api/conf/snmp/mibSupport : set*

api/conf/snmp/mibSupport:

```
{
    "cmd": "set",
    "data": {
        "lgpMIBTrapEnabled": true      ①
    }
} ①
```

① Command to set the lgpMIBTrapEnabled.

▼ **CLI:** set conf snmp mibSupport lgpMIBTrapEnabled

admin> set conf snmp mibSupport lgpMIBTrapEnabled = ARGS

```
set conf snmp mibSupport lgpMIBTrapEnabled = true ①
```

① Command to set the lgpMIBTrapEnabled.

SNMP v1/v2c

The SNMP v1/v2c configuration.

▼ **API:** *GET /api/conf/snmp/v1v2c*

```
GET http://<device-ip>/api/conf/snmp/v1v2c HTTP/1.1

{
    "trap": [],             ①
    "access": [],           ②
    "enabled": false        ③
    "generateTestTraps": {} ④
}
```

① snmp/v1v2c/trap. See SNMP v1/v2c Trap on page 166.

② snmp/v1v2c/access. SeeSNMP v1/v2c Access on page 159.

③ The enable control for SNMP v1/v2c.

④ The SNMP v1/v2c test traps.

▼ **API (cmd):** *api/conf/snmp/v1v2c*

api/conf/snmp/v1v2c: get

```
{
    "trap": [],             ①
    "access": [],           ②
    "enabled": false,       ③
    "generateTestTraps": {} ④
}
```

① snmp/v1v2c/trap. See SNMP v1/v2c Trap on page 166.

② snmp/v1v2c/access. SeeSNMP v1/v2c Access on the facing page.

③ The enable control for SNMP v1/v2c.

④ The SNMP v1/v2c test traps.

▼ **CLI:** get conf snmp v1v2c

user> get conf snmp v1v2c

```
trap:                ①
    ...
access:              ②
    ...
enabled: false       ③
generateTestTraps: ~ ④
```

① snmp/v1v2c/trap. See SNMP v1/v2c Trap on page 166.

② snmp/v1v2c/access. See SNMP v1/v2c Access on the facing page.

③ The enable control for SNMP v1/v2c.

④ The SNMP v1/v2c test traps.

The enabled field is settable.

▼ **API:** *POST /api/conf/snmp/v1v2c*

```
POST http://<device-ip>/api/conf/snmp/v1v2c HTTP/1.1
Content-Type: application/json

{
    "enabled": false
}   ①
```

① Command to enable SNMP v1/v2c.

▼ **API (cmd):** *api/conf/snmp/v1v2c : set*

api/conf/snmp/v1v2c:

```
{
  "cmd": "set",
  "data": {
  "enabled": false
 }
} ①
```

① Command to enable SNMP v1/v2c.

▼ **CLI:** set conf snmp v1v2c enabled

admin> set conf snmp v1v2c enabled = ARGS

```
set conf snmp v1v2c enabled = false ①
```

① Command to enable SNMP v1/v2c.

SNMP v1/v2c Access

The SNMP v1/v2c access configuration. This is used to configure the community and access mode.

▼ **API:** *GET /api/conf/snmp/v1v2c/access*

```
GET http://<device-ip>/api/conf/snmp/v1v2c/access HTTP/1.1

[
  {
    "trustedAccess": [], ①
    "access": "read",    ②
    "community": "a#822",③
    "id": "vpn1"         ④
  }
]
```

① snmp/v1v2c/access/trustedAccess. See SNMP v1/v2c Trusted Access on page 163.

② The access rights for this access entry.

③ The v1/v2c community string.

④ The name of the access ID entry.

▼ **API (cmd):** *api/conf/snmp/v1v2c/access*

api/conf/snmp/v1v2c/access: get

```
[
  {
    "trustedAccess": [], ①
    "access": "read",    ②
    "community": "a#822",③
    "id": "vpn1"         ④
  }
]
```

① snmp/v1v2c/access/trustedAccess. See SNMP v1/v2c Trusted Access on page 163.

② The access rights for this access entry.

③ The v1/v2c community string.

④ The name of the access ID entry.

▼ **CLI:** get conf snmp v1v2c access

user> get conf snmp v1v2c access

```
 -  trustedAccess:   ①
    access: read     ②
    community: a#822 ③
    id: vpn1         ④
   ...
```

① snmp/v1v2c/access/trustedAccess. See SNMP v1/v2c Trusted Access on page 163.

② The access rights for this access entry.

③ The v1/v2c community string.

④ The name of the access ID entry.

▼ **API:** *PUT /api/conf/snmp/v1v2c/access*

```
PUT http://<device-ip>/api/conf/snmp/v1v2c/access HTTP/1.1
Content-Type: application/json

{
    "access": "read",     ①
    "community": "a#822", ②
    "port": 162,          ③
    "id": "vpn1"          ④
}
```

① The access rights for this new access entry (read or write).

② The v1/v2c community string.

③ Port value for the new entry when accepted by the device (optional in some configurations).

④ The access entry id; auto-generated if omitted.

▼ **API (cmd):** *api/conf/snmp/v1v2c/access : add*

api/conf/snmp/v1v2c/access:

```
{
    "cmd": "add",
    "data": {
    "access": "read",
    "community": "a#822",
    "port": 162,
    "id": "vpn1"
   }
} ①
```

① Command to add SNMP v1/v2c access entry.

NOTE: "id" field is auto generated if not set.

▼ **CLI:** add conf snmp v1v2c access

admin> add conf snmp v1v2c access = ARGS

```
add conf snmp v1v2c access = {"access": "read", "community": "a#822", "port": 162,
"id": "vpn1"} ①
```

① Command to add SNMP v1/v2c access entry.

NOTE: "id" field is auto generated if not set.

▼ **API:** *DELETE /api/conf/snmp/v1v2c/access/<accessID>*

```
DELETE http://<device-ip>/api/conf/snmp/v1v2c/access/<accessID> HTTP/1.1
```

▼ **API (cmd):** *api/conf/snmp/v1v2c/access/<accessID> : delete*

api/conf/snmp/v1v2c/access/<accessID>:

```
{
  "cmd": "delete"
} ①
```

① Command to delete access entry.

▼ CLI: *delete conf snmp v1v2c access <accessID>*

admin> delete conf snmp v1v2c access <accessID>

```
delete conf snmp v1v2c access vpn1 ①
```

① Command to delete access entry.

The access and community fields are settable.

Example using the "access" field:

▼ API: *POST /api/conf/snmp/v1v2c/access/<accessID>*

```
POST http://<device-ip>/api/conf/snmp/v1v2c/access/<accessID> HTTP/1.1
Content-Type: application/json

{
    "access": "write"
}  ①
```

① Command to set the access rights to Read-Write.

NOTE: The SNMP access entry (accessID) must be defined.

▼ API (cmd): *api/conf/snmp/v1v2c/access/<accessID> : set*

api/conf/snmp/v1v2c/access/<accessID>:

```
{
    "cmd": "set",
    "data": {
        "access": "write"
    }
} ①
```

① Command to set the access rights to Read-Write.

NOTE: The SNMP access entry (accessID) must be defined.

▼ CLI: *set conf snmp v1v2c access <accessID> access*

admin> set conf snmp v1v2c access <accessID> access = ARGS

```
set conf snmp v1v2c access vpn1 access = write ①
```

① Command to set the access rights to Read-Write.

SNMP v1/v2c Trusted Access

The SNMP v1/v2c trusted access configuration. This is used to configure network access rules.

▼ API: *GET /api/conf/snmp/v1v2c/access/<accessID>/trustedAccess*

```
GET http://<device-ip>/api/conf/snmp/v1v2c/access/<accessID>/trustedAccess HTTP/1.1

[
    {
        "prefix": 24,            ①
        "address": "126.4.12.0", ②
        "id": "vpn1"             ③
    }
]
```

① The network prefix when a network is specified in address.

② The IP Address/Network allowed access.

③ The name of the trusted access entry.

▼ API (cmd): *api/conf/snmp/v1v2c/access/<accessID>/trustedAccess*

api/conf/snmp/v1v2c/access/<accessID>/trustedAccess: get

```
[
  {
    "prefix": 24,             ①
    "address": "126.4.12.0",  ②
    "id": "vpn1"              ③
  }
]
```

① The network prefix when a network is specified in address.

② The IP Address/Network allowed access.

③ The name of the trusted access entry.

▼ CLI: *get conf snmp v1v2c access <accessID> trustedAccess*

user> get conf snmp v1v2c access <accessID> trustedAccess

```
-  prefix: 24           ①
   address: 126.4.12.0  ②
   id: vpn1             ③
```

① The network prefix when a network is specified in address.

② The IP Address/Network allowed access.

③ The name of the trusted access entry.

▼ API: *PUT /api/conf/snmp/v1v2c/access/<accessID>/trustedAccess*

```
PUT http://<device-ip>/api/conf/snmp/v1v2c/access/<accessID>/trustedAccess HTTP/1.1
Content-Type: application/json

{
    "prefix": 24,
    "address": "126.4.12.0",
    "id": "vpn1"
}   ①
```

① Command to add SNMP v1/v2c trusted access entry.

NOTE: "id" field is auto generated if not set.

NOTE: The combination of address and prefix must be a valid network address.

▼ API (cmd): *api/conf/snmp/v1v2c/access/<accessID>/trustedAccess : add*

api/conf/snmp/v1v2c/access/<accessID>/trustedAccess:

```
{
  "cmd": "add",
  "data": {
     "prefix": 24,
     "address": "126.4.12.0",
     "id": "vpn1"
     }
} ①
```

① Command to add SNMP v1/v2c trusted access entry.

NOTE: "id" field is auto generated if not set.

NOTE: The combination of address and prefix must be a valid network address.

▼ CLI: *add conf snmp v1v2c access <accessID> trustedAccess*

admin> add conf snmp v1v2c access <accessID> trustedAccess = ARGS

```
add conf snmp v1v2c access vpn1 trustedAccess = {"prefix": "24", "address":
"126.4.12.33", "id": "vpn1"} ①
```

① Command to add SNMP v1/v2c trusted access entry.

NOTE: "id" field is auto generated if not set.

▼ API: *DELETE /api/conf/snmp/v1v2c/access/<accessID>/trustedAccess/<trustedID>*

```
DELETE http://<device-ip>/api/conf/snmp/v1v2c/access/<accessID>/trustedAccess/<trus
tedID> HTTP/1.1
```

▼ API (cmd): *api/conf/snmp/v1v2c/access/<accessID>/trustedAccess/<trustedID> : delete*

api/conf/snmp/v1v2c/access/<accessID>/trustedAccess/<trustedID>:

```
{
  "cmd": "delete"
} ①
```

① Command to delete trusted access entry.

▼ CLI: *delete conf snmp v1v2c access <accessID> trustedAccess <trustedID>*

admin> delete conf snmp v1v2c access <accessID> trustedAccess <trustedID>

```
delete conf snmp v1v2c access vpn1 trustedAccess vpn1①
```

① Command to delete trusted access entry.

The prefix and address fields are settable.

Example using the "address" field:

▼ API: *POST /api/conf/snmp/v1v2c/access/<accessID>/trustedAccess/<trustedID>*

```
POST http://<device-ip>/api/conf/snmp/v1v2c/access/<accessID>/trustedAccess/<truste
dID> HTTP/1.1
Content-Type: application/json

{
    "address": "126.4.12.33"
}   ①
```

① Command to set the trustedAccess address field.

NOTE: The combination of incoming/existing address and prefix must be a valid network address.

▼ API (cmd): *api/conf/snmp/v1v2c/access/<accessID>/trustedAccess/<trustedID> : set*

api/conf/snmp/v1v2c/access/<accessID>/trustedAccess/<trustedID>:

```
{
  "cmd": "set",
  "data": {
    "address": "126.4.12.33"
  }
} ①
```

① Command to set the trustedAccess address field.

NOTE: The combination of incoming/existing address and prefix must be a valid network address.

▼ CLI: *set conf snmp v1v2c access <accessID> trustedAccess <trustedID>*

admin> set conf snmp v1v2c access <accessID> trustedAccess <trustedID> address = ARGS

```
set conf snmp v1v2c access vpn1 trustedAccess vpn1 address = 126.4.212.2 ①
```

① Command to set the trustedAccess address field.

SNMP v1/v2c Trap

The SNMP v1/v2c trap configuration.

▼ API: *GET /api/conf/snmp/v1v2c/trap*

```
GET http://<device-ip>/api/conf/snmp/v1v2c/trap HTTP/1.1

[
    {
        "community": "secret",   ①
        "port": 162,             ②
        "target": "126.4.12.33", ③
        "id": "0"                ④
    }
]
```

① Password string used by the access host to authenticate read and write operations.

② Port used when sending the trap.

③ Network host destination for traps. The trap target may be a hostname or an IP address.

④ The trap entry id.

▼ API (cmd): *api/conf/snmp/v1v2c/trap*

api/conf/snmp/v1v2c/trap: get

```
[
  {
    "community": "secret",   ①
    "port": 162,             ②
    "target": "126.4.12.33", ③
    "id": "0",               ④
   }
]
```

① Password string used by the access host to authenticate read and write operations.

② Port used when sending the trap.

③ Network host destination for traps. The trap target may be a hostname or an IP address.

④ The name of the trusted access entry.

▼ **CLI:** get conf snmp v1v2c trap

user> get conf snmp v1v2c trap

```
-  community: secret   ①
   port: 162           ②
   target: 126.4.12.33 ③
   id: 0               ④
```

① Password string used by the access host to authenticate read and write operations.

② Port used when sending the trap.

③ Network host destination for traps. The trap target may be a hostname or an IP address.

④ The name of the trusted access entry.

▼ **API:** *PUT /api/conf/snmp/v1v2c/trap*

```
PUT http://<device-ip>/api/conf/snmp/v1v2c/trap HTTP/1.1
Content-Type: application/json

{
    "id": "id0",
    "target": "126.4.12.34",
    "port": 162,
    "community": "Vertiv"
} ①
```

① Command to add a trap entry.

NOTE: "id" field is auto generated if not set.

▼ **API (cmd):** *api/conf/snmp/v1v2c/trap : add*

api/conf/snmp/v1v2c/trap:

```
{
"cmd": "add",
"data": {
    "id": "id0",
    "target": "126.4.12.34",
    "port": 162,
    "community": "Vertiv"
    }
} ①
```

① Command to add a trap entry.

NOTE: "id" field is auto generated if not set.

▼ **CLI:** add conf snmp v1v2c trap

admin> add conf snmp v1v2c trap = ARGS

```
add conf snmp v1v2c trap = {"id": id0, "target": 126.4.12.34, "port": 162,
"community": Vertiv} ①
```

① Command to add a trap entry.

NOTE: "id" field is auto generated if not set.

▼ **API:** *DELETE /api/conf/snmp/v1v2c/trap/<id>*

```
DELETE http://<device-ip>/api/conf/snmp/v1v2c/trap/<id> HTTP/1.1
```

▼ **API (cmd):** *api/conf/snmp/v1v2c/trap/<id> : delete*

api/conf/snmp/v1v2c/trap/<id>:

```
{
    "cmd": "delete"
} ①
```

① Command to delete trap entry.

▼ **CLI:** *delete conf snmp v1v2c trap <id>*

admin> delete conf snmp v1v2c trap <id>

```
delete conf snmp v1v2c trap id0 ①
```

① Command to delete trap entry.

The v1v2c trap fields are settable.

▼ **API:** *POST /api/conf/snmp/v1v2c/trap/<id>*

```
POST http://<device-ip>/api/conf/snmp/v1v2c/trap/<id> HTTP/1.1
Content-Type: application/json

{
    "target": "126.4.12.35"
}    ①
```

① Command to set the trap target field.

▼ API (cmd): *api/conf/snmp/v1v2c/trap/<id> : set*

api/conf/snmp/v1v2c/trap/<id>:

```
{
    "cmd": "set",
    "data": {
        "target": "126.4.12.35"
    }
} ①
```

① Command to set the trap target field.

▼ CLI: *set conf snmp v1v2c trap <id>*

admin> set conf snmp v1v2c trap <id> = ARGS

```
set conf snmp v1v2c trap id0 = {"target": 126.4.12.35} ①
```

① Command to set the trap target field.

Send test trap to a configured SNMP v1v2c target.

▼ API: *api/conf/snmp/v1v2c/trap/<id> : sendTest (Admin)*

api/conf/snmp/v1v2c/trap/<id>:

```
{
  "cmd": "sendTest"
} ①
```

① Command to test a configured SNMP v1v2c trap target.

▼ CLI: *sendTest conf snmp v1v2c trap <id>*

admin> sendTest conf snmp v1v2c trap <id>

```
sendTest conf snmp v1v2c trap id0 ①
```

① Command to test a configured SNMP v1v2c trap target.

SNMP v1/v2c Test Traps

Send test trap to all configured SNMP v1/v2c targets.

▼ API: *api/conf/snmp/v1v2c/generateTestTraps : sendTest (Admin)*

api/conf/snmp/v1v2c/generateTestTraps:

```
{
  "cmd": "sendTest"
} ①
```

① Command to test all configured SNMP v1v2c trap targets.

▼ CLI: sendTest conf snmp v1v2c generateTestTraps

admin> sendTest conf snmp v1v2c generateTestTraps

```
sendTest conf snmp v1v2c generateTestTraps ①
```

① Command to test all configured SNMP v1v2c trap targets.

SNMP v3

The SNMP v3 configuration.

▼ API: *GET /api/conf/snmp/v3*

```
GET http://<device-ip>/api/conf/snmp/v3 HTTP/1.1

{
    "user": [],                    ①
    "engineIdText": "",            ②
    "engineIdType": "MAC Address", ③
    "generateTestTraps": {},       ④
    "engineId": "0x8000deadbeef",  ⑤
    "enabled": false               ⑥
}
```

① SNMP v3 User. See SNMP v3 User on page 173.

② The SNMP v3 engine text.

③ The SNMP v3 engine id type.

④ The SNMP v3 test traps.

⑤ The SNMP v3 engine id. When empty the engineId is auto-generated using the MAC Address.

⑥ The SNMP v3 enable control.

▼ **API (cmd):** *api/conf/snmp/v3*

api/conf/snmp/v3: get

```
{
    "user": [],                    ①
    "engineIdText": "",            ②
    "engineIdType": "MAC Address", ③
    "generateTestTraps": {},       ④
    "engineId": "0x8000deadbeef",  ⑤
    "enabled": false               ⑥
}
```

① SNMP v3 User. See SNMP v3 User on page 173.

② The SNMP v3 engine text.

③ The SNMP v3 engine id type.

④ The SNMP v3 test traps.

⑤ The SNMP v3 engine id. When empty the engineId is auto-generated using the MAC Address.

⑥ The SNMP v3 enable control.

▼ **CLI:** get conf snmp v3

user> get conf snmp v3

```
user:                    ①
  ...
engineIdText:            ②
engineIdType: MAC Address ③
generateTestTraps: ~     ④
engineId: 0x8000deadbeef  ⑤
enabled: false           ⑥
```

① SNMP v3 User. See SNMP v3 User on page 173.

② The SNMP v3 engine text.

③ The SNMP v3 engine id type.

④ The SNMP v3 test traps.

⑤ The SNMP v3 engine id. When empty the engineId is auto-generated using the MAC Address.

⑥ The SNMP v3 enable control.

The engineIdText, engineIdType, and enabled fields are settable.

▼ API: *POST /api/conf/snmp/v3*

```
POST http://<device-ip>/api/conf/snmp/v3 HTTP/1.1
Content-Type: application/json

{
    "engineIdType": "MAC Address"
}   ①
```

① Command to set the engine id type field.

▼ API (cmd): *api/conf/snmp/v3 : set (Admin)*

api/conf/snmp/v3:

```
{
  "cmd": "set",
  "data": {
    "engineIdType": "MAC Address"
   }
} ①
```

① Command to set the engine id type field.

▼ CLI: set conf snmp v3

admin> set conf snmp v3 engineIdType = ARGS

```
set conf snmp v3 engineIdType = MAC Address ①
```

① Command to set the engine id type field.

SNMP v3 User

The SNMP v3 user configuration.

▼ API: *GET /api/conf/snmp/v3/user*

```
GET http://<device-ip>/api/conf/snmp/v3/user HTTP/1.1

[
    {
        "target": [],                ①
        "privPassword": null,        ②
        "privType": "des",           ③
        "privPasswordSet": true,     ④
        "authPasswordSet": true,     ⑤
        "authPassword": null,        ⑥
        "authType": "md5",           ⑦
        "access": "read",            ⑧
        "username": "user0",         ⑨
        "enabled": true,             ⑩
        "id": "0"                    ⑪
    }
]
```

① snmp/v3/user/target. See SNMP v3 User Trap Target on page 178.

② The privilige password.

③ The privilege type (NONE, DES, AES).

④ The flag that indicates the privilege password is set.

⑤ The flag that indicates the authentication password is set.

⑥ The authentication password.

⑦ The authentication type.

⑧ The access type for the user.

⑨ The user entry's user name.

⑩ The enable control for the user.

⑪ The auto-generated user entry id.

▼ **API (cmd):** *api/conf/snmp/v3/user*

api/conf/snmp/v3/user: get

```
[
  {
    "target": [],            ①
    "privPassword": null,    ②
    "privType": "des",       ③
    "privPasswordSet": true, ④
    "authPasswordSet": true, ⑤
    "authPassword": null,    ⑥
    "authType": "md5",       ⑦
    "access": "read",        ⑧
    "username": "user0",     ⑨
    "enabled": true,         ⑩
    "id": "0"                ⑪
  }
]
```

① snmp/v3/user/target. See SNMP v3 User Trap Target on page 178.

② The privilige password.

③ The privilege type (NONE, DES, AES).

④ The flag that indicates the privilege password is set.

⑤ The flag that indicates the authentication password is set.

⑥ The authentication password.

⑦ The authentication type.

⑧ The access type for the user.

⑨ The user entry's user name.

⑩ The enable control for the user.

⑪ The auto-generated user entry id.

▼ **CLI:** get conf snmp v3 user

user> get conf snmp v3 user

```
- target:                ①
    ...
  privPassword: null     ②
  privType: des          ③
  privPasswordSet: true  ④
  authPasswordSet: true  ⑤
  authPassword: null     ⑥
  authType: md5          ⑦
  access: read           ⑧
  username: user0        ⑨
  enabled: true          ⑩
  id: 0                  ⑪
```

① snmp/v3/user/target. See SNMP v3 User Trap Target on page 178.

② The privilige password.

③ The privilege type (NONE, DES, AES).

④ The flag that indicates the privilege password is set.

⑤ The flag that indicates the authentication password is set.

⑥ The authentication password.

⑦ The authentication type.

⑧ The access type for the user.

⑨ The user entry's user name.

⑩ The enable control for the user.

⑪ The auto-generated user entry id.

▼ **API:** *PUT /api/conf/snmp/v3/user*

```
PUT http://<device-ip>/api/conf/snmp/v3/user HTTP/1.1
Content-Type: application/json

{
    "enabled": true,
    "username": "user0",
    "access": "read",
    "privType": "des",
    "privPassword": "privP@ssword!123",
    "authType": "md5",
    "authPassword": "authP@ssword!123"
}   ①
```

① Command to add a user entry. Required fields: enabled, username, access, privType, authType.

▼ API (cmd): *api/conf/snmp/v3/user/ : add*

api/conf/snmp/v3/user:

```
{
  "cmd": "add",
  "data": {
      "enabled": true,
      "username": "user0",
      "access": "read",
      "privType": "des",
      "privPassword": "privP@ssword!123",
      "authType": "md5",
      "authPassword": "authP@ssword!123"
     }
} ①
```

① Command to add a user entry. Required fields: enabled, username, access, privType, authType.

▼ CLI: add conf snmp v3 user

admin> add conf snmp v3 user = ARGS

```
add conf snmp v3 user = {"enabled": true, "username": "user0", "access": "read",
"privType": "des", "privPassword": "privP@ssword!123", "authType": "md5",
"authPassword": "authP@ssword!123"} ①
```

① Command to add a user entry. Required fields: enabled, username, access, privType, authType.

▼ API: *DELETE /api/conf/snmp/v3/user/<id>*

```
DELETE http://<device-ip>/api/conf/snmp/v3/user/<id> HTTP/1.1
```

Removes the SNMP v3 user entry identified by <id> in the request path.

▼ API (cmd): *api/conf/snmp/v3/user/<id> : delete*

api/conf/snmp/v3/user/<id>:

```
{
  "cmd": "delete"
} ①
```

① Command to delete user entry.

▼ CLI: *delete conf snmp v3 user <id>*

admin> delete conf snmp v3 user <id>

```
delete conf snmp v3 user 0 ①
```

① Command to delete user entry.

The user fields are settable except "id" and "authPasswordSet".

Example using the "username" field:

▼ API: *POST /api/conf/snmp/v3/user/<id>*

```
POST http://<device-ip>/api/conf/snmp/v3/user/<id> HTTP/1.1
Content-Type: application/json

{
    "username": "user0"
}   ①
```

① Command to set the user name field.

▼ API (cmd): *api/conf/snmp/v3/user/<id> : set*

api/conf/snmp/v3/user/<id>:

```
{
  "cmd": "set",
  "data": {
  "username": "user0"
  }
} ①
```

① Command to set the engine id type field.

▼ CLI: *set conf snmp v3 user <id> username*

admin> set conf snmp v3 user <id> username = ARGS

```
set conf snmp v3 user 0 username = user0 ①
```

① Command to set the engine id type field.

SNMP v3 User Trap Target

The SNMP v3 user target configuration.

▼ API: *GET /api/conf/snmp/v3/user/<id>/target*

```
GET http://<device-ip>/api/conf/snmp/v3/user/<id>/target HTTP/1.1

[
    {
        "target": "126.4.212.100", ①
        "port": 162,               ②
        "id": "0"                  ③
    }
]
```

① The trap target address.

② The trap target port.

③ The auto-generated trap target id.

▼ API (cmd): *api/conf/snmp/v3/user/<id>/target*

api/conf/snmp/v3/user/0/target: get

```
[
  {
    "target": "126.4.212.100", ①
    "port": 162,               ②
    "id": "0"                  ③
  }
]
```

① The trap target address.

② The trap target port.

③ The auto-generated trap target id.

▼ CLI: *get conf snmp v3 user <id> target*

user> get conf snmp v3 user 0 target

```
- target: 126.4.212.100 ①
  port: 162             ②
  id: 0                 ③
```

① The trap target address.

② The trap target port.

③ The auto-generated trap target id.

▼ API: *PUT /api/conf/snmp/v3/user/<userID>/target*

```
PUT http://<device-ip>/api/conf/snmp/v3/user/<userID>/target HTTP/1.1
Content-Type: application/json

{
    "target": "126.4.212.100",
    "port": 162
}   ①
```

① Command to add a SNMP v3 trap target entry.

▼ API (cmd): *api/conf/snmp/v3/user/<userID>/target : add*

api/conf/snmp/v3/user/<userID>/target:

```
{
  "cmd": "add",
  "data": {
     "target": "126.4.212.100",
     "port": 162
   }
} ①
```

① Command to add a SNMP v3 trap target entry.

▼ CLI: *add conf snmp v3 user <userID> target*

admin> add conf snmp v3 user <userID> target = ARGS

```
add conf snmp v3 user 0 target = {"target": "126.4.212.100", "port": 162} ①
```

① Command to add a SNMP v3 trap target entry.

▼ API: *DELETE /api/conf/snmp/v3/user/<userID>/target/<id>*

```
DELETE http://<device-ip>/api/conf/snmp/v3/user/<userID>/target/<id> HTTP/1.1
```

▼ API (cmd): *api/conf/snmp/v3/user/<userID>/target/<id> : delete*

api/conf/snmp/v3/user/<userID>/target/<id>:

```
{
"cmd": "delete"
} ①
```

① Command to delete trap target entry.

