Liebert® IntelliSlot™ Card API

User Guide

Supporting RDU101 and IS-UNITY cards

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

This page intentionally left blank

1 Overview

This document provides the specification for the card API. The API was developed to meet customer requirements for accessing and updating a subset of the card configuration via HTTP using JSON request and response bodies. Per customer requirements, the API is modeled to the existing Vertiv™ Geist™ PDU API.

The API is now also supported on the Unity card, except for the managed device firmware upload, which is only supported on the RDU101.

1.1 API Version

Table 1.1 API Version

Version	Notes
1	Initial release

1.2 Glossary

Table 1.2 Glossary / Compatibility

Network Management Card	Initial Compatible Version
RDU101	RDU101_1.5.0.0_0000010
IS-UNITY-DP	IS-UNITY_8.2.0.0_00125
IS-UNITY-SNMP	IS-UNITY_8.2.0.0_00125

This page intentionally left blank

2 API

2.1 API Usage

The API is designed to provide developers and integrators with an easy-to-use method to communicate with the device. Device actions can be accessed through this API over HTTP or HTTPS using JSON data structures. Security is provided by HTTPS. **While the API is available over HTTP, this should only be used in development and not in production environments.** All requests will be authenticated.

2.1.1 Paths

All API client requests are specified using a path starting with '/api', followed by an optional version, followed by any remaining path to the area of interest.

If the API version of the path is not provided in the request, then the latest version of the API is used.

The following path indicates a command will access network configuration data against API version 1.0.

/api/1.0/conf/network

The following path would reference the same data against the most recent API version.

/api/conf/network

For this document, the version will be omitted.

Any request specifying an unsupported API version will fail with a response code indicating the API version is unsupported.

2.1.2 Requests

Within this document, example requests will have a background color as shown below. Client requests are generally in the form of HTTP POSTs to an API path, with a JSON object in the body, as follows (note that fields are sometimes optional):

```
{
   "token": "a4b3c2d1",
   "cmd": "get",
   "data": {}
}
```

The *token* is obtained via the *login* command. The *cmd* specifies the command to be executed (see API commands on the next page). The *data* provides the parameters for the command. Not all commands have parameters. Parameters will generally be a JSON object, but when setting on a leaf it will be any other datatype.

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

```
{
   “username”: “admin”,
   “password”: “abc123”,
   “cmd”: “get”,
   “data”: {}
}
```

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=abc123&cmd=get*, 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.

HTTP request headers

While technically not required, the client may specify the following HTTP headers:

Content-Type: application/json

Accept: application/json

If the client request has an HTTP body it will be interpreted as a JSON request body.

2.1.3 API commands

Commands are used to get, set, add, and delete data along with providing management functions like login, logout, and reboot. The command can be specified within an API request JSON object or by using URL attributes. If the command is not specified in either of those two methods, the command will be determined based on the HTTP Method.

API request command

The command can be provided within an API request message using the following API form:

/api/auth/<username>

```
{
   “token”: “a4b3c2d1”,
   “cmd”: “set”,
   “data”: {
      “password”: “some2Password$”
   }
}
```

Specifying the *cmd* label/value pair in the API request JSON body has the highest precedence, thus if provided, the value will determine the command.

URL command

The command and authentication fields can be provided as API URL attributes using the following API form:

/api/conf/network?cmd=get&username=admin&password=abc123

HTTP Method

Only if the command is not specified in the API Request or URL formats will the HTTP method then determine the command.

Table 2.1 Command Summary Table

Command	API Request	URL	HTTP Method
login	x	x	-
logout	x	x	-
reboot	x	x	-
get	x	x	GET[1]
set	x	x	POST[1]
add	x	x	PUT[1]
delete	x	x	DELETE[1]
upload	-	x	-
- not supported			
x = supported			

[1]The HTTP Method will determine the command only when the command was not specified via API Request or URL.

2.1.4 Responses

Within this document, example responses will have a background color as shown below. Responses to client requests come back with a JSON-format HTTP body as follows:

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {}
}
```

2.1.5 Gets and sets

Most operations will affect only the node at the path specified, but two operations – “get” and “set” – are also recursively applied to every child under the specified node. Thus, a get on */api/conf* will return a JSON object starting at the depth indicated by the path and traversing down to the leaf objects.

2.1.6 Card API data model

The following is the card API data model. For a complete description of all model elements, see the model element sections in the remainder of the document. The RDU101 card is used in the examples.

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”:
      “sys”: {
         “model”: “RDU1xx Platform”,
         “build”: “RDU101_1.5.0.0_0000006”,
         “version”: “1.5.0.0”,
         “serialNumber”: “02DA”,
         “apiVersion”: “1.0”,
         “label”: “RDU101-Lab3”,
         “name”: “Vertiv RDU101”,
         “contact”: {
            “location”: “Lab 3”,
            “description”: “Lab 3 UPS”
         }
      },
      “conf”: {
         “contact”: {
            “location”: “Lab 3”,
            “description”: “Lab 3 UPS”
         },
         “system”: {
            "displayTemperatureUnits" : "celsius",
            “hostname”: “lab3-rdu101”,
            “label”: “RDU101-Lab3”,
         },
         “time”: {
            “mode": “ntp”,
            “datetime”: “2020-10-15 14:40:40”,
            “zone": “(UTC+00:00) Coordinated Universal Time”,
            “ntpServer1": “192.168.1.10”,
            “ntpServer2": “192.168.1.11”
            “ntpSyncRate": 24,
            “enableSyncToManagedDevice": false,
            “managedDeviceSyncRate": 1
         },
         “network”: {
            “ethernet”: {
               “dhcpOn”: true,
               “dhcpOnIPv6”: true,
               “macAddr”: “98:90:96:d2:14:9e”,
               “order”: 0,
               “label”: “ethernet”,
               “type”: “lan”,
               “enabled”: true,
               “removable”: false,
               “address”: {
                  “0”: {
                     “mutable”: false,
                     “prefix”: 24,
                     “address”: “192.168.4.23”
                  },
                  “1”: {
                     “mutable”: false,
                     “prefix”: 64,
                     “address”: “fe80::53ce:3fd5:5eb2:e9c6”
                     “slaac”: “”,
```

```
                "linklocal": "fe80::209:f5ff:fe20:f65e"
            }
        },
        "route": {
            "0": {
                "mutable": false,
                "interface": "all",
                "gateway": "192.168.4.1",
                "prefix": 0,
                "destination": "0.0.0.0"
            },
            "1": {
                "mutable": false,
                "interface": "all",
                "gateway": "",
                "prefix": 0,
                "destination": "::"
            }
        },
        "dns": {
            "0": {
                "mutable": false,
                "address": "64.233.222.2"
            },
            "1": {
                "mutable": false,
                "address": "64.233.222.7"
            }
        }
    }
},
"snmp": {
    "v1v2cEnabled": true,
    "v3Enabled": true,
    "port": 161,
    "authTrapsEnabled": false,
    "heartbeatTrapInterval": 1440,
    "rfc1628MIBEnabled": true,
    "rfc1628MIBTrapsEnabled": true,
    "lgpMIBEnabled": true,
    "lgpMIBTrapsEnabled": true,
    "lgpMIBSysNotifyTrap": true,
    "incSysNameLocInTraps": false,
    "engineId": "80001F88030009F520F65E",
    "access": {
        "0": {
            "name": "192.168.10.10",
            "type": "read",
            "community": "public"
        },
        "1": {
            "name": "192.168.10.20",
            "type": "write",
            "community": "private"
        }
    },
    "target": {
        "0": {
```

```
                “name”: “192.168.123.1”,
                “port”: 162,
                “trapVersion”: “1”,
                “community”: “trap”
            },
            “1”: {
                “name”: “192.168.123.1”,
                “port”: 162,
                “trapVersion”: “3”
            }
        },
        “user”: {
            “0”: {
                “username”: “initial”,
                “type”: “read”,
                “authType”: “none”,
                “authPassswordSet”: false,
                “authPassword”: null,
                “privType”: “none”,
                “privPasswordSet”: false,
                “privPassword”: null
            },
            “1”: {
                “username”: “manager”,
                “type”: “write”,
                “authType”: “md5”,
                “authPassswordSet”: true,
                “authPassword”: null,
                “privType”: “des”,
                “privPasswordSet”: true,
                “privPassword”: null
            },
            “2”: {
                “username”: “trap”,
                “type”: “trap”,
                “authType”: “md5”,
                “authPassswordSet”: true,
                “authPassword”: null,
                “privType”: “des”,
                “privPasswordSet”: true,
                “privPassword”: null
            }
        }
    }
},
“auth”: {
    “admin”: {
        “enabled”: true,
        “control”: true,
        “admin”: true,
        “language”: “en”
        “password”: null,
        “passwordSet”: true,
        “source”: “local”
    },
    “user1”: {
        “enabled”: true,
        “control”: false,
```

```
            “admin”: false,
            “language”: “en”
            “password”: null,
            “passwordSet”: true,
            “source”: “local”
         },
      }
   }
}
```

2.1.7 Error codes

Table 2.2 Success

Code	Name	Notes
0	Success	Operation has succeeded.
1	Success, reboot required	Operation has succeeded, reboot required for changes to take effect.
2	Success, set datetime scheduled	When a request to set the date/time results in a negative time shift of nine or more seconds this response code will be returned to indicate the setting of the date/time will be scheduled (one second in the future). This is done because a negative time shift of ten or more seconds results in the webserver restarting its worker processes, which would cause the initial API request to not be able to send a response.

