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 relation 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

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.
1.1.0-1	Added transfer API.
1.3.0-1	Removed SNMPv3 trustedAccess and range alignment.
1.4.0-1	• Added equipment API. • Added missing API and CLI examples for add, delete, set, sendTest commands for nodes in /api/conf and /api/auth. • Updated Object, Data, and Notes for some nodes in /api/conf and /api/auth table. • Updated API reponse block with correct reponse for some nodes.
1.4.0-2	• Added switchFirmware command to api/sys • Added upload error codes 7008 - 7011. • Added Service Center Country options to appendix. • Updated range limits for email, Modbus TCP, remote authentication, SNMP v1v2c, SNMP v3, and webserver. • Updated auth passwordRules get response.

1.8 API Version

Table 1.2 API Version

Version	Product	Notes
1.3.0, March 13, 2025	v1.3.x	-

1.9 Glossary

Table 1.3 Glossary/Compatibility

Version	Note
v1	RDU120 running firmware 1.3.0 or newer

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.

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

Client requests are generally in the form of HTTP POSTs to an API path, with a JSON object in the body. Required request attributes are based upon the command. The following is an example request body for a get command that requires an authorization token:

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

① A session ID issued by the server on **logins**. 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 29.

② Possible commands are: "get", "set", "merge", "add", "control", "delete", "ack", "login", "logout", "reset", "sendTest", and "reboot". If no command is supplied in either the JSON object or the query string, one will be determined based on the HTTP method used: POST defaults to "set", GET defaults to "get", PUT defaults to "add", and DELETE defaults to "delete".

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 14.

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 above 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.

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 15 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

API commands are used to specify the API operation to be performed. The following API commands are supported; **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:

Request on /api/conf/email/

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

Response

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

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

Request on /api/conf/email/target

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

Response

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

Request on /api/conf/email/target/0

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

Response

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

And finally at a leaf:

Request on /api/conf/email/target/0/name

```
{
    "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:

Request on /api

```
{
    "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:

Request on /api/conf/email/target/0/name

```
{
    "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.

Request on /api/auth/user

```
{
    "username": "mricis-main",
    "password": "Mricismain123$",
    "cmd": "add",
    "data": {
        "id": "merger1",
        "password": "Mricismain123$",
        "source": "local",
        "scope": null,
        "enabled": true,
        "language": "en",
        "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.

Request on /api/auth/user

```
{
    "username": "mricis-main",
    "password": "Mricismain123$",
    "cmd": "add",
    "data": {
        "id": "merger1",
        "password": "Mricismain123$",
        "source": "local",
        "scope": null,
        "enabled": true,
        "language": "en",
        "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.

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" and that user's language would be changed to English. 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": "mricis-main",
    "password": "Mricismain123$",
    "cmd": "merge",
    "data": {
        "id": "merger",
        "password": "Merger123$",
        "source": "local",
        "scope": null,
        "enabled": true,
        "language": "en",
        "publicKey": [],
        "privilege": "administrator"
    }
}
```

▼ CLI: merge auth

admin> merge auth = ARGS

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

Control

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

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 'mricis-main' administrator account.

Request on /api/auth/user/myuser

```
{
    "username": "mricis-main",
    "password": "Mricismain123$",
    "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.

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.

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.

Request on /api/auth/user/mricis-main

```
{
    "cmd": "login",
    "username": "mricis-main",
    "password": "Mricismain123!"
}
```

Response

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

Attribute	Description
Token	The API token used in subsequent authenticated API calls.
Language	The language configured for this user.
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.

Request on /api/auth/user/user4

```
{
    "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.

Request on /api/sys

```
{
    "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.

Request on /api/conf/email/target/1

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

Response

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

Reboot

The Reboot command is used to reboot the system.

Request on /api/sys

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

Response

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

2.1.5 Filtering

Filters are JSON objects defined under the "filter" key at the top level of the API request object. Filters 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.

Request on /api

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

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

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
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 14, 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
        language: en
        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 14.

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: john.z.smith@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".
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 29.	Array	1 to 17 objects	"guest"	Commands: add, delete, login, logout
id	String			
enabled	Boolean	true, false	true	
privilege	String	"administrator", "control", "view"	"view"	
language	Language Code	"en"	"en"	Can also be set by user
active	Boolean	true, false	false	read-only
password	Password	1 to String Max		Can also be set by user, get returns null
passwordSet	Boolean	true, false		read-only
scope	String		null	
source	String	"local", "remoteAuth", "factory", "unknown"	local	read-only
user/publicKey, see SSH Public Key Authentication on page 33.	Array	1 to 5 objects		User can also add, delete
id	String			read-only
label	String	0 to String Max	"key"	Can also be set by user
key	String	0 to 800 chars		Can also be set by user
conf/password Rules, see Password Rules on page 36.				
allowRepeatCharacters	Boolean	true, false	true	
allowUsernameInclusion	Boolean	true, false	true	
minDigits	Integer	0 to String Max	0	
minLength	Integer	6 to String Max	8	
minSymbols	Integer	0 to String Max	0	
minUppercase	Integer	0 to String Max	0	
conf/scope, see Scope on page 37.	Array	0 to 10 objects		Commands: add, delete
id	String			read-only
name	String	0 to String Max		read-only

Object	Data			Notes
Field	Format	Range	Default	set, add, delete on these objects require admin privilege
filter	JSON Object		{}	
label	String	0 to String Max	auth/scope/ID/name	
remoteAttribute	String	0 to String Max		

The ▼ API: api/auth/user: get on the facing 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/auth/user.

An authenticated user can set their own "language", "publicKey" and "password", but "privilege" and "scope" can only be set by admin privileged users. Restrictions on password security may be configured in Authentication Configuration: /api/auth/conf on page 36. 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: api/auth/user: get on the facing 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 15 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 15 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 15 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 /api/dev. 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 40, /api/dev, Configuration: /api/sys on page 143, and ▼ API: api/auth/user: get on the facing 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 Configuration: /api/sys on page 143.
- **add** an admin user on ▼ API: api/auth/user: get on the facing page.
- **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 15.

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 15.

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". The "language" field is the same as /api/conf/locale/defaultUserLanguage: if either field changes, they both change, see Locale on page 73. If guest is disabled, unauthenticated get commands on all API branches except Configuration: /api/sys on page 143 will return a "1003 - Not Authorized: Not enough permissions" error, as shown in Table 2.2 on page 15. 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: api/conf/remoteAuth/ldap on page 96, ▼ API: api/conf/remoteAuth/tacacs on page 100, and ▼ API: api/conf/remoteAuth/radius on page 98 configuration sections. Remotely authenticated users always use the system language /api/conf/locale/defaultUserLanguage, see Locale on page 73.

▼ **API:** api/auth/user: get

api/auth/user: get

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

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 15. 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 Overrides /api/conf/locale/defaultUserLanguage for this user, see Locale on page 73.

⑤ Indicates whether the user is logged in.

⑥ "set" operations use "Password on page 26" strings, "get" operations return only null.

⑦ Indicates whether the password field has been set.

⑧ Will return "local" for local users, "remoteAuth" for remotely authenticated users, "factory" for factory users, or "unknown" for none of the above.

⑨ api/auth/user/publicKey. See SSH Public Key Authentication on page 33.

⑩ If null, user has full API access, up to their permissions level. If equal to a key in Scope on page 37, restricts user access according to that scope.

▼ **CLI:** get auth user

user> get auth user

