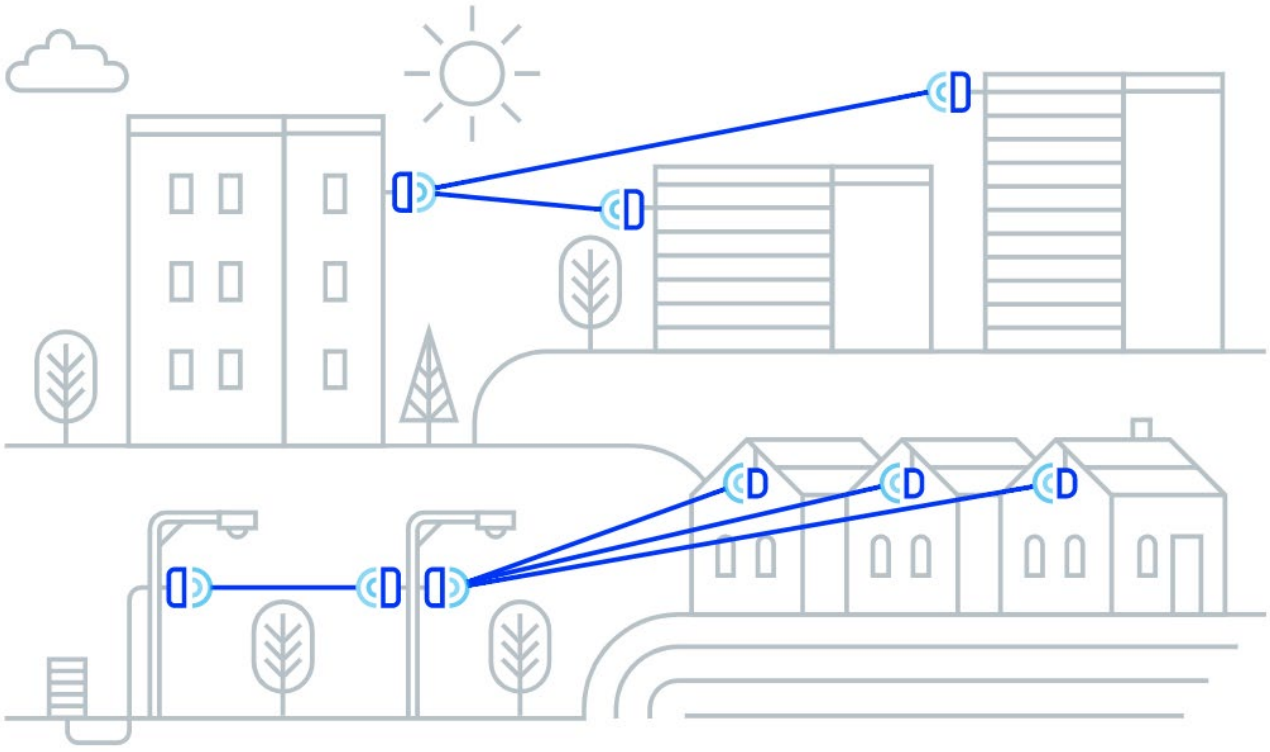

Altowav AltoPlex REST API (Version 002)

Usage Guide

November 7, 2025

Altowav would like to thank all of our staff for their efforts and expertise in development and implementation.

Table of Contents

Version & Date	Revision	Section*
V4.1.0 11-7-25	New endpoints added: • admin/topology_scan • admin/topology_scan_status	topology_scan, sect 5.8 topology_scan_status, sect 5.9
V3.9.1 8-11-25	The optional global parameter `kb_name=KB-XX-XX-XX` has been added to all endpoints. The **kb_name** response data has been added to all endpoints that did not already have it, except for the **device/ll_discovery** endpoint. The cloud_state response data has been added to the **device/device_status** endpoint. **best** and **latest** added to the response description for the **device/gps_status** endpoint.	kb_name, sect 2.5.1 Device status, sect 3.2 GPS Status, sect 3.4
V3.6.0 5-5-25	New endpoints added: • device/mac_filter_status. • device/orientation. • device/resource_utilization	MAC Filter Status, sect 3.9 Orientation, sect 3.13 Resource utilization, sect 3.15
V3.3.1 2-5-25	Updated software upgrade section. Clarified the maximum number of intermediate certificates allowed for client and server certificates.	Software Upgrade, sect 4.5 Install a Client CA certificate, sect 7.2 Install a Client CA certificate, sect 7.2
V3.2.0 12-6-24	New endpoints added for certification authentication: • security/install_client_ca_certificate • security/get_csr • security/install_certificate • security/Altowav Auto-configuration of distribution nodes added: • admin/dn_link_add • admin/dn_link_delete Added new text and key/value pair output formats to **device/ll_discovery**. New response fields bpdufilter and num_rx_bpdu_filtered in device/stp_status.	Install a Client CA certificate, sect 7.2 Download a certificate signing request from the device, sect 7.3 Install a Client CA certificate, sect 7.2 Test client and server authentication, sect 7.5 Add a DN auto-configuration link, sect 5.5 Delete a pending DN auto-configuration link, sect 5.6 Link Layer Discovery, sect 3.7 STP status, sect. 3.13

* Section numbers may shift as new sections are added/deleted. Links to sections remain accurate.

1 Introduction

This document describes the Altowav REST API for Altowav Gen3 RF devices using 802.11ay-based technology. The REST API is a programmatic interface available to manage an Altowav device.

The Altowav embedded web server operates in a secure mode (HTTPS over port 443).

2 Overview

2.1 URL format

The REST API is accessed via a URL that includes the following case-sensitive elements:

- rest -- the word **rest**.
- Version -- currently **v002**.
- Management category -- device, configuration, admin, performance, security.
- Command or item.
- URL query parameters after a **?** – no spaces. Multiple query parameters are separated with the **;** or the **&** character.

Examples of URLs that access the REST API:

- https://<server_name>/rest/v002/device/node_identity
- https://<server_name>/rest/v002/device/syslog?severity=critical
- https://<server_name>/rest/v002/configuration/factory_defaults
- https://<server_name>/rest/v002/admin/reboot
- https://<server_name>/rest/v002/performance/stats
- https://<server_name>/rest/v002/security/password
- https://<server_name>/rest/v002/device/bridge_fdb?interface=kb002;mac=70:88
- https://<server_name>/rest/v002/device/bridge_fdb?interface=kb002&mac=70:88

Notes about HTTPS and <server_name>: The embedded Web server identity is certified via an X509 certificate. By default, the certificate is self-signed, however, you can install a custom certificate signed by a Certificate Authority (CA). See 7.4 Install a server CA certificate.

The Web server identity (i.e. server_name) is of the form "KB-xx-xx-xx" where "x" is hexadecimal digit. Use this server_name to access the REST API. For example:

```
https://KB-36-8A-74.local/rest/v002/device/node_identity
```

This server_name is resolved via mDNS to the IP address of the node. Since the IP address of the node is not part of the certified node identity, attempting to access the REST API via IP address instead of the server_name results in an error like this:

```
curl: (51) SSL: certificate subject name (KB-36-8A-74)
   does not match target host name '192.168.2.17'
```

2.2 HTTP Methods

The HTTP methods allowed by the API will depend on the action requested. The following methods are supported by some of the APIs:

- **GET** - read only data.
- **PATCH** - update values of fields (configuration changes).
- **POST** - create new actions (software update, reboot, etc.)

When an HTTP method is used that is not supported, the HTTP response will have status code of 405 – "Method not allowed".

2.3 Errors

Errors fall into these categories:

- **malformed URL**: A malformed URL results in an HTTP status code in the **400 range** returned by the server.
- **disallowed method**: A disallowed HTTP method results in a HTTP status code of **405** in the response.
- **data error**: A data error results in a **400** level HTTP status in the response. This is described in greater detail below in the description of API calls that send data.
- **cgi lock (556)**: The device is busy processing a lengthy call. Re-issue the call a short time later (100s of milliseconds).
- **server not available (503)**: A daemon process on the device is restarting. Re-issue the call later, waiting at least the time specified in the "Retry-After" value in the 503 error header. For most cases, a 1 second wait is adequate.

Errors in the 400 range have a short description of the error in the header and the response body. In some cases, the body also includes a JSON structure with further information.

503 errors include a "Retry-After" line in the header. The calling application can read this line, and retry after the specified amount of time has passed. Sample response header:

```
< HTTP/1.1 503 Service Not Available
< Retry-After: 2 seconds
< Content-Length: 0
< Cache-Control: public, must-revalidate, proxy-revalidate
< Date: Mon, 04 Dec 2023 21:29:42 GMT
< Server: lighttpd/1.4.54
```

2.4 REST API Summary

The REST API operations are organized in the following categories:

- Device management
- Configuration management
- Administrative control
- Performance management

- Security management

The API URLs are listed here with details in other sections of the document. Links in headings and URLs provide quick access to the details. The "..." at the start of the URL indicates the "https://<server_name>" part of the URL.

2.4.1 Device Management

The API calls in this section do not require authorization.

```
../rest/v002/device/bridge_fdb
../rest/v002/device/device_status
../rest/v002/device/eventlog
../rest/v002/device/gps_status
../rest/v002/device/ignition_status
../rest/v002/device/lan_status
../rest/v002/device/link_quality
../rest/v002/device/ll_discovery
../rest/v002/device/mac_filter_status
../rest/v002/device/management_address
../rest/v002/device/mcs_hist
../rest/v002/device/node_identity
../rest/v002/device/orientation
../rest/v002/device/radio_status
../rest/v002/device/resource_utilization
../rest/v002/device/stp_status
../rest/v002/device/syslog
../rest/v002/device/temperature_stats
../rest/v002/device/topology
```

2.4.2 Configuration Management

The API calls in this section require authorization.

```
../rest/v002/configuration/data (URL used for Getting and Setting config.)
../rest/v002/configuration/factory_defaults
../rest/v002/configuration/software_upgrade
```

2.4.3 Administration

The API calls in this section require authorization.

```
../rest/v002/admin/diagdump
../rest/v002/admin/dn_link_add
../rest/v002/admin/dn_link_delete
../rest/v002/admin/link_bump
../rest/v002/admin/link_scan
../rest/v002/admin/link_scan_status
../rest/v002/admin/locate_device
../rest/v002/admin/power_cycle
../rest/v002/admin/reboot
../rest/v002/admin/topology_scan
../rest/v002/admin/topology_scan_status
```

2.4.4 Performance Management

The API call in this section does not require authorization.

```
../rest/v002/performance/stats
```

2.4.5 Security Management

The API call in this section requires authorization.

```
../rest/v002/security/password
../rest/v002/security/install_client_ca_certificate
../rest/v002/security/get_csr
../rest/v002/security/install_certificate
../rest/v002/security/altowav
```

2.5 Global query parameters

The Altowav REST API includes global parameters that are used by endpoints to validate the identity of the radio and format the response. Not all endpoints accept all global parameters.

2.5.1 kb_name

The `kb_name` parameter, added with software release 3.7.1, is used to validate the identity of the radio. Due to the possibility of IP address reuse in DHCP environments, and the fact that all radios beginning with release 3.6.0 have the same factory default static IP address, it is important to verify the identity of the radio that you are interacting with before performing any PATCH or POST operations.

To do this, include the `kb_name=`*`hostname`* parameter, where *hostname* is the name of the radio in the format KB-XX-XX-XX. For example, to issue a reboot command to the IP address of 10.0.0.2 and verify that the radio being rebooted has the hostname KB-C7-00-01, use the following URL:

```
https://10.0.0.2/rest/v002/admin/reboot?kb_name=KB-C7-00-01
```

If `kb_name` does not match the hostname of the radio, the command will fail and the radio will return http status code 421 (misdirected request).

2.5.2 Output

By default, Altowav REST APIs provide their response in JSON format. For many APIs, you can change the output to text format by using the optional global parameter `output=text`. The ll_discovery endpoint also allows output in name/value pairs by using the optional global parameter `output=keyvalue`.

3 Device Management

This category involves operational status, network topology discovery, diagnostics, and fault isolation.

3.1 Bridge Forwarding Database

Displays the bridge forwarding database.

Optionally, limit the display to a specific Ethernet MAC address or interface. Only the dynamic entries will be listed.

For interface names, the list of valid ports is: eth[n], where n is the port number, and kbrnn where r is a radio number [0-3] and nn identifies a connection for the radio [00-19]. **Tip:** Use /device/lan_status to see available Ethernet ports and use /device/radio_status to look up the kb interface name that corresponds to a specific link or remote unit.

The MAC address must be specified as groups of two hexadecimal digits separated by ":". Filtering for the MAC address will match each instance of the full string listed. For example, “mac=88” will find each device with 88 anywhere in the MAC address and “mac=70:88” will find each device with 70:88 in the MAC address.

3.1.1 Request

bridge_fdb Request

Name	Description
URL	`https://<server_name>/rest/v002/device/bridge_fdb`

URL parameters	`mac=xx[:xx]...`
	`interface=interface_name`
	`kb_name=KB-XX-XX-XX` (see Global query parameters)
	`output=[json|text]` (see Global query parameters)

HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Examples: https://<server_name>/rest/v002/device/bridge_fdb https://<server_name>/rest/v002/device/bridge_fdb?mac=28:92:4a:c8:79:5e https://<server_name>/rest/v002/device/bridge_fdb?mac=70:88:6b https://<server_name>/rest/v002/device/bridge_fdb?interface=kb002 https://<server_name>/rest/v002/device/bridge_fdb?interface=kb002;mac=70:88	

3.1.2 Response

bridge_fdb success Response

Name		Description
Content type	Application/JSON.	
mac	mac_addr	MAC address of this radio.
interface	*name* (for example, `eth1`)	Altowav interface name.
vlan	Lists VLAN when VLANs are enabled. When VLANs are disabled `"0"` is returned. See samples below. Config keyword: vlan.dot1q.mode	
Kb_name	Hostname of the radio (KB-XX-XX-XX).	

bridge_fdb not found Response

Name	Description
Content type	Application/JSON.
error	none found

bridge_fdb error Response

Name	Description
Content type	Application/JSON.
error	bad format mac address 28:92:4a:c8:79:5Z

3.1.3 Sample Response:

This sample shows a JSON version of a ***bridge_fdb success Response*** with VLANs enabled.

```
{
  "BridgeFDB" :
  [
    {
      "interface" : "eth0",
      "mac" : "be:a6:e1:f9:4e:1a",
      "vlan" : "200"
    },
    {
      "interface" : "eth0",
      "mac" : "00:0a:cd:38:ef:ae",
      "vlan" : "200"
    }
  ]
  "kb_name": "KB-C7-00-01
}
```

This sample shows a JSON version of a ***bridge_fdb success Response*** with VLANs disabled.

```
{
  "BridgeFDB" :
  [
    {
      "interface" : "eth0",
      "mac" : "72:88:6b:c6:02:9b",
      "vlan" : "0"
```

```
    },
    {
      "interface" : "eth0",
      "mac" : "80:2d:bf:fa:fe:0c",
      "vlan" : "0"
    },
    {
      "interface" : "eth0",
      "mac" : "00:0a:cd:38:ef:ae",
      "vlan" : "0"
    }
  ]
  "kb_name": "KB-C7-00-01
}
```

3.2 Device Status

Gets the current status of a device. The response includes information about DNS servers, temperature, LED, current time settings and uptime.

3.2.1 Request

device_status Request

Name	Description
URL	`https://<server_name>/rest/v002/device/device_status`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters) **Note:** The `radio_temp=true` parameter is deprecated. Use temperature_stats to poll for temperature.
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Examples: https://<server_name>/rest/v002/device/device_status	

3.2.2 Response

device_status Response

Name	Description
Content type	Application/JSON.
DNS_servers	Comma-separated list of IP addresses, as configured. A list of zero is possible.
cloud_state	The state of the radio's connection to the AltoCommand server. The value is one of: • **Not configured** — The AltoCommand server has not been configured on the radio. • **Disconnected** — The AltoCommand server has been configured on the radio but is not connected. Possible issues include an incorrect URL for the server, network access issues, etc. • **Pending** — The AltoCommand server has been configured and successfully accessed, and the radio is waiting for the server to accept the connection.

	• **Connected** — The radio is successfully connected to the configured AltoCommand server.
description	Description as configured.
kb_name	The hostname of the radio (KB-XX-XX-XX).
led_enable	(`enable` | `disable`) as configured.
location	Location as configured.
time	`2019.07.25-15:29:06` Date followed by time.
uptime_secs	Integer number of seconds since boot.