Table 2.3 Authentication Errors (1000-1999)

Code	Name	Notes
1000	No Admin user configured	At least one Admin user must be configured on the system.
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. Invalid username/password.
1010	Too many sessions	A login request is rejected because the user has reached the maximum number of concurrent sessions.

Table 2.4 JSON Format Errors (2000-2999)

Code	Name	Notes
2000	Malformed JSON	Received JSON is invalid or corrupt. Note this error code may be returned on any API request if the JSON is corrupt (does not parse).
2001	Missing field	An expected field was not found in the JSON structure.

Table 2.5 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.
3004	Unsupported API version	The path indicates an unsupported API version.

Table 2.6 Data Validation Errors (4000-4999)

Code	Name	Notes
4001	Input too long	An input field exceeds the maximum allowed length.
4002	Invalid characters	An input field contains invalid characters.
4004	Invalid Boolean	An input field is not a Boolean when one is expected.
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.
4009	Invalid IP	An input field is not a valid IP address when one is expected.
4011	Invalid username	An input field is an unsupported user name.
4013	Invalid option	An input field contains an invalid option selection.
4014	Invalid date/time	An input field is not a valid date/time when one is expected.
4015	Out of bounds	An input field is out of the allowed bounds for the field or the maximum number of items has been exceeded, e.g. number of users.
4017	Duplicate entry	An input field would create a duplicate when one is not allowed.
4019	Invalid password	An input field is not a valid password.
4020	Read-only	A read-only field exists in a set request.
4021	Input too short	An input field does not meet the minimum allowed length.
4022	Invalid string	An input field is not a string when one is expected.

Table 2.7 Other Errors (5000-5999)

Code	Name	Notes
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.
5003	Not logged in	Logout attempted for user not currently logged in.
5004	Config library error	Unexpected error from configuration library.
5005	API library error	Unexpected error from API library.

Table 2.7 Other Errors (5000-5999) (continued)

Code	Name	Notes
5006	System error	Unexpected system error.
5007	File transfer service error	Unexpected error for the file transfer service during managed device firmware upload.
5008	API request error	Unexpected error during parsing and processing of the API HTTP request.

Table 2.8 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.
6002	NTP mode requires servers	Enabling NTP mode requires at least one NTP server.
6007	Time not settable	Setting datetime requires manual time mode.

2.1.8 Data formats

Table 2.9 Data Formats

Data Format	Details
Boolean	Binary condition. Can only be true or false.
Integer	Number without fractional components.
Float	Number with factional components.
String	Sequence of characters.
Username	A special type of String. Must be 2-30 characters in length, printable ASCII characters (excluding: \:'<>~?#, double quote, and space).
Password	A special type of String. Must be 8-30 characters in length, printable ASCII characters (excluding: \:'<>~?#, double quote, and space). Must contain a combination of upper and lower case, digit and special characters, but not Username.
Array	A sequence of objects, all of the same type.
ID	An identifier for an object of a particular type. When setting, must be a valid reference to an existing object.
Language Code	Two letter language code. Available codes are: “en”
IPv4 Address	Dotted decimal notation of an IPv4 address. Represented as 4 decimal numbers separated by periods.
IPv6 Address	RFC 5952 recommended IPv6 address text representation.
IP Address	An IP address fitting either the IPv4 Address or IPv6 Address formats.
Hostname	Hostname only, not the fully qualified domain name. Max length is 64.
Date/Time	"YYYY-MM-DD HH:MM:SS”

2.2 Authentication

Table 2.10 Authentication: /api/auth

Object		Data			Notes
	Field	Format	Range	Default	*set, add, delete* on these objects require *admin* privilege.
ID			1 to 10 objects ID: 2 to 30 characters		Commands: *get, set, add, delete, login, logout.*
	enabled	Boolean	true, false		
	control	Boolean	true, false		
	admin	Boolean	true, false		
	language	Language Code	"en"	"en"	Read-only as "en" is the only supported language code.
	username		2 to 30 characters		
	password	Password	8 to 30 characters		Returned as null.
	passwordSet	Boolean	TRUE	TRUE	Read-only as all users are required to have a password.
	source	String	"local" or "ldap"		Read-only

The */api/auth* object lists all local system users, remote users with active sessions, and provides a means to authenticate local or remote users.

2.2.1 Privileges

All API requests to the card will be authenticated. The card has three authentication levels: *No Access, General User, and Administrator*. *General User* accounts have read-only access while *Administrator* accounts have read-write access. *No Access* users have no access. Assigning a user to *No Access* effectively suspends that user until their access level is updated.

2.2.2 Security and special rules

The following table maps the three valid combinations of the API *enabled, control,* and *admin* attributes, each of which maps to the card access level. When adding user accounts (*add* command) all three attributes must be supplied and match one of the combinations shown in the table. When updating an existing user account's access level (*set* command), only the attributes to be changed are required in the request (all may be specified), but the resulting combination must be one of the three shown in the table.

Table 2.11 Card and API Access Levels

Card Access Levels	API Access Levels		
	enabled	control	admin
No Access	FALSE	FALSE	FALSE
General User	TRUE	FALSE	FALSE
Administrator	TRUE	TRUE	TRUE

The *language* attribute will always be "en", thus is read-only, as the card only supports English.

An authenticated user can set their own *password*, but *enabled*, *control*, and *admin* can only be set by *admin* privileged users.

The *passwordSet* attribute will always be true, thus is read-only, as all the card users must have a valid password.

Whereas most parts of the system allow unrestricted *get* access to all users, only *admin* privileged users can see */api/auth* in its entirety. Non *admin* users making a *get* request will only see their own configuration. Unlike a Vertiv™ Geist™ PDU, the card does not have a *guest* user.

The Geist™ PDU allows a one-time non-authenticated *add* of an *admin* account if no *admin* accounts exist. The card enforces authentication on all requests, thus an *admin* account must already exist in order to *add* or update users.

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 10 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 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.

The following login example is for local username (ID) "*admin*".

/api/auth/admin

```
{
   “cmd”: “login”,
   “data”: {
      “password”: “abc123DEF!”
   }
}
```

A successful response is shown below.

```
{
   “data”: {
      “token”: “f3e2d1c0”,
      “language”: “en”,
      “admin”: true,
      “control”: true,
      “source”: “local”
   },
   “retCode”: 0,
   “retMsg”: “Success”
}
```

NOTE: The *token* will be 32 characters in length. The *token* values are abbreviated in this document.

Table 2.12 *login* Return Codes

Code	Comments
0	Success. Authentication token returned.
1006	Invalid credential combination. A valid ID was specified in the path, but the password is incorrect, or a non-existent ID was specified in the path.
1010	Too many sessions.
2001	Missing field. The “password” field is missing from the “data” object.
3000	Invalid path.

Command: *logout*

Log out username (ID) specified in the path. Logging out will invalidate all tokens for that user. Non *admin* users may only logout of their own account. *Admin* users may *logout* any account.

The following example logs out user “joe” as the “*admin*” user. The provided token is from the “*admin*” user *login* above.

/api/auth/joe

```
{
   “token”: ”f3e2d1c0”,
   “cmd”: “logout”,
   “data”: {}
}
```

A successful response is shown below.

```
{
   “data”: {},
   “retCode”: 0,
   “retMsg”: “Success”
}
```

Table 2.13 *logout* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1003	Not authorized. Not enough permissions. Returned if a non *admin* user attempts to logout a non-existent */api/auth/ID* path or attempts to logout an existing */api/auth/ID* path that does not match the requesting non admin user.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
5003	Not logged in.

Command: *get*

Only the requesting user is returned for non admin accounts, while all users are returned for *admin* users. The following example shows an admin user get request using an authentication token.