▼ CLI: *delete conf snmp v3 user <userID> target <id>*

admin> delete conf snmp v3 user <userID> target <id>

```
delete conf snmp v3 user 0 target 0 ①
```

① Command to delete trap target entry.

The SNMP v3 trap target fields are settable except "id".

Example using the "target" field:

▼ API: *POST /api/conf/snmp/v3/user/<userID>/target/<id>*

```
POST http://<device-ip>/api/conf/snmp/v3/user/<userID>/target/<id> HTTP/1.1
Content-Type: application/json

{
    "target": "126.4.212.100"
}   ①
```

① Command to set the trap target field.

▼ API (cmd): *api/conf/snmp/v3/user/<userID>/target/<id> : set*

api/conf/snmp/v3/user/<userID>/target/<id>:

```
{
  "cmd": "set",
  "data": {
    "target": "126.4.212.100"
  }
} ①
```

① Command to set the trap target field.

▼ CLI: *set conf snmp v3 user <userID> target <id>*

admin> set conf snmp v3 user <userID> target <id> target = ARGS

```
set conf snmp v3 user 0 target 0 target = 126.4.212.100 ①
```

① Command to set the engine id type field.

Send test trap to a configured SNMP v3 target.

▼ API: *api/conf/snmp/v3/user/<userID>/target/<id> : sendTest (Admin)*

api/conf/snmp/v3/user/<userID>/target/<id>:

```
{
  "cmd": "sendTest"
} ①
```

① Command to test a configured SNMP v3 trap target.

▼ CLI: *sendTest conf snmp v3 user <userID> target <id>*

admin> sendTest conf snmp v3 user <userID> target <id>

```
sendTest conf snmp v3 user 0 target 0 ①
```

① Command to test a configured SNMP v3 trap target.

SNMP v3 Test Traps

Send test trap to all configured SNMP v3 targets.

▼ API: *api/conf/snmp/v3/generateTestTraps : sendTest (Admin)*

api/conf/snmp/v3/generateTestTraps:

```
{
  "cmd": "sendTest"
} ①
```

① Command to test all configured SNMP v3 trap targets.

▼ CLI: sendTest conf snmp v3 generateTestTraps

admin> sendTest conf snmp v3 generateTestTraps

```
sendTest conf snmp v3 generateTestTraps ①
```

① Command to test all configured SNMP v3 trap targets.

2.4.14 SSH

The SSH Configuration.

▼ API: *GET /api/conf/ssh*

```
GET http://<device-ip>/api/conf/ssh HTTP/1.1

{
    "enabled": false, ①
    "port": 22        ②
}
```

① The control to enable SSH connections.

② The SSH server port.

▼ API (cmd): *api/conf/ssh*

api/conf/ssh: get

```
{
    "enabled": false, ①
    "port": 22        ②
}
```

① The control to enable SSH connections.

② The SSH server port.

▼ CLI: get conf ssh

user> get conf ssh

```
enabled: false ①
port: 22       ②
```

① The control to enable SSH connections.

② The SSH server port.

The ssh fields are settable.

Example using the "enabled" field:

▼ **API:** *POST /api/conf/ssh*

```
POST http://<device-ip>/api/conf/ssh HTTP/1.1
Content-Type: application/json

{
    "enabled": true
}   ①
```

① Command to enable/disable ssh.

▼ **API (cmd):** *api/conf/ssh : set*

api/conf/ssh:

```
{
    "cmd": "set",
    "data": {
        "enabled": true
    }
} ①
```

① Command to enable/disable ssh.

▼ **CLI:** set conf ssh enabled

admin> set conf ssh enabled = ARGS

```
set conf ssh enabled = true ①
```

① Command to enable/disable ssh.

The reset command regenerates SSH keys on a server — creating new cryptographic key pairs used for secure authentication.

▼ **API (cmd):** *api/conf/ssh : reset*

api/conf/ssh:

```
{
    "cmd": "reset",
    "data": {
        "target": "key"
    }
} ①
```

① Command to reset ssh.

▼ **CLI:** reset conf ssh

admin> reset conf ssh = ARGS

```
reset conf ssh = { target: key } ①
```

① Command to reset ssh.

2.4.15 System

The system configuration.

▼ **API:** *GET /api/conf/system*

```
GET http://<device-ip>/api/conf/system HTTP/1.1

{
    "contact": "",                ①
    "description": "",            ②
    "label": "",                  ③
    "location": "",               ④
    "factoryAccessEnabled": false ⑤
}
```

① The system contact information.

② The system description information.

③ The system name information.

④ The system location information.

⑤ The factory access enable control.

▼ **API (cmd):** *api/conf/system*

api/conf/system: get

```
{
    "contact": "",                ①
    "description": "",            ②
    "label": "",                  ③
    "location": "",               ④
    "factoryAccessEnabled": false ⑤
}
```

① The system contact information.

② The system description information.

③ The system name information.

④ The system location information.

⑤ The factory access enable control.

▼ **CLI:** get conf system

user> get conf system

```
contact: ""                 ①
description: ""             ②
label: ""                   ③
location: ""                ④
factoryAccessEnabled: false ⑤
```

① The system contact information.

② The system description information.

③ The system name information.

④ The system location information.

⑤ The factory access enable control.

The system fields are settable.

Example using the "location" field:

▼ **API:** *POST /api/conf/system*

```
POST http://<device-ip>/api/conf/system HTTP/1.1
Content-Type: application/json

{
    "location": "Delaware, OH"
}   ①
```

① Command to set system location information.

▼ **API (cmd):** *api/conf/system : set*

api/conf/system:

```
{
    "cmd": "set",
    "data": {
        "location": "Delaware, OH"
    }
} ①
```

① Command to set system location information.

▼ **CLI:** set conf system location

admin> set conf system location = ARGS

```
set conf system location = "Delaware, OH" ①
```

① Command to set system location information.

2.4.16 System Resource

The System Resource configuration.

▼ **API:** *GET /api/conf/systemResource*

```
GET http://<device-ip>/api/conf/systemResource HTTP/1.1

{
    "timeRangeCollection": "5m"     ①
}
```

① The system resource time range collection.

▼ **API (cmd):** *api/conf/systemResource*

api/conf/systemResource: get

```
{
    "timeRangeCollection": "5m"     ①
}
```

① The system resource time range collection.

▼ **CLI:** get conf systemResource

user> get conf systemResource

```
timeRangeCollection: 5m     ①
```

① The system resource time range collection.

2.4.17 Time

The time configuration. For a list of valid time zone entries, see Time Zone on page 325.

▼ API: *GET /api/conf/time*

```
GET http://<device-ip>/api/conf/time HTTP/1.1

{
    "source": "ntp",                    ①
    "mode": "automatic",                ②
    "datetime": "2024-01-12 07:32:57",  ③
    "zone": "UTC",                      ④
    "syncManagedDeviceEnabled": "false" ⑤
    "ntpServer1": "0.pool.ntp.org",     ⑥
    "ntpServer2": "1.pool.ntp.org"      ⑦
}
```

① The time source that last set the system time.

② The control that allows automatic setting of time, or manual setting of time.

③ The control that both reads the current time and is used to set the time manually.

④ The time zone setting.

⑤ The control that allows automatic time writes to the managed device.

⑥ The IP Address/DNS name of an NTP server.

⑦ The IP Address/DNS name of an NTP server.

▼ API (cmd): *api/conf/time*

api/conf/time: get

```
{
    "source": "ntp",                    ①
    "mode": "automatic",                ②
    "datetime": "2024-01-12 07:32:57",  ③
    "zone": "UTC",                      ④
    "syncManagedDeviceEnabled": "false" ⑤
    "ntpServer1": "0.pool.ntp.org",     ⑥
    "ntpServer2": "1.pool.ntp.org"      ⑦
}
```

① The time source that last set the system time.

② The control that allows automatic setting of time, or manual setting of time.

③ The control that both reads the current time and is used to set the time manually.

④ The time zone setting.

⑤ The control that allows automatic time writes to the managed device.

⑥ The IP Address/DNS name of an NTP server.

⑦ The IP Address/DNS name of an NTP server.

▼ **CLI:** get conf time

user> get conf time

```
source: ntp                      ①
mode: automatic                  ②
datetime: 2024-01-12 07:32:57    ③
zone: UTC                        ④
syncManagedDeviceEnabled: false  ⑤
ntpServer1: 0.pool.ntp.org       ⑥
ntpServer2: 1.pool.ntp.org       ⑦
```

① The time source that last set the system time.

② The control that allows automatic setting of time, or manual setting of time.

③ The control that both reads the current time and is used to set the time manually.

④ The time zone setting.

⑤ The control that allows automatic time writes to the managed device.

⑥ The IP Address/DNS name of an NTP server.

⑦ The IP Address/DNS name of an NTP server.

The time fields are settable, except the "source" field which is read-only.

Example using "syncManagedDeviceEnabled" field:

▼ **API:** *POST /api/conf/time*

```
POST http://<device-ip>/api/conf/time HTTP/1.1
Content-Type: application/json

{
    "syncManagedDeviceEnabled": true
}    ①
```

① Command to set the control that allows automatic time writes to the managed device.

▼ **API (cmd):** *api/conf/time : set*

api/conf/time:

```
{
    "cmd": "set",
    "data": {
        "syncManagedDeviceEnabled": true
    }
} ①
```

① Command to set the control that allows automatic time writes to the managed device.

▼ **CLI:** set conf time syncManagedDeviceEnabled

admin> set conf time syncManagedDeviceEnabled = ARGS

```
set conf time syncManagedDeviceEnabled = true ①
```

① Command to set the control that allows automatic time writes to the managed device.

2.4.18 USB

The USB configuration and list of connected devices.

▼ **API:** *GET /api/conf/usb*

```
GET http://<device-ip>/api/conf/usb HTTP/1.1

{
    "enabled": false,  ①
    "dev": []          ②
}
```

① The enable control for the front-panel USB port.

② USB Devices on page 191.

▼ **API (cmd):** *api/conf/usb*

api/conf/usb: get

```
{
    "enabled": false, ①
    "dev": []         ②
}
```

① The enable control for the front-panel USB port.

② USB Devices on page 191.

▼ **CLI:** get conf usb

user> get conf usb

```
enabled: false  ①
dev:            ②
    ...
```

① The enable control for the front-panel USB port.

② USB Devices on page 191.

▼ API: *POST /api/conf/usb*

```
POST http://<device-ip>/api/conf/usb HTTP/1.1
Content-Type: application/json

{
    "enabled": true
}   ①
```

① Command to enable/disable USB.

▼ API (cmd): *api/conf/usb: set*

api/conf/usb:

```
{
    "cmd": "set",
    "data": {
        "enabled": true
    }
} ①
```

① Command to enable/disable USB.

▼ CLI: set conf usb enabled

admin> set conf usb enabled = ARGS

```
set conf usb enabled = true ①
```

① Command to enable/disable USB.

USB Devices

The list of connected USB devices.

▼ API: *GET /api/conf/usb/dev*

```
GET http://<device-ip>/api/conf/usb/dev HTTP/1.1

[
    {
        "id": "0",            ①
        "vendorId": "",       ②
        "productId": "",      ③
        "bcdDevice": "",      ④
        "manufacturer": "",   ⑤
        "product": "",        ⑥
        "serial": "",         ⑦
        "conf": {}            ⑧
    }
]
```

① An auto-generated id for the device entry.

② The device vendor id.

③ The product identifier.

④ The device release or version number.

⑤ The name of the device manufacturer.

⑥ The product description.

⑦ The product serial number.

⑧ USB Device Configuration on page 193.

▼ API (cmd): *api/conf/usb/dev*

api/conf/usb/dev: get

```
[
  {
    "id": "0",          ①
    "vendorId": "",     ②
    "productId": "",    ③
    "bcdDevice": "",    ④
    "manufacturer": "", ⑤
    "product": "",      ⑥
    "serial": "",       ⑦
    "conf": {}          ⑧
  }
]
```

① An auto-generated id for the device entry.

② The device vendor id.

③ The product identifier.

④ The device release or version number.

⑤ The name of the device manufacturer.

⑥ The product description.

⑦ The product serial number.

⑧ USB Device Configuration on the facing page.

▼ **CLI:** get conf usb dev

user> get conf usb dev

```
-  id: 0          ①
   vendorId:      ②
   productId:     ③
   bcdDevice:     ④
   manufacturer:  ⑤
   product:       ⑥
   serial:        ⑦
   conf:          ⑧
    ...
```

① An auto-generated id for the device entry.

② The device vendor id.

③ The product identifier.

④ The device release or version number.

⑤ The name of the device manufacturer.

⑥ The product description.

⑦ The product serial number.

⑧ USB Device Configuration on the facing page.

USB Device Configuration

The configuration settings for the USB device. All are read only.

▼ API: *GET /api/conf/usb/dev/<id>/conf*

```
GET http://<device-ip>/api/conf/usb/dev/<id>/conf HTTP/1.1

{
    "maxPower": "",         ①
    "selfPowered": false,   ②
    "remoteWakeup": false   ③
}
```

① The maximum operating current specified as the number of 2mA units.
Example: maxPower=100 ⇒ means maxCurrent is 200mA.

② This indicates whether the USB device provides its own power.

③ This indicates whether the USB device supports performing remote wakeup.

▼ API (cmd): *api/conf/usb/dev/<id>/conf*

api/conf/usb/dev/<id>/conf: get

```
{
    "maxPower": "",        ①
    "selfPowered": false,  ②
    "remoteWakeup": false, ③
}
```

① The maximum operating current specified as the number of 2mA units.
Example: maxPower=100 ⇒ means maxCurrent is 200mA.

② This indicates whether the USB device provides its own power.

③ This indicates whether the USB device supports performing remote wakeup.

▼ CLI: *get conf usb dev <id> conf*

user> get conf usb dev <id> conf

```
maxPower:          ①
selfPowered: false  ②
remoteWakeup: false ③
```

① The maximum operating current specified as the number of 2mA units.
Example: maxPower=100 ⇒ means maxCurrent is 200mA.

② This indicates whether the USB device provides its own power.

③ This indicates whether the USB device supports performing remote wakeup.

2.4.19 Velocity

The velocity protocol configuration.

▼ **API:** *GET /api/conf/velocity*

```
GET http://<device-ip>/api/conf/velocity HTTP/1.1

{
    "managedDevice": {}, ①
    "ethernet": {},      ②
    "serial": {}         ③
}
```

① Velocity Managed Device. See Velocity Managed Device on the facing page.

② Velocity Managed Device Ethernet. See Velocity Managed Device Ethernet on page 197.

③ Velocity Managed Device Serial. See Velocity Managed Device Serial on page 199.

▼ **API (cmd):** *api/conf/velocity*

api/conf/velocity: get

```
{
    "managedDevice": {}, ①
    "ethernet": {},      ②
    "serial": {}         ③
}
```

① Velocity Managed Device. See Velocity Managed Device on the facing page.

② Velocity Managed Device Ethernet. See Velocity Managed Device Ethernet on page 197.

③ Velocity Managed Device Serial. See Velocity Managed Device Serial on page 199.

▼ **CLI:** get conf velocity

user> get conf velocity

```
managedDevice: ①
    ...
ethernet:      ②
    ...
serial:        ③
    ...
```

① Velocity Managed Device. See Velocity Managed Device on the facing page.

② Velocity Managed Device Ethernet. See Velocity Managed Device Ethernet on page 197.

③ Velocity Managed Device Serial. See Velocity Managed Device Serial on page 199.

Velocity Managed Device

The velocity managed device configuration.

▼ API: *GET /api/conf/velocity/managedDevice*

```
GET http://<device-ip>/api/conf/velocity/managedDevice HTTP/1.1

{
    "lanType": "autodetect-internal",  ①
    "connectStatus": "unsupported",    ②
    "psi": "0.0"                       ③
}
```

① The interface type used to connect to the managed device.

② The connection status.

③ The product sequence ID.

▼ API (cmd): *api/conf/velocity/managedDevice*

api/conf/velocity/managedDevice: get

```
{
    "lanType": "autodetect-internal", ①
    "connectStatus": "unsupported",   ②
    "psi": "0.0"                      ③
}
```

① The interface type used to connect to the managed device.

② The connection status.

③ The product sequence ID.

▼ CLI: get conf velocity managedDevice

user> get conf velocity managedDevice

```
lanType: autodetect-internal ①
connectStatus: unsupported   ②
psi: 0.0                     ③
```

① The interface type used to connect to the managed device.

② The connection status.

③ The product sequence ID.

The lanType field is settable.

▼ **API:** *POST /api/conf/velocity/managedDevice*

```
POST http://<device-ip>/api/conf/velocity/managedDevice HTTP/1.1
Content-Type: application/json

{
    "lanType": "autodetect-internal"
}
```

① Command to set the interface type.

▼ **API (cmd):** *api/conf/velocity/managedDevice : set*

api/conf/velocity/managedDevice:

```
{
  "cmd": "set",
  "data": {
     "lanType": "autodetect-internal"
  }
} ①
```

① Command to set the interface type.

▼ **CLI:** set conf velocity managedDevice lanType

admin> set conf velocity managedDevice lanType = ARGS

```
set conf velocity managedDevice lanType = "autodetect-internal" ①
```

① Command to set the interface type.

Velocity Managed Device Ethernet

The velocity managed device ethernet configuration.

▼ API: *GET /api/conf/velocity/ethernet*

```
GET http://<device-ip>/api/conf/velocity/ethernet HTTP/1.1

{
    "enabled": false,       ①
    "ipAddress": "",        ②
    "port": 47808,          ③
    "networkNumber": 1000   ④
}
```

① The enable control for Velocity/IP. Use this only when connecting to a Velocity/IP capable device, or when using Velocity/IP to read device data.

② The IP address of the Velocity/IP capable device.

③ The TCP Port number used to connect to the Velocity/IP capable device.

④ The Velocity network number.

▼ API (cmd): *api/conf/velocity/ethernet*

api/conf/velocity/ethernet: get

```
{
    "enabled": false,     ①
    "ipAddress": "",      ②
    "port": 47808,        ③
    "networkNumber": 1000 ④
}
```

① The enable control for Velocity/IP. Use this only when connecting to a Velocity/IP capable device, or when using Velocity/IP to read device data.

② The IP address of the Velocity/IP capable device.

③ The TCP Port number used to connect to the Velocity/IP capable device.

④ The Velocity network number.

▼ CLI: get conf velocity ethernet

user> get conf velocity ethernet

```
enabled: false      ①
ipAddress: ""       ②
port: 47808         ③
networkNumber: 1000 ④
```

① The enable control for Velocity/IP. Use this only when connecting to a Velocity/IP capable device, or when using Velocity/IP to read device data.

② The IP address of the Velocity/IP capable device.

③ The TCP Port number used to connect to the Velocity/IP capable device.

④ The Velocity network number.

The ethernet fields are settable.

Example using the "enabled" field:

▼ API: *POST /api/conf/velocity/ethernet*

```
POST http://<device-ip>/api/conf/velocity/ethernet HTTP/1.1
Content-Type: application/json

{
    "enabled": false
}   ①
```

① Command to enable the ethernet interface.

▼ API (cmd): *api/conf/velocity/ethernet : set*

api/conf/velocity/ethernet:

```
{
  "cmd": "set",
  "data": {
    "enabled": false
  }
} ①
```

① Command to enable the ethernet interface.

▼ CLI: set conf velocity ethernet enabled

admin> set conf velocity ethernet enabled = ARGS

```
set conf velocity ethernet enabled = false ①
```

① Command to enable the ethernet interface.

Velocity Managed Device Serial

The velocity managed device serial configuration.

▼ API: *GET /api/conf/velocity/serial*

```
GET http://<device-ip>/api/conf/velocity/serial HTTP/1.1

{
    "baudRate": "38400",                    ①
    "maxAddress": 127,                      ②
    "networkNumber": 1001,                  ③
    "nodeId": 127,                          ④
    "nodeIdAssignmentMethod": "automatic"   ⑤
}
```

① The serial baud rate.

② The maximum node address.

③ The Velocity master network number.

④ The Velocity master node id.

⑤ The Velocity ID node assignment method. Automatic assignement is based upon the card slot-id.

▼ API (cmd): *api/conf/velocity/serial*

api/conf/velocity/serial: get

```
{
    "baudRate": "38400",                    ①
    "maxAddress": 127,                      ②
    "networkNumber": 1001,                  ③
    "nodeId": 127,                          ④
    "nodeIDAssignmentMethod": "automatic"   ⑤
}
```

① The serial baud rate.

② The maximum node address.

③ The Velocity master network number.

④ The Velocity master node id.

⑤ The Velocity ID node assignment method. Automatic assignement is based upon the card slot-id.

▼ **CLI:** get conf velocity serial

user> get conf velocity serial

```
baudRate: 38400                    ①
maxAddress: 127                    ②
networkNumber: 1001                ③
nodeId: 127                        ④
nodeIDAssignmentMethod: automatic  ⑤
```

① The serial baud rate.

② The maximum node address.

③ The Velocity master network number.

④ The Velocity master node id.

⑤ The Velocity ID node assignment method. Automatic assignement is based upon the card slot-id.

The velocity serial fields are settable.

Example using the "baudRate" field:

▼ **API:** *POST /api/conf/velocity/serial*

```
POST http://<device-ip>/api/conf/velocity/serial HTTP/1.1
Content-Type: application/json

{
    "baudRate": 38400
}   ①
```

① Command to set serial baud rate.

NOTE: Allowable values are 38400, 19200, and 9600.

▼ **API (cmd):** *api/conf/velocity/serial : set*

api/conf/velocity/serial:

```
{
    "cmd": "set",
    "data": {
        "baudRate": 38400
    }
} ①
```

① Command to set serial baud rate.

NOTE: Allowable values are 38400, 19200, and 9600.

▼ **CLI:** set conf velocity serial baudRate

admin> set conf velocity serial baudRate = ARGS

```
set conf velocity serial baudRate = 38400 ①
```

① Command to set serial baud rate.

NOTE: Allowable values are 38400, 19200, and 9600.

2.4.20 Web Server

The Web Server configuration.

▼ **API:** *GET /api/conf/webserver*

```
GET http://<device-ip>/api/conf/webserver HTTP/1.1

{
    "redirectEnabled": false ①
    "httpsPort": 443,        ②
    "httpsEnabled": false,   ③
    "httpEnabled": true,     ④
    "httpPort": 80,          ⑤
    "sessionIdleTimeout": 5, ⑥
    "genCertificate": {},    ⑦
    "trustedAccess": []      ⑧
}
```

① The redirectEnabled setting.

② The HTTPS Port setting.

③ The HTTPS Enable setting.

④ The HTTP Enable setting.

⑤ The HTTP Port setting.

⑥ The session idle timeout setting.

⑦ Generate Self-Signed Certificate. See Generate Self-Signed Certificate on page 204.

⑧ Trusted Access. See Trusted Access on page 206.

▼ **API (cmd):** *api/conf/webserver*

api/conf/webserver: get

```
{
    "redirectEnabled": false ①
    "httpsPort": 443,        ②
    "httpsEnabled": false,   ③
    "httpEnabled": true,     ④
    "httpPort": 80,          ⑤
    "sessionIdleTimeout": 5, ⑥
    "genCertificate": {},    ⑦
    "trustedAccess": []      ⑧
}
```

① The redirectEnabled setting.

② The HTTPS Port setting.

③ The HTTPS Enable setting.

④ The HTTP Enable setting.

⑤ The HTTP Port setting.

⑥ The session idle timeout setting.

⑦ Generate Self-Signed Certificate. See Generate Self-Signed Certificate on page 204.

⑧ Trusted Access. See Trusted Access on page 206.

▼ **CLI:** get conf webserver

user> get conf webserver

```
redirectEnabled: false ①
httpsPort: 443         ②
httpsEnabled: false    ③
httpEnabled: true      ④
httpPort: 80           ⑤
sessionIdleTimeout: 5  ⑥
genCertificate:        ⑦
    ...
trustedAccess:         ⑧
    ...
```

① The redirectEnabled setting.

② The HTTPS Port setting.

③ The HTTPS Enable setting.

④ The HTTP Enable setting.

⑤ The HTTP Port setting.

⑥ The session idle timeout setting.

⑦ Generate Self-Signed Certificate. See Generate Self-Signed Certificate on the next page.

⑧ Trusted Access. See Trusted Access on page 206.

The httpEnabled, httpsEnabled, httpPort, httpsPort, redirectEnabled and sessionIdleTimeout fields are settable.

Example using the "httpEnabled" field:

▼ **API:** *POST /api/conf/webserver*

```
POST http://<device-ip>/api/conf/webserver HTTP/1.1
Content-Type: application/json

{
    "httpEnabled": true
}    ①
```

① Command to enable/disable HTTP Enable setting.

▼ **API (cmd):** *api/conf/webserver : set*

api/conf/webserver:

```
{
    "cmd": "set",
    "data": {
        "httpEnabled": true
    }
} ①
```

① Command to enable/disable HTTP Enable setting.

▼ CLI: set conf webserver httpEnabled

admin> set conf webserver httpEnabled = ARGS

```
set conf webserver httpEnabled = true ①
```

① Command to enable/disable HTTP Enable setting.

Generate Self-Signed Certificate

The configuration used when generating the self-signed certificate.

▼ API: *GET /api/conf/webserver/genCertificate*

```
GET http://<device-ip>/api/conf/webserver/genCertificate HTTP/1.1

{
    "organization": "",            ①
    "organizationUnit": "",        ②
    "emailAddress": "",            ③
    "countryCode": "",             ④
    "stateProvince": "",           ⑤
    "cityLocality": "",            ⑥
    "commonName": "",              ⑦
    "ssMode": "Use Default Values" ⑧
}
```

① The organization used in a generated certificate.

② The organization unit used in a generated certificate.

③ The email address used in a generated certificate.

④ The country code used in a generated certificate.

⑤ The state used in a generated certificate.

⑥ The city used in a generated certificate.

⑦ The common name used in a generated certificate.

⑧ The mode by which a certficate will be generated. See Web Server Self-Signed Certificate Generation on page 210.

▼ API (cmd): *api/conf/webserver/genCertificate*

api/conf/webserver/genCertificate: get

```
{
    "organization": "",            ①
    "organizationUnit": "",        ②
    "emailAddress": "",            ③
    "countryCode": "",             ④
    "stateProvince": "",           ⑤
    "cityLocality": "",            ⑥
    "commonName": "",              ⑦
    "ssMode": "Use Default Values" ⑧
}
```

① The organization used in a generated certificate.

② The organization unit used in a generated certificate.

③ The email address used in a generated certificate.

④ The country code used in a generated certificate.

⑤ The state used in a generated certificate.

⑥ The city used in a generated certificate.

⑦ The common name used in a generated certificate.

⑧ The mode by which a certficate will be generated. See Web Server Self-Signed Certificate Generation on page 210.

▼ **CLI:** get conf webserver genCertificate

user> get conf webserver genCertificate

```
organization:              ①
organizationUnit:          ②
emailAddress:              ③
countryCode:               ④
stateProvince:             ⑤
cityLocality:              ⑥
commonName:                ⑦
ssMode: Use Default Values ⑧
```

① The organization used in a generated certificate.

② The organization unit used in a generated certificate.

③ The email address used in a generated certificate.

④ The country code used in a generated certificate.

⑤ The state used in a generated certificate.

⑥ The city used in a generated certificate.

⑦ The common name used in a generated certificate.

⑧ The mode by which a certficate will be generated. See Web Server Self-Signed Certificate Generation on page 210.

The genCertificate fields are settable.

Example using the "organization" field:

▼ **API:** *POST /api/conf/webserver/genCertificate*

```
POST http://<device-ip>/api/conf/webserver/genCertificate HTTP/1.1
Content-Type: application/json

{
    "organization": "Vertiv"
}   ①
```

① Command to set the organization used in a generated certificate.

▼ API (cmd): *api/conf/webserver/genCertificate : set*

api/conf/webserver/genCertificate:

```
{
    "cmd": "set",
    "data": {
        "organization": "Vertiv"
    }
} ①
```

① Command to set the organization used in a generated certificate.