```
-   id: guest                ①
    enabled: true            ②
    privilege: administrator ③
    language: en             ④
    active: false            ⑤
    password: ~              ⑥
    passwordSet: false       ⑦
    source: local            ⑧
    publicKey:               ⑨
        ...
    scope: 1                 ⑩
```

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 15. 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 Overrides /api/conf/locale/defaultUserLanguage for this user, see Locale on page 73.

⑤ Indicates whether the user is logged in.

⑥ "set" operations use "Password on page 26" strings, "get" operations return only null.

⑦ Indicates whether the password field has been set.

⑧ Will return "local" for local users, "remoteAuth" for remotely authenticated users, "factory" for factory users, or "unknown" for none of the above.

⑨ api/auth/user/publicKey. See SSH Public Key Authentication on page 33.

⑩ If null, user has full API access, up to their permissions level. If equal to a key in Scope on page 37, restricts user access according to that scope.

Command: add

▼ **API:** api/auth/user: add (Admin)

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

```
{
    "cmd": "add",
    "data":{
    "id": "admin",                ①
    "password": "abcd1234",       ②
    "language": "en",             ③
    "enabled": true,              ④
    "privilege": "administrator", ⑤
    "scope": null                 ⑥
    }
}
```

① Required field. Must comply with "Username on page 26" rules.

② Required field. Must comply with "Password on page 26" rules.

③ Overrides /api/conf/locale/defaultUserLanguage for this user. See Locale on page 73.

④ If false, attempts to log in as this user will return a "1001 - Not Authorized" error, as shown in **Table 2.2** on page 15. If true, the user has basic "get" permissions.

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

▼ **CLI:** add auth user

admin> add auth user = ARGS

```
add auth user = {"id": "adminCL2", "password": "abc1234", "language": "en", "enabled":
true, "privilege": "administrator", "scope": null} ①②③④⑤⑥⑦
```

① Required fields: id, password.

② ID must comply with "Username on page 26" rules.

③ Password must comply with "Password on page 26" rules.

④ Language overrides /api/conf/locale/defaultUserLanguage for this user. See Locale on page 73.

⑤ 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 15. If true, the user has basic "get" permissions.

⑥ 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.

⑦ Scope is optional, and will default to null if not specified.

Command: delete

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

api/auth/user/<id>: delete (Admin)

```
{
"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/auth/user/ldapadminuser: login

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

login return data

```
{
    "token": "f3e2d1c0",          ①
    "language": "en",
    "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/auth/user/admin: logout *(Same user, in this case "admin")*

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

logout return data

```
{}
```

2.2.5 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 5 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: 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:** api/auth/user/admin/publicKey: add *(admin)*

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

```
{
    "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:** *api/auth/user/admin/publicKey/ID:* delete *(admin)*

api/auth/user/admin/publicKey/ID: delete *(admin)*

```
{
  "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:** 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.

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 94. "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: api/auth/user: get on page 29, Configuration: /api/sys on page 143, 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:** 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:** api/auth/conf/scope: add (Admin)

api/auth/conf/scope: add *(Admin)*

```
{
    "cmd": "add",
    "data": {
      "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 94.

IMPORTANT! CLI add is currently not working for scope.

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

admin> add auth conf scope = ARGS

```
add conf auth scope =
{"filter":{"dev":{"ABC1234567890DEF":{"outlet":{"1":null,"2":null,"3":null}},"G09876
54321HIJKL":null}},"label":"Scope 0","remoteAttribute":""}
①②③
```

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 94.