3.2.3 Sample Response:

This sample shows a JSON version of a ***device_status*** response. **Note,** to poll for a unit's baseband and RF tile temperatures, use temperature_stats.

```
{
  "DNS_servers" : "",
  "description" : "description of this device",
  "cloud_state" : "Connected",
  "kb_name" : "KB-XX-XX-XX"
  "led_enable" : "enable",
  "location" : "location of this device",
  "time" : "2022.01.01-00:22:38",
  "uptime_secs" : 1365
}
```

3.3 EventLog

Returns the event log from the device. The event log is a subset of the syslog, consisting of key events such as Ethernet interface state changes, airlink connections / disconnection, and configuration events.

3.3.1 Request

eventlog Request

Name	Description
URL	`https://<server_name>/rest/v002/device/eventlog`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Example: https://<server_name>/rest/v002/device/eventlog	

3.3.2 Response

eventlog Response

Name	Description
Content type	text/plain

3.4 GPS Status

Returns location information from the GPS device in the K60DN. Some configuration and time synchronization information is also returned. If the 503 error – server not available is issued in response, simply try the call again, waiting at least the time specified in the 503 error header. Typically one second is adequate.

3.4.1 Request

gps_status Request

Name	Description
URL	`https://<server_name>/rest/v002/device/gps_status`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[json|text]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Example: `https://<server_name>/rest/v002/device/gps_status`	

3.4.2 Response

gps_status Response

Name	Description
Content type	Application/JSON.
kb_name	The hostname of the radio (KB-XX-XX-XX).
best The most accurate set of GPS coordinates reported by the GPS module since the radio booted.	
accuracy	A measurement of three-dimensional accuracy for the most accurate GPS data. A number below 50 is considered good, above 50 is considered inaccurate.
altitude	The most accurate measurement of altitude in meters above sea level.
latitude	The most accurate measurement of latitude in decimal degrees, with north positive and south negative.
longitude	The most accurate measurement of longitude in decimal degrees, with east positive and west negative.
timestamp	The time that the GPS data was measured, in seconds since UNIX epoch (midnight, January 1st, 1970). Useful to compare the difference in time between the best and latest measurements.

latest The most recent set of coordinates reported by the GPS module.	
accuracy	A measurement of three-dimensional accuracy for the most recent GPS data. A number below 50 is considered good, above 50 is considered inaccurate.
altitude	The most recent measurement of altitude in meters above sea level.
latitude	The most recent measurement of latitude in decimal degrees, with north positive and south negative.
longitude	The most recent measurement of longitude in decimal degrees, with east positive and west negative.
timestamp	The time that the GPS data was measured, in seconds since UNIX epoch (midnight, January 1st, 1970). Useful to compare the difference in time between the best and latest measurements.
location GPS coordinates that were cached when the radio first synchronized.	
accuracy	A measurement of three-dimensional accuracy for the cached GPS . A number below 50 is considered good, above 50 is considered inaccurate.
altitude	The altitude in meters above sea level.
latitude	The latitude in decimal degrees, with north positive and south negative.
longitude	The longitude in decimal degrees, with east positive and west negative.
synchronized	(`true` | `false`) indicates whether GPS has time synchronization.
wireless_device_gps_sync	(`enable` | `disable`) as configured for GPS timing synchronization. Disable means that internal timing is used. Config keyword: wireless.device.gps_sync

3.4.3 Sample Response:

This sample shows a JSON version of a ***gps_status*** response including the K60DN's location, GPS sync and wireless sync information.


```
{
  "kb_name": "KB-C0-00-01",
  "best": {
    "accuracy": 17.1,
    "altitude": 246.26,
    "latitude": 44.861098333,
    "longitude": -93.360571667,
    "timestamp": 1746637241
  },
  "kb_name": "KB-C6-04-12",
  "latest": {
    "accuracy": 20.9,
    "altitude": 248.72,
    "latitude": 44.861091667,
    "longitude": -93.360531667,
    "timestamp": 1746640729
  },
```

```
  "location": {
    "accuracy": 24.7,
    "altitude": 279.93,
    "latitude": 44.861113333,
    "longitude": -93.36056
  },
  "synchronized": true,
  "wireless_device_gps_sync": "enable"
}
```

3.5 Ignition Status

Returns wireless link ignition status information. The `igCandidates` section shows all links that need to be connected. The `igSchedState` section shows ignition status details for each radio on the device.

3.5.1 Request

ignition_status Request

Name	Description
URL	`https://<server_name>/rest/v002/device/ignition_status`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[json|text]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Example: `https://<server_name>/rest/v002/device/ignition_status`	

3.5.2 Response

ignition_status Response

Name	Description
Content type	Application/JSON.
igCandidates – List of all links that need to be ignited, described with the initiator node name and the link name. Node names and link names specify the local radio with the "`a`" prefix and remote radio with the "`z`" prefix.	
initiatorNodeName	Name of the intitiator.
linkName	Name of the link to be initiated.
igParams	Internal timing parameters. Typically, not useful to end user. Intended for developer use. Parameters include: bfTimeoutSec, enable, linkUpDampenInterval, linkUpInterval.
igSchedState -- Ignition schedule status per radio, including the number of links remaining, the radio's role in the link, the schedule period, time and type.	
haveTorch	(`true` | `false`) indicates whether the ignition app has the synchronization torch, enabling the radio to perform link ignition.
linksRemaining	Quantity of links which need to be ignited on this radio.
radioRole	Description of the radio's link role.

	NONE – no links defined for this radio. CN_INITIATOR – only CN links defined for this radio. DN_INITIATOR – this radio is an initiator in a DN link and may also have CN links. DN_RESPONDER – this radio is a responder in a DN link and may also have CN links.
schedulePeriod	Description of the ignition schedule period. IDLE – no link ignition happens during this time. INITIATE_CN – CN links may be initiated during this time. INITIATE_DN – DN links may be initiated during this time.
scheduleTime	Number of seconds passed in the current schedule period. Only applicable when scheduleType is TIMED. See the following field description.
scheduleType	Description of the schedule type. NONE – No links need to be ignited on this radio. INDEPENDENT – This radio can ignite links without any coordination with other DNs because there are no DN links on the channel. TIMED – This radio follows the timed ignition schedule because there are DN link(s) on the channel. SYNCHRONIZED – The DN link is up so this radio coordinates CN link ignition scheduling with the connected peer DN.
kb_name	The hostname of the radio (KB-XX-XX-XX).
lastIgCandidates	List of links most recently attempted.

3.5.3 Sample Response:

This sample shows an ***ignition_status*** response for two radios with links defined, but not connected.


```
{
  "igCandidates": [
    {
      "initiatorNodeName": "aKB-C6-04-0E",
      "linkName": "link-aKB-C6-04-0E-zcn-KB-C6-04-29"
    },
    {
      "initiatorNodeName": "aKB-C6-04-0E",
      "linkName": "link-aKB-C6-04-0E-zcn-KB-C6-04-28"
    },
    {
      "initiatorNodeName": "aKB-C6-04-0E",
      "linkName": "link-aKB-C6-04-0E-zcn-KB-C6-04-2A"
    },
    {
      "initiatorNodeName": "aKB-C6-04-0E",
      "linkName": "link-aKB-C6-04-0E-zcn-KB-C6-04-35"
    },
    {
      "initiatorNodeName": "aKB-C6-04-0E",
```

```
      "linkName": "link-aKB-C6-04-0E-zdn-04:ce:14:fe:aa:09"
    }
  ],
  "igParams": {
    "bfTimeoutSec": 16,
    "enable": true,
    "linkUpDampenInterval": 15,
    "linkUpInterval": 1
  },
  "igSchedState": {
    "04:ce:14:fe:b2:cf": {
      "haveTorch": false,
      "linksRemaining": 2,
      "radioRole": "DN_RESPONDER",
      "schedulePeriod": "IDLE",
      "scheduleTime": 95,
      "scheduleType": "TIMED"
    },
    "04:ce:14:fe:b2:d9": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    },
    "04:ce:14:fe:b4:4c": {
      "haveTorch": false,
      "linksRemaining": 3,
      "radioRole": "CN_INITIATOR",
      "schedulePeriod": "INITIATE_CN",
      "scheduleTime": 0,
      "scheduleType": "INDEPENDENT"
    },
    "04:ce:14:fe:b4:56": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    }
  },
  "kb_name": "KB-C7-00-01",
  "lastIgCandidates": [
    {
      "initiatorNodeName": "aKB-C6-04-0E",
      "linkName": "link-aKB-C6-04-0E-zcn-KB-C6-04-46"
    }
  ]
}
```


This sample shows the response when the radio with only CN links (`04:ce:14:fe:b4:4c` from the previous sample), has completed ignition.


```
{
  "igCandidates": [
    {
      "initiatorNodeName": "aKB-C6-04-0E",
      "linkName": "link-aKB-C6-04-0E-zcn-KB-C6-04-35"
    }
  ],
```

```
  "igParams": {
    "bfTimeoutSec": 16,
    "enable": true,
    "linkUpDampenInterval": 15,
    "linkUpInterval": 1
  },
  "igSchedState": {
    "04:ce:14:fe:b2:cf": {
      "haveTorch": false,
      "linksRemaining": 1,
      "radioRole": "DN_RESPONDER",
      "schedulePeriod": "INITIATE_CN",
      "scheduleTime": 64,
      "scheduleType": "TIMED"
    },
    "04:ce:14:fe:b2:d9": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    },
    "04:ce:14:fe:b4:4c": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "CN_INITIATOR",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    },
    "04:ce:14:fe:b4:56": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    }
  },
  "kb_name": "KB-C7-00-01",
  "lastIgCandidates": [
    {
      "initiatorNodeName": "aKB-C6-04-0E",
      "linkName": "link-aKB-C6-04-0E-zcn-KB-C6-04-35"
    }
  ]
}
```


A response showing both radios completed ignition.


```
{
  "igCandidates": [],
  "igParams": {
    "bfTimeoutSec": 16,
    "enable": true,
    "linkUpDampenInterval": 15,
    "linkUpInterval": 1
  },
  "igSchedState": {
    "04:ce:14:fe:b2:cf": {
      "haveTorch": false,
      "linksRemaining": 0,
```

```
      "radioRole": "DN_RESPONDER",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    },
    "04:ce:14:fe:b2:d9": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    },
    "04:ce:14:fe:b4:4c": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "CN_INITIATOR",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    },
    "04:ce:14:fe:b4:56": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    }
```


In this sample the synchronization torch has not been passed, (the radio is not allowed to perform ignition, yet).


```
{
  "igCandidates": [
    {
      "initiatorNodeName": "aKB-C6-04-0E",
      "linkName": "link-aKB-C6-04-0E-zcn-KB-C6-04-37"
    }
  ],
  "igParams": {
    "bfTimeoutSec": 16,
    "enable": true,
    "linkUpDampenInterval": 15,
    "linkUpInterval": 1
  },
  "igSchedState": {
    "04:ce:14:fe:b2:cf": {
      "haveTorch": false,
      "linksRemaining": 1,
      "radioRole": "DN_RESPONDER",
      "schedulePeriod": "INITIATE_CN",
      "scheduleTime": 0,
      "scheduleType": "SYNCHRONIZED"
    },
    "04:ce:14:fe:b2:d9": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
```

```
    },
    "04:ce:14:fe:b4:4c": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    },
    "04:ce:14:fe:b4:56": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    }
  },
  "kb_name": "KB-C7-00-01",
  "lastIgCandidates": []
}
```

In this sample the synchronization torch has been passed, (the radio can perform ignition).


```
{
  "igCandidates": [
    {
      "initiatorNodeName": "aKB-C6-04-0E",
      "linkName": "link-aKB-C6-04-0E-zcn-KB-C6-04-37"
    }
  ],
  "igParams": {
    "bfTimeoutSec": 16,
    "enable": true,
    "linkUpDampenInterval": 15,
    "linkUpInterval": 1
  },
  "igSchedState": {
    "04:ce:14:fe:b2:cf": {
      "haveTorch": true,
      "linksRemaining": 1,
      "radioRole": "DN_RESPONDER",
      "schedulePeriod": "INITIATE_CN",
      "scheduleTime": 0,
      "scheduleType": "SYNCHRONIZED"
    },
    "04:ce:14:fe:b2:d9": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    },
    "04:ce:14:fe:b4:4c": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    },
    "04:ce:14:fe:b4:56": {
      "haveTorch": false,
      "linksRemaining": 0,
      "radioRole": "NONE",
      "schedulePeriod": "IDLE",
      "scheduleTime": 0,
      "scheduleType": "NONE"
    }
  },
  "kb_name": "KB-C7-00-01",
  "lastIgCandidates": [
    {
      "initiatorNodeName": "aKB-C6-04-0E",
      "linkName": "link-aKB-C6-04-0E-zcn-KB-C6-04-37"
    }
  ]
}
```

3.6 LAN Status

Gets the current status of all ethernet ports.

3.6.1 Request

lan_status Request

Name	Description
URL	`https://<server_name>/rest/v002/device/lan_status`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.

3.6.2 Response

lan_status Response

Name	Description
Content type	Application/JSON.
kb_name	The hostname of the radio (KB-XX-XX-XX).
Ethernet_Interfaces	
name	ethN – where **N** is the ethernet interface number.
interface number	N - where **N** is the interface number (K60DN: 0-5, K60CN1: 1-2).
interface type	`1GbE` Specifies the type of port.
admin state	(`enabled` | `disabled`) Current admin state of the interface, as configured. Config keyword: ethernet.ethN.admin -- where **N** is the interface number.
run state	(`up` | `down` | `blocked`) Indicates the current run state of the interface.
speed	(`0, 100, 1000, 2500`) Shows port speed.
duplex	`(full | half)` Indicates whether the port is full or half duplex.
PoE	`(none | output)` Shows whether PoE output is enabled on the port.
PoE spec	`PoE | PoE+` Specifies the PoE specification for the port.
PoE admin	`(enabled | disabled)` The current state of PoE admin as configured. Config keyword: ethernet.ethN.poe -- where **N** is the interface number.

3.7 Link Layer Discovery

Returns information about a device's Link Layer neighbors from Link Layer Discovery Protocol (LLDP). The response includes both Altowav and other equipment.

Note: Link layer discovery (***device/ll_discovery)*** is currently available only for AltoPlex devices D621, C420, C410, P621 and P421. However, the response will show other devices that support LLDP (link layer discovery protocol).

If this unit is connected via Ethernet to a managed switch running LLDP, information about the connected switch port is also available.

Alternatively, if this unit is connected via Ethernet to certain transparent bridge devices such as the Altowav Model: AX-PSW-OD-4AT-4C25 switch, then adjacent Altowav devices will be shown as "`Kwikbit device`". If LLDP packets can traverse the switch and pass directly between locally connected Altowav devices, the ***ll_discovery*** response will show the device.

If the device is connected to a switch that does not support forwarding of LLDP data units, nothing will be shown.

The **show=chassis** option returns the local information advertised by this node via LLDP to its link neighbors. Otherwise, ***ll_discovery*** returns information about link neighbors .

ll_discovery Request

Name	Description
URL	`https://<server_name>/rest/v002/device/ll_discovery`
URL parameters	`show=chassis (Optional.) Shows the LLDP information for the local node.` `kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|keyvalue|json]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Application/JSON.
Authentication	Not required.
Example request: https://<server_name>/rest/v002/device/ll_discovery https://<server_name>/rest/v002/device/ll_discovery?output= keyvalue https://<server_name>/rest/v002/device/ll_discovery?show=chassis	

3.7.1 Responses

The response includes either chassis (local node) information or neighboring node(s) information.

ll_discovery Response

Name	Description
Content type	Application/JSON, text/plain, text/key-value

lldp	Indicates discovery via LLDP.
interface includes an array of data for interfaces discovered. For example an interface named "**eth1**" includes the following information.	
via	How discovery occurred. Ex, `LLDP`.
rid	Remote ID.
age	Length of time, since it has been observed.
chassis data returned includes the following attributes describing the device.	
Device name	Discovered device or local device (when **`show=chassis`** is used.) For example, KB-xx-xx-xx the host name of a discovered device.
id data describing the type and value of the device.	
type	local | remote.
value	Device model.
descr	Describes the make of the device.
mgmt-ip	IP address of the remote node may be ipv4 or ipv6 or both.
port data describing the port(s) on the discovered device.	
id data describing the type and value of the port discovered.	
type	Type of data that describes the port. `mac`
value	id value. For example, MAC address for the port.
desc	Description of port discovered.
ttl	Time to live value.