▼ CLI: set conf webserver genCertificate organization

admin> set conf webserver genCertificate organization = ARGS

```
set conf webserver genCertificate organization = "Vertiv" ①
```

① Command to set the organization used in a generated certificate.

Trusted Access

The Web Server trusted access configuration.

▼ API: *GET /api/conf/webserver/trustedAccess*

```
GET http://<device-ip>/api/conf/webserver/trustedAccess HTTP/1.1

[
    {
        "id": "test",             ①
        "address": "10.20.30.40", ②
        "prefix": 24              ③
    }
]
```

① The trusted access entry id.

② The trusted access host/network address.

③ The network prefix.

▼ API (cmd): *api/conf/webserver/trustedAccess*

api/conf/webserver/trustedAccess: get

```
[
    {
        "id": "test",             ①
        "address": "10.20.30.40", ②
        "prefix": 24              ③
    }
]
```

① The trusted access entry id.

② The trusted access host/network address.

③ The network prefix.

▼ CLI: get conf webserver trustedAccess

user> get conf webserver trustedAccess

```
- id: test             ①
  address: 10.20.30.40 ②
  prefix: 24           ③
```

① The trusted access entry id.

② The trusted access host/network address.

③ The network prefix.

▼ API: *PUT /api/conf/webserver/trustedAccess*

```
PUT http://<device-ip>/api/conf/webserver/trustedAccess HTTP/1.1
Content-Type: application/json

{
    "prefix" : 24,
    "address" : "10.20.30.40",
    "id": "test"
}    ①
```

① Command to add a trustedAccess entry.

NOTE: "id" field is auto generated if not set.

▼ API (cmd): *api/conf/webserver/trustedAccess : add*

api/conf/webserver/trustedAccess:

```
{
    "cmd": "add",
    "data": {
        "prefix" : 24,
        "address" : "10.20.30.40",
        "id": "test"
    }
} ①
```

① Command to add a trustedAccess entry.

NOTE: "id" field is auto generated if not set.

▼ CLI: add conf webserver trustedAccess

user> add conf webserver trustedAccess = ARGS

```
add conf webserver trustedAccess = {"prefix": 24, "address": "10.20.30.40", "id":
"test"} ①
```

① Command to add a trustedAccess entry.

NOTE: "id" field is auto generated if not set.

▼ API: *DELETE /api/conf/webserver/trustedAccess/<id>*

```
DELETE http://<device-ip>/api/conf/webserver/trustedAccess/<id> HTTP/1.1
```

▼ API (cmd): *api/conf/webserver/trustedAccess/<id> : delete*

api/conf/webserver/trustedAccess/<id>:

```
{
    "cmd": "delete"
} ①
```

① Command to delete trustedAccess entry.

▼ CLI: *delete conf webserver trustedAccess <id>*

admin> delete conf webserver trustedAccess <id>

```
delete conf webserver trustedAccess test ①
```

① Command to delete trustedAccess entry.

The webserver trustedAccess fields are settable.

▼ API: *POST /api/conf/webserver/trustedAccess/<id>*

```
POST http://<device-ip>/api/conf/webserver/trustedAccess/<id> HTTP/1.1
Content-Type: application/json

{
    "address": "126.4.212.2"
}   ①
```

① Command to set the trustedAccess address field.

▼ API (cmd): *api/conf/webserver/trustedAccess/<id> : set*

api/conf/webserver/trustedAccess/<id>:

```
{
    "cmd": "set",
    "data": {
        "address": "126.4.212.2"
    }
} ①
```

① Command to set the trustedAccess address field.

▼ CLI: *set conf webserver trustedAccess <id>*

admin> set conf webserver trustedAccess <id> = ARGS

```
set conf webserver trustedAccess id0 = {"address": 126.4.212.2} ①
```

① Command to set the trustedAccess address field.

Web Server Self-Signed Certificate Generation

The self-signed certificate is generated when the reset operation is performed on the web server endpoint. The certificate will be generated based upon the Web Server Generate Self-Signed Certificate on page 204.

IMPORTANT! Reset command currently is not working when ssMode is set to "Use Configured Settings".

▼ API (cmd): *api/conf/webserver : reset*

api/conf/webserver:

```
{
    "cmd": "reset",
    "data": {
        "target": "genCertificate"
    }
}    ①
```

① Command to reset generated SSL certificate.

▼ CLI: reset conf webserver

admin> reset conf webserver = ARGS

```
reset conf webserver = { target: genCertificate } ①
```

① Command to reset generated SSL certificate.

2.4.21 YDN23 Server

The YDN23 server configuration settings allow the user to enable/disable the YDN23 server and configure the access, address, and baud rate parameters.

▼ API: *GET /api/conf/ydn23/server*

```
GET http://<device-ip>/api/conf/ydn23/server HTTP/1.1

{
    "enabled": false      ①
    "access": "readOnly", ②
    "address": 1,         ③
    "dataRate": "9600",   ④
}
```

① The YDN23 server mode; enabled (true) or disabled (false).

② The allowed access for a YDN23 client.

③ Device address for the interface.

④ The communications rate (baud rate).

▼ API (cmd): *api/conf/ydn23/server*

api/conf/ydn23/server: get

```
{
    "enabled": false      ①
    "access": "readOnly", ②
    "address": 1,         ③
    "dataRate": "9600",   ④
}
```

① The YDN23 server mode; enabled (true) or disabled (false).

② The allowed access for a YDN23 client.

③ Device address for the interface.

④ The communications rate (baud rate).

▼ CLI: get conf ydn23 server

user> get conf ydn23 server

```
enabled: false     ①
access: readOnly   ②
address: 1         ③
dataRate: 9600     ④
```

① The YDN23 server mode; enabled (true) or disabled (false).

② The allowed access for a YDN23 client.

③ Device address for the interface.

④ The communications rate (baud rate).

The YDN23 Server fields are settable.

Example using the "access" field:

▼ API: *POST /api/conf/ydn23/server*

```
POST http://<device-ip>/api/conf/ydn23/server HTTP/1.1
Content-Type: application/json

{
    "access": "readOnly"
}    ①
```

① Command to set the access field.

▼ API (cmd): *api/conf/ydn23/server*

api/conf/ydn23/server:

```
{
    "cmd": "set",
    "data": {
    "access": "readOnly"
    }
}    ①
```

① Command to set the access field.

▼ CLI: set conf ydn23 server access

admin> set conf ydn23 server access = ARGS

```
set conf ydn23 server access = "readOnly" ①
```

① Command to set the access field.

2.5 System: /api/sys

The Vertiv™ Liebert® IntelliSlot™ RDU120 System API exists to provide status level system information.

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**reset, reboot, switchFirmware on these objects require admin privilege**
sys, see System on the next page				read-only, Commands: reset, reboot, switchFirmw are
alternateFirmwareDate	String	0 to String Max		
alternateFirmwareLabel	String	0 to String Max		
alternateFirmwareVersion	String	0 to String Max		
apiVersion	String	0 to String Max		
build	String	maxLength 8		
component	String	0 to String Max		
currentFirmwareDate	String	0 to String Max		
currentFirmwareLabel	String	0 to String Max		
currentFirmwareVersion	String	0 to String Max		
hostname	String	0 to String Max		
label	String	0 to String Max		
macAddr	String	0 to String Max		
model	String	0 to String Max		
modelNumber	String	0 to String Max		
name	String	0 to String Max		
oem	String	maxLength 3		
picVersion	String	0 to String Max		
platform	String	maxLength 8		
secureKeyLocked	Boolean	true, false		
secureKeyProgrammed	Boolean	true, false		
serialNumber	String	0 to String Max		
ubootVersion	String	0 to String Max		
version	String	0 to String Max		
sys/state, see System State on page 219				read-only
adminExists	Boolean	true, false		
alarmCount	Integer			
component	String	0 to String Max		

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**reset, reboot, switchFirmware on these objects require admin privilege**
dirty	Integer			
guestEnabled	Boolean	true, false		
localTime	String	0 to String Max		
systemTime	Integer			
uptime	Integer			
warnCount	Integer			
sys/state/alarm, see System Alarm State on page 221				read-only
severity	String	"", "alarm", "warning"		
state	String	"none", "inactive", "clear", "acked", "latched", "tripping", "tripped", "clearing"		

2.5.1 System

▼ API: *GET /api/sys*

```
GET http://<device-ip>/api/sys HTTP/1.1

{
    "alternateFirmwareVersion": "1.4.0",                                    ①
    "alternateFirmwareDate": "08/05/2025 09:00:41",                         ②
    "currentFirmwareVersion": "1.4.1",                                      ③
    "currentFirmwareDate": "09/09/2025 15:06:42",                           ④
    "apiVersion": "1.0.0",                                                  ⑤
    "alternateFirmwareLabel": "rdu120-1.4.0-00001-3878ef794",               ⑥
    "picVersion": "1.7",                                                    ⑦
    "name": "rdu120",                                                       ⑧
    "label": "",                                                            ⑨
    "modelNumber": "RDU120",                                                ⑩
    "oem": "VER",                                                           ⑪
    "version": "1.4.1",                                                     ⑫
    "platform": "rdu120",                                                   ⑬
    "state": {},                                                            ⑭
    "component": "ok",                                                      ⑮
    "serialNumber": "420491G10GP2025MAR030564",                             ⑯
    "model": "RDU120",                                                      ⑰
    "macAddr": "00:02:99:34:91:ae",                                         ⑱
    "secureKeyProgrammed": false,                                           ⑲
    "secureKeyLocked": false,                                               ⑳
    "hostname": "rdu120",                                                   ㉑
    "build": "00102-638a6d236",                                             ㉒
    "systemLanguage": "en",                                                 ㉓
    "currentFirmwareLabel": "rdu120-1.4.1-00102-638a6d236",                 ㉔
    "ubootVersion": "2021.04-lf_v2021.04+g628fb6ffdf(Nov 21 2024-19:53:19)" ㉕
}
```

① The alternate firmware version.

② The alternate firmware build date.

③ The current firmware version.

④ The current firmware build date.

⑤ The API version.

⑥ The alternate firmware label.

⑦ The PIC version.

⑧ The configured system name.

⑨ The configured system label.

⑩ The model number.

⑪ The OEM label.

⑫ The system version.

⑬ The platform.

⑭ System State on page 219.

⑮ The state of the component.

⑯ The serial number.

⑰ The model.

⑱ MAC address of the interface.

⑲ The secure key programmed flag.

⑳ The secure key lock flag.

㉑ The hostname of the device.

㉒ The build reference.

㉓ The language displayed on the user interface.

㉔ The current firmware label.

㉕ The U-Boot version.

▼ API (cmd): *api/sys : get*

api/sys: get

```
{
    "alternateFirmwareVersion": "1.4.0",                                    ①
    "alternateFirmwareDate": "08/05/2025 09:00:41",                         ②
    "currentFirmwareVersion": "1.4.1",                                      ③
    "currentFirmwareDate": "09/09/2025 15:06:42",                           ④
    "apiVersion": "1.0.0",                                                  ⑤
    "alternateFirmwareLabel": "rdu120-1.4.0-00001-3878ef794",               ⑥
    "picVersion": "1.7",                                                    ⑦
    "name": "rdu120",                                                       ⑧
    "label": "",                                                            ⑨
    "modelNumber": "RDU120",                                                ⑩
    "oem": "VER",                                                           ⑪
    "version": "1.4.1",                                                     ⑫
    "platform": "rdu120",                                                   ⑬
    "state": {},                                                            ⑭
    "component": "ok",                                                      ⑮
    "serialNumber": "420491G10GP2025MAR030564",                             ⑯
    "model": "RDU120",                                                      ⑰
    "macAddr": "00:02:99:34:91:ae",                                         ⑱
    "secureKeyProgrammed": false,                                           ⑲
    "secureKeyLocked": false,                                               ⑳
    "hostname": "rdu120",                                                   ㉑
    "build": "00102-638a6d236",                                             ㉒
    "systemLanguage": "en",                                                 ㉓
    "currentFirmwareLabel": "rdu120-1.4.1-00102-638a6d236",                 ㉔
    "ubootVersion": "2021.04-lf_v2021.04+g628fb6ffdf(Nov 21 2024-19:53:19)"  ㉕
}
```

① The alternate firmware version.

② The alternate firmware build date.

③ The current firmware version.

④ The current firmware build date.

⑤ The API version.

⑥ The alternate firmware label.

⑦ The PIC version.

⑧ The configured system name.

⑨ The configured system label.

⑩ The model number.

⑪ The OEM label.

⑫ The system version.

⑬ The platform.

⑭ System State on page 219.

⑮ The state of the component.

⑯ The serial number.

⑰ The model.

⑱ MAC address of the interface.

⑲ The secure key programmed flag.

⑳ The secure key lock flag.

㉑ The hostname of the device.

㉒ The build reference.

㉓ The language displayed on the user interface.

㉔ The current firmware label.

㉕ The U-Boot version.

▼ **CLI:** get sys

user> get sys

```
alternateFirmwareVersion: 1.4.0                                    ①
alternateFirmwareDate: 08/05/2025 09:00:41                         ②
currentFirmwareVersion: 1.4.1                                      ③
currentFirmwareDate: 09/09/2025 15:06:42                           ④
apiVersion: 1.0.0                                                  ⑤
alternateFirmwareLabel: rdu120-1.4.0-00001-3878ef794               ⑥
picVersion: 1.7                                                    ⑦
name: rdu120                                                       ⑧
label: ""                                                          ⑨
modelNumber: RDU120                                                ⑩
oem: VER                                                           ⑪
version: 1.4.1                                                     ⑫
platform: rdu120                                                   ⑬
state:                                                             ⑭
...
component: ok                                                      ⑮
serialNumber: 420491G10GP2025MAR030564                             ⑯
model: RDU120                                                      ⑰
macAddr: 00:02:99:34:91:ae                                         ⑱
secureKeyProgrammed: false                                         ⑲
secureKeyLocked: false                                             ⑳
hostname: rdu120                                                   ㉑
build: 00102-638a6d236                                             ㉒
systemLanguage: en                                                 ㉓
currentFirmwareLabel: rdu120-1.4.1-00102-638a6d236                 ㉔
ubootVersion: 2021.04-lf_v2021.04+g628fb6ffdf(Nov 21 2024-19:53:19) ㉕
```

① The alternate firmware version.

② The alternate firmware build date.

③ The current firmware version.

④ The current firmware build date.

⑤ The API version.

⑥ The alternate firmware label.

⑦ The PIC version.

⑧ The configured system name.

⑨ The configured system label.

⑩ The model number.

⑪ The OEM label.

⑫ The system version.

⑬ The platform.

⑭ System State on the facing page.

⑮ The state of the component.

⑯ The serial number.

⑰ The model.

⑱ MAC address of the interface.

⑲ The secure key programmed flag.

⑳ The secure key lock flag.

㉑ The hostname of the device.

㉒ The build reference.

㉓ The language displayed on the user interface.

㉔ The current firmware label.

㉕ The U-Boot version.

System State

The RDU120 system configuration.

▼ API: *GET /api/sys/state*

```
GET http://<device-ip>/api/sys/state HTTP/1.1

{
    "warnCount": 0,                         ①
    "uptime": 4198,                         ②
    "adminExists": true,                    ③
    "guestEnabled": true,                   ④
    "alarmCount": 0,                        ⑤
    "localTime": "2025-09-17 20:53:09",     ⑥
    "component": "ok",                      ⑦
    "dirty": 4,                             ⑧
    "systemTime": 1756241589,               ⑨
    "alarm": {}                             ⑩
}
```

① The number of active warnings.

② The system uptime.

③ The state of an administrator (true if configured, else false).

④ The guest enable flag.

⑤ The number of active alarms.

⑥ The local time.

⑦ The state of the component.

⑧ The dirty flag.

⑨ The system time.

⑩ System Alarm State on page 221.

▼ API cmd): api/sys/state : get

api/sys/state: get

```
{
    "warnCount": 0,                         ①
    "uptime": 4198,                         ②
    "adminExists": true,                    ③
    "guestEnabled": true,                   ④
    "alarmCount": 0,                        ⑤
    "localTime": "2025-09-17 20:53:09",     ⑥
    "component": "ok",                      ⑦
    "dirty": 4,                             ⑧
    "systemTime": 1756241589,               ⑨
    "alarm": {}                             ⑩
}
```

① The number of active warnings.

② The system uptime.

③ The state of an administrator (true if configured, else false).

④ The guest enable flag.

⑤ The number of active alarms.

⑥ The local time.

⑦ The state of the component.

⑧ The dirty flag.

⑨ The system time.

⑩ System Alarm State on the facing page.

▼ **CLI:** get sys state

user> get sys state

```
warnCount: 0                    ①
uptime: 4636                    ②
adminExists: true               ③
guestEnabled: true              ④
alarmCount: 0                   ⑤
localTime: 2025-09-17 20:53:09  ⑥
component: ok                   ⑦
dirty: 4                        ⑧
systemTime: 1756242026          ⑨
alarm:                          ⑩
...
```

① The number of active warnings.

② The system uptime.

③ The state of an administrator (true if configured, else false).

④ The guest enable flag.

⑤ The number of active alarms.

⑥ The local time.

⑦ The state of the component.

⑧ The dirty flag.

⑨ The system time.

⑩ System Alarm State on the facing page.

System Alarm State

The RDU120 system alarm configuration.

▼ API: *GET /api/sys/state/alarm*

```
GET http://<device-ip>/api/sys/state/alarm HTTP/1.1

{
    "severity": "",  ①
    "state": "none"  ②
}
```

① The severity of alarm.

② The system alarm state.

▼ API (cmd): *api/sys/state/alarm : get*

api/sys/state/alarm: get

```
{
    "severity": "",  ①
    "state": "none"  ②
}
```

① The severity of alarm.

② The system alarm state.

▼ CLI: get sys state alarm

user> get sys state alarm

```
"severity": "",  ①
"state": "none"  ②
```

① The severity of alarm.

② The system alarm state.

2.5.2 Commands

The reset command is used to reset the system according to the target value.

The reset command has the following target values:

fullDefaults: Resets all configuration on Configuration: /api/conf on page 55 to factory defaults. It will return success and then be followed by a short period where access to the system will be unavailable as system reboots.

partialDefaults: As the "fullDefaults" option above but does not reset User on page 36, Network on page 114, Web Server on page 201 or Remote Authentication on page 135and does not clear the logs. This will cause portions of the system to reinitialize. It will return success and then be followed by a short period where access to the system will be unavailable.

hostAccessDefaults: Resets the configuration with the User on page 1, Network on page 114, Web Server on page 201, Serial on page 151and Remote Authentication on page 135. Also, does not clear the logs. This will cause portions of the system to reinitialize.

decom: Resets for decommission. As the "fullDefaults" option it will clear configurations but also clear keys, certificates, and logs. SSH access will be disabled. Use with caution.

▼ **API:** *POST /api/sys*

```
POST http://<device-ip>/api/sys HTTP/1.1
Content-Type: application/json

{
    "cmd": "reset",
    "data": {
    "target": "fullDefaults"
    }
}    ①
```

① Command to reset system to one of the four target values (fullDefaults, partialDefaults, hostAccessDefaults, decom).

▼ **API (cmd):** *api/sys : reset (Admin)*

api/sys:

```
{
    "cmd": "reset",
    "data": {
        "target": "fullDefaults"
    }
}    ①
```

① Command to reset system to one of the four target values (fullDefaults, partialDefaults, hostAccessDefaults, decom).

▼ **CLI:** reset sys

admin> reset sys = ARGS

```
reset sys = { target: fullDefaults } ①
```

① Command to reset system to one of the four target values (fullDefaults, partialDefaults, hostAccessDefaults, decom).

The reboot command is used to reboot the system.

▼ **API:** *POST /api/sys*

```
POST http://<device-ip>/api/sys HTTP/1.1
Content-Type: application/json

{
    "cmd": "reboot"
}    ①
```

① Command to reboot the system.

▼ **API (cmd):** *api/sys : reboot (Admin)*

api/sys:

```
{
    "cmd": "reboot"
}    ①
```

① Command to reboot the system.

▼ **CLI:** reboot sys

admin> reboot sys

```
reboot sys ①
```

① Command to reboot the system.

The switchFirmware command is used to switch to the alternate firmware version.

▼ **API:** *api/sys : switchFirmware (Admin)*

api/sys:

```
{
  "cmd": "switchFirmware"
} ①
```

① Command to switch to the alternate firmware version in the system.

▼ **CLI:** switchFirmware sys

admin> switchFirmware sys

```
switchFirmware sys ①
```

① Command to switch to the alternate firmware version in the system.

2.6 Alarm: /api/alarm

Device alarm configuration.

Object		Data			Notes
	Field	Format	Range	Default	set, add, delete on these objects require control privilege
trigger/ID, see Trigger on the next page.			0 to 256 items		Commands: add, delete
	type	String	"high", "low", "status"		
	severity	String	"alarm", "warning"	"alarm"	
	invertValidTime	Boolean	true, false	false	
	latching	Boolean	true, false	false	
	selectedActions	Array (ID)	0 to 32 items (Alarm Action ID, see Action on page 234)	empty Array	
	validTime	ID	Valid Time ID, see Valid Time on page 238	empty ID	
	tripDelay	Integer	0 to 14400 seconds	0	
	path	Dev Path	Device Path, see Device: /api/dev on page 240 or measurement path, see ID/Entity/Measurement on page 254		
	threshold	Float	Float Min to Float Max	0	
	state	String	"clear", "acked", "latched", "tripped", "inactive"		Read-only
	clearDelay	Integer	0 to 14400 Seconds	0	
action/ID, see Action on page 234.			0 to 32 items		Commands: add, delete
	target	ID	Alarm Target ID, see Target on page 232		
	delay	Integer	0 to 14400 Seconds	0	
	repeat	Integer	0 to 14400 Seconds	0	
target/ID, see Target on page 232.			0 to 20 items		Read-only
	name	String	0 to String Max		Read-only
	group	String	0 to String Max		Read-only
	type	String	0 to String Max		Read-only
	enabled	Boolean	true, false		Read-only
	path	API Path	Absolute path for target resource in API		Read-only
validTime/ID, see Valid Time on page 238.			0 to 8 items		Commands: add, delete
	start	Time	00:00 to 24:00	00:00	
	stop	Time	00:00 to 24:00	24:00	
	days	Days	"-------" to "MTWTFSS"	"MTWTFSS"	

2.6.1 Trigger

Alarm triggers represent conditions that the system must monitor as well as actions to perform when something is out of bounds.

Trigger conditions are based on their "type" and "path" and are as follows:

- **status:** Requires a path to a top level device (e.g. "A70004A3BB7F45C3/") and will trip if the state field for the chosen device becomes anything other than "normal".
- **high:** Requires a path to a specific measurement (e.g. "A70004A3BB7F45C3/entity/11/measurement/1") and will trip if the latest value of that measurement is equal or greater than the "threshold". Trigger will clear if current value is less than the "threshold".
- **low:** Requires a path to a specific measurement (e.g. "A70004A3BB7F45C3/entity/11/measurement/1") and will trip if the latest value of that measurement is equal or less than the "threshold". Trigger will clear if current value is greater than the "threshold".

The "validTime" field references a Valid Time ID (see Valid Time on page 238) and is used in conjunction with "invertValidTime" to indicate the times of day when a trigger is active and inactive. When a trigger transitions from active to inactive it immediately goes to an "inactive" state, resets all timers, and cancels any pending or active actions. Once the trigger becomes active again, it immediately goes to the "clear" state and begins processing inputs with no memory of its state prior to being inactive. Leaving this field blank is equivalent to having a valid time that is always active. The "invertValidTime" makes the inactive time ranges into active ones and vice-versa. If the system clock is not set, triggers are always considered active.

The "state" field for any given trigger will have one of the following values:

- **clear:** Default state, condition being monitored is normal. Will transition to "tripped" once the trigger condition is met and the "tripDelay" expires. Transitioning into this state cancels all active or pending selected actions.
- **tripped:** Condition being monitored is outside of expected values. Upon entering this state from "clear", selected actions will be started. Will transition to "acked" if an ack command is received. Once the trigger condition returns to normal and the "clearDelay" expires, will transition to "latched" if the "latching" field is true or to "clear" otherwise. Transitioning to "clear" causes selected actions to send any applicable clear notifications.
- **acked:** Condition being monitored is outside of expected values but the user has acknowledged this. Upon entering this state, active and pending selected actions will be canceled. Will transition to "clear" once the trigger condition returns to normal and the "clearDelay" expires. Transitions to clear regardless of "latching" field.
- **latched:** Condition being monitored is normal but was previously outside of expected values and the user must acknowledge this. Will transition with no delay to "tripped" if the trigger condition is met. Will transition to "clear" if an ack command is received. Selected actions continue to take place while in this state.
- **inactive:** Current time is outside of trigger operating window. All states will transition to "inactive" once the current time falls outside the operating window and will transition to "clear" once the opposite is true. When transitioning into this state all active and pending selected actions are canceled.

Modifying any field in a trigger will cause it to reset. This means that the "state" is set to "clear", all selected pending and active actions are canceled, and all timers are reset. Deleting a trigger will cause all selected pending and active actions to be canceled and all timers to reset.

▼ API: *GET /api/alarm/trigger*

```
GET http://<device-ip>/api/alarm/trigger HTTP/1.1

{
    "17000000EEE2AA12_0_2_overAlarm": {
        "type": "high",                                          ①
        "severity": "alarm",                                     ②
        "invertValidTime": false,                                ③
        "latching": false,                                       ④
        "selectedActions": [],                                   ⑤
        "validTime": "",                                         ⑥
        "tripDelay": 0,                                          ⑦
        "path": "17000000EEE2AA12/entity/0/measurement/2",       ⑧
        "threshold": 18.0,                                       ⑨
        "state": "tripped",                                      ⑩
        "clearDelay": 0                                          ⑪
    }
}
```

① Trigger type.

② Severity of the trigger. Can be "alarm" or "warning".

③ Inverts the "validTime" selection.

④ If true, trigger will latch into the "tripped" state until the ack command is received.

⑤ Array of existing Alarm Action IDs. See Action on page 234.

⑥ Existing Valid Time ID. See Valid Time on page 238. Empty field means alarm is always active.

⑦ Delay in seconds before triggering the "tripDelay" action. If the trigger condition returns to normal before this time elapses, no actions are initiated and this timer is reset.

⑧ Path of object being monitored. Relative to api/dev. See Device: /api/dev on page 240. Requirements vary based on trigger type.

⑨ Value at which trigger will trip. Used in "high" and "low" trigger types.

⑩ Current trigger state. See above.

⑪ Number of seconds a tripped trigger condition must be false before it is considered cleared. If the trigger condition trips again before this time elapses, the trigger returns to the tripped state and this timer is reset. Once it expires, the trigger can be considered clear and any clear actions are initiated.

▼ API (cmd): *api/alarm/trigger : get*

api/alarm/trigger: get

```
{
    "17000000EEE2AA12_0_2_overAlarm": {
        "type": "high",                                          ①
        "severity": "alarm",                                     ②
        "invertValidTime": false,                                ③
        "latching": false,                                       ④
        "selectedActions": [],                                   ⑤
        "validTime": "",                                         ⑥
        "tripDelay": 0,                                          ⑦
        "path": "17000000EEE2AA12/entity/0/measurement/2",       ⑧
        "threshold": 18.0,                                       ⑨
        "state": "tripped",                                      ⑩
        "clearDelay": 0                                          ⑪
    }
}
```

① Trigger type.

② Severity of the trigger. Can be "alarm" or "warning".

③ Inverts the "validTime" selection.

④ If true, trigger will latch into the "tripped" state until the ack command is received.

⑤ Array of existing Alarm Action IDs. See Action on page 234.

⑥ Existing Valid Time ID. See Valid Time on page 238. Empty field means alarm is always active.

⑦ Delay in seconds before triggering the "tripDelay" action. If the trigger condition returns to normal before this time elapses, no actions are initiated and this timer is reset.