Command: delete

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

api/auth/conf/scope/<id>: delete *(admin)*

```
{
"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.

2.4 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 53.				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 54.				set
bbmdAddress	IP Address		""	
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 56.	Array	0 to 5 trustedAccess entries		add, delete, set
id	String	maxLength 32		read-only
address	IP Address			
prefix	Integer	0 to 128	0	
bacnet/server/mstp, see BACnet MSTP on page 58.				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 60.				set
sessionIdleTimeout	Integer	1 to 600	5	minutes
cloud/remoteService, see Cloud on page 62.				set
connectionTestResult	String	"SUCCESS", "FAILURE"		read-only
connectionRetryTime	Integer	30 to 600	30	seconds
serviceCenterCountry	String	Service Center Country	""	
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 Cloud on page 62.				read-only
cloudMessage	String	0 to String Max		
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	0 to String Max		
lastError	String	0 to String Max	""	
lastSuccess	String	0 to String Max		

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 Configuration: /api/conf on page 40.				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	
email, see Email on page 67.				set
passwordSet	Boolean	true, false	false	read-only
port	Integer	1 to 65535	25	
sender	String	maxLength 120		
server	IP Address	maxLength 64		
password	String	maxLength 64		
subjectPrefix	String	maxLength 125		
username	String	maxLength 64		
email/target, see Email Target on page 69.	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 72.				set
enabled	Boolean	true, false	false	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
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 73.				
defaultUserLanguage	String	"en"	"en"	English only
"measurementSystem"	String	"us", "metric"	"us"	set
"systemNotificationsLanguage"	String	"en"	"en"	English only
modbus/server, see Modbus Server on page 75.				set
access	String	"readOnly", "readWrite"	"readOnly "	
mode	String	"disabled", "tcp", "rtu"	"disabled"	
modbus/server/rtu, see Modbus RTU on page 76.				set
baudRate	String	"9600", "19200", "38400"	"9600"	
nodeId	Integer	1 to 247	1	
portParameters	String	"8N2", "8E1", "8O1", "8N1", "8E2", "8O2"	"8N2"	
modbus/server/tcp, see Modbus TCP on page 77.				set
maxClients	Integer	0 to 5	4	
port	Integer	1 to 32767	502	
modbus/server/tcp/trustedAccess, see Modbus TCP Trusted Access on page 78.	Array	0 to 5 trustedAccess entries		add, delete, set
id	String	maxLength 32		read-only
address	IP Address			
prefix	Integer	0 to 128	24	
network, see Network on page 80.				set
hostname	String	maxLength 63	"rdu120-" + macAddr	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
dnsSearchList	String	maxLength 100		
defaultIpEnabled	Boolean	true, false	true	
ip4Enabled	Boolean	true, false	true	
ip6Enabled	Boolean	true, false	true	
ipV4GW	IP Address			
ipV6GW	IP Address			
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 83.	Array	add up to 4 DNS addresses		add, delete
id	String	0 to String Max		read-only
address	IP Address			
mutable	Boolean	true or false	true	read-only
network/route	Array			add, delete
id	String	0 to String Max		read-only
destination	IP Address	maxLength 125	""	
gateway	IP Address	maxLength 125	""	
interface	String	0 to String Max		
mutable	Boolean	true or false		read-only
prefix	Integer	0 to 32		
network/interface/lan, see Network/Interface/LAN on page 87.				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"	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
macAddr	String	maxLength 18		read-only
dhcpEnabled	Boolean	true, false	true	read-only
network/interface/lan/address, see Network/Interface/LAN/Address on page 89.	Array	Up to 8 addresses		add, delete
id	String	0 to String Max		read-only
address	IP Address			
prefix	Integer	0 to 128		
mutable	Boolean	true or false		read-only
network/interface/lan/link, see Network/Interface/LAN/Link on page 91.				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
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 91.				
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 93.				set
enabled	Boolean	true, false	false	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
identity	String	maxLength 32		Cannot be empty if enabled is true
privKeyPassword	String	0 to String Max		
remoteAuth, see Remote Authentication on page 94.				set
mode	String	"none", "ldap", "tacacs", "radius"	"none"	
remoteAuth/ldap, see Remote Authentication on page 94.				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	""	
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 98.				set
adminGroup	String	maxLength 120	"admin"	
authenticationServer1	String	maxLength 120	""	
authenticationServer2	String	maxLength 120	""	
controlGroup	String	maxLength 120	"control"	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
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 100.				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 102.				set
enabled	Boolean	true or false	false	
port	Integer	1 to 65535	514	
target	String	maxlength 120	""	
serial, see Serial on page 103.				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 104.				set
heartBeatTrapInterval	Integer	0, 5, 30, 60, 240, 480, 720, 1440	1440	0 means disabled, minutes
Port	Integer	1 to 65535	161	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
authTrapsEnabled	Boolean	true or false	false	
snmp/mibSupport, see SNMP MIB Support on page 106.				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 107.				set
enabled	Boolean	true or false	false	
snmp/v1v2c/access, see SNMP v1/v2c Access on page 108.	Array	0 to 8 entries		add, delete, set
id	String			read-only
access	String	"read", "write"	"read"	
community	String	maxLength 32	""	
snmp/v1v2c/access/trustedAccess, see SNMP v1/v2c Trusted Access on page 111.	Array	0 to 1 entries		add, delete, set
id	String	maxLength 32		read-only
address	IP Address			
prefix	Integer	0 to 128	24	
snmp/v1v2c/trap, see SNMP v1/v2c Trap on page 113.	Array	0 to 8 entries		add, delete, set, sendTest
id	String			read-only
community	String	maxLength 32	""	
port	Integer	1 to 65535	162	
target	String	maxLength 63	""	
snmp/v1v2c/generateTestTraps				sendTest
snmp/v3, see SNMP v3 on page 116.				set
enabled	Boolean	true or false	false	
engineId	String	maxLength 64	""	read-only

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
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 117.	Array	0 to 8 users		add, delete, set
id	String			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"	
username	String	maxLength 64	""	
snmp/v3/user/target, see SNMP v3 User Trap Target on page 121.	Array	0 to 2 entries		add, delete, set, sendTest
id	String			read-only
port	Integer	1 to 65535	162	
target	String	maxLength 63	""	
snmp/v3/generate TestTraps				sendTest
ssh SSH on page 1	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 125.				set
contact	String	maxLength 120	""	
description	String	maxLength 120	""	
label	String	maxLength 120	"mricis- evk"	Populates /api/sys/name
location	String	maxLength 120	""	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
factoryAccessEnabled	Boolean	true or false	false	Enables factory access functions.
time, see Time on page 126.				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**	"UTC"	
ntpServer1	String	maxLength 64	"0.pool.ntp.org"	
ntpServer2	String	maxLength 64	"1.pool.ntp.org"	
syncManagedDeviceEnabled	Boolean	true or false	false	
usb, see USB on page 128.				set
enabled	Boolean	true or false	false	Enables/disables the front-panel USB
usb/dev, see USB Devices on page 129.	Array			
id	String	0 to String Max		read-only
vendorId	String	0 to String Max		read-only
productId	String	0 to String Max		read-only
bcdDevice	String	0 to String Max		read-only
manufacturer	String	0 to String Max		read-only
product	String	0 to String Max		read-only
serial	String	0 to String Max		read-only
usb/dev/conf, see USB Device Configuration on page 130.				
maxPower	String	0 to String Max		read-only
selfPowered	Boolean			read-only
remoteWakeup	Boolean			read-only
velocity/managedDevice, see Velocity Managed Device on page 132.				set

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
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 133.				set
enabled	Boolean	false		
ip Address	IP Address			
port	Integer	1 to 65535	47808	
networkNumber	Integer	1 to 65534	1000	
velocity/serial, see Velocity Managed Device Serial on page 135.				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 136.				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 138.				set, reset - generate self-signed certificate based upon configuration
organization	String	maxLength 32	""	
organizationUnit	String	maxLength 32	""	

Object	Data			Notes
Field	**Format**	**Range**	**Default**	**add, delete, set, sendTest on these objects require admin privilege**
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 140.	Array			add, set, delete
id	String	maxLength 32		read-only
address	IP Address			
prefix	Integer	0 to 128	0	
webserver-reset, see Web Server Self-Signed Certificate Generation on page 142.				reset - generate self-signed certificate

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:** 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 the next page.

⑧ BACnet MSTP on page 58.

▼ **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 below.

⑧ BACnet MSTP on page 58.

The BACnet server fields are settable.

Example using "deviceName" field:

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

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:** 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 the next page.

▼ **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 next page.

The BACnet server IP fields are settable.

Example using "fdrTtl" field:

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

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:** 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:** 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:** api/conf/bacnet/server/ip/trustedAccess: add (Admin)

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

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

① Command to add a trustedAccess entry.

NOTE: "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 Command to add a trustedAccess entry.

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

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

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