3.7.2 Sample Responses:

This sample shows the output in JSON format:


```
{
  "lldp": {
    "interface": [
      {
        "eth1": {
          "via": "LLDP",
          "rid": "5",
          "age": "0 day, 00:17:57",
          "chassis": {
            "KB-C0-00-00": {
              "id": {
                "type": "local",
                "value": "D621"
              }
            },
          },
          "port": {
            "id": {
              "type": "mac",
              "value": "70:88:6b:c0:00:00"
```

```
            },
            "descr": "eth1",
            "ttl": "120"
          }
        }
      },
      {
        "eth1": {
          "via": "LLDP",
          "rid": "6",
          "age": "0 day, 00:08:17",
          "chassis": {
            "KB-C0-00-01": {
              "id": {
                "type": "mac",
                "value": "70:88:6b:c0:00:01"
              },
              "descr": "Altowav device",
              "mgmt-ip": [
                "192.168.0.143",
                "fe80::7288:6bff:fec0:77"
              ]
            }
          },
          "port": {
            "id": {
              "type": "mac",
              "value": "70:88:6b:c0:00:01
            },
            "descr": "eth1",
            "ttl": "120"
          }
        }
      }
    ]
  }
}
```


This sample shows the same response in text output:

```
$ curl https://<server_name>/rest/v002/device/ll_discovery?output=text
------------------------------------------------------------------------------
LLDP neighbors:
------------------------------------------------------------------------------
Interface:    eth1, via: LLDP, RID: 1, Time: 0 day, 00:06:15
  Chassis:
    ChassisID:    mac 70:88:6b:c0:00:00
    SysName:      KB-C0-00-00
    SysDescr:     Altowav device
    MgmtIP:       192.168.0.2
    MgmtIP:       fe80::7288:6bff:fec0:00
  Port:
    PortID:       mac 70:88:6b:c0:00:00
    PortDescr:    eth1
    TTL:          120

------------------------------------------------------------------------------
Interface:    eth1, via: LLDP, RID: 2, Time: 0 day, 00:06:05
  Chassis:
    ChassisID:    mac 70:88:6b:c0:00:01
    SysName:      KB-C0-00-01
    SysDescr:     Altowav device
    MgmtIP:       192.168.0.3
```

```
    MgmtIP:       fe80::7288:6bff:fec0:01
  Port:
    PortID:       mac 70:88:6b:c0:00:01
    PortDescr:    eth1
    TTL:          120
------------------------------------------------------------------------------
```

This sample shows the response in key=value pairs:

```
curl https://<server_name>/rest/v002/device/ll_discovery?output=keyvalue

lldp.eth1.via=LLDP
lldp.eth1.rid=5
lldp.eth1.age=0 day, 00:17:12
lldp.eth1.chassis.local=D621
lldp.eth1.port.mac=70:88:6b:c0:00:00
lldp.eth1.port.descr=eth1
lldp.eth1.port.ttl=120

lldp.eth1.via=LLDP
lldp.eth1.rid=6
lldp.eth1.age=0 day, 00:07:32
lldp.eth1.chassis.mac=70:88:6b:c0:00:77
lldp.eth1.chassis.name=KB-C0-00-76
lldp.eth1.chassis.descr=Altowav device
lldp.eth1.chassis.mgmt-ip=192.168.0.143
lldp.eth1.chassis.mgmt-ip=fe80::7288:6bff:fec0:77
lldp.eth1.port.mac=70:88:6b:c0:00:77
lldp.eth1.port.descr=eth1
lldp.eth1.port.ttl=120
```

This sample shows a text version of a ***ll_discovery*** response where the local node information is requested. Ex. `https://<server_name>/rest/v002/device/ll_discovery?output=text;show=chassis`

```
local-chassis.chassis.local=nomad
local-chassis.chassis.name=KB-C6-04-F0
local-chassis.chassis.descr=Kwikbit device
local-chassis.chassis.mgmt-ip=192.168.0.138
local-chassis.chassis.mgmt-ip=fe80::7288:6bff:fec6:4f1
```

3.8 Link Quality

Returns data related to link quality for the specified local-to-remote wireless link. The response is data from the radio at each end of the link, including MCS levels and measurements of RSSI and SNR.

The Link Quality API is only available on the K60DN. For the equivalent information from a K60CN1, Use Radio Status.

The response is arranged as an array of radios with an array of link data for each radio nested inside. See sample response below.

Note: Link Quality does not return data for all links, but returns only data for the link between the specified local and remote radios. Use Radio Status to return data for all links.

3.8.1 Request

Specifying the local and remote radios for the link is required.

link_quality Request

Name	Description
URL	`https://<server_name>/rest/v002/device/link_quality`
URL parameters	`local_mac=<mac address>` (Required.) `remote_mac=<mac address>` (Required.) `kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|keyvalue|json]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Example request: (Sample is shown with a line break, but single line entry is the norm.) https://<server_name>/rest/v002/device/link_quality?local_mac=04:ce:14:fe:a5:00; remote_mac=04:ce:14:fc:b5:6b	

3.8.2 Responses

When a link is down, no data is returned from the remote device. See sample responses following the tables.

link_quality Response When Link is Active

Name	Description
Content type	Application/JSON or text.
kb_name	The hostname of the radio (KB-XX-XX-XX) at the local end of the link.

radio data returned includes the following attributes.	
description	Radio description, as configured. Config keyword: wireless.wlanN.description – where **N** is the radio number.
mac	MAC address of radio.
name	Radio interface name (wlan[0-4]).
links data returned includes the following attributes.	
is_alive	(`true` | `false`) Indicates whether the link is up.
remote_mac	MAC address of the remote end of the link.
remote_ipv6_addr	Internally assigned IPv6 link local address, used for Altowav radio management.
remote_ipv4_addr	IP address of the remote node.
remote_kb_name	Host name of the remote node.
local_radio and **remote_radio** data is returned in separate subsections for each link, with the following attributes.	
chn.channel	Radio channel 1-4.
currentLinkState	Current link state (0 is down, 1 is up).
phyPeriodic.rxBeamIdx	Currently selected Rx beam index.
phyPeriodic.txBeamIdx	Currenlty selected Tx beam index.
phystatus.srssi	Signal strength measured in dBm.
phystatus.ssnrEst	Signal to noise ratio.
phyPeriodic.rxBeamAzimuth	Decimal degrees off antenna boresight.
phyPeriodic.rxBeamElevation	Decimal degrees from antenna boresight.
process.uptime.seconds	Shows the increasing run time of radio management daemon process. Useful only to confirm successful inter-process-communication mechanism for this API call. If this number increases on each subsequent call, you are getting valid data samples. (This is not the uptime of the device.)
staPkt.linkUpDataDown	A count of the number of times that the link was up, but unable to pass data plane traffic. A non-zero number shows that a problem occurred at some point in the past.
staPkt.mcs	Transmit MCS level (1-12).
staPkt.perE6	Packet error rate per 100,000.
staPkt.txPowerIndex	Transmission power index (0-31).
tsf.syncModeGps	GPS synchronization. (0 is not synchronized, 1 is synchronized.)

tsf.syncModeRf	RF synchronization. (0 is not synchronized, 1 is synchronized.)

3.8.3 Sample Responses:

This sample shows a JSON version of a ***link_quality*** response for an active link.

Note, the link_quality response uses the same underlying data structure as radio_status. However, when link_quality is called, six values are not returned. Values for linkup_attempts, link_name, remote_ipv4_address, remote_kb_name, remote_node_type and link_uptime will be listed as "0" or blank in the JSON response. Use radio_status to obtain data for these values.


```
{
  "kb_name": "KB-C6-04-0D",
  "radio": [
    {
      "description": "as configured wireless.wlanN.description",
      "mac": "04:ce:14:fe:a5:00",
      "name": "wlan0"

      "links": [
        {
          "is_alive": true,
          "remote_mac": "70:88:6b:c6:04:47",
          "remote_ipv6_addr": " fe80:4567:7288:6bff:fec6:447b:de98:20a4",

          "linkup_attempts": 0,
          "link_name": "",
          "remote_ipv4_addr": "",
          "remote_kb_name": "",
          "remote_node_type": "",

          "local_radio": {
            "chn.channel": 1,
            "currentLinkState": 1,
            "phyPeriodic.rxBeamAzimuth": 4.5,
            "phyPeriodic.rxBeamElevation": -7.5,
            "phyPeriodic.rxBeamIdx": 21,
            "phyPeriodic.txBeamAzimuth": 6.75,
            "phyPeriodic.txBeamElevation": -7.5,
            "phyPeriodic.txBeamIdx": 22,
            "phystatus.srssi": -56,
            "phystatus.ssnrEst": 18,
            "process.uptime.seconds": 74707,
            "staPkt.linkUpDataDown": 2560,
            "staPkt.mcs": 9,
            "staPkt.perE6": 0,
            "staPkt.txPowerIndex": 6,
            "tsf.syncModeGps": 1,
            "tsf.syncModeRf": 0
          },

          "remote_radio": {
            "chn.channel": 1,
            "currentLinkState": 1,
            "phyPeriodic.rxBeamAzimuth": 13.5,
            "phyPeriodic.rxBeamElevation": -7.5,
            "phyPeriodic.rxBeamIdx": 25,
            "phyPeriodic.txBeamAzimuth": 15.75,
            "phyPeriodic.txBeamElevation": -7.5,
            "phyPeriodic.txBeamIdx": 26,
            "phystatus.srssi": -59,
```

```
            "phystatus.ssnrEst": 15,
            "process.uptime.seconds": 82985,
            "staPkt.linkUpDataDown": 4370,
            "staPkt.mcs": 9,
            "staPkt.perE6": 0,
            "staPkt.txPowerIndex": 6,
            "tsf.syncModeGps": 0,
            "tsf.syncModeRf": 1
          }
        }
      ],
    }
  ]
  }
```


This sample shows a JSON ***link_quality*** response for the same device with the link down.


```
{
  "kb_name": "KB-C6-04-0D",
  "radio": [
    {
      "description": "as configured wireless.wlanN.description",
      "mac": "04:ce:14:fe:a5:00",
      "name": "wlan0"

      "links": [
        {
          "is_alive": false,
          "remote_mac": "70:88:6b:c6:04:47",
          "remote_ipv6_addr": " fe80:4567:7288:6bff:fec6:447b:de98:20a4",

          "linkup_attempts": 0,
          "link_name": "",
          "remote_ipv4_addr": "",
          "remote_kb_name": "",
          "remote_node_type": "",

          "local_radio": {
            "chn.channel": 1,
            "currentLinkState": 0,
            "process.uptime.seconds": 4326,
            "tsf.syncModeGps": 1,
            "tsf.syncModeRf": 0
          },
          "remote_radio": {}
        }
      ],
    }
  ]
  }
```

3.9 MAC Filter Status

Returns the status of the radio's MAC filtering.

3.9.1 Request

mac_filter_status Request

Name	Description
URL	`https://<server_name>/rest/v002/device/mac_filter_status`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|keyvalue|json]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.

3.9.2 Response

mac_filter_status Response

Name	Description
Content type	Application/JSON.
kb_name	The hostname of the radio (KB-XX-XX-XX).
ports – data returned for each interface in the local bridge that is not down. Each interface has a subsection which begins with the **interface name** and includes the following attributes.	
filter_ethx_destination_mac	The configured destination MAC address, if any.
filter_ethx_unicast_conversion	Whether the port is configured to convert broadcast and multicast data to unicast. Values are **enabled** .and **disabled**.
filter_ethx_source_mac_limit	The configured number of allowed source MAC addresses. Allowed values are **1**-**10** and **unlimited**.
source_mac_allowlist	Lists all source MAC addresses that are allowed to connect to the Ethernet interface. Not used if `filter_ethx_source_mac_limit` is set to **unlimited**. The source MAC address allow list is created automatically based on the first MAC addresses that connect to the Ethernet interface. For example, if `filter_ethx_source_mac_limit` is set to **5**, the allow list will contain the MAC addresses of the first five devices that connect to the interface. No other MAC addresses will be allowed to connect.

	The allow list can be cleared by rebooting the radio or saving a change to its configuration.

3.9.1 Sample Response

```
{
  "kb_name": "KB-C7-00-00 ,
  "ports": {
    "eth1": {
      "filter_eth1_destination_mac": "10.0.0.1",
      "filter_eth1_unicast_conversion": "enable",
      "filter_eth1_source_mac_limit": "2",
      "source_mac_allowlist": [
        "10:20:30:ab:cd:ef",
        "fe:dc"ba:03:02:01"
      ]
    }
  }
}
```

3.10 Management Address

Returns the IP address in use by the management port. The response includes default gateway and net mask information.

3.10.1 Request

management_address Request

Name	Description
URL	`https://<server_name>/rest/v002/device/management_address`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters).
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Example: https://KB-C6-02-A8.local/rest/v002/device/management_address	

3.10.2 Response

management_address Response

Name	Description
Content type	Application/JSON.
assignProtocol	(`dynamic` | `static`) Shows whether the IP assignment is dynamic or static. Config keyword: network.mgmt.proto
address	The IPv4 address currently being used by the radio. • If assignProtocol is `dynamic`, this is the IP address provided by the DHCP server. • If assignProtocol is `static`, this is the configured static IP (config keyword: network.mgmt.ipaddr).
dfltGateway	The IP address of the default gateway currently being used by the radio. • If assignProtocol is `dynamic`, this is the default gateway provided by the DHCP server. • If assignProtocol is `static`, this is the configured default gateway (config keyword: network.mgmt.gateway). If no gateway is configured, this entry will be blank.
netmask	The network mask currently being used by the radio.

	• If assignProtocol is `dynamic`, this is the netmask provided by the DHCP server. • If assignProtocol is `static`, this is the configured netmask (config keyword: network.mgmt.netmask).
kb_name	The hostname of the radio (KB-XX-XX-XX).

3.10.3 Sample Responses:

This sample shows a ***management_address*** response with a dynamic *assignProtocol*.

```
{
  "assignProtocol" : "dynamic",
  "ipv4" :
  {
     "inUse" :
     {
     "address" : "192.168.0.164",
     "dfltGateway" : "192.168.0.1",
     "netmask" : "255.255.0.0"
     }
  }
  "kb_name"="KB-C7-00-01"
}
```

This sample shows a ***management_address*** response with a static *assignProtocol*.

```
{
  "assignProtocol" : "static",
  "ipv4" :
  {
     "inUse" :
     {
     "address" : "192.168.0.164",
     "dfltGateway" : "192.168.0.1",
     "netmask" : "255.255.0.0"
     }
  }
  "kb_name"="KB-C7-00-01"
}
```

3.11 MCS Histogram

Returns histogram of Rx MCS usage -- traffic per MCS level -- for all links on each radio.

The DN response is arranged as an array of the radios with an array of link data for each radio nested inside. See sample response below.

3.11.1 Request

mcs_hist Request

Name	Description
URL	`https://<server_name>/rest/v002/device/mcs_hist`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters).
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Example: https://<server_name>/rest/v002/device/mcs_hist	

3.11.2 Response

mcs_hist Response

Name	Description
Content type	Application/JSON.
kb_name	The hostname of the radio (KB-XX-XX-XX) at the local end of the link.
radio information is returned for each device listed – 4 radios per DN – with the following attributes.	
name	Radio interface name (wlan[0-4]).
mac	The MAC address of the radio.
Data for each of the radio's **links** is returned in an array within each radio's listing.	
remote_mac	MAC address of the remote end of the link.
remote_kb_name	`dn-04:ce:14:fc:b5:6b` -- "dn-" followed by MAC for remote DN or `KB-C6-04-1B` – KB MAC for remote CNs.
RX-MCS counts	Counts are listed for each MCS level 0-15. See sample response.