/api/auth

```
{
   “token”: ”f3e2d1c0”,
   “cmd”: “get”,
   “data”: {}
}
```

```
{
   “data”: {
      “admin”: {
         “enabled”: true,
         “control”: true,
         “admin”: true,
         “language”: “en”
         “password”: null,
         “passwordSet”: true,
         “source”: “local”
      },
      “user1”: {
         “enabled”: true,
         “control”: false,
         “admin”: false,
         “language”: “en”
         “password”: null,
         “passwordSet”: true,
         “source”: “local”
      },
      “user2”: {
         “enabled”: true,
         “control”: false,
         “admin”: false,
         “language”: “en”
         “password”: null,
         “passwordSet”: true,
         “source”: “local”
      }
   },
   “retCode”: 0,
   “retMsg”: “Success”
}
```

The following example shows a non *admin* user *get* request specifying username and password within the API request body.

/api/auth

```
{
   "username": “user1”,
   “password”: “user1 password”,
   “cmd”: “get”,
   “data”: {}
}
```

```
{
   “data”: {
      “user1”: {
         “enabled”: true,
         “control”: false,
         “admin”: false,
         “language”: “en”
         “password”: null,
         “passwordSet”: true,
         “source”: “local”
      }
   },
   “retCode”: 0,
   “retMsg”: “Success”
}
```

Table 2.14 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1003	Not authorized. Not enough permissions. Returned if a non *admin* user attempts to get a */api/auth/ID* path where ID does not match the non *admin* user.
1006	Invalid credential combination. Password is incorrect.
3000	Path invalid. An invalid attribute was requested, e.g. */api/auth/ID/unknownAttrName*.
3001	Path not found. An admin user attempted to get a */api/auth/ID* on a non-existent user.

Command: *add*

The *add* command adds a local system user. Only *admin* users may issue the *add* command. The following example adds user “user3” using an authentication *token* from an *admin* user.

/api/auth

```
{
   "token": “a8ff7457”,
   “cmd”: “add”,
   “data”: {
      “username”: “user3”,
      “password”: “cba32156$”,
      “language”: “en”,
      “enabled”: true,
      “control”: false,
      “admin”: false
   }
}
```

```
{
   “data”: {},
   “retCode”: 0,
   “retMsg”: “Success”
}
```

Table 2.15 *add* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the request.
1006	Invalid credential combination. Password is incorrect.
2000	JSON invalid. Missing field, unexpected field, unexpected field data type.
3000	Invalid path (not */api/auth*).
4011	Invalid username.
4013	Invalid option. *enabled* is specified and is false, or *control* and *admin* do not have the same value.
4015	Maximum number of users would be exceeded.
4017	Duplicate entry. Username already exists.
4019	Invalid password.

Command: *delete*

The *delete* command deletes a local system user. Only *admin* users may issue the delete command. The following example deletes user “user3” using an authentication *token* from an admin user.

/api/auth/user3

```
{
   "token": “a8ff7457”,
   “cmd”: “delete”,
   “data”: {}
}
```

```
{
   “data”: {},
   “retCode”: 0,
   “retMsg”: “Success”
}
```

Table 2.16 *delete* Return Codes

Code	Comments
0	Success
1000	Must have at least one *admin* user. Attempt was made to delete the only *admin* user.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the request.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path (not */api/auth*).
3001	Path not found. The ID specified in the path is not found.
3003	Request body contains non - empty *data* object.
3013	Field not applicable. if *data* object specified, it is not empty

Command: *set*

The *set* command updates a local system user. When non *admin* users issue a set request, they are limited to updating their *password*. *Admin* users may update any non-read-only attribute. The following example shows non *admin* user “user2” updating their password.

/api/auth/user2

```
{
   "token": “bc984dj5”,
   “cmd”: “set”,
   “data”: {
      “password”: “cat23Dog%”
   }
}
```

```
{
   “data”: {},
   “retCode”: 0,
   “retMsg”: “Success”
}
```

The following example shows an *admin* user updating the *password* of user “user2”, along with upgrading the user to an *admin*.

/api/auth/user2

```
{
   "token": “a8ff7457”,
   “cmd”: “set”,
   “data”: {
      “password”: “noMoreCats%”
      “control”: true,
      “admin”: true
   }
}
```

```
{
   “data”: {},
   “retCode”: 0,
   “retMsg”: “Success”
}
```

Table 2.17 *set* Return Codes

Code	Comments
0	Success
1000	Must have at least one *admin* user. Attempt was made to update the access level of an *admin* user to non *admin* and the user is the the only *admin* user.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted to update something other than their *password* or attempted to update a different user.
1006	Invalid credential combination. Password is incorrect.
2000	Invalid request. No request body or invalid attribute specified in path or request body.
3000	Invalid path.
3001	Path not found. The ID specified in the path is not found.

Table 2.17 set Return Codes (continued)

Code	Comments
3003	Read-only attribute specified.
4013	Invalid option. *enabled* is specified and is false, or *control* and *admin* do not have the same value.
4019	Invalid password.

2.3 System

Table 2.18 System: /api/sys

Object		Data			Notes
	Field	Format	Range	Default	Read-only collection of system information
sys					
	model	String		RDU1xx Platform	Read-only
	build	String			Read-only
	version	String			Read-only
	serialNumber	String			Read-only
	apiVersion	String			Read-only
	label	String		See notes	Defaults to “name” field. Settable at */api/conf/system/label*.
	name	String	“Vertiv card”	See notes	"Vertiv card” always
sys/contact					
	description	String			Read-only at this location. Settable at */api/conf/contact/description*.
	location	String			Read-only at this location. Settable at */api/conf/contact/location*.

The */api/sys* object contains a non-settable collection of generic system information. Some fields are duplicates of other fields elsewhere in the API tree. The */api/sys* branch is readable by all users

Command: *get*

All authenticated users may view the system information contained within */api/sys*.

/api/sys

```
{
   “token”: ”f3e2d1c0”,
   “cmd”: “get”,
   “data”: {}
}
```

```
{
   “data”: {
      “model”: “RDU1xx Platform”,
      “build”: “RDU101_1.5.0.0_0000006”,
      “version”: “1.5.0.0”,
      “serialNumber”: “02DA”,
      “apiVersion”: “1.0”,
      “label”: “RDU101-Lab3”,
      “name”: “Vertiv RDU101”,
      “contact”: {
         “location”: “Lab 3”,
         “description”: “Lab 3 UPS”
      }
   },
   “retCode”: 0,
   “retMsg”: “Success”
}
```

Table 2.19 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.

Command: *reboot*

Only Administrators may issue the reboot command.

/api/sys

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “reboot”,
   “data”: {}
}
```

Following the sending of a successful response, the device will initiate a reboot.

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {}
}
```

Table 2.20 *reboot* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.

2.4 Configuration

2.4.1 System configuration

Table 2.21 System Configuration: /api/conf/system

Object		Data			Notes
	Field	Format	Range	Default	*set* on these objects require *admin* privilege
conf/system					
	label	String	0 to 64 characters. ‘<’ and ‘>’ characters are not valid.	See notes	Defaults to the value of */api/sys/name*. Modification requires reboot of device for change to take effect.
	hostname	Hostname	2 to 64 characters		Modification requires reboot of device for change to take effect.
	displayTemperatureUnits	String	"celsius" or "fahrenheit"	“celsius”	

Command: *get*

All authenticated users may view the system configuration.

/api/conf/system

```
{
   “token”: ”f3e2d1c0”,
   “cmd”: “get”,
   “data”: {}
}
```

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {
      "displayTemperatureUnits" : "celsius",
      “hostname”: “lab3-rdu101”,
      “label”: “RDU101-Lab3”,
   }
}
```

Table 2.22 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.

Command: *set*

Only administrators may update the system configuration. The following example updates the *hostname* attribute.

/api/conf/system

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “hostname”: “lab3a-rdu101”
   }
}
```

```
{
   “data”: {},
   “retCode”: 1,
   “retMsg”: “Success, reboot required”
}
```

Table 2.23 Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
4001	Input string field exceeds max length.
4002	Input string field contains invalid characters.
4009	Invalid IP (hostname).
4021	Input string field does not meet min length.
4022	Input field not a string when expected a string .

2.4.2 Contact configuration

Table 2.24 Contact Configuration: /api/conf/contact

Object		Data			Notes
	Field	Format	Range	Default	set on these objects require *admin* privilege
conf/contact					
	description	String	0 to 50 characters. '<' and '>' characters are not valid.		Modification requires reboot of device for change to take effect.
	location	String	0 to 50 characters. '<' and '>' characters are not valid.		Modification requires reboot of device for change to take effect.
	contactInfo	String	0 to 50 characters. '<' and '>' characters are not valid.		Modification requires reboot of device for change to take effect. Geist API has contactEmail, contactName, and contactPhone fields instead of contactInfo.