```
{
    "cmd": "delete"
} ①
```

① Command to delete 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 trustedAccess entry.

The BACnet IP trustedAccess fields are settable.

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

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:** 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:** api/conf/bacnet/server/mstp: set (Admin)

api/conf/bacnet/server/mstp: set

```
{
    "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:** 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:** api/conf/cli: set (Admin)

api/conf/cli:

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

① Command to set the sessionIdleTimeout in minutes.

▼ **CLI:** set conf cli sesssionIdleTimeout

admin> set conf cli sessionIdleTimeout = ARGS

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

① Command to set the sessionIdleTimeout in minutes.

2.4.3 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**: 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 64.

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

③ Proxy. See Proxy on page 65.

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 the next page.

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

③ Proxy. See Proxy on page 65.

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:** api/conf/cloud/remoteService : set (Admin)

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:** 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:** 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:** api/conf/cloud/remoteService/proxy : set (Admin)

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.

2.4.4 Email

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

▼ **API:** api/conf/email

api/conf/email: get

```
{
    "passwordSet": false, ①
    "port":  25,          ②
    "sender": "",         ③
    "server": "",         ④
    "password": "",       ⑤
    "subjectPrefix": "",  ⑥
    "username": ""        ⑦
    "target":  []         ⑧
}
```

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.

⑥ The Email subject prefix prepended to all email messages.

⑦ The Email username.

⑧ Email Target, see Email Target on page 69.

▼ **CLI:** get conf email

user> get conf email

```
passwordSet: false ①
port: 25           ②
sender: ""         ③
server: ""         ④
password: ""       ⑤
subjectPrefix: ""  ⑥
username: ""       ⑦
target:            ⑧
    ...
```

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.

⑥ The Email subject prefix prepended to all email messages.

⑦ The Email username.

⑧ Email Target, see Email Target on the facing page.

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

Example using "server" field:

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

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:** 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:** api/conf/email/target: add (Admin)

api/conf/email/target:

```
{
    "cmd": "add",
    "data": {
        "id": "test",
        "name": "receive@vertiv.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": "receive@vertiv.com"} ①
```

① Required fields: id, name (email).

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

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:** api/conf/email/target/<id>: set (Admin)

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

```
{
    "cmd": "set",
    "data": {
        "name": "receive@vertiv.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:** api/conf/email/target/<id>: sendTest (Admin)

api/conf/email/target:

```
{
    "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".

2.4.5 LLDP

The Link Layer Discovery Protocol (LLDP) configuration.

▼ **API:** 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:** _api/conf/lldp : set (Admin)

api/conf/lldp:

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

① Command to set enable/disable for the LLDP protocol.

▼ **CLI:** set conf lldp enabled

admin> set conf lldp enabled = ARGS

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

① Command to set enable/disable for the LLDP protocol.

2.4.6 Locale

The locale configuration for the card. Currently language support is English only. Measurements can be US or Metric.

▼ **API:** api/conf/locale

api/conf/locale: get

```
{
    "defaultUserLanguage": "en",         ①
    "measurementSystem": "us",           ②
    "systemNotificationsLanguage": "en" ③
}
```

① The default user language is English.

② The measurement system is US or Metric.

③ The system notifications language is English.

▼ **CLI:** get conf locale

user> get conf locale

```
defaultUserLanguage: en          ①
```

```
measurementSystem: us          ②
systemNotificationsLanguage: en ③
```

① The default user language is English.

② The measurement system is US or Metric.

③ The system notifications language is English.

The get and set commands can be executed on the "measurementSystem" field.

▼ **API:** 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:** api/conf/locale: set (Admin)

api/conf/locale:

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

① Command to set measurement system in US or Metric.

▼ **CLI:** set conf locale measurementSystem

admin> set conf locale measurementSystem = ARGS

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

① Command to set measurement system in US or Metric.

2.4.7 Modbus Server

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

▼ **API:** 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 the next page.

④ Modbus TCP. See Modbus TCP on page 77.

▼ **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 next page.

④ Modbus TCP. See Modbus TCP on page 77.

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

Example using the "access" field:

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

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:** 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:** api/conf/modbus/server/rtu/ : set (Admin)

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:** 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 the next page.

▼ **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 below.

The Modbus Server TCP fields are settable.

Example using the "maxClients" field:

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

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:** 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:** api/conf/modbus/server/tcp/trustedAccess: add (Admin)

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:** api/conf/modbus/server/tcp/trustedAccess/<id>: delete (Admin)

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:** api/conf/modbus/server/tcp/trustedAccess/<id>: set (Admin)

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.

2.4.8 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:** 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 83 for possible states.

⑪ Specifies how the DNS addresses for IPv6 are obtained. See Network/DNS on page 83 for possible states.

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

⑬ Network/DNS. SeeNetwork/DNS on page 83.

⑭ Network/Interface/LAN. See Network/Interface/LAN on page 87.

⑮ Network/Route. See Network/Route on page 85.