3.11.1 Sample Response:

This sample shows a JSON version of a ***mcs_hist*** response with three links on wlan0 and no links on wlan1, wlan2 or wlan3.

```
{
  "kb_name": "KB-C6-04-0D",
```

```
"radio": [
  {
    "name": "wlan0",
    "mac": "04:ce:14:fe:a5:00",
    "links": [
      {
        "remote_mac": "04:ce:14:fc:b5:6b",
        "remote_kb_name": "dn-04:ce:14:fc:b5:6b",
        "RX-MCS counts": {
          "0": 0,
          "1": 0,
          "2": 0,
          "3": 0,
          "4": 0,
          "5": 0,
          "6": 0,
          "7": 0,
          "8": 0,
          "9": 25,
          "10": 2,
          "11": 0,
          "12": 0,
          "13": 0,
          "14": 0,
          "15": 0
        }
      },
      {
        "remote_mac": "70:88:6b:c6:04:47",
        "remote_kb_name": "KB-C6-04-47",
        "RX-MCS counts": {
          "0": 0,
          "1": 0,
          "2": 0,
          "3": 0,
          "4": 0,
          "5": 0,
          "6": 0,
          "7": 0,
          "8": 0,
          "9": 30,
          "10": 1,
          "11": 0,
          "12": 0,
          "13": 0,
          "14": 0,
          "15": 0
        }
      },
      {
        "remote_mac": "70:88:6b:c6:04:48",
        "remote_kb_name": "KB-C6-04-48",
        "RX-MCS counts": {
          "0": 0,
          "1": 0,
          "2": 0,
          "3": 0,
          "4": 0,
          "5": 0,
          "6": 0,
          "7": 0,
          "8": 0,
          "9": 52,
```

```
            "10": 30,
            "11": 0,
            "12": 0,
            "13": 0,
            "14": 0,
            "15": 0
          }
        }
      ]
    },
    {
      "name": "wlan1",
      "mac": "04:ce:14:fe:b4:e6",
      "links": [
      ]
    },
    {
      "name": "wlan2",
      "mac": "04:ce:14:fe:aa:92",
      "links": [
      ]
    },
    {
      "name": "wlan3",
      "mac": "04:ce:14:fe:b5:4d",
      "links": [
      ]
    }
  ]
}
```

3.12 Node Identity

Returns identifying information about the node.

3.12.1 Request

node_identity Request

Name	Description
URL	`https://<server_name>/rest/v002/device/node_identity`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters).
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.

3.12.2 Response

node_identity Response

Name	Description
Content type	Application/JSON.
Ethernet MAC	The MAC address of the br0 (bridge) interface. • For all devices other than the K60DN: This is the same as the MAC address on the device label, except the first octet is 72 rather than 70. • For the K60DN: This is the same as the MAC address on the device label.
HW name	(`K60DN` | `K60cn1`) indicating the hardware model name.
HW rev	Lists the hardware revision number.
HW type code	Hardware type code.
Number Ethernet Interfaces	Number of Ethernet ports.
Number RF Interfaces	Number of radios. Up to 4 for a K60DN.
Part number	Manufacturing part number.
Serial number	Manufacturing serial number
bootloader version	Name and version of boot loader. Ex: U-Boot 2018.09+fsl (Jul 02 2021 - 08:43:46 +0000)-Terragraph Uboot Version: 1.
client	If a client certificate was used to authenticate, this field displays the client certificate common name (CN).

description	Device description, as configured. Config keyword: system.kwikbit.description
location	Device location, as configured. Config keyword: system.kwikbit.location
kb_name	The hostname of the radio (KB-XX-XX-XX).
software	Embedded software revision.
Node role	(`DN` | `CN`) Role of this device.
node type	Identifies the device's node type: • **CN** — Device can only function as a client. • **CN/DN** — Device can be configured as a distribution node or a client, as determined by the wireless.device.role CLI configuration parameter or **Wireless role** on the **Wireless** tab of the WebUI. • **PTP** — Device is a Point-to-Point device.
radio_fw_version	Firmware version number reported by the device's 60 GHz radio.
gps available	Indicates if the device has on-board GPS. • **0** — No GPS is available • **1** — GPS is available.

3.12.3 Sample Response

This sample shows a node_identity response for a K60DN. Note, in this sample long lines are wrapped to the next line. In the JSON response the line is not wrapped.


```
{
     "Ethernet MAC" : "72:88:6B:C6:04:0D",
     "HW name" : "K60DN",
     "HW rev" : 22,
     "HW type code" : 96,
     "Node role" : "DN",
     "Number Ethernet Interfaces" : 6,
     "Number RF Interfaces" : 4,
     "Part number" : "XXXX-XXXX-XXXX-K60DN-LBKA0ZZ1SV1",
     "Serial number" : "AK34045475",
     "bootloader version" : "U-Boot 2018.09+fsl (Jul 02 2021 - 08:43:46
     "client" : "Altowav",
      +0000)-Terragraph Uboot Version: 1.",
     "description" : "as configured system.kwikbit.description",
     "gps available" : 1,
     "location" : "as configured system.kwikbit.location",
     "name" : "KB-C6-04-0D",
     "software" : "0.8.0"
     "node type" : "CN/DN",
     "radio_fw_version" : "10.11.0.108
     "software" : "2.7.3"
  }
```

3.13 Orientation

Returns information about the device's physical orientation, including:

- The forward or backward pitch of the radio (the **tilt**).
- The side-to-side angle of the radio (the **lean**).
- The compass direction that the radio is facing (the **heading**). The heading is only included when the `extended=true` parameter is included in the URL.

This endpoint does not apply to the K60DN or K60CN1.

3.13.1 Request

orientation Request

Name	Description
URL	`https://<server_name>/rest/v002/device/orientation`
URL parameters	`extended=true` `kb_name=KB-XX-XX-XX` (see Global query parameters). `output=[text|json]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.

3.13.2 Response

orientation Response

Name	Description
Content type	Application/JSON or text.
kb_name	The hostname of the radio (KB-XX-XX-XX).
instantaneous information is the point-in-time status reported by the radio about its current orientation. Because of various factors, reported numbers might change over time without the radio actually moving. Generally this variation is less that one degree. AltoPlex radios only report the current point-in-time reading and do not report an average of multiple readings.	
tilt	The forward or backward pitch of the radio, measured in degrees. • Zero (`0`) degrees of pitch means that the radio is upright, facing outward at exactly a 90° angle to the ground (i.e., to the downward vector of gravity). • A positive number indicates that the radio is facing upward (i.e., toward the sky). • A negative number indicates that the radio is facing downward (i.e., toward the ground).

lean	The left or right (side-to-side) lean of the radio (also known as the roll), measured in degrees. • Zero (0) degrees means the radio is upright, with both the left and right sides at exactly a 90° angle to the ground (i.e., to the downward vector of gravity). • A positive number indicates that the radio is leaning to the right, as defined when looking out from the radio. • A negative number indicates that the radio is leaning to the left, as defined when looking out from the radio.
heading	The compass direction the radio reports that it is facing. • Zero (0) degrees means the radio is reporting that it is facing due north. • 180 degrees means the radio is reporting that it is facing due south. • A positive number means that the radio is reporting that it is facing westerly. • A negative number means that the radio is reporting that it is facing easterly. **Note:** Heading is determined by the radio's magnetometer. The magnetometer is sensitive to nearby iron objects, such as metal poles, which can cause the reported heading to be inaccurate. As a result, the reported heading should be used to observe changes in the radio's heading over time (assuming no variation in the amount of iron near the radio), rather than an absolute indicator of the radio's heading. Because of this inaccuracy, heading is only reported when the **extended** parameter is included.

3.13.3 Sample Response

This example curl command shows response when using no parameters.

```
$ curl -k -u admin:admin \
https://KB-C7-00-00.local/rest/v002/device/orientation
{
    "kb_name": "KB-C7-00-00",
    "instantaneous": {
        "tilt": -10.6083,
        "lean": -2.65889
    }
}
```

This example shows a response when using the **extended** parameter.

```
$ curl -k -u admin:admin \
https://KB-C7-00-00.local/rest/v002/device/orientation?extended=true
{
    "kb_name": "KB-C7-00-00",
    "instantaneous": {
        "tilt": -10.6083,
        "lean": -2.65889
        “heading”: -174.021
```

3.14 Radio Status

Returns data for all links defined for the DN queried. The CN returns a limited data set. The response includes measurements of RSSI, SNR, MCS from both local and remote radios for each link.

The response is arranged as an array of the DN's radios with an array of link data for each radio nested inside. See sample response below.

If the 503 error – server not available is issued in response, simply try the call again, waiting at least the time specified in the 503 error header. Typically one second is adequate.

3.14.1 Request

radio_status Request

Name	Description
URL	`https://<server_name>/rest/v002/device/radio_status`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|json]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Examples: https://<server_name>/rest/v002/device/radio_status https://<server_name>/rest/v002/device/radio_status?output=text	

3.14.2 Response

radio_status Response

Name	Description
Content type	Application/JSON or text.
kb_name	The hostname of the radio (KB-XX-XX-XX) at the local end of the link.
radio information is returned for each device listed – 4 radios per DN – with the following attributes.	
description	Radio description, as configured. Config keyword: wireless.wlanN.description -- where **N** is the radio number.
mac	MAC address of the radio.
name	Radio name, (wlan0 - wlan3).

Data for each of the radio's **links** is returned in an array within each radio's listing.	
Dynamic	Indicates the state of a dynamic link: • 0 — indicates a normal link. • 1 — indicates a dynamic link has been configured on the DN initializer and is waiting for the dynamic link to complete. • 2 — indicates that the DN responder is responding to the dynamic link from the DN initializer.
interface	Wireless interface name, (kb000-kb319). The name format is **kbrnn** where **r** is a radio number (0-3) and **nn** identifies a connection for the radio (00-19).
interface_state	(`up` | `down` | `blocked`) Indicates the current run state of the interface.
is_alive	(`true` | `false`) indicates whether the link is up or down.
link_downtime	If the link is down, the number of seconds that have passed since the link went down. Otherwise 0.
link_name	Name of the link, as configured. For example: `link-aKB-C6-04-0D-zdn-04:ce:14:fc:b5:6b` - Where ***a*** indicates the local end of the link and ***z*** indicates the remote end of the link.
link_uptime	Number of seconds the link has been up.
linkup_attempts	Number of link-up attempts.
linkup_attempts_since_last_down	The number of linkup attempts that have been started since the last time the link went down.
remote_ipv4_addr	IPv4 address of remote radio.
remote_ipv6_addr	IPv6 address of remote radio.
remote_kb_name	KB MAC (host name) of remote radio.
remote_mac	MAC address of remote radio.
remote_node_type	Model of remote radio.
local_radio and **remote_radio** data is returned in separate subsections for each link, with the following attributes.	
chn.channel	Channel frequency set for the radio (1-4).
currentLinkState	Current link state (0 is down, 1 is up)
latpcStats.nCW	Count of data blocks received.
latpcStats.nSyn	Count of data blocks received with errors.
latpcStats.synPERQ16	Packet error rate, based on LDPC stats.

phyPeriodic.rxBeamIdx	Receiver (rx) beam index.
phyPeriodic.txBeamIdx	Transmitter (tx) beam index.
phystatus.srssi	Signal strength measured in dBm.
phystatus.ssnrEst	Signal to noise ratio.
process.uptime.seconds	Shows the increasing run time of radio management daemon process. Useful only to confirm successful inter-process-communication mechanism for this API call. If this number increases on each subsequent call, you are getting valid data samples. (This is not the uptime of the device.) See **link_uptime** for length of time the link has been up.
staPkt.linkUpDataDown	A count of the number of times that the link was up, but unable to pass data plane traffic. A non-zero value shows that a problem has occurred at some point.
staPkt.mcs	MCS level for the link (1-12). Tx MCS listed for local_radio, Rx MCS listed for remote_radio.
staPkt.perE6	Packet error rate per 100,000.
staPkt.txPowerIndex	Transmission (tx) power index.
tsf.syncModeGps	GPS synchronization. (0 is not synchronized, 1 is synchronized.)
tsf.syncModeRf	RF synchronization. (0 is not synchronized, 1 is synchronized.)

3.14.3 Sample Responses:

This sample shows a JSON version of a ***radio_status*** response for a DN with 4 radios. Wlan0 has three active links. The other radios have no links.


```
{
  "kb_name": "KB-C6-04-0D",
  "radio": [                                    <-- array of radios
    {
      "description": "Pointing to center of room",
      "mac": "04:ce:14:fe:a5:00",
      "name": "wlan0"

      "links": [                                 <-- array of links
        {
          "interface": "kb001",
          "interface_state": "up",
          "is_alive": true,
          "link_downtime": 0,
          "link_name": "link-aKB-C6-04-0D-zdn-04:ce:14:fc:b5:6b",
          "link_uptime": 91,
          "linkup_attempts": 1,
          "linkup_attempts_since_last_down": 1,
          "remote_ipv4_addr": "192.168.0.51",
          "remote_ipv6_addr": "fe80::6ce:14ff:fefc:b56b",
          "remote_kb_name": "KB-C6-04-1B",
          "remote_mac": "04:ce:14:fc:b5:6b",
          "remote_node_type": "DN",
```

```
      "local_radio": {
        "chn.channel": 1,
        "currentLinkState": 1,
         "latpcStats.nCW": 13540,
         "latpcStats.nSyn": 0,
        "latpcStats.synPERQ16": 0,
         "phyPeriodic.rxBeamIdx": 37,
        "phyPeriodic.txBeamIdx": 3,
        "phystatus.srssi": -48,
        "phystatus.ssnrEst": 25,
        "process.uptime.seconds": 7161,
        "staPkt.linkUpDataDown": 0,
        "staPkt.mcs": 9,
        "staPkt.perE6": 0,
        "staPkt.txPowerIndex": 6,
        "tsf.syncModeGps": 1,
        "tsf.syncModeRf": 0
      },
      "remote_radio": {
        "chn.channel": 1,
        "currentLinkState": 1,
        "latpcStats.nCW": 3058164,
        "latpcStats.nSyn": 0,
         "latpcStats.synPERQ16": 0,
        "phyPeriodic.rxBeamIdx": 26,
        "phyPeriodic.txBeamIdx": 29,
        "phystatus.srssi": -52,
        "phystatus.ssnrEst": 21,
        "process.uptime.seconds": 7322,
        "staPkt.linkUpDataDown": 0,
        "staPkt.mcs": 9,
        "staPkt.perE6": 0,
        "staPkt.txPowerIndex": 6,
        "tsf.syncModeGps": 0,
        "tsf.syncModeRf": 1
      }
    },

    {
      "interface": "kb002",
      "interface_state": "up",
      "is_alive": true,
      "link_name": "link-aKB-C6-04-0D-zcn-KB-C6-04-48",
      "link_uptime": 91,
      "linkup_attempts": 6,
      "remote_ipv4_addr": "192.168.0.168",
      "remote_ipv6_addr": "fe80::7288:6bff:fec6:448",
      "remote_kb_name": "KB-C6-04-48",
      "remote_mac": "70:88:6b:c6:04:48",
      "remote_node_type": "CN",

      "local_radio": {
        "chn.channel": 1,
        "currentLinkState": 1,
        "latpcStats.nCW": 11140,
         "latpcStats.nSyn": 0,
         "latpcStats.synPERQ16": 0,
        "phyPeriodic.rxBeamIdx": 32,
        "phyPeriodic.txBeamIdx": 33,
        "phystatus.srssi": -53,
        "phystatus.ssnrEst": 21,
        "process.uptime.seconds": 7161,
        "staPkt.linkUpDataDown": 0,
```

```
            "staPkt.mcs": 9,
            "staPkt.perE6": 0,
            "staPkt.txPowerIndex": 6,
            "tsf.syncModeGps": 1,
            "tsf.syncModeRf": 0
          },

          "remote_radio": {
            "chn.channel": 1,
            "currentLinkState": 1,
            "latpcStats.nCW": 2836164,
            "latpcStats.nSyn": 0,
             "latpcStats.synPERQ16": 0,
            "phyPeriodic.rxBeamIdx": 34,
            "phyPeriodic.txBeamIdx": 34,
            "phystatus.srssi": -44,
            "phystatus.ssnrEst": 29,
            "process.uptime.seconds": 7325,
            "staPkt.linkUpDataDown": 0,
            "staPkt.mcs": 9,
            "staPkt.perE6": 0,
            "staPkt.txPowerIndex": 6,
            "tsf.syncModeGps": 0,
            "tsf.syncModeRf": 1
          }
        },
        {
          "interface": "kb003",
          "interface_state": "up",
          "is_alive": true,
          "link_name": "link-aKB-C6-04-0D-zcn-KB-C6-04-47",
          "link_uptime": 91,
          "linkup_attempts": 11,
          "remote_ipv4_addr": "192.168.0.117",
          "remote_ipv6_addr": "fe80::7288:6bff:fec6:447",
          "remote_kb_name": "KB-C6-04-47",
          "remote_mac": "70:88:6b:c6:04:47",
          "remote_node_type": "CN",

          "local_radio": {
            "chn.channel": 1,
            "currentLinkState": 1,
            "latpcStats.nCW": 11140,
             "latpcStats.nSyn": 0,
             "latpcStats.synPERQ16": 0,
            "phyPeriodic.rxBeamIdx": 34,
            "phyPeriodic.txBeamIdx": 36,
            "phystatus.srssi": -57,
            "phystatus.ssnrEst": 16,
            "process.uptime.seconds": 7161,
            "staPkt.linkUpDataDown": 0,
            "staPkt.mcs": 9,
            "staPkt.perE6": 0,
            "staPkt.txPowerIndex": 6,
            "tsf.syncModeGps": 1,
            "tsf.syncModeRf": 0
          },
          "remote_radio": {
            "chn.channel": 1,
            "currentLinkState": 1,
            "latpcStats.nCW": 2836164,
            "latpcStats.nSyn": 0,
             "latpcStats.synPERQ16": 0,
```

```
            "phyPeriodic.rxBeamIdx": 31,
            "phyPeriodic.txBeamIdx": 33,
            "phystatus.srssi": -44,
            "phystatus.ssnrEst": 29,
            "process.uptime.seconds": 1338,
            "staPkt.linkUpDataDown": 0,
            "staPkt.mcs": 9,
            "staPkt.perE6": 0,
            "staPkt.txPowerIndex": 6,
            "tsf.syncModeGps": 0,
            "tsf.syncModeRf": 1
          }
        }
      ],
    },
    {
      "description": "radio 1 description not set",
      "mac": "04:ce:14:fe:b4:e6",
      "name": "wlan1",
      "links": [],                          <-- no links here
    },
    {
      "description": "radio 2 description not set",
      "mac": "04:ce:14:fe:aa:92",
      "name": "wlan2",
      "links": [],
    },
    {
      "description": "radio 3 description not set",
      "mac": "04:ce:14:fe:b5:4d",
      "name": "wlan3",
      "links": [],
    }
  ]
}
```