⑧ Path of object being monitored. Relative to api/dev. See Device: /api/dev on page 240. Requirements vary based on trigger type.

⑨ Value at which trigger will trip. Used in "high" and "low" trigger types.

⑩ Current trigger state. See above.

⑪ Number of seconds a tripped trigger condition must be false before it is considered cleared. If the trigger condition trips again before this time elapses, the trigger returns to the tripped state and this timer is reset. Once it expires, the trigger can be considered clear and any clear actions are initiated.

▼ **CLI:** get alarm trigger

user> get alarm trigger

```
17000000EEE2AA12_0_2_overAlarm:
    type: high                                          ①
    severity: alarm                                     ②
    invertValidTime: false                              ③
    latching: false                                     ④
    selectedActions: ~                                  ⑤
    validTime: ""                                       ⑥
    tripDelay: 0                                        ⑦
    path: 17000000EEE2AA12/entity/0/measurement/2       ⑧
    threshold: 18                                       ⑨
    state: tripped                                      ⑩
    clearDelay: 0                                       ⑪
```

① Trigger type.

② Severity of the trigger. Can be "alarm" or "warning".

③ Inverts the "validTime" selection.

④ If true, trigger will latch into the "tripped" state until the ack command is received.

⑤ Array of existing Alarm Action IDs. See Action on page 234.

⑥ Existing Valid Time ID. See Valid Time on page 238. Empty field means alarm is always active.

⑦ Delay in seconds before triggering the "tripDelay" action. If the trigger condition returns to normal before this time elapses, no actions are initiated and this timer is reset.

⑧ Path of object being monitored. Relative to api/dev. See Device: /api/dev on page 240. Requirements vary based on trigger type.

⑨ Value at which trigger will trip. Used in "high" and "low" trigger types.

⑩ Current trigger state. See above.

⑪ Number of seconds a tripped trigger condition must be false before it is considered cleared. If the trigger condition trips again before this time elapses, the trigger returns to the tripped state and this timer is reset. Once it expires, the trigger can be considered clear and any clear actions are initiated.

Command: add

▼ API: *PUT /api/alarm/trigger*

```
PUT http://<device-ip>/api/alarm/trigger HTTP/1.1
Content-Type: application/json

{
    "severity": "alarm",
    "latching": false,
    "type": "low",                                          ①
    "path": "A70004A3BB7F45C3/entity/11/measurement/1",     ①
    "threshold": 0,
    "clearDelay": 0,
    "tripDelay": 0,
    "validTime": "",
    "invertValidTime": false,
    "selectedActions": ["0","1"]
}
```

① Required fields: type, path. selectedActions (see Action on page 234) '0' and '1' must exist, otherwise use '[]'.

▼ API (cmd): *api/alarm/trigger : add (Control)*

api/alarm/trigger: add (Control)

```
{
    "severity": "alarm",
    "latching": false,
    "type": "low",                                          ①
    "path": "A70004A3BB7F45C3/entity/11/measurement/1",     ①
    "threshold": 0,
    "clearDelay": 0,
    "tripDelay": 0,
    "validTime": "",
    "invertValidTime": false,
    "selectedActions": ["0","1"]
}
```

① Required fields: type, path. selectedActions (see Action on page 234)'0' and '1' must exist, otherwise use '[]'.

▼ **CLI:** add alarm trigger ARGS

control> add alarm trigger ARGS

```
add alarm trigger = {severity: alarm, latching: false, type: low, path:
17000000EEE2AA12/entity/0/measurement/2, threshold: 0, clearDelay: 0, tripDelay: 0,
validTime: "", invertValidTime: false, selectedActions: "selectedActions":
["0","1"]} ①
```

① Required fields: type, path. selectedActions (see Action on page 234) '0' and '1' must exist, otherwise use '[]'.

Command: delete

▼ **API:** *DELETE /api/alarm/trigger/17000000EEE2AA12_0_2_overAlarm*

```
DELETE http://<device-ip>/api/alarm/trigger/17000000EEE2AA12_0_2_overAlarm HTTP/1.1

{}
```

▼ **API (cmd):** *api/alarm/trigger/17000000EEE2AA12_0_2_overAlarm : delete (Control)*

api/alarm/trigger/17000000EEE2AA12_0_2_overAlarm: delete (Control)

```
{}
```

▼ **CLI:** *delete alarm trigger 17000000EEE2AA12_0_2_overAlarm*

control> delete alarm trigger 17000000EEE2AA12_0_2_overAlarm

```
~
```

The ack command is used to stop pending or active actions for a given trigger. If the trigger is in the "tripped" state, it will move to the "acked" state where it will remain until it clears. If the trigger is in the "latched" state, it will immediately move to the "clear" state. In both of these cases all pending or currently active selected actions are canceled and this trigger is considered "clear" for all purposes of alarm indications.

Command: ack

▼ API (cmd): *api/alarm/trigger/17000000EEE2AA12_0_2_overAlarm : ack (Control)*

api/alarm/trigger/17000000EEE2AA12_0_2_overAlarm: ack (Control)

```
{}
```

▼ CLI: *ack alarm trigger 17000000EEE2AA12_0_2_overAlarm*

control> ack alarm trigger 17000000EEE2AA12_0_2_overAlarm

```
~
```

2.6.2 Target

List of available action targets in the system. They are referenced by alarm actions (see Action on page 234) which are used by alarm triggers (see Trigger on page 226) to send event notifications.

Targets can have the following types:

- email: Destination email addresses added in email targets to be used for notifications. See Email Target on page 98.
- trap: Destination SNMP trap addresses added in snmp targets to be used for notifications. See SNMP on page 153.

▼ API: *GET /api/alarm/target*

```
GET http://<device-ip>/api/alarm/target HTTP/1.1

{
    "email0": {
        "path": "api/conf/email/target/0", ①
        "group": "email",                  ②
        "type": "email",                   ③
        "enabled": true,                   ④
        "name": "john.z.smith@myorg.com"   ⑤
    }
}
```

① Target path with API hierarchy.

② String to logically sort targets. Can be "email", "trap", or the ID of the physical device the target belongs to.

③ Target type.

④ Target is currently available to be controlled by the alarm system. If false, target can still be selected for alarm actions but they will not be performed. See Action on page 234.

⑤ Name of the target. This will be used in notifications.

▼ **API (cmd):** *api/alarm/target : get*

api/alarm/target: get

```
{
    "email0": {
        "path": "api/conf/email/target/0", ①
        "group": "email",                  ②
        "type": "email",                   ③
        "enabled": true,                   ④
        "name": "john.z.smith@myorg.com"   ⑤
    }
}
```

① Target path with API hierarchy.

② String to logically sort targets. Can be "email", "trap", or the ID of the physical device the target belongs to.

③ Target type.

④ Target is currently available to be controlled by the alarm system. If false, target can still be selected for alarm actions but they will not be performed. See Action on the next page.

⑤ Name of the target. This will be used in notifications.

▼ **CLI:** get alarm target

user> get alarm target

```
email0:
    path: api/conf/email/target/0 ①
    group: email                  ②
    type: email                   ③
    enabled: true                 ④
    name: john.z.smith@myorg.com  ⑤
```

① Target path with API hierarchy.

② String to logically sort targets. Can be "email", "trap", or the ID of the physical device the target belongs to.

③ Target type.

④ Target is currently available to be controlled by the alarm system. If false, target can still be selected for alarm actions but they will not be performed. See Action on the next page.

⑤ Name of the target. This will be used in notifications.

2.6.3 Action

List of user configured actions to be used with alarm triggers. See Trigger on page 226.

▼ API: *GET /api/alarm/action*

```
GET http://<device-ip>/api/alarm/action HTTP/1.1

{
    "0": {
        "delay": 0,         ①
        "target": "email0", ②
        "repeat": 0         ③
    }
}
```

① Number of seconds to wait before performing an action for the first time once an alarm trigger initiates it. Timer is independent for each trigger using the action. Modifying this has no effect on already scheduled delayed actions.

② Alarm Target ID, See Target on page 232.

③ Number of seconds to wait before performing an action after the initial delay expires. Continues to perform the action until the alarm trigger stops it. A repeat of 0 means that actions will only occur the first time. Timer is independent for each trigger using the action. Modifying this has no effect on already scheduled repetitions but updated values apply for any subsequent ones.

▼ API (cmd): *api/alarm/action : get*

api/alarm/action: get

```
{
    "0": {
        "delay": 0,         ①
        "target": "email0", ②
        "repeat": 0         ③
    }
}
```

① Number of seconds to wait before performing an action for the first time once an alarm trigger initiates it. Timer is independent for each trigger using the action. Modifying this has no effect on already scheduled delayed actions.

② Alarm Target ID, See Target on page 232.

③ Number of seconds to wait before performing an action after the initial delay expires. Continues to perform the action until the alarm trigger stops it. A repeat of 0 means that actions will only occur the first time. Timer is independent for each trigger using the action. Modifying this has no effect on already scheduled repetitions but updated values apply for any subsequent ones.

▼ **CLI:** get alarm action

user> get alarm action

```
0:
    delay: 0       ①
    target: email0 ②
    repeat: 0      ③
```

① Number of seconds to wait before performing an action for the first time once an alarm trigger initiates it. Timer is independent for each trigger using the action. Modifying this has no effect on already scheduled delayed actions.

② Alarm Target ID, See Target on page 232.

③ Number of seconds to wait before performing an action after the initial delay expires. Continues to perform the action until the alarm trigger stops it. A repeat of 0 means that actions will only occur the first time. Timer is independent for each trigger using the action. Modifying this has no effect on already scheduled repetitions but updated values apply for any subsequent ones.

Command: add

▼ **API:** *PUT /api/alarm/action*

```
PUT http://<device-ip>/api/alarm/action HTTP/1.1
Content-Type: application/json

{
    "target": "email0", ①
    "delay": 0,
    "repeat": 0
}
```

① Required field: target.

▼ **API (cmd):** *api/alarm/action : add (Control)*

api/alarm/action: add (Control)

```
{
    "target": "email0", ①
    "delay": 0,
    "repeat": 0
}
```

① Required field: target.

▼ **CLI:** add alarm action

control: add alarm action = {target: email0, delay: 0, repeat: 0}

```
~       ①
```

① Required field: target.

Command: delete

▼ **API:** *DELETE /api/alarm/action/0*

```
DELETE http://<device-ip>/api/alarm/action/0 HTTP/1.1

{}
```

▼ **API (cmd):** *api/alarm/action/0 : delete (Control)*

api/alarm/action/0: delete (Control)

```
{}
```

▼ **CLI:** delete alarm action 0

control> delete alarm action 0

```
~
```

Action email format

Any alarm action with an email target will send event messages.

Event line format: Event: <Active|Cleared>: <Alarm|Warning|Message>: <description> [<timestamp>] [Threshold: <value>, Value: <value>]. The Threshold/Value suffix is included only for over/under threshold alarms.

Single event: threshold alarm

```
    ****************************************
    IP Address: 126.4.212.61
    Event: Active: Alarm: Under Temperature [ 2026-03-31 15:41:48 ] Threshold: 75.00F,
Value: 74.98F
    Name: rdu120
    Contact:
    Location:
    Description:
    ****************************************
```

Single event: cleared

```
    ****************************************
    IP Address: 126.4.212.61
    Event: Cleared: Alarm: Sensor Added [ Tue Sep 30 19:02:38 2026(UTC+00:00) ]
    Name: rdu120
    Contact:
    Location:
    Description:
    ****************************************
```

Consolidated report (multiple events)

```
    ****************************************
    IP Address: 126.4.212.61
    Events are listed in order of oldest to newest
    Event: Active: Warning: Over Temperature[1] [ Tue Sep 30 18:56:25 2026(UTC+00:00) ]
Threshold: 75.00F, Value: 76.20F
    Event: Cleared: Warning: Over Temperature[1] [ Tue Sep 30 19:00:25 2026(UTC+00:00) ]
Threshold: 75.00F, Value: 70.10F
    Active Events: 1 Events Active
    Name: rdu120
    Contact:
    Location:
    Description:
    ****************************************
```

2.6.4 Valid Time

List of time ranges for use with alarm triggers, See Trigger on page 226. If an entry is removed, any triggers that reference it are set to be always active. If the system clock is not set, triggers are always considered active and valid times are ignored.

▼ API: *GET /api/alarm/validTime*

```
GET http://<device-ip>/api/alarm/validTime HTTP/1.1

{
    "0": {
        "days": "MTWTF--", ①
        "stop": "17:00",   ②
        "start": "07:00"   ③
    }
}
```

① First letter of selected days in order Monday - Sunday. A '-' is used to represent unselected days.

② 00-23 hour plus minutes or 24:00. 00:00 to 24:00 means all day. Stop must be later than start.

③ 00-23 hour plus minutes or 24:00. 00:00 to 24:00 means all day.

▼ API (cmd): *api/alarm/validTime : get*

api/alarm/validTime: get

```
{
    "0": {
        "days": "MTWTF--", ①
        "stop": "17:00",   ②
        "start": "07:00"   ③
    }
}
```

① First letter of selected days in order Monday - Sunday. A '-' is used to represent unselected days.

② 00-23 hour plus minutes or 24:00. 00:00 to 24:00 means all day. Stop must be later than start.

③ 00-23 hour plus minutes or 24:00. 00:00 to 24:00 means all day.

▼ CLI: get alarm validTime

user> get alarm validTime

```
0:
    days: MTWTF-- ①
    stop: 17:00   ②
    start: 07:00  ③
```

① First letter of selected days in order Monday - Sunday. A '-' is used to represent unselected days.

② 00-23 hour plus minutes or 24:00. 00:00 to 24:00 means all day. Stop must be later than start.

③ 00-23 hour plus minutes or 24:00. 00:00 to 24:00 means all day.

Command: add

▼ API: *PUT /api/alarm/validTime*

```
PUT http://<device-ip>/api/alarm/validTime HTTP/1.1
Content-Type: application/json

{
    "start": "09:00",
    "stop": "17:00",
    "days": "MTWTF--"
}
```

▼ API (cmd): *api/alarm/validTime : add (Control)*

api/alarm/validTime: add (Control)

```
{
    "start": "09:00",
    "stop": "17:00",
    "days": "MTWTF--"
}
```

▼ CLI: *add alarm validTime*

control> add alarm validTime = ARGS

```
add alarm validTime = {"start": "09:00", "stop": "17:00", "days": "MTWTF--"}
```

Command: delete

▼ API: *DELETE /api/alarm/validTime/0*

```
DELETE http://<device-ip>/api/alarm/validTime/0 HTTP/1.1

{}
```

▼ API (cmd): *api/alarm/validTime/0 : delete (Control)*

api/alarm/validTime/0: delete (Control)

```
{}
```

▼ **CLI:** delete alarm validTime 0

control> delete alarm validTime 0

~

2.7 Device: /api/dev

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**set, control, delete, reset on these objects require control privilege**
ID, see Device ID on page 243.		0 to 17 objects		Commands: delete, reset
type	String	"snt", "snh", "snc", "snd", "snl", "remotetemp", "t3hd", "a2d", "thd", "afht3		Read-only
state	String	"normal", "unavailable", "degraded"		Read-only
name	String	1 to 25 chars		Read-only
label	String	0 to 25 chars	dev/ID/na me	
order	Integer	0 to max api/dev object count -1		Read-only
snmpInstance	Integer	1 to max api/dev object count		Read-only
ID/alarm, see ID/Alarm on page 247.				
state	String	"none", "clear", "acked", "latched", "tripped"		Read-only
severity	String	"", "warning", "alarm"		Read-only
ID/analog/ID, see ID/Analog on page 248.				
type	String	"5V","10V","binary"		Read-only
state	String	"normal", "unavailable"		Read-only
value	String	0 to String Max		Read-only
displayValue	String	0 to String Max		Read-only
name	String	1 to 25 chars		Read-only
label	String	0 to 25 chars	dev/ID/analog/ID/name	
order	String	-2,147,483,648 to 2,147,483,647		

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**set, control, delete, reset on these objects require control privilege**
units	String	0 to 7 chars	"V"	Read-only
min	Float	Float Min to Float Max	0	Read-only
max	Float	Float Min to Float Max	10	Read-only
mode	String	"door", "powerFailure", "flood", "wscLeak", "wscFault", "smoke", "customBinary"		
highLabel	String	0 to 25 chars	"on"	
lowLabel	String	0 to 25 chars	"off"	
datalogEnabled	Boolean	true, false	true	
displayEnabled	Boolean	true, false	false	
sensorLabel	String	0 to 48 chars		User Assigned Label
assetTag01	String	0 to 32 chars		
assetTag02	String	0 to 32 chars		
ID/analog/ID/thresholds, see ID/Analog/ID/thresholds on page 251.				
active	String	Float Min to Float Max		
ID/analog/ID/alarm, see ID/Alarm on page 247.				
state	String	"none", "clear", "acked", "latched", "tripped"		Read-only
severity	String	"", "warning", "alarm"		Read-only
ID/entity/ID, see ID/Entity on page 252.				
name	String	1 to 25 chars		Read-only
sensorLabel	String	0 to 48 chars		User Assigned Label
sensorLabelExt	String	0 to 48 chars		User Assigned Label
sensorLabelExt2	String	0 to 48 chars		User Assigned Label
sensorLabelExt3	String	0 to 48 chars		User Assigned Label
assetTag01	String	0 to 32 chars		
assetTag01Ext	String	0 to 32 chars		
assetTag01Ext2	String	0 to 32 chars		

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**set, control, delete, reset on these objects require control privilege**
assetTag01Ext3	String	0 to 32 chars		
assetTag02	String	0 to 32 chars		
assetTag02Ext	String	0 to 32 chars		
assetTag02Ext2	String	0 to 32 chars		
assetTag02Ext3	String	0 to 32 chars		
order	Integer	-2,147,483,648 to 2,147,483,647		
orderExt	Integer	-2,147,483,648 to 2,147,483,647		
orderExt2	Integer	-2,147,483,648 to 2,147,483,647		
orderExt3	Integer	-2,147,483,648 to 2,147,483,647		
ID/entity/ID/alarm, see ID/Alarm on page 247.				
state	String	"none", "clear", "acked", "latched", "tripped"		Read-only
severity	String	"", "warning", "alarm"		Read-only
ID/entity/ID/measurement/ID, see ID/Entity/Measurement on page 254.				
type	String	1 to String Max		Read-only
state	String	"normal", "unavailable"		Read-only
value	String	0 to String Max		Read-only
units	String	0 to 7 chars		Read-only
datalogEnabled	Boolean	true, false	true	
displayEnabled	Boolean	true, false	false	
ID/entity/ID/measurement/ID/thresholds, see ID/Entity/Measurement/ID/thresholds on page 256				
underWarn	String		0	
underWarn	String		0	
overWarn	String		0	
overWarn	String		0	
ID/entity/ID/measurement/ID/alarm, see ID/Alarm on page 247.				
type	String	"", "low", "high", "status"		Read-only

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**set, control, delete, reset on these objects require control privilege**
state	String	"none", "clear", "acked", "latched", "tripped"		Read-only
severity	String	"", "warning", "alarm"		Read-only
ID/layout, see ID/Layout on page 257.				
ID	Array (Dev Path)	1 to N items (Device Path)		Read-only

2.7.1 Device ID

List of devices known to the system. The ID for each device is a unique 16 character string. This ID is used throughout the system as part of device paths to uniquely identify specific fields. Each host may contain one device representing the built in capabilities of the hardware. An additional 16 removable devices are supported. Devices added past the maximum supported quantity are ignored by the system. Each device has a different combination of objects and commands which are detailed in the supported devices section. See Supported Devices on page 258.

The different device "type" values are:

- **afht3:** Airflow, temperature, humidity, and dewpoint sensor. See Airflow, temperature, humidity, and dewpoint sensor on page 258.
- **t3hd:** Triple temperature, humidity, and dewpoint sensor. See Triple temperature, humidity, and dewpoint sensor on page 262.
- **thd:** Temperature, humidity, and dewpoint sensor. See Temperature, humidity, and dewpoint sensor on page 269.
- **remotetemp:** Temperature sensor. See Temperature sensor on page 273.
- **a2d:** Analog to Digital converter. See Analog to Digital converter on page 275.
- **snt:** SN series Temperature sensor. See SN series temperature sensor on page 277.
- **snh:** SN series Humidity sensor. See SN series humidity sensor on page 280.
- **snd:** SN series Door position sensor. See SN series door position sensor on page 282.
- **snl:** SN series Leak detection sensor. See SN series leak detection sensor on page 285.
- **snc:** SN series Dry contact sensor. See SN series dry contact sensor on page 288.

This "state" field will have one of the following options:

- **normal:** The device is communicating as expected.
- **unavailable:** The system has been unable to communicate with the device for at least 10 seconds.
- **degraded:** This state applies to devices that have multiple components. The device is communicating as expected but some component is failing to report.

▼ API: *GET /api/dev*

```
GET http://<device-ip>/api/dev HTTP/1.1

{
    "A70004A3BB7F45C3": {
        "name": "SN Temperature", ①
        "label": "temp sensor",   ②
        "type": "snt",            ③
        "state": "normal",        ④
        "order": 0,               ⑤
        "snmpInstance": 1,        ⑥
        "alarm": {},              ⑦
        "analog": {},             ⑧
        "entity": {},             ⑨
        "layout": {}              ⑩
    }
}
```

① Name of the product.

② User configured name. Defaults to the contents of "name".

③ See above.

④ Valid options are "normal", "unavailable", and "degraded". See above.

⑤ Display order for the GUI.

⑥ Instance number in device table for SNMP. Assigned sequentially based on device type. Gaps left by removed devices are filled as new devices of the same type are discovered.

⑦ Alarm status. See ID/Alarm on page 247.

⑧ Analog object. See ID/Analog on page 248.

⑨ Entity object. See ID/Entity on page 252.

⑩ Layout object. See ID/Layout on page 257.

▼ API (cmd): *api/dev*

api/dev: get

```
{
    "A70004A3BB7F45C3": {
        "name": "SN Temperature",
        "label": "temp sensor",
        "type": "snt",
        "state": "normal",
        "order": 0,
        "snmpInstance": 1,
        "alarm": {},
        "analog": {},
        "entity": {},
        "layout": {}
    }
}
```

▼ CLI: *get dev*

user> get dev

```
---
A70004A3BB7F45C3:
    name: SN Temperature ①
    label: temp sensor   ②
    type: snt            ③
    state: normal        ④
    order: 0             ⑤
    snmpInstance: 1      ⑥
    alarm: ...           ⑦
    analog: ...          ⑧
    entity: ...          ⑨
    layout: ...          ⑩
```

① Name of the product.

② User configured name. Defaults to the contents of "name".

③ See above.

④ Valid options are "normal", "unavailable", and "degraded". See above.

⑤ Display order for the GUI.

⑥ Instance number in device table for SNMP. Assigned sequentially based on device type. Gaps left by removed devices are filled as new devices of the same type are discovered.

⑦ Alarm status. See ID/Alarm on page 247.

⑧ Analog object. See ID/Analog on page 248.

⑨ Entity object. See ID/Entity on page 252.

⑩ Layout object. See ID/Layout on page 257.

Command: delete

Deleting a device will cause all associated triggers, log data, and locally stored configuration to be discarded. If a device is still attached, all consequences of deleting a device will occur and the device will be reinitialized as if it was just discovered.

▼ API: *DELETE /api/dev/A70004A3BB7F45C3 (Control)*

```
DELETE http://<device-ip>/api/dev/A70004A3BB7F45C3 HTTP/1.1
```

▼ API (cmd): *api/dev/A70004A3BB7F45C3 : delete (Control)*

api/dev/A70004A3BB7F45C3:

```
{
"cmd": "delete"
}
```

▼ CLI: delete dev ID

control> delete dev ID

```
~
```

Command: reset

▼ API: *api/dev/A70004A3BB7F45C3: reset (Control)*

api/dev/A70004A3BB7F45C3: reset (Control)

```
{
    "cmd": "reset",
    "target": "defaults" ①
}
```

① Required field. The "defaults" target will cause all locally and remotely stored data on the device to be reset to default values. Additional targets may exist depending on the device type.

▼ CLI: reset dev ID

control> reset dev ID = ARGS

```
    ---
reset dev ID = {target: defaults}    ①
```

① "target" is a required field. The "defaults" target will cause all locally and remotely stored data on the device to reset to default values. Additional targets may exist depending on device type.

2.7.2 ID/Alarm

Representing the alarm state for various objects in the system. It will aggregate all triggers configured on the particular object into a single state and severity, see Trigger on page 226. If no triggers are configured for the object, the state is set to "none" and the severity is a blank string. If there are triggers configured for the object but they are all inactive, the state is set to "inactive" and the severity is a blank string. If all triggers are in the clear state, the state is set to "clear" and the severity is also a blank string. Otherwise, the highest priority combination is shown based on the following table (highest priority at the top):

State	Severity
Tripped	Alarm
Latched	Alarm
Tripped	Warning
Latched	Warning
Acked	Alarm
Acked	Warning
Clear	
Inactive	
None	

▼ API: *api/dev/A70004A3BB7F45C3/alarm: get*

api/dev/A70004A3BB7F45C3/alarm: get

```
{
    "state": "tripped", ①
    "severity": "alarm" ②
}
```

① Values are "none", "clear", "acked", "latched", or "tripped".

② Values are "", "warning", or "alarm".

▼ CLI: get dev A70004A3BB7F45C3 alarm

user> get dev A70004A3BB7F45C3 alarm

```
    ---
state: tripped  ①
severity: alarm ②
```

① Values are "none", "clear", "acked", "latched", or "tripped".

② Values are "", "warning", or "alarm".

2.7.3 ID/Analog

List of objects representing analog inputs in the device. An analog input is a hardware feature that senses a voltage or current depending on their type or configuration. The hardware types are:

- **5 V:** 0 V to 5 V reading range with permanently enabled pull up resistor.
- **10 V:** 0 V to 10 V reading range with selectable enabled pull up resistor and optional current sensing mode. Current mode has a 4 mA to 20 mA reading range.
- **binary:** Closed contact input with permanently enabled pull up resistor.

Measurements will fall under one of the following categories:

- **Binary:** Voltage reading is converted to an on/off value represented as 0 or 1. The threshold to distinguish between these two values is 50% of the voltage scale. The pull up resistor is engaged where available.
- **Voltage:** Voltage reading is linearly scaled to the selected min and max values. A reading of 0 V will correspond the "min" field while the "max" field will correspond to the top of the voltage range. The pull up resistor is disabled where available. Values are displayed with a precision of two decimal places (X.YY).
- **Current:** Reading is linearly scaled to the selected min and max values. A reading of 4 mA will correspond the "min" field while the "max" field will correspond to the top of the current range. Values that fall under the 4 mA minimum are represented as '<' followed by the min if the min value is less than the max one, or '>' followed by max otherwise. The pull up resistor is disabled where available. Values are displayed with a precision of two decimal places (X.YY).

A "mode" field is used to select either a preset configuration or to specify that custom parameters are to be used. The following table shows the implied settings of the preset types along with the default values for custom types and the characteristics for each mode:

Mode	Description	Required Type	Measurement	Min	Max	Units	High Label	Low Label
"door"	Door	5V, 10V, binary	Binary	N/A	N/A	N/A	"Open"	"Closed"
"powerFailure"	Power Failure	5V, 10V	Binary	N/A	N/A	N/A	"OK"	"Failure"
"flood"	Flood	5V, 10V	Binary	N/A	N/A	N/A	"Dry"	"Wet"
"wscLeak"	Leak Detector (Sense)	5V, 10V	Binary	N/A	N/A	N/A	"Wet"	"Dry"
"wscFault"	Leak Detector (Fault)	5V, 10V	Binary	N/A	N/A	N/A	"Fault"	"OK"
"smoke"	Smoke Alarm	5V, 10V	Binary	N/A	N/A	N/A	"Clear"	"Alarm"
"customBinary"	Custom (Binary Mode)	5V, 10V	Binary	N/A	N/A	N/A	"High"	"Low"

Values for fields that are not applicable for a given mode as well as values configured by the user before selecting a preset mode are preserved to whatever they held last. Preset mode configuration for fields will always override user settings. These fields can still be set but the preset configuration will always be returned. Once a manual mode is selected, any parameters configured by the user will be returned instead of the previous preset values. N/A fields for a given preset mode are returned as blank or 0.