Command: *get*

All authenticated users may view the contact configuration.

/api/conf/contact

```
{
   “token”: ”f3e2d1c0”,
   “cmd”: “get”,
   “data”: {}
}
```

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {
      “location”: “Lab 3”,
      “description”: “Lab 3 UPS”,
      “contactInfo”: “John Doe 614-555-5555"
   }
}
```

Table 2.25 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.

Command: *set*

Only administrators may update the contact configuration. The following example updates the *location* attribute.

/api/conf/contact

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “location”: “Lab 3A”
   }
}
```

```
{
   “data”: {},
   “retCode”: 1,
   “retMsg”: “Success, reboot required”
}
```

Table 2.26 *set* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3002	Unknown input field.
4001	Input string field exceeds max length.
4002	Input string contains invalid characters.
4022	Input field not a string when expected a string.

2.4.3 Time configuration

Table 2.27 Time Configuration: /api/conf/time

Object		Data		Notes	
	Field	Format	Range	Default	*set* on these objects require Administrator access.
conf/time					
	mode	String	"ntp", "manual", "modbus", "bacnet", "vms", "life", "remSrvSys"		
	datetime	Date/Time	Format is "YYYY-MM-DD HH:MM:SS" with hours ranging from 0-23.		This field is displayed in local time. Only settable when mode is "manual".
	zone	String	See Time Zone table below	(UTC+00:00) Coordinated Universal Time	
	ntpServer1	Hostname or IP Address	0 to 64 characters		
	ntpServer2	Hostname or IP Address	0 to 64 characters		
	ntpSyncRate	Integer	1, 12, 24	1	Units are hours
	enableSyncToManagedDevice	Boolean	true or false	FALSE	Enable automatic writing of the time to the managed device.
	managedDeviceSyncRate	Integer	1, 12, 24	1	Units are hours

The following table lists the supported labels for the *zone* attribute. The first column lists the *zone* labels used by the card web GUI. The second column lists an alternative set of supported labels.

Table 2.28 Time *Zone* Labels

Time *Zone* Labels (for *zone*)	
RDU101 Web GUI Label	**Alternative Label**
“(GMT) UTC”	"(UTC+00:00) Coordinated Universal Time"
“(GMT-12:00) International Date Line West”	"(UTC-12:00) International Date Line West"
“(GMT-11:00) UTC-11"	"(UTC-11:00) UTC-11"
“(GMT-10:00) Hawaii”	"(UTC-10:00) Hawaii"
“(GMT-09:00) Alaska”	"(UTC-09:00) Alaska"
“(GMT-08:00) Pacific Time (US and Canada), Baja California”	"(UTC-08:00) Pacific Time (US and Canada), Baja California"
“(GMT-07:00) Arizona”	"(UTC-07:00) Arizona"
“(GMT-07:00) Chihuahua, La Paz, Mazatlan”	"(UTC-07:00) Chihuahua, La Paz, Mazatlan"
“(GMT-07:00) Mountain Time (US and Canada)”	"(UTC-07:00) Mountain Time (US and Canada)"
“(GMT-06:00) Central America”	"(UTC-06:00) Central America"
“(GMT-06:00) Central Time (US and Canada)”	"(UTC-06:00) Central Time (US and Canada)"
“(GMT-06:00) Guadalajara, Mexico City, Monterrey”	"(UTC-06:00) Guadalajara, Mexico City, Monterrey"
“(GMT-06:00) Saskatchewan”	"(UTC-06:00) Saskatchewan"
“(GMT-05:00) Bogota, Lima, Quito”	"(UTC-05:00) Bogota, Lima, Quito"
“(GMT-05:00) Eastern Time (US and Canada)”	"(UTC-05:00) Eastern Time (US and Canada)"
“(GMT-05:00) Indiana (East)”	"(UTC-05:00) Indiana (East)"
“(GMT-04:30) Caracas”	"(UTC-04:30) Caracas"
“(GMT-04:00) Asuncion”	"(UTC-04:00) Asuncion"
“(GMT-04:00) Atlantic Time (Canada)”	"(UTC-04:00) Atlantic Time (Canada)"
“(GMT-04:00) Cuiaba”	"(UTC-04:00) Cuiaba"
“(GMT-04:00) Georgetown, La Paz, Manaus, San Juan”	"(UTC-04:00) Georgetown, La Paz, Manaus, San Juan"
“(GMT-04:00) Santiago”	"(UTC-04:00) Santiago"
“(GMT-03:30) Newfoundland”	"(UTC-03:30) Newfoundland"
“(GMT-03:00) Brasilia”	"(UTC-03:00) Brasilia"
“(GMT-03:00) Buenos Aires”	"(UTC-03:00) Buenos Aires"
“(GMT-03:00) Cayenne, Fortaleza”	"(UTC-03:00) Cayenne, Fortaleza"
“(GMT-03:00) Greenland”	"(UTC-03:00) Greenland"
“(GMT-03:00) Montevideo”	"(UTC-03:00) Montevideo"
“(GMT-03:00) Salvador”	"(UTC-03:00) Salvador"
“(GMT-02:00) UTC-2"	“(UTC-02:00) Coordinated Universal Time-02”

Table 2.28 Time Zone Labels (continued)

Time *Zone* Labels (for *zone*)	
RDU101 Web GUI Label	**Alternative Label**
“(GMT-01:00) Azores”	"(UTC-01:00) Azores"
“(GMT-01:00) Cape Verde Is.”	"(UTC-01:00) Cape Verde Is."
“(GMT) Casablanca”	"(UTC+00:00) Casablanca"
“(GMT) Dublin, Edinburgh, Lisbon, London”	"(UTC+00:00) Dublin, Edinburgh, Lisbon, London"
“(GMT) Monrovia, Reykjavik”	"(UTC+00:00) Monrovia, Reykjavik"
“(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna”	"(UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna"
“(GMT+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague”	"(UTC+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague"
“(GMT+01:00) Brussels, Copenhagen, Madrid, Paris”	"(UTC+01:00) Brussels, Copenhagen, Madrid, Paris"
“(GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb”	"(UTC+01:00) Sarajevo, Skopje, Warsaw, Zagreb"
“(GMT+01:00) West Central Africa”	"(UTC+01:00) West Central Africa"
“(GMT+01:00) Windhoek”	"(UTC+01:00) Windhoek"
“(GMT+02:00) Athens, Bucharest, Istanbul”	"(UTC+02:00) Athens, Bucharest, Istanbul"
“(GMT+02:00) Beirut”	"(UTC+02:00) Beirut"
“(GMT+02:00) Cairo”	"(UTC+02:00) Cairo"
“(GMT+02:00) Harare, Pretoria”	"(UTC+02:00) Harare, Pretoria, Tripoli"
“(GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius”	"(UTC+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius"
“(GMT+02:00) Jerusalem”	"(UTC+02:00) Jerusalem"
“(GMT+02:00) Nicosia”	"(UTC+02:00) Nicosia, Damascus"
“(GMT+03:00) Baghdad”	"(UTC+03:00) Baghdad, Amman"
“(GMT+03:00) Kaliningrad, Minsk”	"(UTC+03:00) Kaliningrad, Minsk"
“(GMT+03:00) Kuwait, Riyadh”	"(UTC+03:00) Kuwait, Riyadh"
“(GMT+03:00) Nairobi”	"(UTC+03:00) Nairobi"
“(GMT+03:30) Tehran”	"(UTC+03:30) Tehran"
“(GMT+04:00) Abu Dhabi, Muscat”	"(UTC+04:00) Abu Dhabi, Muscat"
“(GMT+04:00) Baku”	"(UTC+04:00) Baku"
“(GMT+04:00) Moscow, St. Petersburg, Volgograd”	"(UTC+04:00) Moscow, St. Petersburg, Volgograd"
“(GMT+04:00) Port Louis”	"(UTC+04:00) Port Louis"
“(GMT+04:00) Tbilisi”	"(UTC+04:00) Tbilisi"
“(GMT+04:00) Yerevan”	"(UTC+04:00) Yerevan"
“(GMT+04:30) Kabul”	"(UTC+04:30) Kabul"
“(GMT+05:00) Islamabad, Karachi, Tashkent”	"(UTC+05:00) Islamabad, Karachi, Tashkent"
“(GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi”	"(UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi"