This sample shows a text version of a ***radio_status*** response for a case where a DN has three links, on radio Wlan0, and no other links.

```
KwikbitRadioStatus: KB-C6-04-0D
wlan0, 04:ce:14:fe:a5:00, "Pointing to center of room"  <-- radio: wlan0
 Link: link-aKB-C6-04-0D-zdn-04:ce:14:fc:b5:6b          <-- first link
  remote_node_type DN
  remote_mac 04:ce:14:fc:b5:6b
  remote_kb_name dn-04:ce:14:fc:b5:6b
  remote_ipv4_addr 192.168.0.51
  remote_ipv6_addr fe80::6ce:14ff:fefc:b56b
  interface kb001
  interface_status up
  is_alive 1
  linkup_attempts 1
  local_radio:
   chn.channel = 1
   currentLinkState = 1
   latpcStats.nCW = 13540,
   latpcStats.nSyn = 0,
   latpcStats.synPERQ16 = 0,
   phyPeriodic.rxBeamIdx = 37
   phyPeriodic.txBeamIdx = 3
   phystatus.srssi = -48
   phystatus.ssnrEst = 24
   process.uptime.seconds = 7849
```

```
   staPkt.linkUpDataDown = 0
   staPkt.mcs = 9
   staPkt.perE6 = 0
   staPkt.txPowerIndex = 6
   tsf.syncModeGps = 1
   tsf.syncModeRf = 0
  remote_radio:
   chn.channel = 1
   currentLinkState = 1
   latpcStats.nCW = 2836164,
   latpcStats.nSyn = 0,
   latpcStats.synPERQ16 = 0,
   phyPeriodic.rxBeamIdx = 26
   phyPeriodic.txBeamIdx = 29
   phystatus.srssi = -52
   phystatus.ssnrEst = 22
   process.uptime.seconds = 8009
   staPkt.linkUpDataDown = 0
   staPkt.mcs = 9
   staPkt.perE6 = 0
   staPkt.txPowerIndex = 6
   tsf.syncModeGps = 0
   tsf.syncModeRf = 1
Link: link-aKB-C6-04-0D-zcn-KB-C6-04-48            <-- second link
 remote_node_type CN
 remote_mac 70:88:6b:c6:04:48
 remote_kb_name KB-C6-04-48
 remote_ipv4_addr 192.168.0.168
 remote_ipv6_addr fe80::7288:6bff:fec6:448
 interface kb002
 interface_status up
 is_alive 1
 linkup_attempts 6
 local_radio:
  chn.channel = 1
  currentLinkState = 1
  latpcStats.nCW = 13540,
  latpcStats.nSyn = 0,
  latpcStats.synPERQ16 = 0,
  phyPeriodic.rxBeamIdx = 32
  phyPeriodic.txBeamIdx = 33
  phystatus.srssi = -53
  phystatus.ssnrEst = 21
  process.uptime.seconds = 7849
  staPkt.linkUpDataDown = 0
  staPkt.mcs = 9
  staPkt.perE6 = 0
  staPkt.txPowerIndex = 6
  tsf.syncModeGps = 1
  tsf.syncModeRf = 0
 remote_radio:
  chn.channel = 1
  currentLinkState = 1
  latpcStats.nCW = 2836164,
  latpcStats.nSyn = 0,
  latpcStats.synPERQ16 = 0,
  phyPeriodic.rxBeamIdx = 34
  phyPeriodic.txBeamIdx = 34
  phystatus.srssi = -44
  phystatus.ssnrEst = 29
  process.uptime.seconds = 8012
  staPkt.linkUpDataDown = 0
  staPkt.mcs = 9
```

```
   staPkt.perE6 = 0
   staPkt.txPowerIndex = 6
   tsf.syncModeGps = 0
   tsf.syncModeRf = 1
 Link: link-aKB-C6-04-0D-zcn-KB-C6-04-47            <-- third link
  remote_node_type CN
  remote_mac 70:88:6b:c6:04:47
  remote_kb_name KB-C6-04-47
  remote_ipv4_addr 192.168.0.117
  remote_ipv6_addr fe80::7288:6bff:fec6:447
  interface kb003
  interface_status up
  is_alive 1
  linkup_attempts 11
  local_radio:
   chn.channel = 1
   currentLinkState = 1
   latpcStats.nCW = 13540,
   latpcStats.nSyn = 0,
   latpcStats.synPERQ16 = 0,
   phyPeriodic.rxBeamIdx = 34
   phyPeriodic.txBeamIdx = 36
   phystatus.srssi = -58
   phystatus.ssnrEst = 16
   process.uptime.seconds = 7849
   staPkt.linkUpDataDown = 0
   staPkt.mcs = 9
   staPkt.perE6 = 0
   staPkt.txPowerIndex = 6
   tsf.syncModeGps = 1
   tsf.syncModeRf = 0
  remote_radio:
   chn.channel = 1
   currentLinkState = 1
   latpcStats.nCW = 2836164,
   latpcStats.nSyn = 0,
   latpcStats.synPERQ16 = 0,
   phyPeriodic.rxBeamIdx = 31
   phyPeriodic.txBeamIdx = 33
   phystatus.srssi = -44
   phystatus.ssnrEst = 29
   process.uptime.seconds = 2025
   staPkt.linkUpDataDown = 0
   staPkt.mcs = 9
   staPkt.perE6 = 0
   staPkt.txPowerIndex = 6
   tsf.syncModeGps = 0
   tsf.syncModeRf = 1
wlan1, 04:ce:14:fe:b4:e6, "radio 1 description not set"  <-- radio: wlan1
wlan2, 04:ce:14:fe:aa:92, "radio 2 description not set"  <-- radio: wlan2
wlan3, 04:ce:14:fe:b5:4d, "radio 3 description not set"  <-- radio: wlan3
```

3.15 Resource utilization

Returns CPU load and memory information.

The CPU load section displays the number of CPUs and load averages over 1, 5, and 15 minute intervals. A higher load average than the number of CPUs indicates a heavy load. The memory statistics section displays statistics about physical memory: total, used and free memory, in kilobytes.

The optional extended parameter includes cumulative CPU usage since the last reboot. These numbers can be used to calculate CPU usage percentages for each type (idle, waiting, and work) over a desired time interval. The extended memory statistics includes additional information about physical memory, and also information about swap memory.

3.15.1 Request

resource_utilization request

Name	Description
URL	`https://<server_name>/rest/v002/device/resource_utilization`
URL parameters	`extended=true` `kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|json]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Examples: https://<server_name>/rest/v002/device/resource_utilization https://<server_name>/rest/v002/device/resource_utilization?extended=true	

3.15.1 Response

resource_utilization response

Name	Description
Content type	Application/JSON or text.
kb_name	The hostname of the radio (KB-XX-XX-XX).
cpu_num	The number of CPUs.
load_avg – Average CPU load. A higher load average than the number of CPUs indicates a heavy load.	
load_avg_1min	Average CPU load over one minute.
load_avg_5min	Average CPU load over five minutes.

load_avg_15min	Average CPU load over 15 minute.
cumulative_cpu_usage_raw – Displayed when `extended` parameter is included. The numbers displayed are the total number of ticks — hundredths of a second — aggregate for all CPUs since the last reboot.	
cpu_raw_user	The number of ticks spent processing user-level code.
cpu_raw_nice	The number of ticks spent processing low priority user-level code.
cpu_raw_system	The number of ticks spent processing system-level code.
cpu_raw_idle	The number of ticks spent idle.
cpu_raw_wait	The number of ticks spent waiting for I/O.
cpu_raw_interrupt	The number of ticks spent processing hardware interrupts.
cpu_raw_softIRQ	The number of ticks spent processing software interrupts.
memory_statistics – Memory usage in kilobytes. The **physical** memory section refers to actual physical memory, while the **swap** section (displayed with `extended` parameter is included) refers to swap space.	
mem_total	Total amount, in kilobytes, of physical memory.
mem_used	Amount, in kilobytes, of used memory.
mem_free	Amount, in kilobytes, of unused memory.
mem_shared	Amount, in kilobytes, of memory allocated as shared.
mem_buffer	Amount, in kilobytes, of memory allocated for use as memory buffers.
mem_cached	Amount, in kilobytes, of memory allocated for use as cached memory.
mem_avail	Amount, in kilobytes, of memory available for allocation without using swapping. This is dfferent than free memory because it includes memory buffers and cached memory that can be freed without using swap space.
swap_total	Total amount, in kilobytes, of swap space.
swap_used	Amount, in kilobytes, of swap space being used.
swap_avail	Amount, in kilobytes, of swap space that is unused and available.

3.15.1 Sample Responses:

This sample shows a JSON version of a ***resource_utilization*** response.

```
{
  "kb_name": "KB-C7-00-01",
  "cpu_statistics": {
     "cpu_num": "4",
     "load_avg": {
             "load_avg_1min": "1.04",
             "load_avg_5min": "0.61",
             "load_avg_15min": "0.25"
     }
     },
  "memory_statistics": {
     "physical": {
             "mem_total": "503376 kB",
             "mem_used": "209644 kB",
             "mem_free": "208900 kB"
     }
  }
}
```

This sample shows a text version of a ***resource_utilization*** response using the `extended` parameter:

```
  kb_name: KB-C7-00-01
  cpu_statistics:
    cpu_num: 4
    load_avg:
      load_avg_1min: 1.10
      load_avg_5min: 1.07
      load_avg_15min: 1.01
    cumulative_cpu_usage_raw:
      cpu_raw_user: 9914
      cpu_raw_nice: 0
      cpu_raw_system: 9925
      cpu_raw_idle: 1120440
      cpu_raw_wait: 0
      cpu_raw_interrupt: 0
      cpu_raw_softIRQ: 436
  memory_statistics:
    physical:
      mem_total: 498256 kB
      mem_used: 216940 kB
      mem_free: 199944 kB
      mem_shared: 81288 kB
      mem_buffer: 0 kB
      mem_cached: 81372 kB
      mem_avail: 198060 kB
    swap:
      swap_total: 0 kB
      swap_used: 0 kB
      swap_avail: 0 kB
```

3.16 STP Status

Returns data for spanning tree protocol (STP) status.

3.16.1 Request

stp_status Request

Name	Description
URL	`https://<server_name>/rest/v002/device/stp_status`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|json]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Examples: https://<server_name>/rest/v002/device/stp_status https://<server_name>/rest/v002/device/stp_status?output=text	

3.16.2 Response

stp_status response

Name	Description
Content type	Application/JSON or text.
kb_name	The hostname of the radio (KB-XX-XX-XX).
network_stp_admin	Enable or disable STP. Values: (`enable|disable`). Config keyword: network.stp.admin
bridge_id	MAC address of the local bridge.
network_stp_priority	Used to determine which device will serve as the root of the STP bridge. The device with the lowest priority will serve as the root. The priority configured here is a multiplier; to determine the actual STP priority, multiply by 4096.
designated_root	MAC address of the bridge that is the root of the spanning tree.
root_port	Interface name of the local bridge port that leads to the root. Returns a blank when the local bridge is the spanning tree root.

path_cost	Represents the distance from the spanning tree root node in a way that also factors in the speed of the link(s) between the node being examined and the root node.
force_protocol_version	Version of the spanning tree protocol. A value of "`stp`" indicates that non-Altowav equipment which does not support RSTP is participating in the spanning tree.
time_since_topology_change	Number of seconds since the last spanning tree topology change was observed by the local bridge.
topology_change_count	Number of spanning tree topology changes observed by the local bridge.
ports – data returned for each interface in the local bridge that is not down. Each interface has a subsection which begins with the **interface name** and includes the following attributes.	
state	Indicates the port's current stp state. Values: (`listening`|`learning`|`forwarding`|`blocking`).
remote_mac	MAC address of radio on the other end of the wireless link. This is blank for all Ethernet ports.
edge	Indicates the port is at the edge of the spanning tree and nothing connected to the port is participating in the spanning tree. Values: (`yes` | `no`) .
cost	Used to determine the preferred path to the root. The path with the lowest cumulative cost is used.
bpdufilter	Indicates whether the Bridge Protocol Data Unit (BPDU) filter is enabled. The BPDU filter prevents STP packets from being forwarded, which allows for separate networks to be isolated from participating in the same STP environment. When enabled, the filter is applied whether or not network_stp_admin is enabled. Values: (`yes`|`no`) . This filter applies to Ethernet ports only. Wireless interfaces will always have a value of `no`.
num_rx_bpdu_filtered	The number of BPDU packets that have been filtered and dropped. Applies to Ethernet ports only. Wireless interfaces will always have a value of `0`.

3.16.3 Sample Responses:

This sample shows a JSON version of an ***stp_status*** response showing two wireless interfaces with different states and an Ethernet port that is at the edge of the spanning tree.


```
{
  "kb_name": "KB-C6-04-11",
  "network_stp_admin": "enable",
  "bridge_id": "70:88:6B:C6:04:11",
  "bridge_priority": "8",
  "designated_root": "70:88:6B:C6:04:0E",
  "root_port": "kb000",
  "path_cost": "20000",
  "force_protocol_version": "rstp",
  "time_since_topology_change": "374",
  "topology_change_count": "1",
  "ports": {
    "kb200": {
      "state": "blocking",
      "remote_mac": "04:ce:14:fe:b2:d9",
      "edge": "no
      "cost": "123456"
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
    },
    "kb000": {
      "state": "forwarding",
      "remote_mac": "04:ce:14:fe:b2:cf",
      "edge": "no"
      "cost": "20000"
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
    },
    "eth5": {
      "state": "forwarding",
      "remote_mac": "",
      "edge": "yes"
      "cost": "20000"
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
    }
  }
}
```


This sample shows a JSON version of an ***stp_status*** response where this node is the root of the spanning tree.


```
{
  "kb_name": "KB-C6-04-0E",
  "network_stp_admin": "enable",
  "bridge_id": "70:88:6B:C6:04:0E",
  "bridge_priority": "8",
  "designated_root": "70:88:6B:C6:04:0E",
  "root_port": "",
  "path_cost": "0",
  "force_protocol_version": "rstp",
  "time_since_topology_change": "312",
  "topology_change_count": "7",
  "ports": {
    "kb300": {
      "state": "forwarding",
      "remote_mac": "04:ce:14:fe:b5:82",
```

```
      "edge": "no"
      "cost": "20000"
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
    },
    "kb000": {
      "state": "forwarding",
      "remote_mac": "04:ce:14:fe:aa:09",
      "edge": "no"
      "cost": "20000"
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
    },
    "kb001": {
      "state": "forwarding",
      "remote_mac": "70:88:6b:c6:04:46",
      "edge": "no"
      "cost": "20000"
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
    },
    "eth0": {
      "state": "forwarding",
      "remote_mac": "",
      "edge": "yes"
      "cost": "20000"
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
    },
    "eth5": {
      "state": "forwarding",
      "remote_mac": "",
      "edge": "yes"
      "cost": "20000"
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
    }
  }
}
```


This sample shows a text version of an ***stp_status*** response where spanning tree is disabled, so many values are blank.


```
  "kb_name": "KB-C0-01-30",
  "network_stp_admin": "disable",
  "bridge_id": "",
  "bridge_priority": "",
  "designated_root": "",
  "root_port": "",
  "path_cost": "",
  "force_protocol_version": "",
  "time_since_topology_change": "",
  "topology_change_count": "",
  "ports": {
    " kb000": {
      "state": "forwarding",
      "remote_mac": "",
      "edge": "",
      "cost": ""
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
```

```
    " eth2": {
      "state": "forwarding",
      "remote_mac": "",
      "edge": "",
      "cost": ""
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
    " eth1": {
      "state": "forwarding",
      "remote_mac": "",
      "edge": "",
      "cost": ""
      "bpdufilter": "no",
      "num_rx_bpdu_filtered": "0"
```

3.17 Syslog

Returns the syslog messages from the device. By default all messages are returned. Otherwise, to limit messages specify "info", "warning", or "critical". (Normally Altowav logging consists only of these three priority values. Under certain conditions debug or other messages may be generated.)