▼ API: *api/dev/A70004A3BB7F45C3/analog: get*

api/dev/A70004A3BB7F45C3/analog: get

```
{
    "0": {
        "name": "Door 2",          ①
        "label": "Door 2",         ②
        "type": "binary",          ③
        "state": "normal",         ④
        "value": "0",              ⑤
        "units": "",               ⑥
        "min": 0.0,                ⑦
        "max": 1.0,                ⑧
        "mode": "door",            ⑨
        "highLabel": "Open",       ⑩
        "lowLabel": "Closed",      ⑪
        "datalogEnabled": true,    ⑫
        "displayEnabled": false,   ⑬
        "displayValue": "Closed",  ⑭
        "sensorLabel": "",         ⑮
        "assetTag01": "",          ⑯
        "assetTag02": "",          ⑰
        "order": 0                 ⑱
    }
}
```

① Name of the analog input.

② User configured name. Defaults to the contents of "name".

③ Hardware type of input. Can be "10 V", "5 V", or "binary". Determines which modes are allowed.

④ Current analog measurement value status. Can be "normal" if there are no issues or "unavailable" if the value cannot be obtained.

⑤ String representation of current value. Shown as 0 or 1 for binary modes or scaled from min to max for voltage and current modes. If "state" is "unavailable", will show last known value or "" if no known values exist.

⑥ User configurable unit string. Used for voltage and current modes.

⑦ Lower end of scale for voltage and current modes. Represents the value when reading 0 V.

⑧ Upper end of scale for voltage and current modes. Represents the value when reading max input voltage.

⑨ Determines how to interpret the measured analog values. See above.

⑩ Label to be used when displaying a 1 in binary modes.

⑪ Label to be used when displaying a 0 in binary modes.

⑫ Determines if this measurement is to be logged in the datalog or not.

⑬ Determines if this measurement is to be displayed on any external displays. Field is only available on units with a display.

⑭ For binary modes, if the current value is 0, then the low label is given. Otherwise the high label is used. For all other modes, it will be the same string as "value".

⑮ User assigned label.

⑯ User assigned asset tag.

⑰ User assigned asset tag.

⑱ Order in which this input is displayed in the UI. Lower numbers come first.

▼ **CLI:** get dev A70004A3BB7F45C3 analog

user> get dev A70004A3BB7F45C3 analog

```
    ---
0:
    name: door 2          ①
    label: door 2         ②
    type: binary          ③
    state: normal         ④
    value: ""             ⑤
    units: ""             ⑥
    min: 0                ⑦
    max: 1                ⑧
    mode: door            ⑨
    highLabel: Open       ⑩
    lowLabel: Closed      ⑪
    datalogEnabled: true  ⑫
    displayEnabled: false ⑬
    displayValue: Closed  ⑭
    sensorLabel: ""       ⑮
    assetTag01: ""        ⑯
    assetTag02: ""        ⑰
    order: 0              ⑱
```

① Name of the analog input.

② User configured name. Defaults to the contents of "name".

③ Hardware type of input. Can be "10 V", "5 V" or "binary". Determines which modes are allowed.

④ Current analog measurement value status. Can be "normal" if there are no issues or "unavailable" if the value cannot be obtained.

⑤ String representation of current value. Shown as 0 or 1 for binary modes or scaled from min to max for voltage and current modes. If "state" is "unavailable", will show last known value or "" if no known values exist.

⑥ User configurable unit string. Used for voltage and current modes.

⑦ Lower end of scale for voltage and current modes. Represents the value when reading 0 V.

⑧ Upper end of scale for voltage and current modes. Represents the value when reading max input voltage.

⑨ Determines how to interpret the measured analog values. See above.

⑩ Label to be used when displaying a 1 in binary modes.

⑪ Label to be used when displaying a 0 in binary modes.

⑫ Determines if this measurement is to be logged in the datalog or not.

⑬ Determines if this measurement is to be displayed on any external displays. Field is only available on units with a display.

⑭ For binary modes, if the current value is 0, then the low label is given. Otherwise the high label is used. For all other modes, it will be the same string as "value".

⑮ User assigned label.

⑯ User assigned asset tag.

⑰ User assigned asset tag.

⑱ Order in which this input is displayed in the UI. Lower numbers come first.

ID/Analog/ID/thresholds

A threshold for analog (binary) sensors. If a value is over the threshold then it's active, otherwise inactive.

Name	Description
"active"	A threshold for active state

▼ API: *api/dev/A70004A3BB7F45C3/analog/0/thresholds: get*

api/dev/A70004A3BB7F45C3/layout: get

```
{
    "active": "1.00" ①
}
```

① A threshold for active state.

▼ CLI: get dev A70004A3BB7F45C3 analog 0 thresholds

user> get dev A70004A3BB7F45C3 analog 0 thresholds

```
    ---
active: 1 ①
```

① A threshold for active state.

2.7.4 ID/Entity

List of objects representing measurement groups in the device. An entity will represent different logical groupings based on the device type. See Supported Devices on page 258.

▼ API: *api/dev/A70004A3BB7F45C3/entity: get*

api/dev/A70004A3BB7F45C3/entity: get

```
{
    "0": {
        "name": "SN Temperature",    ①
        "order": 2,                  ②
        "orderExt": -1,              ③
        "orderExt2": -1,
        "orderExt3": -1,
        "sensorLabel": "",           ④
        "sensorLabelExt": "",        ⑤
        "sensorLabelExt2": "",
        "sensorLabelExt3": "",
        "assetTag01": "",            ⑥
        "assetTag01Ext": "",
        "assetTag01Ext2": "",
        "assetTag01Ext3": "",
        "assetTag02": "",            ⑦
        "assetTag02Ext": "",
        "assetTag02Ext2": "",
        "assetTag02Ext3": "",
        "alarm": {},                 ⑧
        "measurement": {},           ⑨
    }
}
```

① Name of the entity

② Order of measurement/0. Lower numbers will be displayed first.

③ Order of measurement/1 thru measurement/3 respectively if available. Lower numbers will be displayed first. '-1' will be assigned if not available.

④ User configured name of measurement/0. Defaults to the contents of "name". This field is optional and not present in all entities. If present, it will be settable.

⑤ User assigned tags for measurement/0 thru measurement/3 respectively if available. This field is optional and not present in all entities.

⑥ Asset tags 01 for for measurement/0 thru measurement/3 respectively if available. This field is optional and not present in all entities.

⑦ Asset tags 02 for for measurement/0 thru measurement/3 respectively if available. This field is optional and not present in all entities.

⑧ Alarm status. See ID/Alarm on page 247.

⑨ Measurement object. See ID/Entity/Measurement on page 254.

▼ **CLI:** get dev A70004A3BB7F45C3 entity

user> get dev A70004A3BB7F45C3 entity

```
  ---
0:
  name: SN Temperature     ①
  order: 2                 ②
  orderExt: -1             ③
  orderExt2: -1
  orderExt3: -1
  sensorLabel: ""          ④
  sensorLabelExt: ""       ⑤
  sensorLabelExt2: ""
  sensorLabelExt3: ""
  assetTag01: ""           ⑥
  assetTag01Ext: ""
  assetTag01Ext2: ""
  assetTag01Ext3: ""
  assetTag02: ""           ⑦
  assetTag02Ext: ""
  assetTag02Ext2: ""
  assetTag02Ext3: ""
  alarm: ⑧
    ...
  measurement:             ⑨
    ...
```

① Name of the entity.

② Order of measurement/0. Lower numbers will be displayed first.

③ Order of measurement/1 thru measurement/3 respectively if available. Lower numbers will be displayed first. '-1' will be assigned if not available.

④ User configured name of measurement/0. Defaults to the contents of "name". This field is optional and not present in all entities. If present, it will be settable.

⑤ User assigned tags for measurement/0 thru measurement/3 respectively if available. This field is optional and not present in all entities.

⑥ Asset tags 01 for for measurement/0 thru measurement/3 respectively if available. This field is optional and not present in all entities.

⑦ Asset tags 02 for for measurement/0 thru measurement/3 respectively if available. This field is optional and not present in all entities.

⑧ Alarm status. See ID/Alarm on page 247.

⑨ Measurement object. See ID/Entity/Measurement on the next page.

ID/Entity/Measurement

List of objects representing measurements of the device. The "type" field will determine what kind of measurement to expect and will have one of the following values:

Type	Name	Min	Max	Units	Precision	Notes
"temperature"	"temperature"	- 40	254	degree Celsius	X.YY	The "units" field specifies if value is in degrees Celsius or Fahrenheit
"airflow"	"Airflow"	0	100	N/A	X	Values less than 20 represent still air. 100 represents rushing air. Will show "calibrating" while initializing
"humidity"	"Humidity"	100	100	percent (%)	X	
"dewpoint"	"Dewpoint"	254	254	degree Celsius	X.YY	The "unit" field specifies if value is in degrees Celsius or Fahrenheit

▼ API: *api/dev/A70004A3BB7F45C3/entity/0/measurement: get*

api/dev/A70004A3BB7F45C3/entity/0/measurement: get

```
{
    "0": {
        "type": "temperature",      ①
        "value": "5.43",            ②
        "units": "C",               ③
        "state": "normal",          ④
        "alarm": {}                 ⑤
        "datalogEnabled": true,     ⑥
        "displayEnabled": false     ⑦
        "thresholds": {}            ⑧
    }
}
```

① Measurement type. See above.

② String representation of the current value of the measurement. Precision depends on measurement type. If "state" is "unavailable", will show last known value or "" if no known values exist.

③ Optional field that overrides measurement units given by type.

④ Current measurement status. Can be "normal" if there are no issues or "unavailable" if the measurement cannot be obtained.

⑤ Alarm status. See ID/Alarm on page 247.

⑥ Determines if this measurement is to be logged in the datalog or not.

⑦ Determines if this measurement is to be displayed on any external displays. Field is only available on units with a display.

⑧ Entity measurement thresholds contains thresholds for the measurement. See ID/Entity/Measurement/ID/thresholds on page 256.

▼ **CLI:** get dev A70004A3BB7F45C3 entity 0 measurement

user> get dev A70004A3BB7F45C3 entity 0 measurement

```
 ---
0:
  type: temperature       ①
  value: 5.43             ②
  units: C                ③
  state: normal           ④
  alarm:                  ⑤
    ...
  datalogEnabled: true    ⑥
  displayEnabled: false   ⑦
  thresholds: {}          ⑧
    ...
```

① Measurement type. See above.

② String representation of the current value of the measurement. Precision depends on measurement type. If "state" is "unavailable", will show last known value or "" if no known values exist.

③ Optional field that overrides measurement units given by type.

④ Current measurement status. Can be "normal" if there are no issues or "unavailable" if the measurement cannot be obtained.

⑤ Alarm status. See ID/Alarm on page 247.

⑥ Determines if this measurement is to be logged in the datalog or not.

⑦ Determines if this measurement is to be displayed on any external displays. Field is only available on units with a display.

⑧ Entity measurement thresholds contains thresholds for the measurement. See ID/Entity/Measurement/ID/thresholds on the next page.

ID/Entity/Measurement/ID/thresholds

List of objects representing thresholds for an entity measurement. These thresholds are available for temperature, humidity, dewpoint and airflow sensors.

Name	Description
"underWarn"	Under tempearture/humidity/dewpoint/airflow warning threshold
"underWarn"	Under tempearture/humidity/dewpoint/airflow alarm threshold
"overWarn"	Over tempearture/humidity/dewpoint/airflow warning threshold
"overWarn"	Over tempearture/humidity/dewpoint/airflow alarm threshold

▼ API: *api/dev/A70004A3BB7F45C3/entity/0/measurement/0/thresholds: get*

api/dev/A70004A3BB7F45C3/layout: get

```
{
    "underWarn": "68.0",  ①
    "underAlarm": "64.4", ②
    "overWarn": "95.0",   ③
    "overAlarm": "98.6"   ④
}
```

① Under warning threshold

② Under alarm threshold

③ Over warning threshold

④ Over alarm threshold

▼ CLI: get dev A70004A3BB7F45C3 entity 0 measurement 0 thresholds

user> get dev A70004A3BB7F45C3 entity 0 measurement 0 thresholds

```
---
underWarn: 68                  ①
underAlarm: 64.40000000000001  ②
overWarn: 95                   ③
overAlarm: 98.59999999999999   ④
```

① Under warning threshold

② Under alarm threshold

③ Over warning threshold

④ Over alarm threshold

2.7.5 ID/Layout

List of objects that control the display of entities or set hierarchical relationships within the device. The ID for each layout entry can be either a numeric string (for top level groups) or a device path (for groups belonging to another object). Each layout entry is composed of exactly one array containing a number of device paths. The contents of the array can be seen as components of the ID object. Every device must have at least one layout entry with the ID "0" to represent the top level group. Device paths can only be identified as a components once. Any device path IDs must appear as a component to some other ID but numeric string IDs are not required to be components. Any other entries are optional. Not all entity or analog must be represented in the layout. All device paths must belong to the parent device.

▼ API: *api/dev/A70004A3BB7F45C3/layout: get*

api/dev/A70004A3BB7F45C3/layout: get

```
{
    "0": ["entity/0", "entity/1", "entity/2"],         ①
    "entity/0" : ["entity/3", "entity/4", "entity/5"], ②
    "1": ["entity/6", "entity/7", "entity/8"]          ③
}
```

① Top level entry identified by "0". Entities 0, 1, and 2 make up the top level object

② Entity 0 is composed of entities 3, 4, and 5

③ A second top level group composed of entities 6, 7, and 8 also exists

▼ CLI: get dev A70004A3BB7F45C3 layout

user> get dev A70004A3BB7F45C3 layout

```
---
0: [entity/0, entity/1, entity/2]        ①
entity/0: [entity/3, entity/4, entity/5] ②
1: [entity/6, entity/7, entity/8]        ③
```

① Top level entry identified by "0". Entities 0, 1, and 2 make up the top level object

② Entity 0 is composed of entities 3, 4, and 5

③ A second top level group composed of entities 6, 7, and 8 also exists

2.7.6 Supported Devices

Airflow, temperature, humidity, and dewpoint sensor

This device measures airflow, temperature, humidity, and dewpoint. When powered on, the airflow sensing element must go through an initialization period and will report "calibrating" until done.

▼ API: *api/dev: get*

api/dev: get


```
"00000000EFBDC212": {
    "state": "unavailable",
    "label": "RTAFHD3",
    "name": "RTAFHD3",
    "type": "afht3",
    "alarm": {
        "severity": "",
        "state": "none"
    },
    "layout": {
        "0": [
            "entity/0"
        ]
    },
    "order": 2,
    "snmpInstance": 1,
    "entity": {
        "0": {
        "orderExt3": 1,
        "orderExt2": 0,
        "orderExt": 2,
        "assetTag01Ext3": "Airflow Sensor Asset Tag 01",
        "assetTag01Ext2": "",
        "assetTag01Ext": "",
        "order": 3,
        "assetTag02Ext3": "Airflow Sensor Asset Tag 02",
        "assetTag02": "Temperature Sensor Asset Tag 02",
        "assetTag01": "Temperature Sensor Asset Tag 01",
        "sensorLabelExt3": "Airflow Sensor User Assigned Label",
        "sensorLabelExt2": "Dewpoint Sensor User Assigned Label",
        "assetTag02Ext2": "",
        "assetTag02Ext": "",
        "sensorLabelExt": "Relative Humidity Sensor User Assigned Label",
        "sensorLabel": "Temperature Sensor User Assigned Label",
        "name": "RTAFHD3",
        "measurement": {
            "2": {
                "thresholds": {
                    "underWarn": "39.2",
                    "underAlarm": "35.6",
                    "overWarn": "60.8",
                    "overAlarm": "64.4"
                },
                "datalogEnabled": true,
                "value": "49.30",
```

```
            "state": "unavailable",
            "units": "F",
            "displayEnabled": false,
            "type": "dewpoint",
            "alarm": {
                "type": "high",
                "severity": "alarm",
                "state": "tripped"
            }
        },
        "3": {
            "thresholds": {
                "underWarn": "20.0",
                "underAlarm": "10.0",
                "overWarn": "80.0",
                "overAlarm": "90.0"
            },
            "datalogEnabled": true,
            "value": "28",
            "state": "unavailable",
            "units": "",
            "displayEnabled": false,
            "type": "airflow",
            "alarm": {
                "type": "high",
                "severity": "alarm",
                "state": "clear"
            }
        },
        "1": {
            "thresholds": {
                "underWarn": "34.0",
                "underAlarm": "30.0",
                "overWarn": "56.0",
                "overAlarm": "60.0"
            },
            "datalogEnabled": true,
            "value": "36",
            "state": "unavailable",
            "units": "%",
            "displayEnabled": false,
            "type": "humidity",
            "alarm": {
                "type": "high",
                "severity": "warning",
                "state": "clear"
            }
        },
        "0": {
            "thresholds": {
                "underWarn": "68.0",
                "underAlarm": "64.4",
                "overWarn": "95.0",
                "overAlarm": "98.6"
            },
            "datalogEnabled": true,
            "value": "78.44",
```

```
                "state": "unavailable",
                "units": "F",
                "displayEnabled": false,
                "type": "temperature",
                "alarm": {
                    "type": "high",
                    "severity": "alarm",
                    "state": "tripped"
                }
            }
        },
        "alarm": {
            "severity": "",
            "state": "none"
            }
        }
    }
}
```


▼ **CLI:** get dev

user> get dev


```
 ---
00000000EFBDC212:
  state: unavailable
  label: RTAFHD3
  name: RTAFHD3
  type: afht3
  alarm:
   severity: ""
   state: none
  layout:
    0:
      - entity/0
  order: 2
  snmpInstance: 1
  entity:
   0:
     orderExt3: 1
     orderExt2: 0
     orderExt: 2
     assetTag01Ext3: Airflow Sensor Asset Tag 01
     assetTag01Ext2: ""
     assetTag01Ext: ""
     order: 3
     assetTag02Ext3: Airflow Sensor Asset Tag 02
     assetTag02: Temperature Sensor Asset Tag 02
     assetTag01: Temperature Sensor Asset Tag 01
     sensorLabelExt3: Airflow Sensor User Assigned Label
     sensorLabelExt2: Dewpoint Sensor User Assigned Label
     assetTag02Ext2: ""
     assetTag02Ext: ""
     sensorLabelExt: Relative Humidity Sensor User Assigned Label
     sensorLabel: Temperature Sensor User Assigned Label
```

```
name: RTAFHD3
measurement:
  2:
    thresholds:
      underWarn: 39.2
      underAlarm: 35.6
      overWarn: 60.8
      overAlarm: 64.40000000000001
    datalogEnabled: true
    value: ""
    state: unavailable
    units: F
    displayEnabled: false
    type: dewpoint
    alarm:
      type: ""
      severity: ""
      state: none
  3:
    thresholds:
      underWarn: 20
      underAlarm: 10
      overWarn: 80
      overAlarm: 90
   datalogEnabled: true
   value: ""
   state: unavailable
   units: ""
   displayEnabled: false
   type: airflow
   alarm:
     type: ""
     severity: ""
     state: none
  1:
    thresholds:
      underWarn: 34
      underAlarm: 30
      overWarn: 56
      overAlarm: 60
    datalogEnabled: true
    value: ""
    state: unavailable
    units: "%"
    displayEnabled: false
    type: humidity
    alarm:
      type: ""
      severity: ""
      state: none
  0:
    thresholds:
      underWarn: 68
      underAlarm: 64.40000000000001
      overWarn: 95
      overAlarm: 98.59999999999999
    datalogEnabled: true
```

```
          value: ""
          state: unavailable
          units: F
          displayEnabled: false
          type: temperature
          alarm:
            type: ""
            severity: ""
            state: none
        alarm:
          severity: ""
          state: none
```

Triple temperature, humidity, and dewpoint sensor

This device measures built in temperature, humidity, and dewpoint as well as an additional pair of removable temperature probes. If any of the removable temperature probes is detected as missing, the device state will set to "degraded".

▼ API: *api/dev: get*

api/dev: get

```
{
    "02000000EEF13312": {
        "state": "unavailable",
        "label": "GT3HD",
        "name": "GT3HD",
        "type": "t3hd",
        "alarm": {
            "severity": "",
            "state": "none"
        },
        "layout": {
            "0": [
                "entity/0",
                "entity/1",
                "entity/2"
            ]
        },
        "order": 11,
        "snmpInstance": 1,
        "entity": {
            "2": {
                "orderExt": -1,
                "order": 0,
                "assetTag02Ext2": "",
                "assetTag02Ext": "",
                "assetTag02": "temp 1 Asset Tag 02",
                "sensorLabelExt": "",
                "sensorLabelExt2": "",
                "sensorLabel": "temp 1 User Assigned Label",
                "orderExt2": -1,
                "assetTag01": "temp 1 Asset Tag 01",
                "label": "Temp B",
```

```
            "assetTag01Ext": "",
            "name": "Temp B",
            "assetTag01Ext2": "",
            "measurement": {
                "0": {
                    "thresholds": {
                        "underWarn": "68.0",
                        "underAlarm": "64.4",
                        "overWarn": "95.0",
                        "overAlarm": "98.6"
                    },
                    "datalogEnabled": true,
                    "value": "74.08",
                    "state": "unavailable",
                    "units": "F",
                    "displayEnabled": false,
                    "type": "temperature",
                    "alarm": {
                        "type": "high",
                        "severity": "alarm",
                        "state": "tripped"
                    }
                }
            },
            "alarm": {
                "severity": "",
                "state": "none"
            }
        },
        "1": {
            "orderExt": -1,
            "order": 1,
            "assetTag02Ext2": "",
            "assetTag02Ext": "",
            "assetTag02": "temp 2 Asset Tag 02",
            "sensorLabelExt": "",
            "sensorLabelExt2": "",
            "sensorLabel": "temp 2 User Assigned Label",
            "orderExt2": -1,
            "assetTag01": "temp 2 Asset Tag 01",
            "label": "Temp A",
            "assetTag01Ext": "",
            "name": "Temp A",
            "assetTag01Ext2": "",
            "measurement": {
                "0": {
                    "thresholds": {
                        "underWarn": "68.0",
                        "underAlarm": "64.4",
                        "overWarn": "95.0",
                        "overAlarm": "98.6"
                    },
                    "datalogEnabled": true,
                    "value": "75.09",
                    "state": "unavailable",
                    "units": "F",
                    "displayEnabled": false,
```

```
                "type": "temperature",
                "alarm": {
                    "type": "high",
                    "severity": "alarm",
                    "state": "tripped"
                }
            }
        },
        "alarm": {
            "severity": "",
            "state": "none"
        }
    },
    "0": {
        "orderExt": 3,
        "order": 4,
        "assetTag02Ext2": "dew asset tag 02",
        "assetTag02Ext": "hum Asset Tag 02",
        "assetTag02": "temp Asset Tag 02",
        "sensorLabelExt": "hum User Assigned Label",
        "sensorLabelExt2": "dew user assigned label",
        "sensorLabel": "temp User Assigned Label",
        "orderExt2": 2,
        "assetTag01": "temp Asset Tag 01",
        "label": "Internal",
        "assetTag01Ext": "hum Asset Tag 01",
        "name": "Internal",
        "assetTag01Ext2": "dew asset tag 01",
        "measurement": {
            "2": {
                "thresholds": {
                    "underWarn": "39.2",
                    "underAlarm": "35.6",
                    "overWarn": "60.8",
                    "overAlarm": "64.4"
                },
                "datalogEnabled": true,
                "value": "47.84",
                "state": "unavailable",
                "units": "F",
                "displayEnabled": false,
                "type": "dewpoint",
                "alarm": {
                    "type": "high",
                    "severity": "alarm",
                    "state": "tripped"
                }
            },
            "1": {
                "thresholds": {
                    "underWarn": "34.0",
                    "underAlarm": "30.0",
                    "overWarn": "56.0",
                    "overAlarm": "60.0"
                },
                "datalogEnabled": true,
                "value": "38",
```

```
                        "state": "unavailable",
                        "units": "%",
                        "displayEnabled": false,
                        "type": "humidity",
                        "alarm": {
                            "type": "high",
                            "severity": "warning",
                            "state": "clear"
                        }
                    },
                    "0": {
                        "thresholds": {
                            "underWarn": "68.0",
                            "underAlarm": "64.4",
                            "overWarn": "95.0",
                            "overAlarm": "98.6"
                        },
                        "datalogEnabled": true,
                        "value": "75.16",
                        "state": "unavailable",
                        "units": "F",
                        "displayEnabled": false,
                        "type": "temperature",
                        "alarm": {
                            "type": "high",
                            "severity": "alarm",
                            "state": "tripped"
                        }
                    }
                },
                "alarm": {
                    "severity": "",
                    "state": "none"
                }
            }
        }
    }
}
```

▼ **CLI:** get dev

user> get dev

```
 ---
02000000EEF13312:
  state: unavailable
  label: GT3HD
  name: GT3HD
  type: t3hd
  alarm:
    severity: ""
    state: none
  layout:
    0:
      - entity/0
      - entity/1
      - entity/2
  order: 11
  snmpInstance: 1
  entity:
    2:
      orderExt: -1
      order: 0
      assetTag02Ext2: ""
      assetTag02Ext: ""
      assetTag02: temp 1 Asset Tag 02
      sensorLabelExt: ""
      sensorLabelExt2: ""
      sensorLabel: temp 1 User Assigned Label
      orderExt2: -1
      assetTag01: temp 1 Asset Tag 01
      label: Temp B
      assetTag01Ext: ""
      name: Temp B
      assetTag01Ext2: ""
      measurement:
        0:
          thresholds:
          underWarn: 68
          underAlarm: 64.40000000000001
          overWarn: 95
          overAlarm: 98.59999999999999
        datalogEnabled: true
        value: ""
        state: unavailable
        units: F
        displayEnabled: false
        type: temperature
        alarm:
          type: ""
          severity: ""
          state: none
    alarm:
      severity: ""
      state: none
  1:
```

```
    orderExt: -1
    order: 1
    assetTag02Ext2: ""
    assetTag02Ext: ""
    assetTag02: temp 2 Asset Tag 02
    sensorLabelExt: ""
    sensorLabelExt2: ""
    sensorLabel: temp 2 User Assigned Label
    orderExt2: -1
    assetTag01: temp 2 Asset Tag 01
    label: Temp A
    assetTag01Ext: ""
    name: Temp A
    assetTag01Ext2: ""
    measurement:
      0:
        thresholds:
          underWarn: 68
          underAlarm: 64.40000000000001
          overWarn: 95
          overAlarm: 98.59999999999999
        datalogEnabled: true
        value: ""
        state: unavailable
        units: F
        displayEnabled: false
        type: temperature
        alarm:
        type: ""
        severity: ""
        state: none
  alarm:
    severity: ""
    state: none
0:
  orderExt: 3
  order: 4
  assetTag02Ext2: dew asset tag 02
  assetTag02Ext: hum Asset Tag 02
  assetTag02: temp Asset Tag 02
  sensorLabelExt: hum User Assigned Label
  sensorLabelExt2: dew user assigned label
  sensorLabel: temp User Assigned Label
  orderExt2: 2
  assetTag01: temp Asset Tag 01
  label: Internal
  assetTag01Ext: hum Asset Tag 01
  name: Internal
  assetTag01Ext2: dew asset tag 01
  measurement:
    2:
      thresholds:
        underWarn: 39.2
        underAlarm: 35.6
        overWarn: 60.8
        overAlarm: 64.40000000000001
      datalogEnabled: true
```

```
      value: ""
      state: unavailable
      units: F
      displayEnabled: false
      type: dewpoint
      alarm:
        type: ""
        severity: ""
        state: none
    1:
      thresholds:
        underWarn: 34
        underAlarm: 30
        overWarn: 56
        overAlarm: 60
      datalogEnabled: true
      value: ""
      state: unavailable
      units: "%"
      displayEnabled: false
      type: humidity
      alarm:
      type: ""
      severity: ""
      state: none
  0:
    thresholds:
      underWarn: 68
      underAlarm: 64.40000000000001
      overWarn: 95
      overAlarm: 98.59999999999999
    datalogEnabled: true
    value: ""
    state: unavailable
    units: F
    displayEnabled: false
    type: temperature
    alarm:
      type: ""
      severity: ""
      state: none
alarm:
  severity: ""
  state: none
```

Temperature, humidity, and dewpoint sensor

This device measures temperature, humidity, and dewpoint.