Table 2.28 Time Zone Labels (continued)

Time *Zone* Labels (for *zone*)	
RDU101 Web GUI Label	**Alternative Label**
“(GMT+05:30) Sri Jayawardenepura”	"(UTC+05:30) Sri Jayawardenepura"
“(GMT+05:45) Kathmandu”	"(UTC+05:45) Kathmandu"
“(GMT+06:00) Astana”	"(UTC+06:00) Astana"
“(GMT+06:00) Dhaka”	"(UTC+06:00) Dhaka"
“(GMT+06:00) Ekaterinburg”	"(UTC+06:00) Ekaterinburg"
“(GMT+06:30) Yangon”	"(UTC+06:30) Yangon"
“(GMT+07:00) Bangkok, Hanoi, Jakarta”	"(UTC+07:00) Bangkok, Hanoi, Jakarta"
“(GMT+07:00) Novosibirsk”	"(UTC+07:00) Novosibirsk"
“(GMT+08:00) Beijing, Chongqing, Hong Kong, Urumqi”	"(UTC+08:00) Beijing, Chongqing, Hong Kong, Urumqi"
“(GMT+08:00) Krasnoyarsk”	"(UTC+08:00) Krasnoyarsk"
“(GMT+08:00) Kuala Lumpur, Singapore”	"(UTC+08:00) Kuala Lumpur, Singapore"
“(GMT+08:00) Perth”	"(UTC+08:00) Perth"
“(GMT+08:00) Taipei”	"(UTC+08:00) Taipei"
“(GMT+08:00) Ulaanbaatar”	"(UTC+08:00) Ulaanbaatar"
“(GMT+09:00) Irkutsk”	"(UTC+09:00) Irkutsk"
“(GMT+09:00) Osaka, Sapporo, Tokyo”	"(UTC+09:00) Osaka, Sapporo, Tokyo"
“(GMT+09:00) Seoul”	"(UTC+09:00) Seoul"
“(GMT+09:30) Adelaide”	"(UTC+09:30) Adelaide"
“(GMT+09:30) Darwin”	"(UTC+09:30) Darwin"
“(GMT+10:00) Brisbane”	"(UTC+10:00) Brisbane"
“(GMT+10:00) Canberra, Melbourne, Sydney”	"(UTC+10:00) Canberra, Melbourne, Sydney"
“(GMT+10:00) Guam, Port Moresby”	"(UTC+10:00) Guam, Port Moresby"
“(GMT+10:00) Hobart”	"(UTC+10:00) Hobart"
“(GMT+10:00) Yakutsk”	"(UTC+10:00) Yakutsk"
“(GMT+11:00) Solomon Is., New Caledonia”	"(UTC+11:00) Solomon Is., New Caledonia"
“(GMT+11:00) Vladivostok”	"(UTC+11:00) Vladivostok"
“(GMT+12:00) Auckland, Wellington”	"(UTC+12:00) Auckland, Wellington"
“(GMT+12:00) Fiji”	"(UTC+12:00) Fiji"
“(GMT+12:00) Magadan”	"(UTC+12:00) Magadan"
“(GMT+12:00) UTC+12”	"(UTC+12:00) UTC+12"
“(GMT+13:00) Nuku'alofa”	"(UTC+13:00) Nuku'alofa"
“(GMT+13:00) Samoa”	"(UTC+13:00) Samoa"
“(GMT+14:00) Kiritimati, Christmas Island, Kiribati”	"(UTC+14:00) Kiritimati, Christmas Island, Kiribati

Command: *get*

All authenticated users may view the time information.

/api/conf/time

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “get”,
   “data”: {}
}
```

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {
      “mode": “ntp”,
      “datetime”: “2020-10-15 14:40:40”,
      “zone": “(UTC+00:00) Coordinated Universal Time”,
      “ntpServer1": “192.168.1.10”,
      “ntpServer2": “192.168.1.11”
      “ntpSyncRate": 24,
      “enableSyncToManagedDevice": false,
      “managedDeviceSyncRate": 1
   }
}
```

Table 2.29 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.

Command: *set*

Only administrators may update the time information. The following rules and notes regarding updating time information should be considered:

Setting the time *mode* to "ntp" requires a valid entry in at least one of the NTP servers.

When setting the *datetime*, all the following apply:

- The *mode* may not be updated in the same request. This is because a time source (*mode*) change takes 8-9 seconds for the card to recognize the time source change.
- The current time source (*mode*) must be “manual”

- The *zone* may not be updated in the same request.

The following example sets the time *mode* to “manual”.

/api/conf/time

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “mode”: “manual”
   }
}
```

```
{
   “data”: {},
   “retCode”: 0,
   “retMsg”: “Success”
}
```

Table 2.30 *set* Return Codes

Code	Comments
0	Success
1	Success, reboot required. Updating ntp servers requires a reboot.
2	Success, set datetime scheduled. This will occur when setting the *datetime* backwards 9 or more seconds. This indicates the set datetime request has been received, but was not immediately executed. The setting of datetime was scheduled to occur one second in the future. This is to allow the API response to be sent out prior to setting the datetime, which in this negative time shift case, will cause a restart of the webserver.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3002	Unknown input field.
4001	Input exceeds max length.
4004	Input boolean field not a boolean.
4006	Input integer field not an integer.
4013	Invalid option. An input field contains an invalid option selection.
4014	Invalid datetime.

Table 2.30 set Return Codes (continued)

Code	Comments
4022	Input string field not a string.
6002	• NTP mode requires at least one non-blank server. • Returned if both *ntpServer1* and *ntpServer2* are not configured and *mode* is already “ntp” or is being set to “ntp”
6007	• datetime is being set but mode is not “manual”. • The request would have resulted in changes to both *datetime* and *zone*. • The request would have resulted in changes to both *datetime* and *mode*.

The following example sets the *datetime*.

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “datetime”: “2021-02-04 15:45:00”
   }
}
```

```
{
   “data”: {},
   “retCode”: 0,
   “retMsg”: “Success”
}
```

The following example sets the *datetime* where the requested *datetime* is in the past by nine or more seconds. In this case, the attempt to update the *datetime* will occur one second following the request.

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “datetime”: “2021-02-04 05:45:00”
   }
}
```

```
{
   “data”: {},
   “retCode”: 2,
   “retMsg”: “Success, set datetime scheduled”
}
```

2.4.4 Network

Network: /api/conf/network

The card network configuration is contained within the *ethernet* (ID) node of */api/conf/network*.

- Geist API has a single “dhcpOn” setting that applies to both IPv4 and IPv6. The card API has separate DHCP settings for IPv4 and IPv6.
- There is a fixed length list of two ethernet addresses at */api/conf/network/ethernet/address*, one for IPv4 (ID 0) and one for IPv6 (ID 1). Geist API uses variable length list.
- There is a fixed length list of two routes at */api/conf/network/ethernet/route*, one for IPv4 (ID 0) and one for IPv6 (ID 1). Geist API uses variable length list.
- There is a variable length list of dns objects at */api/conf/network/ethernet/dns*.

Table 2.31 Variable List

Object		Data			Notes
	Field	Format	Range	Default	*set* on these objects require Administrator access.
network/ethernet					
	type	String	“lan”	“lan”	Read-only
	label	String	“ethernet”	“ethernet”	Read-only
	order	Integer	0	0	Read-only
	removable	Boolean	FALSE	FALSE	Read-only. The ethernet interface is not removable.
	enabled	Boolean	TRUE	TRUE	Read-only. The ethernet interface is always enabled.
	macAddr	String			Read-only
	dhcpOn	Boolean	true, false	TRUE	
	dhcpOnIPv6	Boolean	true, false	TRUE	
network/ethernet/address			2 objects		
network/ethernet/address/0					IPv4 address
	mutable	Boolean	true, false	FALSE	Read-only. false if DHCP enabled, true otherwise.
	prefix	Integer	0 to 32	24	Number of bits in subnet mask.
	address	IP Address	IPv4 address		Read-only if DHCP enabled.
network/ethernet/address/1					IPv6 address
	mutable	Boolean	true, false	FALSE	Read-only. false if DHCP enabled, true otherwise.
	prefix	Integer	0 to 128	64	

Table 2.31 Variable List (continued)

Object		Data			Notes
	Field	**Format**	**Range**	**Default**	*set* on these objects require Administrator access.
	address	IP Address	IPv6 Address	"::"	Read-only if DHCP enabled.
	slaac	IP Address	IPv6 Address		Read-only
	linklocal	IP Address	IPv6 Address		Read-only
network/ethernet/route			2 objects		
network/ethernet/route/0					IPv4 route
	mutable	Boolean	true, false	FALSE	Read-only. false if DHCP enabled, true otherwise.
	interface	String	"all"	"all"	Read-only. The card has single default route.
	gateway	IP Address	IPv4 Address	""	By default, no gateway is set.
	prefix	Integer	0	0	Read-only. The card has single default route.
	destination	IP Address	"0.0.0.0"	"0.0.0.0"	Read-only. The card has single default route.
network/ethernet/route/1					IPv6 route
	mutable	Boolean	true, false	FALSE	Read-only. false if DHCP enabled, true otherwise.
	interface	String	"all"	"all"	Read-only. The card has single default route.
	gateway	IP Address	IPv6 Address	""	By default, no gateway is set.
	prefix	Integer	0	0	Read-only. The card has single default route.
	destination	IP Address	"::"	"::"	Ready-only. The card has single default route.
network/ethernet/dns/ID			0 to 2 objects		IPv4 and IPv6 DNS Servers
	address	IP Address	IPv4 or IPv6 Address		
	mutable	Boolean	true, false	FALSE	Read-only. The card defaults both IPv4 and IPv6 DNS server source as Automatic.

Command: *get*

All authenticated users may view the network information.

/api/conf/network