▼ **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 the facing page for possible states.

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

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

⑬ Network/DNS. SeeNetwork/DNS on the facing page.

⑭ Network/Interface/LAN. See Network/Interface/LAN on page 87.

⑮ Network/Route. See Network/Route on page 85.

The network fields are settable except "ipv4DnsSource" and "ipv6DnsSource" which are read-only.

Example using "hostname" field:

▼ **API:** api/conf/network : set (Admin)

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 up to four DNS 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. A reset to defaults will delete all IDs. 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: 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: api/conf/network/dns : add (Admin)

api/conf/network/dns:

```
{
  "cmd": "add",
  "data": {
      "address": "8.8.4.4"
   }
} ①
```

① Command to add network dns.

▼ **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.

▼ **API:** api/conf/network/dns/<id>: delete (Admin)

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:** 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:** api/conf/network/route: add (Admin)

api/conf/network/route:

```
{
    "cmd": "add",
    "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:** api/conf/network/route/<id> : delete (Admin)

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:** 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
    "dhcpEnabled": true,            ⑥
    "address": [],                  ⑦
    "link": {},                     ⑧
    "dot1x": {}                     ⑨
}
```

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 Indicates if DHCP is enabled for this interface.

⑦ Network/Interface/LAN/Address. See Network/Interface/LAN/Address on page 89.

⑧ Network/Interface/LAN/Link. See Network/Interface/LAN/Link on page 91.

⑨ Network/Interface/LAN/Dot1x . See Network/Interface/LAN/Dot1x on page 93.

▼ **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
dhcpEnabled: true            ⑥
address:                     ⑦
    ...
link:                        ⑧
    ...
dot1x:                       ⑨
    ...
```

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 Indicates if DHCP is enabled for this interface.

⑦ Network/Interface/LAN/Address. See Network/Interface/LAN/Address on the facing page.

⑧ Network/Interface/LAN/Link. See Network/Interface/LAN/Link on page 91.

⑨ Network/Interface/LAN/Dot1x . See Network/Interface/LAN/Dot1x on page 93.

The interface lan fields are settable except "removable", "name", and "macAddr" which are read-only.

Example using "label" field:

▼ **API:** api/conf/network/interface/lan : set (Admin)

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.

▼ **API:** 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:** api/conf/network/interface/lan/address : add (Admin)

api/conf/network/interface/lan/address:

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

① Required fields: prefix, address.

▼ **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.

▼ **API:** api/conf/network/interface/lan/address/<id> : delete (Admin)

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:** 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:** api/conf/network/interface/lan/link : set (Admin)

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:** 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:** api/conf/network/interface/lan/dot1x: set (Admin)

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.

2.4.9 Remote Authentication

The Remote Authentication configuration settings allow the user to configure RADIUS, TACACS+, or LDAP remote authentication.

▼ **API:** 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 96.

③ Radius on page 98.

④ TACACS on page 100.

▼ **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 page 96.

③ Radius on page 98.

④ TACACS on page 100.

The mode field is settable.

▼ **API:** api/conf/remoteAuth : set (Admin)

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:** 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:** api/conf/remoteAuth/ldap: set (Admin)

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:** 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:** api/conf/remoteAuth/radius: set (Admin)

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:** 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:** api/conf/remoteAuth/tacacs: set (Admin)

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.

2.4.10 Remote Syslog

The Remote Syslog configuration settings allow the user to enable and configure remote syslog client.

▼ **API:** api/conf/remoteSyslog

api/conf/remoteSyslog: get

```
{
    "enabled": false, ①
    "port": 514,      ②
    "target": ""      ③
}
```

① Syslog enable control.

② Syslog port number.

③ Syslog target server address.

▼ **CLI:** get conf remoteSyslog

user> get conf remoteSyslog

```
enabled: false ①
port: 514      ②
target: ""     ③
```

① Syslog enable control.

② Syslog port number.

③ Syslog target server address.

The remoteSyslog fields are settable.

Example using the "target" field:

▼ **API:** api/conf/remoteSyslog: set (Admin)

api/conf/remoteSyslog:

```
{
    "cmd": "set",
    "data": {
        "target": "0.0.0.0"
    }
} ①
```

① Command to set syslog target server address.

▼ **CLI:** set conf remoteSyslog target

admin> set conf remoteSyslog target = ARGS

```
set conf remoteSyslog target = "0.0.0.0" ①
```

① Command to set syslog target server address.

2.4.11 Serial

The Serial configuration settings for the console port.

▼ **API:** 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:** api/conf/serial: set (Admin)

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.

2.4.12 SNMP

The SNMP configuration settings for SNMP v1, v2c, and v3.

▼ **API:** 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 116.

② snmp/v1v2c. See SNMP v1/v2c on page 107.

③ snmp/mibSupport. See SNMP MIB Support on page 106.

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 116.

② snmp/v1v2c. See SNMP v1/v2c on page 107.

③ snmp/mibSupport. See SNMP MIB Support on the next 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:** api/conf/snmp/port: set (Admin)

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:** 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:** api/conf/snmp/mibSupport: set (Admin)

api/conf/snmp/mibSupport: set

```
{
    "cmd": "set",
    "data": {
        "lgpMIBTrapEnabled": true
    }
} 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:** 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 113.

② snmp/v1v2c/access. SeeSNMP v1/v2c Access on the next 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 113.

② snmp/v1v2c/access. See SNMP v1/v2c Access below.

③ The enable control for SNMP v1/v2c.

④ The SNMP v1/v2c test traps.

The enabled field is settable.

▼ **API:** api/conf/snmp/v1v2c : set (Admin)

api/conf/snmp/v1v2c: set

```
{
  "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:** 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 111.

② The access rights for this access entry.

③ The v1/v2c community string.

④ The name of the access 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 111.

② The access rights for this access entry.

③ The v1/v2c community string.

④ The name of the access ID entry.

▼ **API:** api/conf/snmp/v1v2c/access : add (Admin)

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:** api/conf/snmp/v1v2c/access/<accessID> : delete (Admin)

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:** api/conf/snmp/v1v2c/access/<accessID> : set (Admin)

api/conf/snmp/v1v2c/access/<accessID>: set

```
{
    "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:** api/conf/snmp/v1v2c/access/<accessID>/trustedAccess

api/conf/snmp/v1v2c/access/<accessID>/trustedAccess: get

```
[
  {
    "prefix": 24,             ①
    "address": "126.4.12.33", 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: 126.4.12.33 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:** api/conf/snmp/v1v2c/access/<accessID>/trustedAccess : add (Admin)

api/conf/snmp/v1v2c/access/<accessID>/trustedAccess:

```
{
  "cmd": "add",
  "data": {
     "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.