▼ API: *api/dev: get*

api/dev: get


```
{
    "04000000EED8DC12": {
        "state": "unavailable",
        "label": "GTHD",
        "name": "GTHD",
        "type": "thd",
        "alarm": {
            "severity": "",
            "state": "none"
        },
        "layout": {
            "0": [
                "entity/0"
            ]
        },
        "order": 12,
        "snmpInstance": 1,
        "entity": {
            "0": {
                "orderExt3": -1,
                "orderExt2": 0,
                "orderExt": 1,
                "assetTag01Ext3": "",
                "assetTag01Ext2": "GTHDDewpoint Sensor Asset Tag 01",
                "assetTag01Ext": "GTHD Relative Humidity Asset 1",
                "order": 2,
                "assetTag02Ext3": "",
                "assetTag02": "GTHD Temperature Sensor Asset 02",
                "assetTag01": "GTHD Temperature Sensor Asset 01",
                "sensorLabelExt3": "",
                "sensorLabelExt2": "GTHD Dewpoint Sensor User Assigned Label",
                "assetTag02Ext2": "GTHDDewpoint Sensor Asset Tag 02",
                "assetTag02Ext": "GTHD Relative Humidity Asset 2",
                "sensorLabelExt": "GTHD Relative Humidity Sensor User Assigned Labe",
                "sensorLabel": "GTHD Temperature Sensor User Assigned Label",
                "name": "GTHD",
                "measurement": {
                    "2": {
                        "thresholds": {
                            "underWarn": "39.2",
                            "underAlarm": "35.6",
                            "overWarn": "60.8",
                            "overAlarm": "64.4"
                        },
                        "datalogEnabled": true,
                        "value": "47.89",
                        "state": "unavailable",
                        "units": "F",
                        "displayEnabled": false,
```

```
                        "type": "dewpoint",
                        "alarm": {
                            "type": "high",
                            "severity": "alarm",
                            "state": "tripped"
                        }
                    },
                    "1": {
                        "thresholds": {
                            "underWarn": "34.0",
                            "underAlarm": "30.0",
                            "overWarn": "56.0",
                            "overAlarm": "60.0"
                        },
                        "datalogEnabled": true,
                        "value": "38",
                        "state": "unavailable",
                        "units": "%",
                        "displayEnabled": false,
                        "type": "humidity",
                        "alarm": {
                            "type": "high",
                            "severity": "alarm",
                            "state": "clear"
                        }
                    },
                    "0": {
                        "thresholds": {
                            "underWarn": "68.0",
                            "underAlarm": "64.4",
                            "overWarn": "95.0",
                            "overAlarm": "98.6"
                        },
                        "datalogEnabled": true,
                        "value": "75.22",
                        "state": "unavailable",
                        "units": "F",
                        "displayEnabled": false,
                        "type": "temperature",
                        "alarm": {
                            "type": "high",
                            "severity": "alarm",
                            "state": "tripped"
                        }
                    }
                },
                "alarm": {
                    "severity": "",
                    "state": "none"
                }
            }
        }
    }
}
```

▼ **CLI:** get dev

user> get dev

```
 ---
04000000EED8DC12:
  state: unavailable
  label: GTHD
  name: GTHD
  type: thd
  alarm:
    severity: ""
    state: none
  layout:
    0:
      - entity/0
  order: 12
  snmpInstance: 1
  entity:
    0:
      orderExt3: -1
      orderExt2: 0
      orderExt: 1
      assetTag01Ext3: ""
      assetTag01Ext2: GTHDDewpoint Sensor Asset Tag 01
      assetTag01Ext: GTHD Relative Humidity Asset 1
      order: 2
      assetTag02Ext3: ""
      assetTag02: GTHD Temperature Sensor Asset 02
      assetTag01: GTHD Temperature Sensor Asset 01
      sensorLabelExt3: ""
      sensorLabelExt2: GTHD Dewpoint Sensor User Assigned Label
      assetTag02Ext2: GTHDDewpoint Sensor Asset Tag 02
      assetTag02Ext: GTHD Relative Humidity Asset 2
      sensorLabelExt: GTHD Relative Humidity Sensor User Assigned Labe
      sensorLabel: GTHD Temperature Sensor User Assigned Label
      name: GTHD
      measurement:
        2:
          thresholds:
            underWarn: 39.2
            underAlarm: 35.6
            overWarn: 60.8
            overAlarm: 64.40000000000001
          datalogEnabled: true
          value: ""
          state: unavailable
          units: F
          displayEnabled: false
          type: dewpoint
          alarm:
            type: ""
            severity: ""
            state: none
        1:
          thresholds:
            underWarn: 34
```

```
          underAlarm: 30
          overWarn: 56
          overAlarm: 60
        datalogEnabled: true
        value: ""
        state: unavailable
        units: "%"
        displayEnabled: false
        type: humidity
        alarm:
          type: ""
          severity: ""
          state: none
      0:
        thresholds:
          underWarn: 68
          underAlarm: 64.40000000000001
          overWarn: 95
          overAlarm: 98.59999999999999
        datalogEnabled: true
        value: ""
        state: unavailable
        units: F
        displayEnabled: false
        type: temperature
        alarm:
          type: ""
          severity: ""
          state: none
    alarm:
      severity: ""
      state: none
```

Temperature sensor

This device measures temperature.

▼ API: *api/dev: get*

api/dev: get


```
{
    "88000007FA557328": {
        "state": "unavailable",
        "label": "SRT",
        "name": "SRT",
        "type": "remotetemp",
        "alarm": {
            "severity": "",
            "state": "none"
        },
        "layout": {
            "0": [
                "entity/0"
            ]
        },
        "order": 3,
        "snmpInstance": 2,
        "entity": {
            "0": {
                "orderExt3": -1,
                "orderExt2": -1,
                "orderExt": -1,
                "assetTag01Ext3": "",
                "assetTag01Ext2": "",
                "assetTag01Ext": "",
                "order": 0,
                "assetTag02Ext3": "",
                "assetTag02": "",
                "assetTag01": "",
                "sensorLabelExt3": "",
                "sensorLabelExt2": "",
                "assetTag02Ext2": "",
                "assetTag02Ext": "",
                "sensorLabelExt": "",
                "sensorLabel": "",
                "name": "SRT",
                "measurement": {
                    "0": {
                        "thresholds": {
                            "underWarn": "68.0",
                            "underAlarm": "64.4",
                            "overWarn": "95.0",
                            "overAlarm": "98.6"
                        },
                        "datalogEnabled": true,
                        "value": "77.34",
                        "state": "unavailable",
                        "units": "F",
                        "displayEnabled": false,
```

```
                        "type": "temperature",
                        "alarm": {
                            "type": "high",
                            "severity": "alarm",
                            "state": "tripped"
                        }
                    }
                },
                "alarm": {
                    "severity": "",
                    "state": "none"
                }
            }
        }
    }
}
```


▼ **CLI:** get dev

user> get dev


```
 ---
88000007FA557328:
  state: normal
  label: SRT
  name: SRT
  type: remotetemp
  alarm:
    severity: ""
    state: none
  layout:
    0:
      - entity/0
  order: 3
  snmpInstance: 2
  entity:
    0:
      orderExt3: -2147483648
      orderExt2: -2147483648
      orderExt: -2147483648
      assetTag01Ext3: ""
      assetTag01Ext2: ""
      assetTag01Ext: ""
      order: 0
      assetTag02Ext3: ""
      assetTag02: ""
      assetTag01: ""
      sensorLabelExt3: ""
      sensorLabelExt2: ""
      assetTag02Ext2: ""
      assetTag02Ext: ""
      sensorLabelExt: ""
      sensorLabel: ""
      name: SRT
      measurement:
```

```
        0:
          thresholds:
             underWarn: 68
             underAlarm: 64.40000000000001
             overWarn: 95
             overAlarm: 98.59999999999999
           datalogEnabled: true
           value: 74.41
           state: normal
           units: F
           displayEnabled: false
           type: temperature
           alarm:
             type: high
             severity: warning
             state: clear
       alarm:
         severity: ""
         state: none
```

Analog to Digital converter

This device measures a single analog input port.

▼ API: *api/dev: get*

api/dev: get

```
{
    "72ACB66A375D1FD2": {
        "order": 1,
        "snmpInstance": 1,
        "state": "unavailable",
        "label": "A2D",
        "name": "A2D",
        "type": "a2d",
        "alarm": {
            "severity": "",
            "state": "none"
        },
        "layout": {
            "0": [
                "analog/0"
            ]
        },
        "analog": {
            "0": {
                "order": 0,
                "assetTag01": "A2D Sensor Asset Tag 01",
                "sensorLabel": "A2D User Assigned Label",
                "displayValue": "High",
                "displayEnabled": false,
                "mode": "customBinary",
                "highLabel": "High",
```

```
                "value": "1",
                "min": 0.0,
                "state": "unavailable",
                "type": "10V",
                "max": 1.0,
                "label": "Custom (Binary Mode)",
                "thresholds": {
                    "active": "2.00"
                },
                "assetTag02": "A2D Sensor Asset Tag 02",
                "datalogEnabled": true,
                "lowLabel": "Low",
                "name": "Custom (Binary Mode)",
                "units": "",
                "alarm": {
                    "type": "",
                    "severity": "",
                    "state": "none"
                }
            }
        }
    }
}
```


▼ **CLI:** get dev

user> get dev


```
 ---
72ACB66A375D1FD2:
  order: 1
  snmpInstance: 1
  state: unavailable
  label: A2D
  name: A2D
  type: a2d
  alarm:
    severity: ""
    state: none
  layout:
    0:
      - analog/0
  analog:
    0:
      order: 0
      assetTag01: A2D Sensor Asset Tag 01
      sensorLabel: A2D User Assigned Label
      displayValue: Low
      displayEnabled: false
      mode: customBinary
      highLabel: High
      value: ""
      min: 0
      state: unavailable
      type: 10V
```

```
      max: 1
      label: Custom (Binary Mode)
      thresholds:
        active: 2
      assetTag02: A2D Sensor Asset Tag 02
      datalogEnabled: true
      lowLabel: Low
      name: Custom (Binary Mode)
      units: ""
      alarm:
        type: ""
        severity: ""
        state: none
```

SN series temperature sensor

This device measures temperature.

▼ API: *api/dev: get*

api/dev: get


```
{
    "400000000374B042": {
        "state": "unavailable",
        "label": "SN Temperature",
        "name": "SN Temperature",
        "type": "snt",
        "alarm": {
            "severity": "",
            "state": "none"
        },
        "layout": {
            "0": [
                "entity/0"
            ]
        },
        "order": 10,
        "snmpInstance": 1,
        "entity": {
            "0": {
                "orderExt3": -1,
                "orderExt2": -1,
                "orderExt": -1,
                "assetTag01Ext3": "",
                "assetTag01Ext2": "",
                "assetTag01Ext": "",
                "order": 2,
                "assetTag02Ext3": "",
                "assetTag02": "",
                "assetTag01": "",
                "sensorLabelExt3": "",
                "sensorLabelExt2": "",
                "assetTag02Ext2": "",
```

```
                "assetTag02Ext": "",
                "sensorLabelExt": "",
                "sensorLabel": "",
                "name": "SN Temperature",
                "measurement": {
                    "0": {
                        "thresholds": {
                            "underWarn": "68.0",
                            "underAlarm": "64.4",
                            "overWarn": "95.0",
                            "overAlarm": "98.6"
                        },
                        "datalogEnabled": true,
                        "value": "75.31",
                        "state": "unavailable",
                        "units": "F",
                        "displayEnabled": false,
                        "type": "temperature",
                        "alarm": {
                            "type": "high",
                            "severity": "alarm",
                            "state": "tripped"
                        }
                    }
                },
                "alarm": {
                    "severity": "",
                    "state": "none"
                }
            }
        }
    }
}
```


▼ **CLI:** get dev

user> get dev


```
 ---
400000000374B042:
  state: unavailable
  label: SN Temperature
  name: SN Temperature
  type: snt
  alarm:
    severity: ""
    state: none
  layout:
    0:
      - entity/0
  order: 10
  snmpInstance: 1
  entity:
    0:
      orderExt3: -1
```

```
orderExt2: -1
orderExt: -1
assetTag01Ext3: ""
assetTag01Ext2: ""
assetTag01Ext: ""
order: 2
assetTag02Ext3: ""
assetTag02: ""
assetTag01: ""
sensorLabelExt3: ""
sensorLabelExt2: ""
assetTag02Ext2: ""
assetTag02Ext: ""
sensorLabelExt: ""
sensorLabel: ""
name: SN Temperature
measurement:
  0:
    thresholds:
      underWarn: 68
      underAlarm: 64.40000000000001
      overWarn: 95
      overAlarm: 98.59999999999999
    datalogEnabled: true
    value: ""
    state: unavailable
    units: F
    displayEnabled: false
    type: temperature
    alarm:
      type: ""
      severity: ""
      state: none
alarm:
  severity: ""
  state: none
```

SN series humidity sensor

This device measures relative humidity.

▼ API: *api/dev: get*

api/dev: get


```
{
    "F2000000E7F33326": {
        "state": "unavailable",
        "label": "SN Relative Humidity",
        "name": "SN Relative Humidity",
        "type": "snh",
        "alarm": {
            "severity": "",
            "state": "none"
        },
        "layout": {
            "0": [
                "entity/0"
            ]
        },
        "order": 7,
        "snmpInstance": 1,
        "entity": {
            "0": {
                "orderExt3": -1,
                "orderExt2": -1,
                "orderExt": -1,
                "assetTag01Ext3": "",
                "assetTag01Ext2": "",
                "assetTag01Ext": "",
                "order": 1,
                "assetTag02Ext3": "",
                "assetTag02": "",
                "assetTag01": "123",
                "sensorLabelExt3": "",
                "sensorLabelExt2": "",
                "assetTag02Ext2": "",
                "assetTag02Ext": "",
                "sensorLabelExt": "",
                "sensorLabel": "",
                "name": "SN Relative Humidity",
                    "measurement": {
                        "1": {
                            "thresholds": {
                                "underWarn": "34.0",
                                "underAlarm": "30.0",
                                "overWarn": "56.0",
                                "overAlarm": "60.0"
                            },
                            "datalogEnabled": true,
                            "value": "39",
                            "state": "unavailable",
                            "units": "%",
                            "displayEnabled": false,
```

```
                            "type": "humidity",
                            "alarm": {
                                "type": "high",
                                "severity": "alarm",
                                "state": "clear"
                            }
                        }
                    },
                    "alarm": {
                        "severity": "",
                        "state": "none"
                    }
                }
            }
        }
}
```


▼ **CLI:** get dev

user> get dev


```
 ---
F2000000E7F33326:
  state: unavailable
  label: SN Relative Humidity
  name: SN Relative Humidity
  type: snh
  alarm:
    severity: ""
    state: none
  layout:
    0:
      - entity/0
  order: 7
  snmpInstance: 1
  entity:
    0:
      orderExt3: -1
      orderExt2: -1
      orderExt: -1
      assetTag01Ext3: ""
      assetTag01Ext2: ""
      assetTag01Ext: ""
      order: 1
      assetTag02Ext3: ""
      assetTag02: ""
      assetTag01: 123
      sensorLabelExt3: ""
      sensorLabelExt2: ""
      assetTag02Ext2: ""
      assetTag02Ext: ""
      sensorLabelExt: ""
      sensorLabel: ""
      name: SN Relative Humidity
      measurement:
```

```
        1:
          thresholds:
            underWarn: 34
            underAlarm: 30
            overWarn: 56
            overAlarm: 60
          datalogEnabled: true
          value: ""
          state: unavailable
          units: "%"
          displayEnabled: false
          type: humidity
          alarm:
            type: ""
            severity: ""
            state: none
      alarm:
        severity: ""
        state: none
```

SN series door position sensor

This device measures 2 closed contact inputs used in detecting door position. The analog "mode" fields for this device are always "door" and these fields are not changeable.

▼ API: *api/dev: get*

api/dev: get

```
{
    "6A00000013E9F220": {
        "order": 6,
        "snmpInstance": 1,
        "state": "unavailable",
        "label": "SN Door",
        "name": "SN Door",
        "type": "snd",
        "alarm": {
            "severity": "",
            "state": "none"
        },
        "layout": {
            "1": [
                "analog/1"
            ],
            "0": [
                "analog/0"
            ]
        },
        "analog": {
            "1": {
                "order": -1,
                "assetTag01": "",
                "sensorLabel": "",
```

```
                "displayValue": "Closed",
                "displayEnabled": false,
                "mode": "door",
                "highLabel": "Open",
                "value": "0",
                "min": 0.0,
                "state": "unavailable",
                "type": "binary",
                "max": 1.0,
                "label": "Door 2",
                "thresholds": {
                    "active": "2.00"
                },
                "assetTag02": "",
                "datalogEnabled": true,
                "lowLabel": "Closed",
                "name": "Door 2",
                "units": "",
                "alarm": {
                    "type": "",
                    "severity": "",
                    "state": "none"
                }
            },
            "0": {
                "order": 0,
                "assetTag01": "",
                "sensorLabel": "",
                "displayValue": "Closed",
                "displayEnabled": false,
                "mode": "door",
                "highLabel": "Open",
                "value": "0",
                "min": 0.0,
                "state": "unavailable",
                "type": "binary",
                "max": 1.0,
                "label": "Door 1",
                "thresholds": {
                    "active": "2.00"
                },
                "assetTag02": "",
                "datalogEnabled": true,
                "lowLabel": "Closed",
                "name": "Door 1",
                "units": "",
                "alarm": {
                    "type": "",
                    "severity": "",
                    "state": "none"
                }
            }
        }
    }
}
```

▼ **CLI:** get dev

user> get dev

```
 ---
6A00000013E9F220:
  order: 6
  snmpInstance: 1
  state: unavailable
  label: SN Door
  name: SN Door
  type: snd
  alarm:
    severity: ""
    state: none
  layout:
   1:
     - analog/1
   0:
     - analog/0
  analog:
    1:
      order: -1
      assetTag01: ""
      sensorLabel: ""
      displayValue: Closed
      displayEnabled: false
      mode: door
      highLabel: Open
      value: ""
      min: 0
      state: unavailable
      type: binary
      max: 1
      label: Door 2
      thresholds:
        active: 2
      assetTag02: ""
      datalogEnabled: true
      lowLabel: Closed
      name: Door 2
      units: ""
      alarm:
        type: ""
        severity: ""
        state: none
    0:
      order: 0
      assetTag01: ""
      sensorLabel: ""
      displayValue: Closed
      displayEnabled: false
      mode: door
      highLabel: Open
      value: ""
      min: 0
      state: unavailable
```

```
      type: binary
      max: 1
      label: Door 1
      thresholds:
        active: 2
      assetTag02: ""
      datalogEnabled: true
      lowLabel: Closed
      name: Door 1
      units: ""
      alarm:
        type: ""
        severity: ""
        state: none
```

SN series leak detection sensor

This device detects the presence of water along a wire. The analog "mode" fields for this device are always "snLeak" and "snFault" and these fields are not changeable.

▼ API: *api/dev: get*

api/dev: get


```
{
    "EE0030000003AC7E": {
        "order": 9,
        "snmpInstance": 2,
        "state": "normal",
        "label": "SN Leak Detector",
        "name": "SN Leak",
        "type": "snl",
        "alarm": {
            "severity": "",
            "state": "none"
        },
        "layout": {
            "0": [
                "analog/0"
            ]
        },
        "analog": {
            "1": {
                "order": -1,
                "assetTag01": "",
                "sensorLabel": "",
                "displayValue": "OK",
                "displayEnabled": false,
                "mode": "snFault",
                "highLabel": "Fault",
                "value": "0",
                "min": 0.0,
                "state": "normal",
                "type": "binary",
```

```
                "max": 1.0,
                "label": "Fault status",
                "thresholds": {
                    "active": "2.00"
                },
                "assetTag02": "",
                "datalogEnabled": true,
                "lowLabel": "OK",
                "name": "Fault Status",
                "units": "",
                "alarm": {
                    "type": "",
                    "severity": "",
                    "state": "none"
                }
            },
            "0": {
                "order": 0,
                "assetTag01": "",
                "sensorLabel": "",
                "displayValue": "OK",
                "displayEnabled": false,
                "mode": "snLeak",
                "highLabel": "Wet",
                "value": "0",
                "min": 0.0,
                "state": "normal",
                "type": "binary",
                "max": 1.0,
                "label": "Leak detection",
                "thresholds": {
                    "active": "1.00"
                },
                "assetTag02": "",
                "datalogEnabled": true,
                "lowLabel": "OK",
                "name": "Leak Detection",
                "units": "",
                "alarm": {
                    "type": "high",
                    "severity": "alarm",
                    "state": "clear"
                }
            }
        }
    }
}
```

▼ **CLI:** get dev

user> get dev

```
 ---
EE0030000003AC7E:
  order: 9
  snmpInstance: 2
  state: unavailable
  label: SN Leak Detector
  name: SN Leak
  type: snl
  alarm:
    severity: ""
    state: none
  layout:
    0:
      - analog/0
  analog:
    1:
      order: -1
      assetTag01: ""
      sensorLabel: ""
      displayValue: OK
      displayEnabled: false
      mode: snFault
      highLabel: Fault
      value: ""
      min: 0
      state: unavailable
      type: binary
      max: 1
      label: Fault status
      thresholds:
        active: 2
      assetTag02: ""
      datalogEnabled: true
      lowLabel: OK
      name: Fault Status
      units: ""
      alarm:
        type: ""
        severity: ""
        state: none
    0:
      order: 0
      assetTag01: ""
      sensorLabel: ""
      displayValue: OK
      displayEnabled: false
      mode: snLeak
      highLabel: Wet
      value: ""
      min: 0
      state: unavailable
      type: binary
      max: 1
```

```
        label: Leak detection
        thresholds:
          active: 1
        assetTag02: ""
        datalogEnabled: true
        lowLabel: OK
        name: Leak Detection
        units: ""
        alarm:
          type: ""
          severity: ""
          state: none
```

SN series dry contact sensor

This device is designed to assist in monitoring conditions within a rack and can be placed in any area to read door status.

▼ API: *api/dev: get*

api/dev: get


```
{
    "7C0000000DDF4D20": {
        "order": 5,
        "snmpInstance": 2,
        "state": "unavailable",
        "label": "SN Contact",
        "name": "SN Contact",
        "type": "snc",
        "alarm": {
            "severity": "",
            "state": "none"
        },
        "layout": {
            "2": [
                "analog/2"
            ],
            "1": [
                "analog/1"
            ],
            "0": [
                "analog/0"
            ]
        },
        "analog": {
            "2": {
                "order": -1,
                "assetTag01": "",
                "sensorLabel": "",
                "displayValue": "Open",
                "displayEnabled": false,
                "mode": "contact",
                "highLabel": "Open",
                "value": "1",
```

```
            "min": 0.0,
            "state": "unavailable",
            "type": "binary",
            "max": 1.0,
            "label": "Contact 3",
            "thresholds": {
                "active": "2.00"
            },
            "assetTag02": "",
            "datalogEnabled": true,
            "lowLabel": "Closed",
            "name": "Contact 3",
            "units": "",
            "alarm": {
                "type": "",
                "severity": "",
                "state": "none"
            }
        },
        "1": {
            "order": -1,
            "assetTag01": "",
            "sensorLabel": "",
            "displayValue": "Open",
            "displayEnabled": false,
            "mode": "contact",
            "highLabel": "Open",
            "value": "1",
            "min": 0.0,
            "state": "unavailable",
            "type": "binary",
            "max": 1.0,
            "label": "Contact 2",
            "thresholds": {
                "active": "2.00"
            },
            "assetTag02": "",
            "datalogEnabled": true,
            "lowLabel": "Closed",
            "name": "Contact 2",
            "units": "",
            "alarm": {
                "type": "",
                "severity": "",
                "state": "none"
            }
        },
        "0": {
            "order": 0,
            "assetTag01": "",
            "sensorLabel": "",
            "displayValue": "Open",
            "displayEnabled": false,
            "mode": "contact",
            "highLabel": "Open",
            "value": "1",
            "min": 0.0,
```

```
                "state": "unavailable",
                "type": "binary",
                "max": 1.0,
                "label": "Contact 1",
                "thresholds": {
                    "active": "2.00"
                },
                "assetTag02": "",
                "datalogEnabled": true,
                "lowLabel": "Closed",
                "name": "Contact 1",
                "units": "",
                "alarm": {
                    "type": "",
                    "severity": "",
                    "state": "none"
                }
            }
        }
    }
}
```


▼ **CLI:** get dev

user> get dev


```
 ---
7C0000000DDF4D20:
  order: 5
  snmpInstance: 2
  state: unavailable
  label: SN Contact
  name: SN Contact
  type: snc
  alarm:
    severity: ""
    state: none
  layout:
    2:
      - analog/2
    1:
      - analog/1
    0:
      - analog/0
  analog:
    2:
      order: -1
      assetTag01: ""
      sensorLabel: ""
      displayValue: Closed
      displayEnabled: false
      mode: contact
      highLabel: Open
      value: ""
      min: 0
```

```
      state: unavailable
      type: binary
      max: 1
      label: Contact 3
      thresholds:
        active: 2
      assetTag02: ""
      datalogEnabled: true
      lowLabel: Closed
      name: Contact 3
      units: ""
      alarm:
        type: ""
        severity: ""
        state: none
    1:
      order: -1
      assetTag01: ""
      sensorLabel: ""
      displayValue: Closed
      displayEnabled: false
      mode: contact
      highLabel: Open
      value: ""
      min: 0
      state: unavailable
      type: binary
      max: 1
      label: Contact 2
      thresholds:
        active: 2
      assetTag02: ""
      datalogEnabled: true
      lowLabel: Closed
      name: Contact 2
      units: ""
      alarm:
        type: ""
        severity: ""
        state: none
    0:
      order: 0
      assetTag01: ""
      sensorLabel: ""
      displayValue: Closed
      displayEnabled: false
      mode: contact
      highLabel: Open
      value: ""
      min: 0
      state: unavailable
      type: binary
      max: 1
      label: Contact 1
      thresholds:
        active: 2
      assetTag02: ""
```

```
      datalogEnabled: true
      lowLabel: Closed
      name: Contact 1
      units: ""
      alarm:
        type: ""
        severity: ""
        state: none
```

2.8 Special File Transfers: /Transfer, SCP

These special file transfer features are available via API and/or SCP. Please note that SSH must be enabled in order to use SCP commands.

2.8.1 Factory Support Package: logs.enc

The logs.enc file.

Downloads an encrypted diagnostic package that can be sent to technical support personnel. This is not processed as a regular API request.

The package can be downloaded by performing an HTTP GET on the /transfer/logs.enc path.

Downloads require user authentication to be passed in as part of the query string and the user must have administrator privileges.

▼ API: *GET /transfer/logs.enc*

```
GET http://<device-ip>/transfer/logs.enc HTTP/1.1
```

▼ SCP: *logs.enc*

```
scp <admin>@<your-device-ip>:logs.enc .
```

2.8.2 Ethpcap Download

The ethpcap (ethernet traffic) file.