```
{
   "token": "f3e2d1c0",
   "cmd": "get",
   "data": {}
}
```

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {
      “ethernet”: {
         “dhcpOn”: true,
         “dhcpOnIPv6”: true,
         “macAddr”: “98:90:96:d2:14:9e”,
         “order”: 0,
         “label”: “ethernet”,
         “type”: “lan”,
         “enabled”: true,
         “removable”: false,
         “address”: {
            “0”: {
               “mutable”: false,
               “prefix”: 24,
               “address”: “192.168.4.23”
            },
            “1”: {
               “mutable”: false,
               “prefix”: 64,
               “address”: “fe80::53ce:3fd5:5eb2:e9c6”,
               “slaac”: “”,
               “linklocal”: “fe80::209:f5ff:fe20:f65e”
            }
         },
         “route”: {
            “0”: {
               “mutable”: false,
               “interface”: “all”,
               “gateway”: “192.168.4.1”,
               “prefix”: 0,
               “destination”: “0.0.0.0”
            },
            “1”: {
               “mutable”: false,
               “interface”: “all”,
               “gateway”: “”,
               “prefix”: 0,
               “destination”: “::”
            }
         },
         “dns”: {
            “0”: {
               “mutable”: false,
               “address”: “64.233.222.2”
            },
            “1”: {
               “mutable”: false,
               “address”: “64.233.222.7”
            }
         }
      }
   }
}
```

get Notes

- The returned *address* objects will have the *mutable* attribute set to true if the *address* was statically configured, or false if address obtained from DHCP (IPv4) or Automatic (IPv6).
- The returned *route* objects will have the *mutable* attribute set to true if the *gateway* was statically configured, or false if *gateway* obtained from DHCP (IPv4) or Automatic (IPv6).
- The returned *dns* objects will have the *mutable* attribute set to true if the *dns* object was added manually, or false if the information was obtained automatically.

Table 2.32 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.

Command: *set*

Only administrators may update the network information.

The following example enables IPv4 and IPv6 DHCP.

/api/conf/network/ethernet

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “dhcpOn”: true,
      “dhcpOnIPv6”: true,
   }
}
```

The following example disables IPv4 and IPv6 DHCP and sets IPv4 address.

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “dhcpOn”: false,
      “dhcpOnIPv6”: false,
      “address”: {
         “0”: {
            “prefix”: 24,
            “address”: “192.168.4.25”
         }
      }
   }
}
```

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {}
}
```

Table 2.33 *set* Return Codes

Code	Comments
0	Success
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3001	Path not found.
3002	Unknown input field.
4004	Input boolean field not a boolean.
4005	Out of range. An input field falls outside the valid range for the field.
4006	Input integer field not an integer.
4009	Invalid IP. An input field is not a valid IP address when one is expected.
4013	Invalid option. An input field contains an invalid option selection.
4020	Input field is read-only.
4022	Input string field not a string.
6000	Inconsistent state. The command would leave the system in an inconsistent state so it is rejected. -IPv4 DHCP is disabled and IPv4 address (index 0) and/or route is/are not configured. -IPv6 DHCP is disabled and IPv6 address (index 1) is not configured. -IPv4 DNS primary and secondary servers are the same. -IPv6 DNS primary and secondary server are the same.

Command *add*: /api/conf/network/ethernet/dns

Supports adds and deletes for up to two DNS addresses. Address field can be IPv4 or IPv6, for a total of two DNS addresses. DHCP acquired DNS addresses returned in get commands 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.

Only administrators may update the network DNS information.

The following example adds an IPv4 DNS address.

/api/conf/network/ethernet/dns

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “add”,
   “data”: {
      “address”: “192.168.4.66”
   }
}
```

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {}
}
```

add Notes

- If IPv4 address is added, then The card will disable IPv4 Automatic mode.
- If IPv6 address is added, then The card will disable IPv6 Automatic mode.

Table 2.34 *add* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3002	Unknown input field.
4009	Invalid IP. An input field is not a valid IP address when one is expected.
4015	Out of bounds. Number of entries would be exceeded.
4022	Input string field not a string.

Command *delete*: /api/conf/network/ethernet/dns/ID

Only administrators may delete network DNS entries.

The following example deletes the DNS entry at ID “0”.

/api/conf/network/ethernet/dns/0

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “delete”,
   “data”: {}
}
```