▼ **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:** api/conf/snmp/v1v2c/access/<accessID>/trustedAccess/<trustedID> : delete (Admin)

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:** api/conf/snmp/v1v2c/access/<accessID>/trustedAccess/<trustedID> : set (Admin)

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.

▼ **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:** 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:** api/conf/snmp/v1v2c/trap: add (Admin)

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:** api/conf/snmp/v1v2c/trap/<id>: delete (Admin)

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:** api/conf/snmp/v1v2c/trap/<id>: set (Admin)

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:** 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 the facing page.

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 the facing page.

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:** 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:** 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. SeeSNMP v3 User Trap Target on page 121.

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. SeeSNMP v3 User Trap Target on page 121.

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:** api/conf/snmp/v3/user/ : add (Admin)

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:** api/conf/snmp/v3/user/<id> : delete (Admin)

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:** api/conf/snmp/v3/user/<id> : set (Admin)

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:** 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: api/conf/snmp/v3/user/<userID>/target : add (Admin)

api/conf/snmp/v3/user/0/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:** api/conf/snmp/v3/user/<userID>/target/<id> : delete (Admin)

api/conf/snmp/v3/user/<userID>/target/<id>:

```
{
"cmd": "delete"
} ①
```

① Command to delete trap 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 entry.

The SNMP v3 trap target fields are settable except "id".

Example using the "target" field:

▼ **API:** api/conf/snmp/v3/user/<userID>/target/<id> : set (Admin)

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.

2.4.13 SSH

The SSH Configuration.

▼ **API:** 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:** api/conf/ssh/enabled: set (Admin)

api/conf/ssh/enabled:

```
{
    "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 turn enable/disable ssh.

The reset command regenerates SSH keys on a server — creating new cryptographic key pairs used for secure authentication.

▼ **API:** api/conf/ssh: reset (Admin)

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.

2.4.14 System

The system configuration.

▼ **API:** 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:** api/conf/system: set (Admin)

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.

2.4.15 Time

The time configuration. For a list of valid time zone entries, see Time Zone on page 161.

▼ **API:** 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:** api/conf/time: set (Admin)

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 system location = ARGS

```
set conf time syncManagedDeviceEnabled = true ①
```

① Command to set the control that allows automatic time writes to the managed device.

2.4.16 USB

The USB configuration and list of connected devices.

▼ **API:** 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 the facing page.

▼ **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 the facing page.

▼ **API:** api/conf/usb: set (Admin)

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:** 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 next 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 below.

USB Device Configuration

The configuration settings for the USB device. All are read only.

▼ **API:** 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.

2.4.17 Velocity

The velocity protocol configuration.

▼ **API:** api/conf/velocity

api/conf/velocity: get

```
{
    "managedDevice": {}, ①
    "ethernet": {},      ②
    "serial": {}         ③
}
```

① Velocity Managed Device. See Velocity Managed Device below.

② Velocity Managed Device Ethernet. See Velocity Managed Device Ethernet on the facing page.

③ Velocity Managed Device Serial. See Velocity Managed Device Serial on page 135.

▼ **CLI:** get conf velocity

user> get conf velocity

```
managedDevice: ①
    ...
ethernet:      ②
    ...
serial:        ③
    ...
```

① Velocity Managed Device. See Velocity Managed Device below.

② Velocity Managed Device Ethernet. See Velocity Managed Device Ethernet on the facing page.

③ Velocity Managed Device Serial. See Velocity Managed Device Serial on page 135.

Velocity Managed Device

The velocity managed device configuration.

▼ **API:** 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:** api/conf/velocity/managedDevice : set (Admin)

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:** 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:** api/conf/velocity/ethernet : set (Admin)

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:** api/conf/velocity/serial

api/conf/velocity/serial: get

```
{
    "baudRate": "38400",                ①
    "maxAddress": 127,                  ②
    "networkNumber": 1001,              ③
    "nodeId": 127,                      ④
    "nodeAssignmentMethod": "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
nodeAssignmentMethod: 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 slotid.

The velocity serial fields are settable.

Example using the "baudRate" field:

▼ **API:** api/conf/velocity/serial: set (Admin)

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 <1> Command to set serial baud rate.

NOTE: Allowable values are 38400, 19200 and 9600.

2.4.18 Web Server

The Web Server configuration.

▼ **API:** 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 the next page.

⑧ Trusted Access, See Trusted Access on page 140.

▼ **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 140.

The httpEnabled, httpsEnabled, httpPort, httpsPort, redirectEnabled and sessionIdleTimeout fields are settable.

Example using the "httpEnabled" field:

▼ **API:** api/conf/webserver: set (Admin)

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:** 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. Web Server Self-Signed Certificate Generation on page 142.

▼ **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. Web Server Self-Signed Certificate Generation on page 142.

The genCertificate fields are settable.

Example using the "organization" field:

▼ **API:** api/conf/webserver/genCertificate: set (Admin)

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:** api/conf/webserver/trustedAccess

api/conf/webserver/trustedAccess: get

```
[
  {
    "id": "test",          ①
    "address": "0.0.0.0", 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: 0.0.0.0 ②
  prefix: 24       ③
```

① The trusted access entry id.

② The trusted access host/network address.

③ The network prefix.

▼ **API:** api/conf/webserver/trustedAccess: add (Admin)

api/conf/webserver/trustedAccess:

```
{
    "cmd": "add",
    "data": {
        "prefix" : 24,
        "address" : "0.0.0.0",
        "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": "0.0.0.0", "id":
"test"} ①
```

① Command to add a trustedAccess entry.