Downloads the card's ethernet traffic file and saves as a .pcap file. The .pcap file contains raw packet data captured from the specified network interface, enabling users to analyze network traffic for debugging, monitoring, or security purposes.

The package can be downloaded by performing an HTTP GET on the /transfer/ethpcap path.

Downloads require user authentication to be passed in as part of the query string and the user must have administrator privileges.

▼ API: *GET /transfer/ethpcap*

```
GET http://<device-ip>/transfer/ethpcap HTTP/1.1
```

2.8.3 Backup Download

Downloads a configuration backup file. This is not processed as a regular API request. This file contains all user configured settings in the API tree in an encrypted form and is unique to the device that produced it. The file can be downloaded by performing an HTTP GET on the /transfer/backup path. Downloads do not require user authentication. The name of the downloaded file is "_transfer_backup-XXXXXXXXXX".

▼ API: *GET /transfer/backup*

```
GET http://<device-ip>/transfer/backup HTTP/1.1
```

▼ SCP: download backup

```
scp <admin>@<your-device-ip>:backup .
```

2.8.4 Backup Upload

Uploads a configuration backup file. This is not processed as a regular API request. The file is uploaded by performing an HTTP POST command on the /transfer/backup path. All POSTs to this path are assumed to be an upload command and require user authentication to be passed in as part of the query string (see API Usage on page 5), and the user must have administrator privileges. Input must be encoded as multipart/form-data with a single component called "file". The actual file name being used is ignored. Output is a regular API JSON response with an error code 0 if the upload was successful or another error code otherwise, see Error Codes on page 22. The uploaded file must have been downloaded from the same unit. Uploading a file belonging to a different unit will return an error. Once a file has been successfully uploaded, all user configuration parameters will be replaced with those found in the backup file.

▼ API: *POST /transfer/backup*

```
{
POST http://<your-device-ip>/transfer/backup \
-H "Content-Type: multipart/form-data" \
-u <admin>:<your-password> \
-F "file=@/path/to/your/_transfer_backup-XXXXXXXXXX"
}   ①
```

① Command to upload a backup file that restore the system.

▼ SCP: upload backup

```
scp <backupFile> <admin>@<your-device-ip>:backup
```

2.8.5 Bootlog Download

The boot log file.

Downloads file through an HTTP GET on the "transfer/bootlog" path. If running the command on a terminal, be sure to redirect the output to a file.

On Postman, there is "save file" button to write the output to a file. Downloads require user authentication to be passed in as part of the query string and the user must have administrator privileges.

▼ API: *GET /transfer/bootlog*

```
GET http://<device-ip>/transfer/bootlog HTTP/1.1
```

2.8.6 Firmware Update

Uploads a new firmware file to the system. Upon successful upload, the system will reboot to apply the new firmware version.

It requires authentication to be sent in through the query string ("username"/"password"); authentication can also be sent via token parameter on the following endpoint: '/transfer/firmware?token=1234'. Token is generated by accessing the API path '/api/auth/user/<username>'. When token is used to authenticate, the '-u' option is not needed.

Input must be encoded as multipart/form-data with a single component called "file".

Firmware upload file must match the platform and OEM currently running.

Uploading a file that violates these restrictions will return an error.

Once a firmware update is in progress, it cannot be interrupted. Any subsequent upload attempts will be blocked.

▼ API: *POST /transfer/firmware*

```
{
POST http://<your-device-ip>/transfer/firmware \
-H "Content-Type: multipart/form-data" \
-u <admin>:<your-password> \
-F "file=@/path/to/your/file.firmware"
}   ①
```

① Command to upload a firmware file that updates the system.

▼ SCP: firmware

```
scp <file.firmware> <admin>@<your-device-ip>:firmware
```

2.8.7 SSL Certificate Upload

Uploads a custom SSL certificate to replace the default. All subsequent HTTPS requests will use the new certificate, unless a full or partial reset to defaults restores it to the default certificate.

It requires authentication to be sent in through the query string ("username"/"password"). The HTTP body is in the form of the multipart form post, with two parts:

- "file", which is an SSL certificate file of either PFX, CER, CRT, or PEM type. This file must contain the certificate and private key to be used.
- "password", which is needed when the file is password secured. Can be an empty string.

The response will be a standard JSON response with retCode 0 upon success, or 7004, 7005, or 7007 on failure, see Error Codes on page 22. Possible failure conditions include malformed file or insufficient security on the uploaded file or firmware update already in progress.

▼ API: *POST /transfer/sslcert*

```
{
POST http://<your-device-ip>/transfer/sslcert \
-H "Content-Type: multipart/form-data" \
-u <admin>:<your-password> \
-F "file=@/path/to/your/cert.pfx" \
-F "password=<file-password>"
}   ①
```

① Command to upload a custom SSL certificate.

▼ SCP: sslcert

```
scp <ssl.crt> <admin>@<your-device-ip>:sslcert
```

2.8.8 802.1X Upload

Uploads the necessary certificates and private key for IEEE 802.1X authentication. The user must provide the following:

- "caCert", which is used to verify the authentication server's certificate. The certificate must be in PEM format.
- "clientCert", which is the client certificate. The certificate can be in PEM, CER, CRT, or PFX format. This file must contain the certificate and private key to be used.
- "privKeyPwd", which is the password for the client private key.

Uploading a file that violates these restrictions will return an error, see Error Codes on page 22.

▼ API: *POST /transfer/dot1x*

```
{
POST http://<your-device-ip>/transfer/dot1x \
-H "Content-Type: multipart/form-data" \
-u <admin>:<your-password> \
-F "caCert=@/path/to/your/caCert.pem" \
-F "clientCert=@/path/to/your/clientCert.pem" \
-F "privKeyPwd=<PrivKeyPassword>"
}   ①
```

① Command to upload 802.1X files.

When SCP is used, "caCert" and "clientCert" files should be zipped up along with the "dot1x.json" file so that a single file can be uploaded.

- "dot1x.json" file that contains information about "caCert" and "clientCert" and "privKeyPwd", which is the password for the client private key. For example:

```
{
    "caCert": "<CA_cert_filename>",
    "clientCert": "<client_cert_filename>",
    "privKeyPwd": "<Private_key_password>"
}
```

- Create a tar.gz file that contains "caCert", "clientCert" and "dot1x.json" files, for example:

```
tar -czvf <dot1xFiles.tar.gz> <CA_cert_filename> <client_cert_filename> dot1x.json
```

Uploading a file that violates these restrictions will return an "Unknown" error. Check log for a specific error message.

▼ SCP: dot1x

```
scp <dot1xFiles.tar.gz> <admin>@<your-device-ip>:dot1x
```

2.8.9 Equipment Firmware Update

Uploads a new firmware file to the equipment (device).

It requires authentication to be sent in through the query string ("username"/"password"); authentication can also be sent via token parameter on the following endpoint: '/transfer/equipment/firmware?token=1234'. Token is generated by accessing the API path '/api/auth/user/<username>'. When token is used to authenticate, the '-u' option is not needed.

Input must be encoded as multipart/form-data with a single component called "file".

Uploading a file that violates these restrictions will return an error.

Once a firmware update is in progress, it cannot be interrupted. Any subsequent upload attempts will be blocked.

After the POST succeeds (the upload request is accepted), poll GET /equipment/firmware (see Firmware on page 320) on the equipment API to read verification and update status until the operation completes or reports an error in the firmware status fields.

▼ API: *POST /transfer/equipment/firmware*

```
{
POST http://<your-device-ip>/transfer/equipment/firmware \
-H "Content-Type: multipart/form-data" \
-u <admin>:<your-password> \
-F "file=@/path/to/your/file.firmware"
}   ①
```

① Command to upload a firmware file that updates the equipment.

2.8.10 PLD Import

Uploads an encrypted PLD package file to the system.

It requires authentication to be sent in through the query string ("username"/"password").

Input must be encoded as multipart/form-data with a single component called "file".

The uploaded file must be an encrypted tar.gz file (.tar.gz.enc).

Uploading a file that violates these restrictions will return an error.

▼ API: *POST /transfer/pldImport*

```
{
POST http://<your-device-ip>/transfer/pldImport \
-H "Content-Type: multipart/form-data" \
-u <admin>:<your-password> \
-F "file=@/path/to/your/file.tar.gz.enc"
}   ①
```

① Command to upload an encrypted PLD package file.

2.8.11 Remote Syslog mTLS Certificate Upload

Uploads certificate material for remote syslog over TLS (mTLS).

The device exposes a single path, /transfer/remoteSyslog, with behavior selected by the multipart form field auth.

All POSTs use multipart/form-data and require user authentication in the query string (see API Usage on page 5); the user must have administrator privileges.

On success, the JSON response uses retCode 0, see Error Codes on page 22).

Failures return a standard JSON error body. Detailed diagnostics are written to device logs.

Multipart validation errors (for example missing auth, wrong auth value, missing required file parts, or for auth=target an empty target id) result in HTTP status 417 Expectation Failed with a JSON error (typically retCode in the validation range, see Error Codes on page 22).

Other failures use the usual HTTP status for API JSON responses.

Internal or non-reportable server-side faults may yield retCode 5000 with minimal data; use device logs for diagnosis in those cases., see Error Codes on page 22.

Syslog client PEM (Mutual TLS)

To enable mutual TLS (mTLS) authentication, use this endpoint to upload an unencrypted .PEM bundle containing the X.509 public key certificate and its corresponding unencrypted private key.

The syslog subsystem in the RDU120 then uses this material to prove its identity and establish a secure connection to remote syslog servers downstream.

The user must provide the following:

- "clientPem", which is the client PEM bundle.
  - This file must contain the client's X.509 certificate and its corresponding unencrypted private key.
  - The file must be in .PEM format.
  - The .PEM file must be unencrypted.
- "auth=client", which identifies the operation as a client PEM upload.

Any upload that does not meet these requirements will be rejected.

Invalid or unusable client PEM material (after validation) is reported with retCode 7015 (Invalid syslog TLS client certificate file)., see Error Codes on page 22.

▼ *Client PEM Validation Rules*

Client PEM uploads are validated on receipt. A successful upload is accepted silently. When validation finds a problem, one of two outcomes follows:

- Blocking errors — the upload is rejected because the issue would prevent the device from starting or establishing TLS. The failure is reported with retCode 7015 (Invalid syslog TLS client certificate file) and the validator writes diagnostic detail to the device log, see Error Codes on page 22.
- Logged warnings — the upload is accepted, but the validator records a message in the device log because the material is likely to be rejected by some remote syslog servers during the TLS handshake. The operator should replace the certificate before relying on the connection.

Operators should consult the device log after a failed upload, or when there is no observed connectivity with a TLS-based remote syslog destination.

Blocking conditions (upload rejected)

- The uploaded file cannot be read as a PEM container or as an X.509 certificate (corrupted or in an unrecognizable format).
- The bundle does not contain a BEGIN CERTIFICATE PEM block; no client certificate was found in the bundle.
- The bundle does not include a private key in an accepted PEM form alongside the certificate.
- The private key uses PKCS#8 or legacy PEM encryption; only plaintext key material is accepted.
- The private key does not load with an empty passphrase (treated as still encrypted or unusable); may co-occur with the previous condition.
- The bundled private key does not match the leaf certificate, or multiple certificates make the leaf ambiguous.
- No readable X.509 certificate could be obtained from the PEM (shared validation path for client and CA PEM material). The file might be a PEM container, but it does not contain an X.509 certificate.

Logged-warning conditions (upload accepted; see device log)

- The client leaf asserts **CA:true** in Basic Constraints (it is also a CA certificate). TLS servers downstream typically reject web-client-authentication certificates with the **CA:true** attribute for security reasons; the operator should replace the certificate with a non-CA leaf.
- Extended Key Usage does not include **TLS Web Client Authentication**. Some remote syslog servers tolerate this; others will reject the handshake. The operator should replace the certificate with one that declares the client-auth EKU for portability.

▼ **API:** *transfer/remoteSyslog (auth=client)*

transfer/remoteSyslog: POST (client PEM / mTLS identity)

```
{
POST http://<your-device-ip>/transfer/remoteSyslog \
-H "Content-Type: multipart/form-data" \
-u <admin>:<your-password> \
-F "auth=client" \
-F "clientPem=@/path/to/your/client-cert-and-key.pem"
}   ①
```

① Command to upload a client PEM bundle.

When SCP is used, upload the same PEM bundle as for the clientPem form field in the POST API. The destination after the colon must be exactly remoteSyslog with no additional path (not remoteSyslog/). This is equivalent to POST /transfer/remoteSyslog with auth=client and the clientPem part. If the upload is rejected, the retCode is the same as for the equivalent HTTP POST for the same fault, see Error Codes on page 22.

▼ **SCP:** remoteSyslog

```
scp <path/to/client-cert-and-key.pem> <admin>@<your-device-ip>:remoteSyslog
```

Example when the uploaded client PEM is rejected:

```
scp <invalid-client-pem> <admin>@<ip>:remoteSyslog
(<admin>@<ip>) Password:
<invalid-client-pem>
100% 2094 330.2KB/s 00:00
7015: Invalid syslog client certificate file
```

Remote syslog target PEM (Server authentication)

To authenticate a remote syslog server downstream, use this endpoint to upload a valid, unencrypted CA certificate in .PEM format (which may be self-signed).

This allows the syslog subsystem in the RDU120 to verify a remote syslog server's identity during the TLS handshake and maintain a trusted connection.

The uploaded file must be a valid, unencrypted CA in .PEM format. If the certificate is improperly formatted or encrypted, the upload will be rejected and the client will be unable to verify the server's authenticity.

The user must provide the following: * "caPem", which is the CA PEM bundle. This file must be a valid, unencrypted CA in .PEM format. The file must be in .PEM format. ** The file must be unencrypted. * "auth=target", which identifies the operation as a target PEM upload. * "target", which is the id of a remote syslog target that is configured for TLS transport.

Invalid or unusable CA PEM material is reported with retCode 7014 (Invalid syslog TLS CA certificate file), see Error Codes on page 22.

When auth=target, the form field target must be the id of a remote syslog target that is configured for TLS transport.

If the id is unknown, refers to a non-TLS (for example UDP) target, or is otherwise not eligible for TLS trust material, the operation fails with retCode 4013 (Invalid option), see Error Codes on page 22.

▼ *CA PEM Validation Rules*

Target CA PEM uploads are validated on receipt. A successful upload is accepted silently. When validation finds a problem, one of two outcomes follows:

- Blocking errors — the upload is rejected because the issue would prevent the device from starting, or because the CA cannot be installed at all. CA-content failures are reported with retCode 7014 (Invalid syslog TLS CA certificate file), see Error Codes on page 22; a target field that does not identify an eligible TLS target is reported with retCode 4013 (Invalid option), see Error Codes on page 22. In both cases the validator writes diagnostic detail to the device log.
- Logged warnings — the upload is accepted, but the validator records a message in the device log because the material is likely to prevent or affect the TLS handshake to the remote syslog server. The operator should replace the CA before relying on the connection.

Operators should consult the device log after a failed upload, or when there is no observed connectivity with a TLS-based remote syslog destination.

Blocking conditions (upload rejected)

- The target form field does not identify a remote syslog target that is configured for TLS on the device (the id is unknown, the target uses non-TLS transport such as UDP, or it is otherwise not eligible for TLS trust material). This case is reported with retCode 4013 rather than 7014, see Error Codes on page 22.
- The uploaded CA does not assert CA:true in Basic Constraints (it cannot be used as a CA trust anchor).

- No readable X.509 certificate could be obtained from the PEM (shared validation path for client and CA PEM material). The file might be a PEM container, but it does not contain an X.509 certificate.

Logged-warning conditions (upload accepted; see device log)

- The CA does not declare Certificate Sign or CRL Sign under Key Usage. Strict TLS clients may refuse to honor it as a trust anchor; the operator should replace the certificate with one that declares the appropriate Key Usage.
- The CA includes TLS Web Client Authentication in Extended Key Usage. That EKU is the wrong profile for verifying a remote syslog server and may cause strict TLS clients to reject the handshake; the operator should replace the CA with one whose EKU permits server authentication.
- The CA certificate's subject is empty. Some TLS stacks reject anonymous trust anchors during chain validation; the operator should replace it with a CA that has a populated subject.
- The CA certificate's issuer is empty. Some TLS stacks reject CAs whose issuer cannot be matched during chain validation; the operator should replace it with a CA that has a populated issuer.

▼ API: *transfer/remoteSyslog (auth=target)*

transfer/remoteSyslog: POSTS (CA PEM / server authentication)

```
{
POST http://<your-device-ip>/transfer/remoteSyslog \
-H "Content-Type: multipart/form-data" \
-u <admin>:<your-password> \
-F "auth=target" \
-F "target=<target-id>" \
-F "caPem=@/path/to/your/caCert.pem"
}  ①
```

① Command to upload a CA PEM bundle.

When SCP is used, upload the same PEM file as for the caPem form field in the POST API. The destination after the colon is remoteSyslog/ followed by the target id (the same value as the target form field for that TLS remote syslog destination). The id must be a valid object id (the id of a remote syslog target with TLS enabled); remoteSyslog/ with nothing after the slash is rejected. This is equivalent to POST /transfer/remoteSyslog with auth=target, target=<target-id>, and the caPem part. If the upload is rejected, the retCode is the same as for the equivalent HTTP POST for the same fault, see Error Codes on page 22.

▼ SCP: *remoteSyslog/<target-id>*

```
scp <path/to/caCert.pem> <admin>@<your-device-ip>:remoteSyslog/<target-id>
```

Example when the uploaded CA PEM is rejected:

```
scp <invalid-ca-pem> <admin>@<ip>:remoteSyslog/0
(<admin>@<ip>) Password:
<invalid-ca-pem>
100% 3363 802.2KB/s 00:00
7014: Invalid syslog CA certificate file
```

Example when the destination target id is not valid for this operation:

```
scp <ca-upload-pem> <admin>@<ip>:remoteSyslog/invalid-id
(<admin>@<ip>) Password:
<ca-upload-pem>
100% 3363 921.7KB/s 00:00
4013: Invalid option
```

3 Equipment

3.1 Equipment API Usage

The equipment API conforms to the RDU120 Web API with the exception of the root path. All equipment requests are performed on a path starting with /equipment/ instead of /api/ (an example usage would be http://<ip address>/equipment/psi). Otherwise, the behavior of nodes in the path is unchanged from the regular API. The equipment API exists to allow the managed device's configuration and data to be read and configured. It is a protected API whereby the API user must be authenticated before being able to read or modify the configuration.

IMPORTANT! The API will replace the forward slash '/' with a hyphen '-' for all point names in Web UI.

3.2 Equipment API

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**set on these objects require admin privilege**
psi	String			read-only
report, see Report on page 305.				
tag	String			read-only
mmCnt	String			read-only
report/ID/point, see Point on page 309.				
id	String			read-only
access	String			read-only
tag	String			read-only
desc	String			read-only
type	String			read-only
units	String			read-only
report/ID/point/ID/value, see Value on page 312				
id	String			read-only
value	String			
report/ID/event, see Event on page 311				
id	String			read-only
tag	String			read-only
desc	String			read-only
report/ID/event/ID/value, see Value on page 312				
id	String			read-only
value	String			read-only
report/ID/children, see Children on page 307				

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**set on these objects require admin privilege**
id	String			read-only
tag	String			read-only
mmCnt	String			read-only
activeEvent, see Active Event on page 314				
id	String			read-only
tag	String			read-only
desc	String			read-only
activeEvent/ID/value, see Value on page 312				
id	String			read-only
value	String			read-only
firmware, see Firmware on page 320				
hostFileSize	Integer			read-only
hostFilename	String			read-only
additionalInfo	String			read-only
statusDescription	String			read-only
statusCode	Integer			read-only

3.2.1 Equipment

▼ API: *GET /equipment/*

```
GET http://<device-ip>/equipment/ HTTP/1.1

{
    "psi": "2.34",     ①
    "report": [],      ②
    "activeEvent": []  ③
}
```

① The product sequence identifier of the equipment.

② The reports of the equipment. See Report on the facing page.

③ The active events of the equipment. See Active Event on page 314.

▼ API (cmd): *equipment : get*

equipment: get

```
{
   "psi": "2.34",     ①
   "report": [],      ②
   "activeEvent": []  ③
}
```

① The product sequence identifier of the equipment.

② The reports of the equipment. See Report below.

③ The active events of the equipment. See Active Event on page 314.

▼ CLI: *{"cmd": "get", "path": "equipment"}*

user> {"cmd": "get", "path": "equipment"}

```
psi: 2.34     ①
report:       ②
  ...
activeEvent:  ③
  ...
```

① The product sequence identifier of the equipment.

② The reports of the equipment. See Report below.

③ The active events of the equipment. See Active Event on page 314.

3.2.2 Report

▼ API: *GET /equipment/report*

```
GET http://<device-ip>/equipment/report HTTP/1.1

[
    {
        "children": [],     ①
        "point": [],        ②
        "event": [],        ③
        "mmCnt": "1",       ④
        "tag": "16384",     ⑤
        "id": "input"       ⑥
    }
]
```

① Report children. See Children on page 307.

② Report data points. See Point on page 309.

③ Report events. See Event on page 311.

④ The multi-module count, indicating how many values there should be for each data point. See Value on page 312.

⑤ The report tag.

⑥ The ID of the report.

▼ **API (cmd):** *equipment/report*

equipment/report: get

```
[
    {
        "children": [],    ①
        "point": [],       ②
        "event": [],       ③
        "mmCnt": "1",      ④
        "tag": "16384",    ⑤
        "id": "input"      ⑥
    }
]
```

① Report children. See Children on the facing page.

② Report data points. See Point on page 309.

③ Report events. See Event on page 311.

④ The multi-module count, indicating how many values there should be for each data point. See Value on page 312.

⑤ The report tag.

⑥ The ID of the report.

▼ **CLI:** *{"cmd": "get", "path": "equipment/report"}*

user> {"cmd": "get", "path": "equipment/report"}

```
- children:    ①
    ...
  point:       ②
    ...
  event:       ③
    ...
  mmCnt: 1     ④
  tag: 16384   ⑤
  id: input    ⑥
```

① Report children. See Children on the facing page.

② Report data points. See Point on page 309.

③ Report events. See Event on page 311.

④ The multi-module count, indicating how many values there should be for each data point. See Value on page 312.

⑤ The report tag.

⑥ The ID of the report.

Children

The report children follow the same structure as the top-level reports and can have its own children if applicable. See Report on page 305.

▼ API: *GET /equipment/report/<id>/children*

```
GET http://<device-ip>/equipment/report/<id>/children HTTP/1.1

[
    {
        "children": [],            ①
        "point": [],               ②
        "event": [],               ③
        "mmCnt": "1",              ④
        "tag": "26403",            ⑤
        "id": "batteryStatistics"  ⑥
     }
]
```

① Report children. See Children above.

② Report data points. See Point on page 309.

③ Report events. See Event on page 311.

④ The maximum multi-module count, indicating the maximum amount of values there can be for each data point. See Value on page 312.

⑤ The report tag.

⑥ The ID of the report.

▼ API (cmd): *equipment/report/<id>/children*

equipment/report/<id>/children: get

```
[
    {
        "children": [],           ①
        "point": [],              ②
        "event": [],              ③
        "mmCnt": "1",             ④
        "tag": "26403",           ⑤
        "id": "batteryStatistics" ⑥
    }
]
```

① Report children. See Children above.

② Report data points. See Point on page 309.

③ Report events. See Event on page 311.

④ The maximum multi-module count, indicating the maximum amount of values there can be for each data point. See Value on page 312.

⑤ The report tag.

⑥ The ID of the report.

▼ **CLI:** *{"cmd": "get", "path": "equipment/report/<id>/children"}*

user> {"cmd": "get", "path": "equipment/report/<id>/children"}

```
- children:              ①
   ...
  point:                 ②
   ...
  event:                 ③
   ...
  mmCnt: 1               ④
  tag: 26403             ⑤
  id: batteryStatistics  ⑥
```

① Report children. See Children on the previous page.

② Report data points. See Point on the facing page.

③ Report events. See Event on page 311.

④ The maximum multi-module count, indicating the maximum amount of values there can be for each data point. See Value on page 312.

⑤ The report tag.

⑥ The ID of the report.

Point

▼ API: *GET /equipment/report/<id>/point*

```
GET http://<device-ip>/equipment/report/<id>/point HTTP/1.1

[
 {
  "value": [],                                                                    ①
  "units": "VDC",                                                                 ②
  "type": "uint16",                                                               ③
  "desc": "The voltage between the positive and negative terminals of the DC bus." ④
  "tag": "6321",                                                                  ⑤
  "access": "read-only",                                                          ⑥
  "id": "upsDcInputVoltage"                                                       ⑦
 }
]
```

① Point values. See Value on page 312.

② The unit of measure.

③ The data type.

④ The description.

⑤ The tag.

⑥ The access rights.

⑦ The ID of the point object.

▼ API (cmd): *equipment/report/<id>/point*

equipment/report/<id>/point: get

```
[
 {
  "value": [],                                                                    ①
  "units": "VDC",                                                                 ②
  "type": "uint16",                                                               ③
  "desc": "The voltage between the positive and negative terminals of the DC bus.",④
  "tag": "6321",                                                                  ⑤
  "access": "read-only",                                                          ⑥
  "id": "upsDcInputVoltage"                                                       ⑦
 }
]
```

① Point values. See Value on page 312.

② The unit of measure.

③ The data type.

④ The description.

⑤ The tag.

⑥ The access rights.

⑦ The ID of the point object.

▼ CLI: *{"cmd": "get", "path": "equipment/report/<id>/point"}*

user> {"cmd": "get", "path": "equipment/report/<id>/point"}

```
- value:                                                                          ①
   ...
  units: VDC                                                                      ②
  type: uint16                                                                    ③
  desc: The voltage between the positive and negative terminals of the DC bus.    ④
  tag: 6321                                                                       ⑤
  access: read-only                                                               ⑥
  id: UPS DC input voltage                                                        ⑦
```

① Point values. See Value on page 312.

② The unit of measure.

③ The data type.

④ The description.

⑤ The tag.

⑥ The access rights.

⑦ The ID of the point object.

Event

▼ API: *GET /equipment/report/<id>/event*

```
GET http://<device-ip>/equipment/report/<id>/event HTTP/1.1

[
 {
  "value": [],                                                                    ①
  "desc": "The voltage between the positive and negative terminals of the DC bus.", ②
  "tag": "4382",                                                                  ③
  "id": "systemInputCurrentImbalance"                                             ④
 }
]
```

① Event values. See Value on the next page.

② The description.

③ The tag.

④ The ID of the event object.

▼ API (cmd): *equipment/report/<id>/event*

equipment/report/<id>/event: get

```
[
 {
  "value": [],                                                                    ①
  "desc": "The voltage between the positive and negative terminals of the DC bus.", ②
  "tag": "4382",                                                                  ③
  "id": "systemInputCurrentImbalance"                                             ④
 }
]
```

① Event values. See Value on the next page.

② The description.

③ The tag.

④ The ID of the event object.

▼ CLI: *{"cmd": "get", "path": "equipment/report/<id>/event"}*

user> {"cmd": "get", "path": "equipment/report/<id>/event"}

```
- value:
①
    ...
  desc: The voltage between the positive and negative terminals of the DC bus. ②
  tag: 4382                                                                    ③
  id: systemInputCurrentImbalance                                              ④
```

① Event values. See Value below.

② The description.

③ The tag.

④ The ID of the event object.

Value

The following structure is the same for points, events, and active events. See Point on page 309, Event on the previous page, and Active Event on page 314.

▼ API: *GET /equipment/report/<id>/point/<id>/value*

```
GET http://<device-ip>/equipment/report/<id>/point/<id>/value HTTP/1.1

[
    {
        "value": "off", ①
        "id": "[1]"     ②
    }
]
```

① The value of the data.

② The hierarchial ID of the object, based on the index of the report it resides in.