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {}
}
```

delete Notes

- If IPv4 address is deleted and no more IPv4 DNS entries exist, then the card will enable IPv4 *Automatic* mode.
- If IPv6 address is deleted and no more IPv6 DNS entries exist, then the card will enable IPv6 *Automatic* mode.

Table 2.35 *delete* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3001	Path not found. ID of target specified in path does not exist.

2.4.5 SNMP

Configure the device to work as a server for Simple Network Management Protocol. Setting both "v1v2cEnabled" and "v3Enabled" to false will turn off SNMP support.

Table 2.36 SNMP

Object		Data			Notes
	Field	**Format**	**Range**	**Default**	*set* on these objects require Administrator access.
snmp					
	v1v2cEnabled	Boolean	true or false	TRUE	Note: The card defaults this to true, but it defaults the Protocols/SNMP Protocol setting to disabled.
	v3Enabled	Boolean	true or false	TRUE	Note: The card defaults this to true, but it defaults the Protocols/SNMP Protocol setting to disabled.
	port	Integer	161	161	Read-only
	authTrapsEnabled	Boolean	true or false	FALSE	
	heartbeatTrapInterval	Integer	0, 5, 30, 60, 240, 480, 720, 1440	1440	Units are minutes. Zero indicates heartbeat trap intervals are disabled.
	rfc1628MIBEnabled	Boolean	true or false	TRUE	
	rfc1628MIBTrapsEnabled	Boolean	true or false	TRUE	
	lgpMIBEnabled	Boolean	true or false	TRUE	
	lgpMIBTrapsEnabled	Boolean	true or false	TRUE	
	lgpMIBSysNotifyTrap	Boolean	true or false	TRUE	
	incSysNameLocInTraps	Boolean	true or false	FALSE	Include SysName and SysLocation in traps
	engineId	String	1 to 27 characters		• Read-only (matches Geist API). • The card defaults to generating engineId from source MAC address.
snmp/access/ID			0 to 20 v1/v2c access objects		Note: Access objects will have ID 0 – 19.
	name	Hostname or IP Address	"0.0.0.0" to accept queries from any IPv4 host, "::" to accept queries from any IPv6 host, or leave blank to accept either. Max length of 125 characters		
	type	String	"read", "write"		"read" allows read-only access while "write" allows read-write access.
	community	String	• 0 to 32 characters • Printable characters are allowed except for <, >, comma, semi-colon, and space.		Returned in clear text.
snmp/target/ID			0 to 20 v1 trap targets. 0 to n v3 trap targets where total length of v3 target names is < 126 – n.		See note for "trapVersion" regarding v2 support. Note: v1 trap targets will have ID 0 – 19, while v3 traps will have ID starting at 20.
	name	Hostname or	Max length of 125 characters		V1/ trap names each can be up to 125,

Table 2.36 SNMP (continued)

Object		Data			Notes
	Field	**Format**	**Range**	**Default**	*set* on these objects require Administrator access.
		IP Address			while the sum of V3 trap name lengths must be 126 – n, where n is the number of V3 targets.
	port	Integer	1 to 65535	162	All v3 targets must have the same port. v1 and v2c targets may have different ports.
	trapVersion	String	"1", "3"	"1"	• Geist supports version 1, 2c, and 3. The RDU places the targets in either SNMPv1 Trap or SNMPv3 User. The RDU can't distinguish between versions 1 and 2 (once stored), so 2 will not be supported. • It is not valid to update (set command) the trapVersion.
	community	String	• 0 to 32 characters • Printable characters are allowed except for <, >, comma, semi-colon, and space.		Returned in clear text. This field applies only to v1 targets.
snmp/user/ID			3 objects		• SNMPv3 users. Three fixed entries for each user type: read-only, read-write, and trap. • Read is object 0, write is object 1, and trap is object 2.
	username	String	• 0 to 32 characters • Printable characters are allowed with the exception of <, >, colon, semi-colon, space, tab, double-quote and question mark.		The card has no default. Will be blank on get until a set updates each object.
	type	String	"read", "write", "trap"		Read-only
	authType	String	"none", "md5", "sha1"	"none"	
	authPasswordSet	Boolean	true, false		Read-only
	authPassword	String	• 8 to 64 characters • Printable characters are allowed with the exception of <, >, colon, semi-colon, space, tab, double-quote and question mark.		Returned as null.
	privType	String	"none", "des", "aes"	"none"	
	privPasswordSet	Boolean	true, false		Read-only
	privPassword	String	• 8 to 64 characters • Printable characters are allowed with the exception of <, >, colon, semi-colon, space, tab, double-quote and question mark.		Returned as null.

Command: *get*

All authenticated users may view the snmp information.

/api/conf/snmp

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “get”,
   “data”: {}
}
```


```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {
      “v1v2cEnabled”: true,
      “v3Enabled”: true,
      “port”: 161,
      “authTrapsEnabled”: false,
      “heartbeatTrapInterval”: 1440,
      “rfc1628MIBEnabled”: true,
      “rfc1628MIBTrapsEnabled”: true,
      “lgpMIBEnabled”: true,
      “lgpMIBTrapsEnabled”: true,
      “lgpMIBSysNotifyTrap”: true,
      “incSysNameLocInTraps”: false,
      “engineId”: “80001F88030009F520F65E“,
      “access”: {
         “0”: {
            “name”: “192.168.10.10”,
            “type”: “read”,
            “community”: “public”
         },
         “1”: {
            “name”: “192.168.10.20”,
            “type”: “write”,
            “community”: “private”
         }
      },
      “target”: {
         “0”: {
            “name”: “192.168.123.1”,
            “port”: 162,
            “trapVersion”: “1”,
            “community”: “trap”
         },
         “1”: {
            “name”: “192.168.123.1”,
            “port”: 162,
            “trapVersion”: “3”
         }
```

```
      },
      “user”: {
         “0”: {
            “username”: “”,
            “type”: “read”,
            “authType”: “none”,
            “authPassswordSet”: false,
            “authPassword”: null,
            “privType”: “none”,
            “privPasswordSet”: false,
            “privPassword”: null
         },
         “1”: {
            “username”: “”,
            “type”: “write”,
            “authType”: “md5”,
            “authPassswordSet”: false,
            “authPassword”: null,
            “privType”: “des”,
            “privPasswordSet”: false,
            “privPassword”: null
         },
         “2”: {
            “username”: “”,
            “type”: “trap”,
            “authType”: “md5”,
            “authPassswordSet”: false,
            “authPassword”: null,
            “privType”: “des”,
            “privPasswordSet”: false,
            “privPassword”: null
         }
      }
   }
}
```


Table 2.37 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.

Command: *set*

Only administrators may update the snmp information. The following example enables snmp v3.

/api/conf/snmp

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “v3Enabled”: true
   }
}
```

```
{
   “retCode”: 1,
   “retMsg”: “Success, reboot required”,
   “data”: {}
}
```

Table 2.38 *set* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3001	Path not found.
3002	Unknown input field.
4001	Input string field exceeds max length.
4002	Input string field contains invalid characters.
4004	Input boolean field not a boolean.
4005	Input field value out of range.
4006	Input integer field not an integer.
4020	Input field is read-only.
4021	Input string field does not meet min length.
4022	Invalid string field not a string.
6000	Inconsistent state. The command would leave the system in an inconsistent state so it is rejected. • V3 user *privType* is not *none* and *authType* is *none*.

SNMP v1/v2c access

A list of access IPs or host names for v1/v2c SNMP access.

Command: *get*

All authenticated users may view the snmp v1/v2c access objects.

/api/conf/snmp/access

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “get”,
   “data”: {}
}
```

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {
      “0”: {
         “name”: “192.168.10.10”,
         “type”: “read”,
         “community”: “public”
      },
      “1”: {
         “name”: “192.168.10.20”,
         “type”: “write”,
         “community”: “private”
      }
   }
}
```

Table 2.39 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3001	Path not found.

Command: *add*

Only administrators may add snmp v1/v2c access objects.

/api/conf/snmp/access

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “add”,
   “data”: {
      “name”: “192.168.10.30”,
      “type”: “write”,
      “community”: “secret”
   }
}
```

```
{
   “retCode”: 1,
   “retMsg”: “Success, reboot required”,
   “data”: {}
}
```

Table 2.40 *add* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
2001	Expected input field missing.
3000	Invalid path.
3001	Path not found.
4001	Input string field exceeds max length.
4005	Input field is out of range.
4009	Invalid IP address.
4013	Invalid option. An input field contains an invalid option selection.
4015	Out of bounds. Maximum number of elements exceeded.

Command: *delete*

Only administrators may delete snmp access objects. The following example deletes ID “1” access.

/api/conf/snmp/access/1

```
{
   “token”: “f3e2d1c0”,
```

```
   “cmd”: “delete”,
   “data”: {}
}
```

```
{
   “retCode”: 1,
   “retMsg”: “Success, reboot required”,
   “data”: {}
}
```

Table 2.41 *delete* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3001	Path not found. ID of access specified in path does not exist.

Command: *set*

Only administrators may update snmp access objects. The following example updates the type of ID “1” access object.

/api/conf/snmp/access/1

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “type”: “write”
   }
}
```

```
{
   “retCode”: 1,
   “retMsg”: “Success, reboot required”,
   “data”: {}
}
```

Table 2.42 *set* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3001	Path not found. ID of access specified in path does not exist.
4001	Input string field exceeds max length.
4005	Input field value out of range.
4009	Invalid IP address.
4013	Invalid option. An input field contains an invalid option selection.
4022	Invalid string field not a string.

SNMP target

A list of recipient IPs or host names for SNMP traps.

Command: *get*

All authenticated users may view the snmp targets.

/api/conf/snmp/target

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “get”,
   “data”: {}
}
```

```
{
   “retCode”: 0,
   “retMsg”: “Success”,
   “data”: {
      “0”: {
         “name”: “192.168.123.1”,
         “port”: 162,
         “trapVersion”: “1”
      },
      “1”: {
         “name”: “192.168.123.1”,
         “port”: 162,
         “trapVersion”: “3”
      },
   }
}
```

Table 2.43 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3001	Path not found.

Command: *add*

Only administrators may add snmp targets. The following example adds a v1 trap target.

Version 1 targets will use the community string provided. Version 3 targets will use the community string corresponding to the *snmp/user/2* object.

/api/conf/snmp/target

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “add”,
   “data”: {
      “name”: “192.168.4.23”,
      “port”: 162,
      “trapVersion”: “1”,
      “community”: “trap”
   }
}
```

```
{
   “retCode”: 1,
   “retMsg”: “Success, reboot required”,
   “data”: {}
}
```

Table 2.44 *add* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
2001	Expected input field missing.
3000	Invalid path.
3001	Path not found.
4001	Input string field exceeds max length.
4005	Input field is out of range.
4009	Invalid IP address.
4013	Invalid option. An input field contains an invalid option selection.
4015	Out of bounds. Maximum number of elements exceeded.

Command: *delete*

Only administrators may delete snmp targets. The following example deletes ID “1” target.

/api/conf/snmp/target/1

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “delete”,
   “data”: {}
}
```

```
{
   “retCode”: 1,
   “retMsg”: “Success, reboot required”,
   “data”: {}
}
```

Table 2.45 *Delete* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3001	Path not found. ID of target specified in path does not exist.

Command: *set*

Only administrators may update snmp targets. The following example updates the port of ID “1” target.