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

▼ **API:** api/conf/webserver/trustedAccess/<id>: delete (Admin)

api/conf/webserver/trustedAccess/<id>: delete

```
{
    "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:** api/conf/webserver/trustedAccess/<id>: set (Admin)

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 138.

IMPORTANT! Reset command currently is not working when ssMode is set to "Use Configured Settings".

▼ **API:** api/conf/webserver: reset (Admin)

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.

2.5 Configuration: /api/sys

The Vertiv™ Liebert® IntelliSlot™ RDU120 System API exists to provide status level system information for the Vertiv™ Liebert® IntelliSlot™ RDU120.

Object	Data			Notes
Field	Format	Range	Default	
adminExists	Boolean			
alarmCount	Integer			
alternateFirwmwareDate	String			
alternateFirmwareLabel	String			
alternateFirmwareVersion	String			
apiVersion	String			
build	String			
component	String			
currentFirmwareDate	String			
currentFirmwareLabel	String			
currentFirmwareVersion	String			
dirty	Integer			
guestEnabled	Boolean			
label	String			
localTime	String			
model	String			
modelNumber	String			
name	String			
oem	String			
partNumber	String			
picVersion	String			
platform	String			
secureKeyLocked	Boolean			
secureKeyProgrammed	Boolean			
serialNumber	String			
systemTime	Integer			
ubootVersion	String			
uptime	Integer			
version	String			
warnCount	Integer			

▼ **API:** api/sys

api/sys: get

```
{
    "adminExists": true,                                                          ①
    "alarmCount": 0,                                                              ②
    "alternateFirmwareDate": "09/12/2024 14:41:34",                               ③
    "alternateFirmwareLabel": "mric-is-1.1.1-02006-c2b5547a8",                    ④
    "alternateFirmwareVersion": "1.1.1",                                          ⑤
    "apiVersion": "0.0.1",                                                        ⑥
    "build": "02007-c2b5547a8",                                                   ⑦
    "component": "ok",                                                            ⑧
    "currentFirmwareDate": "09/12/2024 15:17:11",                                 ⑨
    "currentFirmwareLabel": "mric-is-1.1.1-02007-c2b5547a8",                      ⑩
    "currentFirmwareVersion": "1.1.1",                                            ⑪
    "dirty": 0,                                                                   ⑫
    "guestEnabled": true,                                                         ⑬
    "label": "",                                                                  ⑭
    "localTime": "2024-09-12 20:17:04",                                           ⑮
    "model": "I-15",                                                              ⑯
    "modelNumber": "unset",                                                       ⑰
    "name": "mricis-evk",                                                         ⑱
    "oem": "VER",                                                                 ⑲
    "partNumber": "unset",                                                        ⑳
    "picVersion": "1.7",                                                          ①
    "platform": "mric-is",                                                        ②
    "secureKeyLocked": false,                                                     ③
    "secureKeyProgrammed": true,                                                  ④
    "serialNumber": "unset",                                                      ⑤
    "systemTime": 1726172224,                                                     ⑥
    "ubootVersion": "2021.04-lf_v2021.04+gf13185d46c(Sep 10 2024-15:02:49)",      ⑦
    "uptime": 1879,                                                               ⑧
    "version": "1.1.1",                                                           ⑨
    "warnCount": 0                                                                ⑩
}
```

① The state of an administrator (true if configured, else false)

② The number of active alarms.

③ The alternate firmware build date.

④ The alternate firmware label.

⑤ The alternate firmware version.

⑥ The API version.

⑦ The build reference.

⑧ The state of the component.

⑨ The current firmware build date.

⑩ The current firmware label.

⑪ The current firmware version.

⑫ The dirty flag.

⑬ The guest enable flag.

⑭ The configured system label.

⑮ The local time.

⑯ The model.

⑰ The model number.

⑱ The configured system name.

⑲ The OEM label.

⑳ The part number.

1. The PIC version.
2. The platform.
3. The secure key lock flag.
4. The secure key programmed flag.
5. The serial number.
6. The system time.
7. The U-Boot version.
8. The system uptime.
9. The system version.
10. The number of active warnings.

▼ **CLI:** get sys

user> get sys

```
alternateFirmwareVersion: 1.1.1                                    ①
currentFirmwareVersion: 1.1.1                                      ②
currentFirmwareDate: 09/13/2024 13:45:02                           ③
currentFirmwareLabel: mric-is-1.1.1-02012-c2b5547a8                ④
secureKeyProgrammed: true                                          ⑤
apiVersion: 0.0.1                                                  ⑥
picVersion: 1.7                                                    ⑦
label: ""                                                          ⑧
version: 1.1.1                                                     ⑨
partNumber: unset                                                  ⑩
name: mricis-evk                                                   ⑪
build: 02012-c2b5547a8                                             ⑫
oem: VER                                                           ⑬
platform: mric-is                                                  ⑭
secureKeyLocked: false                                             ⑮
modelNumber: unset                                                 ⑯
serialNumber: unset                                                ⑰
alternateFirmwareDate: 09/13/2024 12:52:04                         ⑱
component: ok                                                      ⑲
alternateFirmwareLabel: mric-is-1.1.1-02011-c2b5547a8              ⑳
model: I-15                                                        ①
warnCount: 0                                                       ②
uptime: 243612                                                     ③
```

```
adminExists: true                                                    ④
guestEnabled: true                                                   ⑤
alarmCount: 0                                                        ⑥
localTime: 2024-09-16 13:53:22                                       ⑦
dirty: 0                                                             ⑧
systemTime: 1726494802                                               ⑨
ubootVersion: 2021.04-lf_v2021.04+gf13185d46c(Sep 10 2024-15:02:49) ⑩
```

① The alternate firmware version.

② The current firmware version.

③ The current firmware build date.

④ The current firmware label.

⑤ The secure key programmed flag.

⑥ The API version.

⑦ The PIC version.

⑧ The configured system label.

⑨ The system version.

⑩ The part number.

⑪ The configured system name.

⑫ The build reference.

⑬ The OEM label.

⑭ The platform.

⑮ The secure key lock flag.

⑯ The model number.

⑰ The serial number.

⑱ The alternate firmware build date.

⑲ The state of the component.

⑳ The alternate firmware label.

1. The model.
2. The number of active warnings.
3. The system uptime.
4. The state of an administrator (true if configured, else false).
5. The guest enable flag.
6. The number of active alarms.
7. The local time.
8. The dirty flag.

9. The system time.
10. The U-Boot version.

Commands

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

▼ **API:** api/sys: reset (Admin)

api/sys:

```
{
    "cmd": "reset",
    "data": {
        "target": "fullDefaults"
    }
} ①
```

① Command to reset card to one of the five target values (fullDefaults, partialDefaults, hostAccessDefaults, eventLog, decom).

▼ **CLI:** reset sys

admin> reset sys = ARGS

```
reset sys = { target: fullDefaults } ①
```

① Command to reset card to one of the five target values (fullDefaults, partialDefaults, hostAccessDefaults, eventLog, decom).

The reboot command is used to reboot the card.

▼ **API:** api/sys: reboot (Admin)

api/sys:

```
{
    "cmd": "reboot"
} ①
```

① Command to reboot the card.

▼ **CLI:** reboot sys

admin> reboot sys

```
reboot sys ①
```

① Command to reboot the card.

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 card.

▼ **CLI:** switchFirmware sys

admin> switchFirmware sys

```
switchFirmware sys ①
```

① Command to switch to the alternate firmware version in the card.

2.6 Special File Transfers: /Transfer

2.6.1 Bootlog Download: /transfer/bootlog

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 authentication to be sent in through the query string ("username"/"password").

2.6.2 Factory Support Package: /transfer/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 authentication to be sent in through the query string ("username"/"password").

2.6.3 Factory Support Package: /transfer/ethpcap

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 authentication to be sent in through the query string ("username"/"password").

2.6.4 Firmware Update: /transfer/firmware

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 a progress, it cannot be interrupted. Any subsequent upload attempts will be blocked.

▼ **API:** transfer/firmware

transfer/firmware: post

```
{
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.

2.6.5 SSL Certificate Upload: /transfer/sslcert

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. Possible failure conditions include malformed file or insufficient security on the uploaded file or firmware update already in progress.

▼ **API:** transfer/sslcert

transfer/sslcert: post

```
{
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.

2.6.6 802.1X Upload: /transfer/dot1x

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 15.

▼ **API:** transfer/dot1x

transfer/dot1x: post

```
{
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/combined-clientCert.pem" \
-F "privKeyPwd=<PrivKeyPassword>"
} ①
```

① Command to upload 802.1X files.

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. It is a protected API whereby the API user must be authenticated before being able to read 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**	
psi	String			
report, see Report on page 154.				
tag	String			
mmCnt	String			
report/ID/point, see Point on page 156				
id	String			
access	String			
tag	String			
desc	String			
type	String			
units	String			
report/ID/point/ID/value, see Value on page 158				
id	String			
value	String			
report/ID/event, see Event on page 157				
id	String			
tag	String			
desc	String			
report/ID/event/ID/value, see Event on page 157				
id	String			
value	String			
report/ID/children, see Children on page 155				

Object	Data			Notes
Field	**Format**	**Range**	**Default**	
id	String			
tag	String			
mmCnt	String			
activeEvent, see Active Event on page 158				
id	String			
tag	String			
desc	String			
activeEvent/ID/value, see Value on page 158				
id	String			
value	String			

3.2.1 Equipment

▼ **API:** equipment

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 on the next page

③ The active events of the equipment. See Active Event on page 158.

▼ **CLI:** get equipment

user> get 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 on the next page.

③ The active events of the equipment. See Active Event on page 158.

Report

▼ **API:** 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 156.

③ Report events. See Event on page 157.

④ The multi-module count, indicating how many values there should be for each data point. See Value on page 158.

⑤ The report tag.

⑥ The ID of the report.

▼ **CLI:** get equipment report

user> get 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 156.

③ Report events. See Event on page 157.

④ The multi-module count, indicating how many values there should be for each data point. See Value on page 158.

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.

▼ **API:** equipment/report/<id>/children

equipment/report/<id>/children: get

```
{
  "children": [],            ①
  "point": [],               ②
  "event": [],               ③
  "mmCnt": "1",              ④
  "tag": "26403"             ⑤
  "id": "Battery Statistics" ⑥
}
```

① Report children. See Children above.

② Report data points. See Point on the next page.

③ Report events. See Event on page 157.

④ The maximum multi-module count, indicating the maximum amount of values there can be for each data point. See Value on page 158.

⑤ The report tag.

⑥ The ID of the report.

▼ **CLI:** get equipment report <id> children

user> get equipment report <id> children

```
children:              ①
 ...
point:                 ②
 ...
event:                 ③
 ...
mmCnt: 1               ④
tag: 26403             ⑤
id: Battery Statistics ⑥
```

① Report children. See Children above.

② Report data points. See Point on the next page.

③ Report events. See Event on page 157.

④ The maximum multi-module count, indicating the maximum amount of values there can be for each data point. See Value on page 158.

⑤ The report tag.

⑥ The ID of the report.

Point

▼ **API:** 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": "UPS DC input voltage"
  ⑦
}
```

① Point values. Value on page 158.

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:** get equipment report <id> point

user> get 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 1.

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:** 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": "System Input Current Imbalance"                                          ④
}
```

① Event values. See Value on page 1.

② The description.

③ The tag.

④ The ID of the event object.

▼ **CLI:** get equipment report <id> event

user> get equipment report <id> event

```
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 Event values. See Value on page 1.

② The description.

③ The tag.

④ The ID of the event object.

Value

▼ **API:** equipment/report/<id>/point/<id>/value

equipment/report/<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:** get equipment report <id> point <id> value

user> get 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.

3.2.2 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 the previous pageand Value above for more information.

▼ **API:** 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": "System Input Current Imbalance"                                          ④
}
```

① Active event values. See Value above.

② The description.

③ The tag.

④ The ID of the event object.

▼ **CLI:** get equipment activeEvent

user> get 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 the previous page.

② The description.

③ The tag.

④ The ID of the event object.

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

Via Leonardo Da Vinci 8 Zona Industriale Tognana

35028 Piove Di Sacco (PD) Italy

Asia

7/F, Dah Sing Financial Centre

3108 Gloucester Road, Wanchai

Hong Kong

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

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_REVA_10-25