▼ API (cmd): *equipment/report/<id>/point/<id>/value*

equipment/report/<id>/point/<id>/value: get

```
[
    {
        "value": "off", ①
        "id": "[1]"     ②
    }
]
```

① The value of the data.

② The hierarchial ID of the object, based on the index of the report it resides in.

▼ CLI: *{"cmd": "get", "path": "equipment/report/<id>/point/<id>/value"}*

user> {"cmd": "get", "path": "equipment/report/<id>/point/<id>/value"}

```
- value: off ①
  id: [1]    ②
```

① The value of the data.

② The hierarchial ID of the object, based on the index of the report it resides in.

▼ API: *POST /equipment/report/<id>/point/<id>/value/<id>*

```
POST http://<device-ip>/equipment/report/<id>/point/<id>/value/<id> HTTP/1.1
Content-Type: application/json

{
    "value": "off"
}   ①
```

① The value of the data.

▼ API (cmd): *equipment/report/<id>/point/<id>/value/<id> : set*

equipment/report/<id>/point/<id>/value/<id>:

```
{
    "cmd": "set",
    "data": {
        "value": "off" ①
    }
}
```

① The value of the data.

▼ CLI: *{"cmd": "set", "path": "equipment/report/<id>/point/<id>/value/<id>"}*

user> {"cmd": "set", "path": "equipment/report/<id>/point/<id>/value/<id>", "data": {"value": ARGS}}

```
{"cmd": "set", "path": "equipment/report/<id>/point/<id>/value/<id>", "data":
{"value": "off"}} ①
```

① The value of the data.

3.2.3 Active Event

Active events contain any objects from the equipment's events that contains a value other than "Normal". Only the value objects that are not "Normal" will be displayed. See Event on page 311 and Value on page 312 for more information.

▼ API: *GET /equipment/activeEvent*

```
GET http://<device-ip>/equipment/activeEvent HTTP/1.1

[
 {
  "value": [],                                                                    ①
  "desc": "The voltage between the positive and negative terminals of the DC bus.",②
  "tag": "4382",                                                                  ③
  "id": "systemInputCurrentImbalance"                                             ④
 }
]
```

① Active event values. See Value on page 312.

② The description.

③ The tag.

④ The ID of the event object.

▼ API (cmd): *equipment/activeEvent*

equipment/activeEvent: get

```
[
 {
  "value": [],                                                                    ①
  "desc": "The voltage between the positive and negative terminals of the DC bus.",②
  "tag": "4382",                                                                  ③
  "id": "systemInputCurrentImbalance"                                             ④
 }
]
```

① Active event values. See Value on page 312.

② The description.

③ The tag.

④ The ID of the event object.

▼ CLI: *{"cmd": "get", "path": "equipment/activeEvent"}*

user> {"cmd": "get", "path": "equipment/activeEvent"}

```
value:                                                                    ①
 ...
desc: The voltage between the positive and negative terminals of the DC bus. ②
tag: 4382                                                                 ③
id: System Input Current Imbalance                                        ④
```

① Active event values. See Value on page 312.

② The description.

③ The tag.

④ The ID of the event object.

3.2.4 Multi-Module Information

GET the multi-module data of an <id>.

▼ API: *GET /equipment/report/outletGroup[mm]*

```
GET http://<device-ip>/equipment/report/outletGroup[mm] HTTP/1.1

{
    "children": [],    ①
    "point": [],       ②
    "event": [],       ③
    "mmCnt": "4",      ④
    "tag": "16392"     ⑤
}
```

① Report children. See Children on page 307.

② Report data points. See Point on page 309.

③ Report events. See Event on page 311.

④ The maximum multi-module count, indicating the maximum amount of values there can be for each data point. See Value on page 312.

⑤ The tag.

▼ **API (cmd):** *equipment/report/outletGroup[mm]*

equipment/report/outletGroup[mm]: get

```
{
    "children": [],    ①
    "point": [],       ②
    "event": [],       ③
    "mmCnt": "4",      ④
    "tag": "16392"     ⑤
}
```

① Report children. See Children on page 307.

② Report data points. See Point on page 309.

③ Report events. See Event on page 311.

④ The maximum multi-module count, indicating the maximum amount of values there can be for each data point. See Value on page 312.

⑤ The tag.

▼ **CLI:** *{"cmd": "get", "path": "equipment/report/outletGroup[mm]"}*

user> {"cmd": "get", "path": "equipment/report/outletGroup[mm]"}

```
children: ~   ①
point:        ②
...
event: ~      ③
mmCnt: 4      ④
tag: 16392    ⑤
```

① Report children. See Children on page 307.

② Report data points. See Point on page 309.

③ Report events. See Event on page 311.

④ The maximum multi-module count, indicating the maximum amount of values there can be for each data point. See Value on page 312.

⑤ The tag.

GET the values of an <id> in a report point.

▼ API: *GET /equipment/report/Battery/children/lithiumBatteryGroup[mm]/point/batteryStateOfHealth*

```
GET http://<device-ip>/equipment/report/Battery/children/lithiumBatteryGroup[mm]/po
int/batteryStateOfHealth HTTP/1.1

{
    "access": "read-only",              ①
    "units": "%",                       ②
    "type": "uint16",                   ③
    "value": [                          ④
        {
            "value": "87",
            "id": "[7]"
        },
        {
            "value": "79",
            "id": "[8]"
        },
        {
            "value": "90",
            "id": "[4]"
        },
        {
            "value": "91",
            "id": "[10]"
        },
        ...
    ],
    "desc": "Battery State of Health", ⑤
    "tag": "8341"                       ⑥
}
```

① The access rights.

② The unit of measure.

③ The data type.

④ The Battery State of Health values.

⑤ The description.

⑥ The tag.

▼ API: *equipment/report/Battery/children/lithiumBatteryGroup[mm]/point/batteryStateOfHealth*

equipment/report/Battery/children/lithiumBatteryGroup[mm]/point/batteryStateOfHealth: get

```
{
    "access": "read-only",              ①
    "units": "%",                       ②
    "type": "uint16",                   ③
    "value": [                          ④
        {
            "value": "87",
            "id": "[7]"
        },
        {
            "value": "79",
            "id": "[8]"
        },
        {
            "value": "90",
            "id": "[4]"
        },
        {
            "value": "91",
            "id": "[10]"
        },
        ...
    ],
    "desc": "Battery State of Health", ⑤
    "tag": "8341"                       ⑥
}
```

① The access rights.

② The unit of measure.

③ The data type.

④ The Battery State of Health values.

⑤ The description.

⑥ The tag.

▼ CLI: *{"cmd": "get", "path":*

"*equipment/report/battery/children/lithiumBatteryGroup[mm]/point/batteryStateOfHealth"}*

user> {"cmd": "get", "path":

"equipment/report/battery/children/lithiumBatteryGroup[mm]/point/batteryStateOfHealth"}

```
access: read-only              ①
units: "%"                     ②
type: uint16                   ③
value:                         ④
- value: 87
id: "[7]"
- value: 79
id: "[8]"
- value: 90
id: "[4]"
- value: 91
id: "[10]"
...
desc: Battery State of Health ⑤
tag: 8341                      ⑥
```

① The access rights.

② The unit of measure.

③ The data type.

④ The Battery State of Health values.

⑤ The description.

⑥ The tag.

3.2.5 Firmware

Reads verification and update status for equipment (device) firmware.

Firmware is uploaded with HTTP POST to /transfer/equipment/firmware; see Equipment Firmware Update for authentication, multipart form requirements, and restrictions, see Equipment Firmware Update on page 297. After upload, poll this GET periodically to track verification progress and final status (statusDescription, statusCode, and additionalInfo).

▼ API:*GET /equipment/firmware*

```
GET http://<device-ip>/equipment/firmware HTTP/1.1

{
    "hostFileSize": 0,           ①
    "hostFilename": "",          ②
    "additionalInfo": "",        ③
    "statusDescription": "Idle", ④
    "statusCode": 0              ⑤
}
```

① Firmware update host file size.

② Firmware update host filename.

③ Firmware update additional info.

④ Firmware update status description.

⑤ Firmware update status code.

▼ API (cmd):*equipment/firmware*

equipment/firmware: get

```
GET http://<device-ip>/equipment/firmware HTTP/1.1

{
    "hostFileSize": 0,           ①
    "hostFilename": "",          ②
    "additionalInfo": "",        ③
    "statusDescription": "Idle", ④
    "statusCode": 0              ⑤
}
```

① Firmware update host file size.

② Firmware update host filename.

③ Firmware update additional info.

④ Firmware update status description.

⑤ Firmware update status code.

If a non-firmware or otherwise invalid image is uploaded, verification can fail while the GET request itself still succeeds (retCode 0, retMsg Success). The failure is indicated by the firmware status fields inside data, for example:

▼ API: *GET /equipment/firmware (invalid firmware file)*

```
GET http://<device-ip>/equipment/firmware HTTP/1.1

{
    "data": {
        "hostFileSize": 42023,                                    ①
        "hostFilename": "Event Log.csv",                          ②
        "additionalInfo": "Image verify Failed / Update Abort",   ③
        "statusDescription": "Error",                             ④
        "statusCode": 3                                           ⑤
    },
    "retCode": 0,                                                 ⑥
    "retMsg": "Success"                                           ⑦
}
```

① Size in bytes of the uploaded file on the host.

② Original filename of the uploaded file.

③ Device-provided detail for the failure (wording may vary).

④ High-level firmware update status for this attempt.

⑤ Numeric firmware update status code for this attempt.

⑥ API result code for the GET; 0 means the status read succeeded, not that the image was valid. See Error Codes on page 22.

⑦ API result message for the GET.

▼ API (cmd): *equipment/firmware (invalid firmware file)*

equipment/firmware: get (invalid firmware file)

```
{
    "data": {
        "hostFileSize": 42023,                                    ①
        "hostFilename": "Event Log.csv",                          ②
        "additionalInfo": "Image verify Failed / Update Abort",   ③
        "statusDescription": "Error",                             ④
        "statusCode": 3                                           ⑤
        },
        "retCode": 0,                                             ⑥
        "retMsg": "Success"                                       ⑦
}
```

① Size in bytes of the uploaded file on the host.

② Original filename of the uploaded file.

③ Device-provided detail for the failure (wording may vary).

④ High-level firmware update status for this attempt.

⑤ Numeric firmware update status code for this attempt.

⑥ API result code for the GET; 0 means the status read succeeded, not that the image was valid. See Error Codes on page 22.

⑦ API result message for the GET.

▼ CLI: *{"cmd": "get", "path": "equipment/firmware"}*

user> {"cmd": "get", "path": "equipment/firmware"}

```
hostFileSize: 0          ①
hostFilename: ""         ②
additionalInfo: ""       ③
statusDescription: Idle  ④
statusCode: 0            ⑤
```

① Firmware update host file size.

② Firmware update host filename.

③ Firmware update additional info.

④ Firmware update status description.

⑤ Firmware update status code.

Appendices

Appendix A: Technical Support and Contacts

A.1 Technical Support/Service in the United States

Vertiv Group Corporation

24x7 dispatch of technicians for all products.

1-800-543-2378

Liebert® Thermal Management Products

1-800-543-2778

Liebert® Channel Products

1-800-222-5877

Liebert® AC and DC Power Products

1-800-543-2378

A.2 Locations

United States

Vertiv Headquarters

505 N Cleveland Ave

Westerville, OH, 43082, USA

Europe

Victor-von-Bruns Strasse 21,

8212 Neuhausen am Rheinfall, Switzerland

Asia

Singapore

151 Lorong Chuan, Lobby D #05-04

New Tech Park, Singapore 556741

India

Vertiv Energy Private Limited

Plot No. C 20, Road No. 19

Wagle Industrial Estate, MIDC

Thane (West), Maharashtra 400604, India

China

Vertiv Technology Co., Limited

Floors 1–4 and 6–10,

Building B2, Nanshan I Park

No. 1001 Xueyuan Road, Nanshan District

Shenzhen, Guangdong 518055, China

Appendix B: Time Zone

Appendix is a collection of supplementary information. The following time zones are valid options from the Time Zone Database (tzdata), version 2021a.

▼ Time Zones

- Africa/Abidjan
- Africa/Accra
- Africa/Addis_Ababa
- Africa/Algiers
- Africa/Asmara
- Africa/Asmera
- Africa/Bamako
- Africa/Bangui
- Africa/Banjul
- Africa/Bissau
- Africa/Blantyre
- Africa/Brazzaville
- Africa/Bujumbura
- Africa/Cairo
- Africa/Casablanca
- Africa/Ceuta
- Africa/Conakry
- Africa/Dakar
- Africa/Dar_es_Salaam
- Africa/Djibouti
- Africa/Douala
- Africa/El_Aaiun
- Africa/Freetown
- Africa/Gaborone
- Africa/Harare
- Africa/Johannesburg
- Africa/Juba
- Africa/Kampala
- Africa/Khartoum
- Africa/Kigali
- Africa/Kinshasa
- Africa/Lagos
- Africa/Libreville
- Africa/Lome
- Africa/Luanda

- Africa/Lubumbashi
- Africa/Lusaka
- Africa/Malabo
- Africa/Maputo
- Africa/Maseru
- Africa/Mbabane
- Africa/Mogadishu
- Africa/Monrovia
- Africa/Nairobi
- Africa/Ndjamena
- Africa/Niamey
- Africa/Nouakchott
- Africa/Ouagadougou
- Africa/Porto-Novo
- Africa/Sao_Tome
- Africa/Timbuktu
- Africa/Tripoli
- Africa/Tunis
- Africa/Windhoek
- America/Adak
- America/Anchorage
- America/Anguilla
- America/Antigua
- America/Araguaina
- America/Argentina/Buenos_Aires
- America/Argentina/Catamarca
- America/Argentina/ComodRivadavia
- America/Argentina/Cordoba
- America/Argentina/Jujuy
- America/Argentina/La_Rioja
- America/Argentina/Mendoza
- America/Argentina/Rio_Gallegos
- America/Argentina/Salta
- America/Argentina/San_Juan
- America/Argentina/San_Luis
- America/Argentina/Tucuman
- America/Argentina/Ushuaia
- America/Aruba
- America/Asuncion
- America/Atikokan
- America/Atka

- America/Bahia
- America/Bahia_Banderas
- America/Barbados
- America/Belem
- America/Belize
- America/Blanc-Sablon
- America/Boa_Vista
- America/Bogota
- America/Boise
- America/Buenos_Aires
- America/Cambridge_Bay
- America/Campo_Grande
- America/Cancun
- America/Caracas
- America/Catamarca
- America/Cayenne
- America/Cayman
- America/Chicago
- America/Chihuahua
- America/Coral_Harbour
- America/Cordoba
- America/Costa_Rica
- America/Creston
- America/Cuiaba
- America/Curacao
- America/Danmarkshavn
- America/Dawson
- America/Dawson_Creek
- America/Denver
- America/Detroit
- America/Dominica
- America/Edmonton
- America/Eirunepe
- America/El_Salvador
- America/Ensenada
- America/Fort_Nelson
- America/Fort_Wayne
- America/Fortaleza
- America/Glace_Bay
- America/Godthab
- America/Goose_Bay

- America/Grand_Turk
- America/Grenada
- America/Guadeloupe
- America/Guatemala
- America/Guayaquil
- America/Guyana
- America/Halifax
- America/Havana
- America/Hermosillo
- America/Indiana/Indianapolis
- America/Indiana/Knox
- America/Indiana/Marengo
- America/Indiana/Petersburg
- America/Indiana/Tell_City
- America/Indiana/Vevay
- America/Indiana/Vincennes
- America/Indiana/Winamac
- America/Indianapolis
- America/Inuvik
- America/Iqaluit
- America/Jamaica
- America/Jujuy
- America/Juneau
- America/Kentucky/Louisville
- America/Kentucky/Monticello
- America/Knox_IN
- America/Kralendijk
- America/La_Paz
- America/Lima
- America/Los_Angeles
- America/Louisville
- America/Lower_Princes
- America/Maceio
- America/Managua
- America/Manaus
- America/Marigot
- America/Martinique
- America/Matamoros
- America/Mazatlan
- America/Mendoza
- America/Menominee

- America/Merida
- America/Metlakatla
- America/Mexico_City
- America/Miquelon
- America/Moncton
- America/Monterrey
- America/Montevideo
- America/Montreal
- America/Montserrat
- America/Nassau
- America/New_York
- America/Nipigon
- America/Nome
- America/Noronha
- America/North_Dakota/Beulah
- America/North_Dakota/Center
- America/North_Dakota/New_Salem
- America/Nuuk
- America/Ojinaga
- America/Panama
- America/Pangnirtung
- America/Paramaribo
- America/Phoenix
- America/Port-au-Prince
- America/Port_of_Spain
- America/Porto_Acre
- America/Porto_Velho
- America/Puerto_Rico
- America/Punta_Arenas
- America/Rainy_River
- America/Rankin_Inlet
- America/Recife
- America/Regina
- America/Resolute
- America/Rio_Branco
- America/Rosario
- America/Santa_Isabel
- America/Santarem
- America/Santiago
- America/Santo_Domingo
- America/Sao_Paulo

- America/Scoresbysund
- America/Shiprock
- America/Sitka
- America/St_Barthelemy
- America/St_Johns
- America/St_Kitts
- America/St_Lucia
- America/St_Thomas
- America/St_Vincent
- America/Swift_Current
- America/Tegucigalpa
- America/Thule
- America/Thunder_Bay
- America/Tijuana
- America/Toronto
- America/Tortola
- America/Vancouver
- America/Virgin
- America/Whitehorse
- America/Winnipeg
- America/Yakutat
- America/Yellowknife
- Antarctica/Casey
- Antarctica/Davis
- Antarctica/DumontDUrville
- Antarctica/Macquarie
- Antarctica/Mawson
- Antarctica/McMurdo
- Antarctica/Palmer
- Antarctica/Rothera
- Antarctica/South_Pole
- Antarctica/Syowa
- Antarctica/Troll
- Antarctica/Vostok
- Arctic/Longyearbyen
- Asia/Aden
- Asia/Almaty
- Asia/Amman
- Asia/Anadyr
- Asia/Aqtau
- Asia/Aqtobe

- Asia/Ashgabat
- Asia/Ashkhabad
- Asia/Atyrau
- Asia/Baghdad
- Asia/Bahrain
- Asia/Baku
- Asia/Bangkok
- Asia/Barnaul
- Asia/Beirut
- Asia/Bishkek
- Asia/Brunei
- Asia/Calcutta
- Asia/Chita
- Asia/Choibalsan
- Asia/Chongqing
- Asia/Chungking
- Asia/Colombo
- Asia/Dacca
- Asia/Damascus
- Asia/Dhaka
- Asia/Dili
- Asia/Dubai
- Asia/Dushanbe
- Asia/Famagusta
- Asia/Gaza
- Asia/Harbin
- Asia/Hebron
- Asia/Ho_Chi_Minh
- Asia/Hong_Kong
- Asia/Hovd
- Asia/Irkutsk
- Asia/Istanbul
- Asia/Jakarta
- Asia/Jayapura
- Asia/Jerusalem
- Asia/Kabul
- Asia/Kamchatka
- Asia/Karachi
- Asia/Kashgar
- Asia/Kathmandu
- Asia/Katmandu

- Asia/Khandyga
- Asia/Kolkata
- Asia/Krasnoyarsk
- Asia/Kuala_Lumpur
- Asia/Kuching
- Asia/Kuwait
- Asia/Macao
- Asia/Macau
- Asia/Magadan
- Asia/Makassar
- Asia/Manila
- Asia/Muscat
- Asia/Nicosia
- Asia/Novokuznetsk
- Asia/Novosibirsk
- Asia/Omsk
- Asia/Oral
- Asia/Phnom_Penh
- Asia/Pontianak
- Asia/Pyongyang
- Asia/Qatar
- Asia/Qostanay
- Asia/Qyzylorda
- Asia/Rangoon
- Asia/Riyadh
- Asia/Saigon
- Asia/Sakhalin
- Asia/Samarkand
- Asia/Seoul
- Asia/Shanghai
- Asia/Singapore
- Asia/Srednekolymsk
- Asia/Taipei
- Asia/Tashkent
- Asia/Tbilisi
- Asia/Tehran
- Asia/Tel_Aviv
- Asia/Thimbu
- Asia/Thimphu
- Asia/Tokyo
- Asia/Tomsk

- Asia/Ujung_Pandang
- Asia/Ulaanbaatar
- Asia/Ulan_Bator
- Asia/Urumqi
- Asia/Ust-Nera
- Asia/Vientiane
- Asia/Vladivostok
- Asia/Yakutsk
- Asia/Yangon
- Asia/Yekaterinburg
- Asia/Yerevan
- Atlantic/Azores
- Atlantic/Bermuda
- Atlantic/Canary
- Atlantic/Cape_Verde
- Atlantic/Faeroe
- Atlantic/Faroe
- Atlantic/Jan_Mayen
- Atlantic/Madeira
- Atlantic/Reykjavik
- Atlantic/South_Georgia
- Atlantic/St_Helena
- Atlantic/Stanley
- Australia/ACT
- Australia/Adelaide
- Australia/Brisbane
- Australia/Broken_Hill
- Australia/Canberra
- Australia/Currie
- Australia/Darwin
- Australia/Eucla
- Australia/Hobart
- Australia/LHI
- Australia/Lindeman
- Australia/Lord_Howe
- Australia/Melbourne
- Australia/NSW
- Australia/North
- Australia/Perth
- Australia/Queensland
- Australia/South

- Australia/Sydney
- Australia/Tasmania
- Australia/Victoria
- Australia/West
- Australia/Yancowinna
- Brazil/Acre
- Brazil/DeNoronha
- Brazil/East
- Brazil/West
- CET
- CST6CDT
- Canada/Atlantic
- Canada/Central
- Canada/Eastern
- Canada/Mountain
- Canada/Newfoundland
- Canada/Pacific
- Canada/Saskatchewan
- Canada/Yukon
- Chile/Continental
- Chile/EasterIsland
- Cuba
- EET
- EST
- EST5EDT
- Egypt
- Eire
- Etc/GMT
- Etc/GMT+0
- Etc/GMT+1
- Etc/GMT+10
- Etc/GMT+11
- Etc/GMT+12
- Etc/GMT+2
- Etc/GMT+3
- Etc/GMT+4
- Etc/GMT+5
- Etc/GMT+6
- Etc/GMT+7
- Etc/GMT+8
- Etc/GMT+9

- Etc/GMT-0
- Etc/GMT-1
- Etc/GMT-10
- Etc/GMT-11
- Etc/GMT-12
- Etc/GMT-13
- Etc/GMT-14
- Etc/GMT-2
- Etc/GMT-3
- Etc/GMT-4
- Etc/GMT-5
- Etc/GMT-6
- Etc/GMT-7
- Etc/GMT-8
- Etc/GMT-9
- Etc/GMT0
- Etc/Greenwich
- Etc/UCT
- Etc/UTC
- Etc/Universal
- Etc/Zulu
- Europe/Amsterdam
- Europe/Andorra
- Europe/Astrakhan
- Europe/Athens
- Europe/Belfast
- Europe/Belgrade
- Europe/Berlin
- Europe/Bratislava
- Europe/Brussels
- Europe/Bucharest
- Europe/Budapest
- Europe/Busingen
- Europe/Chisinau
- Europe/Copenhagen
- Europe/Dublin
- Europe/Gibraltar
- Europe/Guernsey
- Europe/Helsinki
- Europe/Isle_of_Man
- Europe/Istanbul

- Europe/Jersey
- Europe/Kaliningrad
- Europe/Kiev
- Europe/Kirov
- Europe/Lisbon
- Europe/Ljubljana
- Europe/London
- Europe/Luxembourg
- Europe/Madrid
- Europe/Malta
- Europe/Mariehamn
- Europe/Minsk
- Europe/Monaco
- Europe/Moscow
- Europe/Nicosia
- Europe/Oslo
- Europe/Paris
- Europe/Podgorica
- Europe/Prague
- Europe/Riga
- Europe/Rome
- Europe/Samara
- Europe/San_Marino
- Europe/Sarajevo
- Europe/Saratov
- Europe/Simferopol
- Europe/Skopje
- Europe/Sofia
- Europe/Stockholm
- Europe/Tallinn
- Europe/Tirane
- Europe/Tiraspol
- Europe/Ulyanovsk
- Europe/Uzhgorod
- Europe/Vaduz
- Europe/Vatican
- Europe/Vienna
- Europe/Vilnius
- Europe/Volgograd
- Europe/Warsaw
- Europe/Zagreb

- Europe/Zaporozhye
- Europe/Zurich
- Factory
- GB
- GB-Eire
- GMT
- GMT+0
- GMT-0
- GMT0
- Greenwich
- HST
- Hongkong
- Iceland
- Indian/Antananarivo
- Indian/Chagos
- Indian/Christmas
- Indian/Cocos
- Indian/Comoro
- Indian/Kerguelen
- Indian/Mahe
- Indian/Maldives
- Indian/Mauritius
- Indian/Mayotte
- Indian/Reunion
- Iran
- Israel
- Jamaica
- Japan
- Kwajalein
- Libya
- MET
- MST
- MST7MDT
- Mexico/BajaNorte
- Mexico/BajaSur
- Mexico/General
- NZ
- NZ-CHAT
- Navajo
- PRC
- PST8PDT

- Pacific/Apia
- Pacific/Auckland
- Pacific/Bougainville
- Pacific/Chatham
- Pacific/Chuuk
- Pacific/Easter
- Pacific/Efate
- Pacific/Enderbury
- Pacific/Fakaofo
- Pacific/Fiji
- Pacific/Funafuti
- Pacific/Galapagos
- Pacific/Gambier
- Pacific/Guadalcanal
- Pacific/Guam
- Pacific/Honolulu
- Pacific/Johnston
- Pacific/Kiritimati
- Pacific/Kosrae
- Pacific/Kwajalein
- Pacific/Majuro
- Pacific/Marquesas
- Pacific/Midway
- Pacific/Nauru
- Pacific/Niue
- Pacific/Norfolk
- Pacific/Noumea
- Pacific/Pago_Pago
- Pacific/Palau
- Pacific/Pitcairn
- Pacific/Pohnpei
- Pacific/Ponape
- Pacific/Port_Moresby
- Pacific/Rarotonga
- Pacific/Saipan
- Pacific/Samoa
- Pacific/Tahiti
- Pacific/Tarawa
- Pacific/Tongatapu
- Pacific/Truk
- Pacific/Wake

- Pacific/Wallis
- Pacific/Yap
- Poland
- Portugal
- ROC
- ROK
- Singapore
- Turkey
- UCT
- US/Alaska
- US/Aleutian
- US/Arizona
- US/Central
- US/East-Indiana
- US/Eastern
- US/Hawaii
- US/Indiana-Starke
- US/Michigan
- US/Mountain
- US/Pacific
- US/Samoa
- UTC
- Universal
- W-SU
- WET
- Zulu

Appendix C: Service Center Country

Used to help identify a remote service card during initial startup. The following are options available for service center country.

▼ Service Center Country

- Argentina
- Australia
- Austria
- Azerbaijan
- Botswana
- Brazil
- Bulgaria
- Canada
- Chile
- China
- Czech Republic
- France
- Greece
- Germany
- India
- Italy
- Japan
- Kazakhstan
- Lesotho
- Mexico
- Mozambique
- Netherlands
- Other
- Poland
- Portugal
- Romania
- Russia
- Saudi Arabia
- Singapore
- Slovakia
- South Africa
- Spain
- Sweden
- Thailand
- Turkey

- United Arab Emirates
- United Kingdom
- United States

This page intentionally left blank

Connect with Vertiv on Social Media

 https://www.facebook.com/vertiv/

 https://www.instagram.com/vertiv/

 https://www.linkedin.com/company/vertiv/

 https://www.x.com/Vertiv/

Vertiv.com | Vertiv Headquarters, 505 N Cleveland Ave, Westerville, OH 43082 USA

AV-50017_REVB_09-26