/api/conf/snmp/target/1

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “port”: 163
   }
}
```

```
{
   “retCode”: 1,
   “retMsg”: “Success, reboot required”,
   “data”: {}
}
```

Table 2.46 *set* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3001	Path not found. ID of target specified in path does not exist.
4001	Input string field exceeds max length.
4005	Input field value out of range.
4006	Input integer field not an integer.
4013	Invalid option. An input field contains an invalid option selection.
4020	Input field is read-only.
4021	Input string field does not meet min length.
4022	Invalid string field not a string.

SNMP V3 user

List of credentials for SNMPv3 users that have access to the device. The list has three fixed entries, one for each user type: read-only, read-write, and trap.

Table 2.47 Type of User

Authentication (authType)	Encryption (privType)
“none”	“none”
“md5”, “sha1”	“none”
“md5”, “sha1”	“des”, “aes”

Each user has security settings for authentication and encryption. If authentication is enabled, then "authType" will be "md5" or "sha1" and "authPassword" will be set. For encryption, "privType" will be "des" or "aes" and "privPassword" will be set. Authentication is required for encryption. The acceptable combinations for authentication and privacy are listed in the table above.

The following table has the defaults for each user type.

Table 2.48 User Type Sample

type	username	authType	authPassword	privType	privPassword
“read”		“none”		“none”	
“write”		“none”		“none”	
“trap”		“none”		“none”	

Modifying the "username" field by itself will cause authentication and privacy types and passwords to be reset to "none" and blank respectively. Changing any of the two types by themselves will also cause the associated password to be set to blank.

“*username*” field is blank by default. A set command must be issued on a user object in order to set *username* and other attributes.

Command: *get*

All authenticated users may view the snmp users.

The following example assumes the user objects for *read*, *write*, and *trap* have all previously had a set issued against them to set their *usernames*.

/api/conf/snmp/user

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “get”,
   “data”: {}
}
```

```
{
   “retCode”: 1,
   “retMsg”: “Success, reboot required”,
   “data”: {
      “0”: {
         “username”: “user”,
         “type”: “read”,
         “authType”: “none”,
         “authPassswordSet”: false,
         “authPassword”: null,
         “privType”: “none”,
         “privPasswordSet”: false,
         “privPassword”: null
      },
      “1”: {
         “username”: “manager”,
         “type”: “write”,
         “authType”: “md5”,
         “authPassswordSet”: true,
         “authPassword”: null,
         “privType”: “des”,
         “privPasswordSet”: true,
```

```
            “privPassword”: null
         },
         “2”: {
            “username”: “trap”,
            “type”: “trap”,
            “authType”: “md5”,
            “authPasswordSet”: true,
            “authPassword”: null,
            “privType”: “des”,
            “privPasswordSet”: true,
            “privPassword”: null
         }
      }
}
```


Table 2.49 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.

Command: *set*

Only administrators may update snmp users. The following example updates the username of ID “0” user.

/api/conf/snmp/user/0


```
{
   “token”: “f3e2d1c0”,
   “cmd”: “set”,
   “data”: {
      “username”: "read-user”
   }
}
```



```
{
   “retCode”: 1,
   “retMsg”: “Success, reboot required”,
   “data”: {}
}
```

Table 2.50 *set* Return Codes

Code	Comments
1	Success, reboot required for changes to take effect.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
3001	Path not found. ID of user specified in path does not exist.
4001	Input string field exceeds max length.
4005	Input field value out of range.
4013	Invalid option. An input field contains an invalid option selection.
4020	Input field is read-only.
4021	Input string field does not meet min length.
4022	Invalid string field not a string.
6000	Inconsistent state. The command would leave the system in an inconsistent state so it is rejected. • V3 user *privType* is not *none* and *authType* is *none*.

2.5 Managed Device

2.5.1 Firmware upload

Firmware Upload: */device/transfer/firmware*

The API supports upload of firmware to the managed device. The firmware upload is initiated by the API *upload* command. This command first uploads the file from the host to the card. Next, a transfer of the file from the card to the managed device is initiated. At this point a response is returned. The client may then initiate a series of *get* commands to monitor the status of the upload.

NOTE: The managed device firmware upload is not supported on the IS-UNITY cards.

Table 2.51 Firmware Sample

Object		Data			Notes
	Field	Format	Range	Default	
device/transfer/firmware					
	statusCode	Integer	0 – Idle 1 – In Progress 2 – Complete 3 – Error 4 - Timeout		Read-only.
	statusDescription	String	“Idle”, “In Progress”, “Complete”, “Error”, “Timeout”		Read-only.

Table 2.51 Firmware Sample (continued)

Object		Data			Notes
	Field	Format	Range	Default	
	hostFilename	String			Read-only. Name of file uploaded from host.
	hostFileSize	Integer			Read-only. Size of file uploaded from host.
	additionalInfo	String			Populated with additional information for error or timeout condition.

Command: *upload*

Uploads firmware to the managed device. Only administrators may initiate a firmware upload. This request does not receive a regular API JSON object. The user authentication information (username and password, or token) and the command are required to be passed in as part of the query string. The input must be encoded as multipart/form-data with a single component called “file”. Output is a regular API JSON response with an error code of 0 if the upload was successfully initiated. Once an upload is successfully initiated a get command may be issued to determine the upload status.

HTTP header example:

```
POST /api/device/transfer/firmware?token=12345678&cmd=upload HTTP/1.1Content-Type:
multipart/form-data; boundary=----ABCDEFGHIJKLMNOPQRSTUVWXYZ
----ABCDEFGHIJKLMNOPQRSTUVWXYZ
Content-Disposition: form-data; name="file"; filename="fwUpload.bin"Content-Type:
application/octet-stream....
```

/api/device/transfer/firmware

```
{
   “retCode”: 0,
   “retMsg”: “Success [ Firmware upload started ]”
}
```

Table 2.52 *upload* Return Codes

Code	Comments
0	Success, upload has been initiated.
1002	Session expired. Provided token is invalid.
1003	Not enough permissions. A non *admin* user attempted the command.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.
5002	System busy.
5005	API library error. Socket error.
5006	System error. Insufficient space on card to upload file.
5007	File transfer service error.
5008	API request error. Error parsing and processing the HTTP request.

Command: *get*

The get command retrieves the status of the upload.

/api/device/transfer/firmware

```
{
   “token”: “f3e2d1c0”,
   “cmd”: “get”
}
```

Invoking a get prior to initiating an upload you will receive the following response:

```
{
   "data": {
      "statusCode": 0,
      "statusDescription": "Idle",
      "hostFilename": "",
      "hostFileSize": 0,
      "additionalInfo": ""
   },
   "retCode": 0,
   "retMsg": "Success"
}
```

An in-progress upload returns the following:

```
{
   "data": {
      "statusCode": 1,
      "statusDescription": "In progress",
      "hostFilename": "fwUpload.bin",
      "hostFileSize": 2767977,
      "additionalInfo": ""
   },
   "retCode": 0,
   "retMsg": "Success"
}
```

When the upload is complete the following is returned on a *get*:

```
{
   "data": {
      "statusCode": 2,
      "statusDescription": "Complete",
      "hostFilename": "fwUpload.bin",
      "hostFileSize": 2767977,
      "additionalInfo": ""
   },
```

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

If an error occurs the following is returned on a *get*. Note *additionalInfo* is populated.

```
{
   "data": {
      "statusCode": 3,
      "statusDescription": "Error",
      "hostFilename": "get",
      "hostFileSize": 178,
      "additionalInfo": "Error locking file transfer service"
   },
   "retCode": 0,
   "retMsg": "Success"
}
```

Finally, if a timeout occurs, which indicates neither a complete nor error status has been received from the managed device within seven minutes, the following is returned on a *get*:

```
{
   "data": {
      "statusCode": 4,
      "statusDescription": "Timeout",
      "hostFilename": "get",
      "hostFileSize": 178,
      "additionalInfo": ""
   },
   "retCode": 0,
   "retMsg": "Success"
}
```

NOTE: Once an upload finishes (complete, error, or timeout), subsequent gets will continue to return the final status (until another upload is initiated or the card has been rebooted).

NOTE: The Complete status indicates the firmware was uploaded to the managed device. The managed device will reset itself at this point.

Table 2.53 *get* Return Codes

Code	Comments
0	Success
1002	Session expired. Provided token is invalid.
1006	Invalid credential combination. Password is incorrect.
3000	Invalid path.

Connect with Vertiv on Social Media


https://www.facebook.com/vertiv/


https://www.instagram.com/vertiv/


https://www.linkedin.com/company/vertiv/


https://www.twitter.com/Vertiv/

Vertiv.com | Vertiv Headquarters, 1050 Dearborn Drive, Columbus, OH, 43085, USA

SL-70969_REVA_09-22