3.17.1 Request

syslog Request

Name	Description
URL	`https://<server_name>/rest/v002/device/syslog`
URL parameters	`severity=[info | warning | critical]` `kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Examples: To return all messages. https://<server_name>/rest/v002/device/syslog To return info, warning, and critical messages. https://<server_name>/rest/v002/device/syslog?severity=info To return warning and critical messages. https://<server_name>/rest/v002/device/syslog?severity=warning To return critical messages. https://<server_name>/rest/v002/device/syslog?severity=critical	

3.17.2 Response

syslog Response

Name	Description
Content type	text/plain

3.18 Temperature Stats

Returns the temperature statistics for the queried device, using degrees Celcius (C˚). The response includes an array of the device's radios with baseband temperature and the temperature of each RF tile. The response also includes the “in case temperature” and “processor temperature” from the SoC (system on a chip). Not all devices have all available temperatures; in cases where a temperature is not available, a response of “-273” is returned. See sample responses below. Use temperature_stats to poll for temperature instead of the deprecated, “radio_temp” option in device_status.

3.18.1 Request

temperature_stats Request

Name	Description
URL	`https://<server_name>/rest/v002/device/temperature_stats`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|json]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Examples: https://<server_name>/rest/v002/device/temperature_stats https://<server_name>/rest/v002/device/temperature_stats?output=text	

3.18.2 Response

temperature_stats Response

Name	Description
Content type	Application/JSON or text.
kb_name	The hostname of the radio (KB-XX-XX-XX).
radio_devices information is returned for the 4 radios per DN – with the following attributes.	
mac	MAC address of the radio.
name	Radio name. DN radios are named `wlan0` – `wlan3`.
temperature_data includes the following baseband and RF tile temperatures for each radio.	
iftemperature	Baseband temperature in C˚.
rftemperatureN	RF tile temperature in C˚ where **N** is the tile number 0-3.

3.18.1 Sample Responses:

This sample shows a JSON version of a ***temperature_stats*** response for a K60DN with 4 radios. Temperatures are given in degrees Celcius (C˚).


```
{
  "kb_name": "KB-C6-04-0D",
  "radio_devices": [
    {
      "mac": "04:ce:14:0e:a5:00",
      "name": "wlan0",
      "temperature_data": {
        "iftemperature": 58,
        "rftemperature0": 58,
        "rftemperature1": 53,
        "rftemperature2": 54,
        "rftemperature3": 56
      }
    },
    {
      "mac": "04:ce:14:fe:b4:e6",
      "name": "wlan1",
      "temperature_data": {
        "iftemperature": 51,
        "rftemperature0": 22,
        "rftemperature1": 49,
        "rftemperature2": 33,
        "rftemperature3": 28
      }
    },
    {
      "mac": "04:ce:14:fe:aa:92",
      "name": "wlan2",
      "temperature_data": {
        "iftemperature": 53,
        "rftemperature0": 35,
        "rftemperature1": 47,
        "rftemperature2": 39,
        "rftemperature3": 39
      }
    },
    {
      "mac": "04:ce:14:fe:b5:4d",
      "name": "wlan3",
      "temperature_data": {
        "iftemperature": 57,
        "rftemperature0": 25,
        "rftemperature1": 45,
        "rftemperature2": 33,
        "rftemperature3": 42
      }
    }
  ],
  "temperature_data": {
    "temperature_case": 53,
    "temperature_processor": 56
  }
}
```

This sample shows a JSON version of a ***temperature_stats*** response for a D621. Note, the response "-273" is used to show a non-existent temperature.


```
{
  "kb_name": "KB-C7-00-00",
  "radio_devices": [
    {
      "mac": "70:88:6b:c7:08:08",
      "name": "wlan0",
      "temperature_data": {
        "iftemperature": 71,
        "rftemperature0": 70,
        "rftemperature1": 57,
        "rftemperature2": -273,
        "rftemperature3": -273
      }
    }
  ],
  "temperature_data": {
    "temperature_case": -273,
    "temperature_processor": -273
  }
}
```


This sample shows a text version of a ***temperature_stats*** response for a D621.

```
KwikbitTemperatureStats: KB-C7-08-08
 temperature_case = -273
 temperature_processor = -273
wlan0, 70:88:6b:c7:08:08
 iftemperature = 71
 rftemperature0 = 70
 rftemperature1 = 57
 rftemperature2 = -273
 rftemperature3 = -273
```

3.19 Topology

Extracts the topology document from a DN. This document provides a list of defined links that describe the topology of the network.

If the 503 error – server not available is issued in response, simply try the call again after a second, waiting at least the time specified in the 503 error header. Typically one second is adequate.

Note: The topology API is not related to the topology_scan API, which performs a scan on a local device to discover nearby radios without knowing their MAC addresses or GPS locations.

3.19.1 Request

topology Request

Name	Description
URL	`https://<server_name>/rest/v002/device/topology`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|json]` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Not required.
Examples: https://<server_name>/rest/v002/device/topology https://<server_name>/rest/v002/device/topology?output=text	

3.19.2 Responses

The response document includes four sections, config, links, nodes and sites. In the response data an “a” is used to indicate the local DN and “z” indicates the remote node on the link.

topology Response

Name	Description
config – This section is not used.	
kb_name	The hostname of the radio (KB-XX-XX-XX).
links – This section lists the links defined on this DN. Each link is a connection between two nodes.	
name	Name of the link, as configured. For example: `link-aKB-C6-04-0D-zdn-04:ce:14:fc:b5:6b` - Where ***a*** indicates the local end of the link and ***z*** indicates the remote end of the link.
a_node_mac	MAC address of the local end of the link.

a_node_name	Host name (KB name) of the radio at the local end of the link, preceded by the letter ***a.*** Ex: `aKB-C6-04-0D`
dynamic	Indicates the state of a dynamic link: • 0 — indicates a normal link. • 1 — indicates a dynamic link has been configured on the DN initializer and is waiting for the dynamic link to complete. • 2 — indicates that the DN responder is responding to the dynamic link from the DN initializer.
is_alive	(`true` | `false`) Indicates whether the link is up or down.
link_type	A value of **`1`** means wireless.
Linkdown_time	The system uptime, in seconds, when either: • The link last went down, or • The link was first added to configuration.
linkup_attempts	Number of attempts to create link.
linkup_attempts_since_last_down	The number of linkup attempts that have been started since the last time the link went down.
linkup_time	The system uptime, in seconds, when the link last came up.
z_node_mac	MAC address of the remote end of the link.
z_node_name	Name of the radio at the local end of the link, preceded by the letter ***z.*** Ex of DN at remote end of link: `zdn-04:ce:14:fc:b5:6b` If remote DN cannot be reached, the remote MAC address is returned.
nodes - This section lists data about the nodes at the endpoints of the links. Each device (DN or CN) is a single node. **Note:** A single DN node has four radios. A CN node has one radio.	
ant_azimuth	Azimuth of antenna.
ant_elevation	Elevation of antenna.
mac_addr	MAC address for the node.
name	Name of the node preceded by ***a*** for local node, and ***z*** for remote node.
node_type	A value of `1` means CN, and `2` means DN.
pop_node	Unused. Always set to `false`.
site_name	Lists the node's site name.
status	(`1` | `2` | `3`) Where, `1` = offline, `2` = online, `3` = online initiator

wlan_mac_addrs	Lists MAC address for all radios on the node. Lists [ ] if not known.
sites - This section is automatically generated. Site data describes the location of a node. Only the local DN site has data, others can be ignored.	
name	The name of the site.
location -- This subsection in sites lists data for the following attributes when GPS is enabled.	
accuracy	Accuracy of GPS in meters.
altitude	Altitude of local DN node site in meters.
latitude	Latitude of local DN node site in degrees.
longitude	Longitude of local DN node site in degrees.

3.19.3 Sample Responses:

This sample shows a JSON version of a ***topology*** response for a case where a DN has two links, both are down.


```
{
  "config": {},

  "links": [
    {
      "a_node_mac": "04:ce:14:fe:a5:00",
      "a_node_name": "aKB-C6-04-0D",
      "is_alive": false,
      "link_type": 1,
      "linkdown_time": 53,
      "linkup_attempts": 0,
      "linkup_attempts_since_last_down": 3,
      "linkup_time": 52,
      "name": "link-aKB-C6-04-0D-zcn-KB-C6-02-A8",
      "z_node_mac": "70:88:6b:c6:02:a8",
      "z_node_name": "zcn-KB-C6-02-A8"
    },
    {
      "a_node_mac": "04:ce:14:fe:a5:00",
      "a_node_name": "aKB-C6-04-0D",
      "is_alive": false,
      "link_type": 1,
      "linkup_attempts": 0,
      "name": "link-aKB-C6-04-0D-zdn-04:ce:14:fc:b5:6b",
      "z_node_mac": "04:ce:14:fc:b5:6b",
      "z_node_name": "zdn-04:ce:14:fc:b5:6b"
    }
  ],
  "name": "auto-topo-matic",                          <-- Topology name
  "nodes": [
    {
      "ant_azimuth": 0,
      "ant_elevation": 0,
      "mac_addr": "34:ef:b6:8a:12:ac",
      "name": "aKB-C6-04-0D",
      "node_type": 2,
      "pop_node": false,
      "site_name": "site_here",
      "status": 3,
```

```
      "wlan_mac_addrs": [
        "04:ce:14:fe:a5:00",
        "04:ce:14:fe:b4:e6",
        "04:ce:14:fe:aa:92",
        "04:ce:14:fe:b5:4d"
      ]
    },
    {
      "ant_azimuth": 0,
      "ant_elevation": 0,
      "mac_addr": "70:88:6b:c6:02:a8",
      "name": "zcn-KB-C6-02-A8",
      "node_type": 1,
      "pop_node": false,
      "site_name": "wlan0site_CN1",
      "status": 1,
      "wlan_mac_addrs": []
    },
    {
      "ant_azimuth": 0,
      "ant_elevation": 0,
      "mac_addr": "04:ce:14:fc:b5:6b",
      "name": "zdn-04:ce:14:fc:b5:6b",
      "node_type": 2,
      "pop_node": false,
      "site_name": "wlan0site_DN0",
      "status": 1,
      "wlan_mac_addrs": []
    }
  ],
  "sites": [
    {
      "location": {
        "accuracy": 39.9,
        "altitude": 290.01,
        "latitude": 44.861175,
        "longitude": -93.361218333
      },
      "name": "site_here"
    },
    {
      "location": {
        "accuracy": 40000000,
        "altitude": 0,
        "latitude": 0,
        "longitude": 0
      },
      "name": "wlan0site_CN1"
    },
    {
      "location": {
        "accuracy": 40000000,
        "altitude": 0,
        "latitude": 0,
        "longitude": 0
      },
      "name": "wlan0site_DN0"
    }
  ]
}
```

This sample shows a text version of a ***topology*** response for a case where a DN has two links, (both down) and 3 nodes. The first link is to CN, the second link is to a DN. The nodes listed are the local node (DN), a remote node (CN), and another remote node (DN).

Note: The text response does not include the config or site section. The config section is unused and the site information is combined with the node section data.

```
link_name: link-aKB-C6-04-0D-zcn-KB-C6-02-A8
  a_node:  aKB-C6-04-0D   04:ce:14:fe:a5:00
  z_node:  zcn-KB-C6-02-A8   70:88:6b:c6:02:a8
  linkup_attempts: 0
  is_alive: 0
link_name: link-aKB-C6-04-0D-zdn-04:ce:14:fc:b5:6b
  a_node:  aKB-C6-04-0D   04:ce:14:fe:a5:00
  z_node:  zdn-04:ce:14:fc:b5:6b   04:ce:14:fc:b5:6b
  linkup_attempts: 0
  is_alive: 0

node_name:           aKB-C6-04-0D  34:ef:b6:8a:12:ac
  wlan_mac_addrs: 04:ce:14:fe:a5:00
  wlan_mac_addrs: 04:ce:14:fe:b4:e6
  wlan_mac_addrs: 04:ce:14:fe:aa:92
  wlan_mac_addrs: 04:ce:14:fe:b5:4d
  Location: accuracy: 39.9
    altitude: 290.01
    latitude: 44.8612
    longitude: -93.3612
node_name:  zcn-KB-C6-02-A8  70:88:6b:c6:02:a8
node_name:  zdn-04:ce:14:fc:b5:6b  04:ce:14:fc:b5:6b
```

4 Configuration Management

This category examines and changes the value of configuration parameters.

To avoid unwanted subsystem restarts, pass a collection of configuration sets on a single API call, rather than a separate call for each individual parameter.

4.1 List-type parameters

Some configuration parameters, “CN_responder” and “VLAN membership” are list-type parameters, meaning their value may be a list of elements. All other parameters are considered non-list parameters.

List-type parameters have additional information about the list in Get responses and have additional handling features for Configuration Setting.

Response Differences

See the for list-type parameters include:

- “is_uci_list” : “true” and “max_list_length” fields are included to describe the list. These items are not included in the response for non-list type parameters.
- The “value” field may be a list of elements.

4.2 Configuration Parameters - Get

Retrieves configuration parameters. Configuration parameter data includes:

- Parameter name
- Current configurated value
- With the extended parameter (`output=ext`), additional information about configuration parameters is included, such as the default value, a description of the parameter, and the variable type used by the parameter.

Parameter keys are organized by package, section and option. For example, the parameter keyword "wireless.wlan0.description" is in the “wireless” package, “wlan0” section, with the option “description”.

Package names for parameter keys		
auth	ethernet	network
system	vlan	wireless

Use the `key=` URL option to filter the retrieved information. After the `key=` option, specify a parameter key or any leading portion of a key to match. For example, `key=wireless` returns all keys that start with "wireless”. See examples in the *configuration Request (GET)* table, below.

The field “applies if” may appear within parameter information. It indicates the parameter does not apply unless the condition is met. (Even if the parameter doesn't apply, it can still be viewed and set.) The format of the “applies if” field is: <configuration-parameter><operator><value>.

For example, (parameter "network.mgmt.ipaddr") "applies_if" : "network.mgmt.proto=static" means the parameter "network.mgmt.ipaddr" only applies when the network.mgmt.proto is set to 'static'. The following table describes the valid operators for “applies if”.

Operators for “applies if”	Description
=	Equal to
=!	Not equal to
=~	Contains

4.2.1 Request

configuration Request (GET)

Name	Description
URL	`https://<server_name>/rest/v002/configuration/data`
URL parameters	`key=<pattern match>` – see the key matching examples below. `output=ext` – requests extended output, which includes paramInfo (legacy behavior). Without this option, a shorter response is returned with keyword and value, but no paramInfo structure. `kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Required.
Examples: To output the full configuration data response: https://<server_name>/rest/v002/configuration/data To limit the output to those items in the `network` section: https://<server_name>/rest/v002/configuration/data?key=system To limit the output to those items in the `network.mgmt` section: https://<server_name>/rest/v002/configuration/data?key=network.mgmt To limit the output to only the `network.mgmt.ipaddr` configuration parameter: https://<server_name>/rest/v002/configuration/data?key=network.mgmt.ipaddr	

4.2.2 Response

Response data varies slightly depending on the device type. The following is response data for the D621:

configuration Response (GET)

Name	Description
Content type	Application/JSON.
ethernet.eth1.admin	Ethernet Port 1 Port Enable
filter.eth1.destination_mac	Ethernet Port 1 destination MAC address
filter.eth1.source_mac_limit	Ethernet Port 1 MAC Limit
filter.eth1.unicast_conversion	Ethernet Port 1 unicast conversion
network.dhcrelay.admin	DHCP Relay Agent Enable
network.dhcrelay.circuitid	DHCP Relay Agent Circuit ID Type
network.dhcrelay.eth1_circuit	Ethernet Port 1 Host Access

network.dns.servers	DNS IP List
network.mgmt.gateway	Network Gateway (static)
network.mgmt.ipaddr	IP Address (static)
network.mgmt.netmask	Network Mask (static)
network.mgmt.proto	IP Assignment Method
network.ntp.server	NTP servers
network.stp.admin	Spanning Tree Protocol (STP) Enable
network.stp.eth1_bpdufilter	STP Ethernet port 1 BPDU filter
network.stp.priority	STP Bridge Priority
network.stp.wlan0_cost	STP Radio 0 Port Path Cost
system.device.led_enable	Link state LED
system.kwikbit.description	Description
system.kwikbit.location	Location
system.snmp.agent	SNMP Agent Enable
system.snmp.notify	SNMPv2 Notification Enable
system.snmp.notify_dest_community	SNMPv2 Notification Community
system.snmp.notify_dest_host	SNMPv2 Notification Destination
system.snmp.notify_dest_port	SNMPv2 Notification port
system.snmp.rocommunity	SNMP Read-only Community
vlan.dot1q.mode	VLAN 802.1q mode
vlan.eth1.accepted_frames	Ethernet Port 1 802.1q Accepted Frame Types
vlan.eth1.membership	Ethernet Port 1 802.1q Membership
vlan.eth1.port_isolation	Ethernet Port 1 Isolation
vlan.eth1.pvid	Ethernet Port 1 802.1q PVID
vlan.mgmt.vid	Management 802.1q VLAN ID
vlan.wireless.port_isolation	Wireless Port Isolation
wifi.ap.admin	Diagnostic WiFi Access Point
wifi.ap.ipaddr	WiFi AP IP address(static)
wifi.ap.password	WiFi AP password
wifi.ap.ssid	WiFi AP SSID
wireless.device.gps_sync	GPS Synchronization
wireless.device.role	Wireless Role

wireless.scan.auto_pbf	Auto PBF enable
wireless.scan.auto_pbf_interval	PBF scan interval minutes
wireless.wlan0.CN_responder	Radio 0 CN Responder
wireless.wlan0.DN_responder	Radio 0 DN Responder
wireless.wlan0.channel	Radio 0 Channel
wireless.wlan0.description	Radio 0 Description
wireless.wlan0.golay	Radio 0 Golay Index
wireless.wlan0.polarity	Radio 0 Polarity
kb_name	The hostname of the radio (KB-XX-XX-XX).

4.2.3 Sample Responses:

This sample shows a JSON version of a ***configuration/data*** response using options for pattern matching and the shorter, default version of output.

Request: https://kb-c6-04-0d.local/rest/v002/configuration/data?key=system.kwikbit

```
{
  "configItems" :
  [
    {
      "key" : "system.kwikbit.location",
      "value" : "SW Lab, south wall, upper shelf"
    },
    {
      "key" : "system.kwikbit.description",
      "value" : "TK DN 1"
    }
  ]
  kb_name=KB-C7-00-01
}
```

This sample shows a JSON version of a ***configuration/data*** response using options for pattern matching and extended output (legacy behavior).

Request: https://kb-c6-04-0d.local/rest/v002/configuration/data?key=system.kwikbit\;output=ext

```
{
  "configItems" :
  [
    {
      "key" : "system.kwikbit.location",
      "paramInfo" :
      {
        "default" : "system location not set",
        "display_name" : "Location",
        "max_length" : 130,
        "min_length" : 0,
        "name" : "system.kwikbit.location",
        "type" : "string"
      },
      "value" : "SW Lab, south wall, upper shelf"
    },
    {
```

```
      "key" : "system.kwikbit.description",
      "paramInfo" :
      {
        "default" : "system description not set",
        "display_name" : "Description",
        "max_length" : 130,
        "min_length" : 0,
        "name" : "system.kwikbit.description",
        "type" : "string"
      },
      "value" : "TK Hub ONE"
    }
  ]
  kb_name=KB-C7-00-01
}
```


4.3 Configuration Parameters - Set

To set a configuration parameter, send an HTTP PATCH to the configuration URL.

The data consists of a JSON array of "configItems". Each "configItems" has a "key" and a "value". See Configuration Request Data (PATCH) as an example. Only the changing parameters need to be sent.

Changing list-type parameters with list_action: Using the special feature `list_action` in the URL enables the addition or deletion of individual elements of list-type parameters, without changing the remaining elements in the list. List_action is used with list-type parameters only. See the JSON Doc Samples for more detail about using list_action.

See the examples in ***configuration Request (PATCH)*** below and Configuration Request Data (PATCH).

4.3.1 Request

configuration Request (PATCH)

Name	Description
URL	`https://<server_name>/rest/v002/configuration/data`
URL parameters	`list_action=[del | add]` Deletes or adds individual elements to a list-type parameter without affecting the other elements in the list. Can only be used for list-type parameters. If used with a non list-type parameter, errors are returned. `kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	PATCH
Request data	see Configuration Request Data (PATCH)
Content type	Application/JSON.
Authentication	Required.
URL Examples: To change configuration per the JSON "data" file: https://<server_name>/rest/v002/configuration/data	

To change configuration per the JSON “data” file, using the list_action feature to delete individual elements from list-type parameters:

https://<server_name>/rest/v002/configuration/data?list_action=del

To change configuration per the JSON “data” file, using the list_action feature to add individual elements to list-type parameters:

https://<server_name>/rest/v002/configuration/data?list_action=add

4.3.1.1 Configuration Request Data (PATCH)

4.3.1.1.1 Allowed characters

The set of characters allowed as "value" are:

- Lowercase and uppercase letters: a-z.
- Numbers: 0-9.
- Space (value may need to be quoted if whitespace is present)
- The following characters:

```
` ~ ! @ # $ % ^ * ( ) _ + = { } [ ] | ; : ? . , / -
```

- The following characters are NOT allowed:

```
' " \  < > &
```

4.3.1.1.2 Sample JSON documents

This sample sets a static IP on a radio unit:

```
"configItems" :
[
  {
    "key" : "network.mgmt.proto",
    "value" : "static"
  },
  {
    "key" : "network.mgmt.ipaddr",
    "value" : "10.10.0.71"
  },
]
}
```

These samples show different ways to configure list-type parameters with or without the `list_action=` parameter in the URL to add or delete.

Note: `list_action=` cannot be used in the URL with JSON configuration docs that include any non-list-type parameters.

Submitting the following JSON doc with no `list_action=` in the URL, all the existing CN_responders are replaced with the values shown.

```
{
 "configItems" :
 [
   {
     "key" : "wireless.wlan3.CN_responder",
     "value" : "kb-c6-00-00 kb-c6-00-01" }     <--- note 2 values here
 ]
}
```

The sample below shows different results depending on how the `list_action`= URL parameter is used.

- If submitted without `list_action=add`, any existing CN_responder values would be replaced and the second value would overwrite the first value for the final configuration.
- Submitted with `list_action=add`, both values are added to the existing list of CN_responders.
- With `list_action=del` in the URL, both values are removed from the existing list of CN_responders, without affecting other elements of the list.

```
{
   "configItems" :
   [
     {
       "key" : "wireless.wlan3.CN_responder", <--- note same key used twice
       "value" : "kb-c6-00-00"
     },
     {
       "key" : "wireless.wlan3.CN_responder", <--- note same key used twice
       "value" : "kb-c6-00-01"
     }
   ]
}
```

4.3.2 Response (PATCH)

For list-type parameters, adding an element that already exists to a list does not cause an error. However, deleting an element that does not exist in the list or will return an error. Including `list_action` with non list-type parameters in the URL will also return an error.

configuration Response (PATCH)

Name	Description
Content type	Application/JSON.
Data	Only on field errors; no data on success
Errors	See Configuration Response Error Data (PATCH)

4.3.2.1 Configuration Response Error Data (PATCH)

If invalid data is sent, the HTTP status code returned will be 400:

HTTP/1.1 400 Bad Request

The data returned will include the invalid field and the error message. For example, this sample shows the data returned when the value assigned was outside of the valid integer range for VID.

```
{
  "code" : 602,
  "fieldErrors" :
  [
    {
      "code" : 811,
      "field" : "vlan.mgmt.vid",
      "message" : "value is out of the valid range"
    }
  ],
  "message" : "validation error"
}
```

This sample shows the data returned when the when the value assigned to VID was the wrong type, a string instead of an integer.

```
{
  "code" : 602,
  "fieldErrors" :
  [
    {
      "code" : 803,
      "field" : "vlan.mgmt.vid",
      "message" : "bad input parameter detected"
    }
  ],
  "message" : "validation error"
}
```

This sample shows the data returned when a bad value is found for an element of a list-type parameter. Specifically, this case shows one bad KB MAC name for wireless.wlan0.CN_responder. If there are multiple errors for a single value list, they will all be listed in errorvalues.

```
{
  "code" : 602,
  "fieldErrors" :
  [
    {
      "code" : 810,
      "errorvalues" :
              [
                "KB-C6-02-7"
              ],
              "field" : "wireless.wlan0.CN_responder",
      "message" : "'KB-C6-02-7' is not a valid Kwikbit device name"
    }
  ],
  "message" : "validation error"
}
```

This sample shows the header and data returned for an error due to `list_action` being used for a non list-type parameter.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Error: Non-list type parameter invalid w/list action
Cache-Control: public, must-revalidate, proxy-revalidate
Content-Length: 104
Date: Thu, 07 Dec 2023 20:54:37 GMT
Server: lighttpd/1.4.54

{
   "code" : 601,
   "fieldErrors" : [],
   "message" : "Non-list type parameter invalid w/list action"
}
```

4.4 Restore Factory Settings

Restore configuration to the original factory settings. This function also resets the user password to **admin** for firmware version 2.7.3 or later and reboots the device. (**Note:** For firmware version 1.8.1 and earlier, the password resets to **kwikbit**.)

4.4.1 Request

restore factory defaults Request

Name	Description
URL	`https://<server_name>/rest/v002/configuration/factory_defaults`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	POST
Request data	Requires an empty string. For example, in curl, include the option `-d ""`.
Content type	Not applicable.
Authentication	Required.
Example: https://<server_name>/rest/v002/configuration/factory_defaults	

4.4.2 Response

restore factory defaults Response

Name	Description
Content type	Not applicable.

4.5 Software Upgrade

Software upgrade is performed from the REST API by uploading a software image to the radio. You can also use this API to check the status of the upgrade.

4.5.1 Request

software_upgrade Request

Name	Description
URL	`https://<server_name>/rest/v002/configuration/software_upgrade`
Request header	For upgrade file upload: Content-Type: application/octet-stream For status: None.
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP method	POST
Request data	For upgrade file upload: Upgrade binary is sent as the data. For status: The status command is sent.
Content type	application/octet-stream.
Authentication	Required.
Examples:	Example upgrade command: https://<server_name>/rest/v002/configuration/software_upgrade \ -X POST \ -H "Content-Type:application/octet-stream" \ -H "X-File-Name: < path_and_filename_of_upgrade_binary >" \ --data-binary @"<path_and_filename_of_upgrade_binary>" Example that returns upgrade status: https://<server_name>/rest/v002/configuration/software_upgrade \ -X POST \ -d '{"command": "status"}'

4.5.2 Response

software_upgrade Response

Name	Description
Content type	Application/JSON.
Data	status: One of starting, downloading, upgrading, rebooting, idle, upgrade succeeded, upgrade failed. upgrade-partition: LinuxA or LinuxB. `running-sw-version`: The software version currently running on the radio. `new-sw-version`: The software version that the radio will be upgraded to. `error`: The reason for failure when upgrade status is `upgrade failed`. upgrade-running: yes or no.

4.5.1 Sample Responses:

JSON response from an API call that uploads an upgrade file to the radio:

```
{
  "status":"starting",
  "running-sw-version":"3.2.0",
  "upgrade-running":"yes"
}
```

JSON response from a status call during the upgrade:

```
{
  "status":"upgrading",
  "upgrade-partition":"LinuxA",
  "running-sw-version":"3.2.0",
  "new-sw-version":"3.3.0",
  "upgrade-running":"yes"
}
```

Example JSON response from a status call when the upgrade is unsuccessful:

```
{
  "status":"upgrade failed",
  "running-sw-version":"3.2.0",
  "new-sw-version":"3.3.0",
  "error":"SW image does not support this HW type: 0090",
  "upgrade-running":"no"
}
```

JSON response from a status call after the upgrade has succeeded and the device is preparing to reboot:

```
{
  "status":"rebooting",
  "upgrade-partition":"LinuxA",
  "running-sw-version":"3.2.0",
  "new-sw-version":"3.3.0",
  "upgrade-running":"yes"
}
```

While the device is rebooting, status calls will no longer be able to reach the device. After it has completed rebooting, a status call will return the status of `idle`:

```
{
  "status":"idle",
  "upgrade-running":"no"
}
```

5 Administration

This category administers node behavior.

Administration includes functions related to deployment and operation of the system: to control the basic function of the node, to change the state of the device.

5.1 Download a Diagnostic File

Return diagnostic information from an Altowav unit. The diagnostic information is downloaded into a file named:

<hostname>-diag-<date/time>

Altowav customer support (https://support.altowav.com) will require this file in order to debug problems that may occur with Altowav units.

5.1.1 Request

diagnostic dump Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/diagdump`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	GET
Request data	None.
Content type	Not applicable.
Authentication	Required.
Example: https://<server_name>/rest/v002/admin/diagdump	

5.1.2 Response

diagnostic dump Response

Name	Description
Content type	text/plain
Content-Disposition	attachment; filename=<hostname>-diag-<date/time>

5.1 Add a DN auto-configuration link

Adds a DN auto-configuration link, which is used to establish a link to a remote DN responder without requiring that the DN responder be pre-configured. The remote DN responder must be in a factory default state and cannot have any configuration performed on it prior to responding to a dynamic link request.

Note: Changing the channel, Golay code, or polarity on the DN initiator will cancel any pending dynamic links.

5.1.1 Request

dn_link_add Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/dn_link_add`
URL parameters	`wlan=x` (Required; the value is the radio number from 0 to 3, for example, `wlan=0`. For devices with a single radio, the radio number is 0.) `dn_responder=<MAC_address>` (Required; the value is the MAC address of the remote device, in colon-separated hexadecimal notation.) `kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP method	POST
Request data	Requires an empty string. For example, in curl, include the option `-d ""`.
Content type	Application/ JSON.
Authentication	Required.
Example: https://<server_name>/rest/v002/admin/dn_link_add?wlan=0;dn_responder=70:88:6B:C0:00:01	

5.1.2 Response

dn_link_add Response

Name	Description
Content type	Text/plain
cache-control	The caching policy associated with the certificate.
content-length	The number of bytes of data in the body of the request.
date	The server date and time.
server	The name and version of the web server.

5.1 Delete a pending DN auto-configuration link

Removes an existing DN auto-configuration link. This will only remove the link prior to the DN responder responding to the incoming link request from the DN initiator. Once the link has been established, is API will have no effect.

5.1.1 Request

dn_link_delete Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/dn_link_delete`
URL parameters	`wlan=x` (Required; the value is the radio number from 0 to 3, for example, `wlan=0`. For devices with a single radio, the radio number is 0.) `dn_responder=<MAC_address>` (Required; the value is the MAC address of the remote device, in colon-separated hexadecimal notation.) `kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP method	POST
Request data	Requires an empty string. For example, in curl, include the option `-d ""`.
Content type	Application/ JSON.
Authentication	Required.
Example: https://<server_name>/rest/v002/admin/dn_link_delete?wlan=0;dn_responder=70:88:6B:C0:00:01	

5.1.2 Response

dn_link_delete Response

Name	Description
Content type	Text/plain
cache-control	The caching policy associated with the certificate.
content-length	The number of bytes of data in the body of the request.
date	The server date and time.
server	The name and version of the web server.

5.2 Link Bump

Link Bump temporarily disconnects the link in order to force a re-evaluation of the TX beams, RX beams and TX power. The link reconnects after a short period of time.

The link to bump is specified with a pair of radio MAC addresses, the `local_mac` and `remote_mac`.

- **`local_mac`** is the MAC address of one of the radios in the local device.
- **`remote_mac`** is the MAC address of the radio at the remote end of the airlink.

5.2.1 Request

link_bump Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/link_bump`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters), "
HTTP method	POST
Request data	`{"local_mac" : "xx:xx:xx:xx:xx:xx", "remote_mac" : "xx:xx:xx:xx:xx:xx"}` (required).
Content type	Not applicable.
Authentication	Required.
Example: https://<server_name>/rest/v002/admin/link_bump { "local_mac" : "04:ce:14:0e:a5:00", "remote_mac" : "70:88:6b:c6:04:48" }	

5.2.2 Response

link_bump Response

Name	Description
Data	No data returned.

5.3 Link Scan

Link Scan performs a relative PBF (periodic beam forming) scan on the specified link. A pair of radio MAC addresses, the `local_mac` and `remote_mac`, specifies which link to re-beamform.

- **`local_mac`** is the MAC address of one of the radios in the local device.
- **`remote_mac`** is the MAC address of the radio at the remote end of the airlink.

The relative PBF scan is performed first in one direction (from local to remote), then separately in the other direction (remote to local). The system schedules each scan at a precisely synchronized time. After the scans complete, the radios return scan status reports. Most scans complete within 30 seconds, but if multiple scans are pending it may take 60 seconds or longer.

Access status information on link scans through *admin/link_scan_status*. The two "token" values returned from *link_scan* are passed with the *link_scan_status* call to specify which link scan to report. For example, *link_scan* returns token values of 7 and 8. Those tokens are then specified in *link_scan_status ?token=7,8* to return the status information from the *link scan*.

To minimize potential interference, avoid performing link scans simultaneously on nearby nodes. Use *admin/link_scan_status* to determine whether the returned information for the local device includes `"status": "COMPLETE"`.

Note: GPS synchronization must be enabled to perform a link_scan, (GPS sync config keyword: wireless.device.gps_sync).

5.3.1 Request

link_scan Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/link_scan`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|json]` (see Global query parameters)
HTTP method	POST
Request data	`{"local_mac" : "xx:xx:xx:xx:xx:xx", "remote_mac" : "xx:xx:xx:xx:xx:xx"}` (required).
Content type	Application/JSON.
Authentication	Required.
Example: https://<server_name>/rest/v002/admin/link_scan { "local_mac" : "04:ce:14:0e:a5:00", "remote_mac" : "70:88:6b:c6:04:48" }	

5.3.2 Response

link_scan Response

Name	Description
Content type	Application/JSON or text.
kb_name	The hostname of the radio (KB-XX-XX-XX).
scanResp An array of two scan responses.	
message	Diagnostic message returned from internal Scan application.
success	`(true | false)` Indicates whether or not the scan is scheduled.
token	Scan ID.

5.3.3 Sample Response:

This sample shows a JSON version of a ***link_scan*** response including the scan schedule and scan ID information.

```
{
 "scanResp": [
  {
    "message": "Scheduled PBF scan 1"
    "success": true,
    "token": 1
  },
  {
    "message": "Scheduled PBF scan 2"
    "success": true,
    "token": 2
  }  ]
}
```

This sample shows a text version of a ***link_scan*** response.

```
Scheduled PBF scan 3
success: true
token: 3
Scheduled PBF scan 4
success: true
token: 4
```

5.4 Link Scan Status

Return the status of a previous relative PBF (periodic beam forming) scan. Specify which previous scan by passing the pair of tokens returned from a previously issued *admin/link_scan*.

A relative PBF scan is scheduled for a precisely synchronized time. The status returned by this REST call changes as the scheduled time expires and scan results are returned by the radios. For example, the JSON field `nresponsesWaiting` appears only when scan completion is pending. In addition, while the scan is scheduled and pending, the `responses` structure is empty { }. Data populates the `responses` structures as scan responses arrive.

DN-DN links return one `responses` structure. DN-CN links return two `responses` structures.

The PBF scan is complete when all `responses` structures include `"status": "COMPLETE"`. However, a `COMPLETE` status for the local device indicates the link scan is sufficiently complete to run another without interference.

5.4.1 Request

link_scan_status Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/link_scan_status`
URL parameters	`token=n,n` (Required.) Comma-separated "token" values from *link_scan.* `kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|json]` (see Global query parameters)
HTTP method	GET
Request data	None.
Content type	Application/ JSON.
Authentication	Required.
Example: https://<server_name>/rest/v002/admin/link_scan_status?token=3,4	

5.4.2 Response

link_scan_status Response

Name	Description
Content type	Application/ JSON or text.
kb_name	The hostname of the radio (KB-XX-XX-XX).
scans An array of two scan responses for the tokens specified.	
nresponsesWaiting	Number of scan responses pending.
scanType	Indicates relative PBF scan.
txNode	Radio MAC address of transmitting device.

responses An array of responses for completed link scans, arranged by the radio MAC address of the originating radio.	
newBeam	Beam index after link scan is complete.
oldBeam	Beam index before the link scan.
status	Completion status for the relative PBF initiated by link scan.
token	Token identifying the scan response.

5.4.3 Sample Responses:

This JSON sample shows a link_scan_status response when the scan is scheduled and results are pending. Note Tokens 13 and 14, as shown in this sample's URL, specify the scan results to list.

https://<server_name>/rest/v002/admin/link_scan_status?token=13,14.


```
{
 "scans": {
    "13": {
      "nresponsesWaiting": 2,
      "responses": {},
      "scanType": "PBF",
      "txNode": "04:ce:14:0e:a5:00"
    },
    "14": {
      "nresponsesWaiting": 2,
      "responses": {},
      "scanType": "PBF",
      "txNode": "70:88:6b:c6:04:48"
    }
  }
}
```


For the same URL as above, this sample shows a response after relative PBF completion.


```
{
  "scans": {
    "13": {
      "responses": {
        "04:ce:14:0e:a5:00": {
          "newBeam": 48,
          "oldBeam": 48,
          "radioMac": "04:ce:14:0e:a5:00",
          "status": "COMPLETE",
          "token": 13,
        },
        "70:88:6b:c6:04:48":
          "newBeam": 62,
          "oldBeam": 63,
          "radioMac": "70:88:6b:c6:04:48",
          "status": "COMPLETE",
          "token": 13
        }
      },
      "scanType": "PBF",
      "txNode": "04:ce:14:0e:a5:00"
    },
    "14": {
```

```
      "responses": {
        "04:ce:14:0e:a5:00": {
          "newBeam": 45,
          "oldBeam": 45,
          "radioMac": "04:ce:14:0e:a5:00",
          "status": "COMPLETE",
          "token": 14
        },
        "70:88:6b:c6:04:48": {
          "newBeam": 63,
          "oldBeam": 62,
          "radioMac": "70:88:6b:c6:04:48",
          "status": "COMPLETE",
          "token": 14,
        }
      },
      "scanType": "PBF",
      "txNode": "70:88:6b:c6:04:48"
    }
  }
}
```


This sample shows the response for a completed link scan, when text output is selected.

https://<server_name>/rest/v002/admin/link_scan_status?token=13,14?output=text

```
scanStatus.scans[13] txNode:04:ce:14:0e:a5:00
  scanType:PBF
  responses: nodename:70:88:6b:c6:04:48
    status: COMPLETE
    token: 13
    radioMac: 70:88:6b:c6:04:48
    newBeam: 62
    oldBeam: 63
  responses: nodename:04:ce:14:0e:a5:00
    status: COMPLETE
    token: 13
    radioMac: 04:ce:14:0e:a5:00
    newBeam: 48
    oldBeam: 48

scanStatus.scans[14] txNode:70:88:6b:c6:04:48
  scanType:PBF
  responses: nodename:04:ce:14:0e:a5:00
    status: COMPLETE
    token: 14
    radioMac: 04:ce:14:0e:a5:00
    newBeam: 45
    oldBeam: 45
  responses: nodename:70:88:6b:c6:04:48
    status: COMPLETE
    token: 14
    radioMac: 70:88:6b:c6:04:48
    newBeam: 63
    oldBeam: 62
```

5.5 Locate Node

Blinks the device's lights in a distinctive flashing pattern so that a particular unit can be identified.

5.5.1 Request

locate_device Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/locate_device`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP method	POST
Request data	`{ "action": "on" }` or `{ "action": "off" }` (If no parameter is specified, the action defaults to on.) Also requires an empty string. For example, in curl, include the option `-d ""`.
Content type	application/ JSON.
Authentication	Required.
Examples: https://<server_name>/rest/v002/admin/locate_device https://<server_name>/rest/v002/admin/locate_device { "action": "off"}	

5.5.2 Response

locate_device Response

Name	Description
Content type	Not applicable - no data returned.
Data	No data returned.

5.6 Power Cycle Ethernet Interface

The Power Cycle API is only available on the K60DN. It cycles the power on the specified Ethernet interface such that the PoE output on the interface goes off and then back on. Optionally provide the amount of time the interface stays off. The default off time is 1 second and the valid range is from 0.2 to 10.0 seconds.

The following is a list of the types of possible command errors:

- The command is missing a specified interface name.
- The specified interface is invalid.
- The specified interface does not have a PoE output.
- The specified interface has a PoE output but the PoE output is disabled.
- The specified off time is not a floating-point or integer number.
- The specified off time is out of the valid range.

Network Interfaces with PoE Outputs

Unit Type	PoE Output Interfaces
K60DN	eth1, eth2, eth3, eth4
D423	eth2, eth3

5.6.1 Request

power_cycle Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/power_c`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP method	POST
Request data	`{ "interface": "eth2" }` (Required; see the valid interface list above.) `{ "off_time": <value> }` (Optional; int/float seconds off time.)
Content type	Application/JSON.
Authentication	Required.
Examples: https://<server_name>/rest/v002/admin/power_cycle { "interface": "eth2"} https://<server_name>/rest/v002/admin/power_cycle { "interface": "eth3", "off_time": 2.5}	

5.6.2 Response

power_ cycle Response

Name	Description
Content type	Not applicable.
Data	No data returned - Note that the header is returned before the power cycle.

5.7 Reboot Node

Reboots the node immediately.

5.7.1 Request

reboot Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/reboot`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP method	POST
Request data	Requires an empty string. For example, in curl, include the option `-d ""`.
Content type	Not applicable.
Authentication	Required.
Examples: https://<server_name>/rest/v002/admin/reboot	

5.7.2 Response

reboot Response

Name	Description
Content type	Not applicable.
Data	No data returned – Note: The header will be returned before reboot takes effect.

5.8 Topology Scan

Topology scan performs a scan on a local device to discover nearby radios without knowing their MAC addresses or GPS locations. A local MAC address is used to identify the radio on the local device that will be used for the scan.

Note: For AltoPlex devices that have only one radio, the MAC address for the wlan0/Radio 0 interface should be used. This is also the MAC address on the device label.

The topology scan will return values for up to 20 responders. Additional responders after the first 20 will be ignored. The scan will return the following information:

- MAC address of the responding device.
- Signal-to-Noise ratio (SNR) for both the local and remote radios:
- Beam angles for both the local and remote radios.

The topology scan will timeout if it hasn't completed within 15 second. It takes approximately 11 seconds for the scan to complete, and an additional two to three seconds for the initiator to be removed from each responder's station list. During this time, additional scans cannot be performed, including a link scan.

The topology scan returns a token value. Use topology scan status to view the status of the topology scan; if the topology scan is complete, topology scan status will return the results of the scan.

Note: GPS synchronization must be enabled to perform a topology scan.

5.8.1 Request

topology_scan Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/topology_scan`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|json]` (see Global query parameters)
HTTP method	POST
Request data	`"local_mac" : "<colon_hexadecimal_mac_address>"` (required).
Content type	Application/JSON.
Authentication	Required.
Example: https://<server_name>/rest/v002/admin/topology_scan '{ "local_mac" : "70:88:6b:c7:00:00"}'	

5.8.2 Response

topology_scan Response

Name	Description
Content type	Application/JSON or text.
kb_name	The hostname of the device (KB-XX-XX-XX).
message	Diagnostic message returned from internal scan application.
success	`(true | false)` Indicates whether or not the scan is scheduled.
token	The token ID of the scan, to be used by topology scan status to view the status of the scan.

5.8.3 Sample Response:

This sample shows a JSON version of a topology scan response, including the token to be used by topology scan status.

```
{
 "scanResp": [
  {
    "message": "Scheduled topology scan 1"
    "success": true,
    "token": 1
  }
}
```

This sample shows a text version of a topology scan response.

```
Scheduled topology scan 1
success: true
token: 1
```

5.9 Topology Scan Status

Returns the status of a previous topology scan. If the topology scan is complete, returns the results of the scan. When a topology scan is scheduled, it returns a token, which is specified as part of the topology scan status API.

A topology scan is scheduled for a precisely synchronized time. The status returned by topology scan status changes as the scan completes and scan results are returned by the radios. For example, the JSON field `nresponsesWaiting` appears only when scan completion is pending. In addition, while the scan is scheduled and pending, the `responses` structure is empty { }. Data populates the `responses` structure as scan responses arrive.

The topology scan is complete when the `responses` structure lists the status as `COMPLETE`. The `COMPLETE` status indicates that another scan (including a link scan) can be run without interference.

5.9.1 Request

link_scan_status Request

Name	Description
URL	`https://<server_name>/rest/v002/admin/topology_scan_status`
URL parameters	`token=n` (Required.) Returned from topology scan. `kb_name=KB-XX-XX-XX` (see Global query parameters) `output=[text|json]` (see Global query parameters)
HTTP method	GET
Request data	None.
Content type	Application/ JSON.
Authentication	Required.
Example: https://<server_name>/rest/v002/admin/topology_scan_status?token=1	

5.9.2 Response

link_scan_status Response

Name	Description
Content type	Application/ JSON or text.
kb_name	The hostname of the device (KB-XX-XX-XX).
nresponsesWaiting	Indicates the scan response is pending.
responses An array of responses for completed topology scans. If the scan response is pending, this array will be empty.	
radioMAC	The MAC address of the local radio.
status	Completion status for the scan.
token	The token as entered in the URL parameter.

topoResps	An array of responses to the topology scan. Each response is identified by a number and provides the following information for both the local radio and the remote radio.
local radio	Array of date about the beam formed from the local radio.
bestSnr	The SNR value, in dB, of the beam with the best link quality.
rxBeamAzimuth	The azimuth angle of the rx beam.
rxBeamElevation	The elevation angle of the rx beam.
txBeamAzimuth	The azimuth angle of the tx beam.
txBeamElevation	The elevation angle of the tx beam.
remote radio	Array of data about the beam formed from the remote radio.
bestSnr	The SNR value, in dB, of the beam with the best link quality.
rxBeamAzimuth	The azimuth angle of the rx beam.
rxBeamElevation	The elevation angle of the rx beam.
txBeamAzimuth	The azimuth angle of the tx beam.
txBeamElevation	The elevation angle of the tx beam.
respMAC	The MAC address of the responding radio.
scanType	Indicates `TOPO` (topology) scan.
txNode	Radio MAC address of initiating device.

5.9.3 Sample Responses:

This JSON sample shows a topology_scan_status response when the scan is scheduled and results are pending.

https://<server_name>/rest/v002/admin/topology_scan_status?token=1.


```
{
  "kb_name": "KB-C7-00-00",
  "scans": {
    "1": {
      "nResponsesWaiting": 1,
      "responses": {},
      "scanType": "TOPO",
      "txNode": "70:88:6b:c7:00:00"
    }
  }
}
```

This JSON sample shows the topology scan status response after the scan has completed. The response lists two responding radios.

https://<server_name>/rest/v002/admin/topology_scan_status?token=1.


```
{
  "kb_name": "KB-C7-00-00",
  "scans": {
    "1": {
      "responses":
        "04:ce:14:fe:b3:27": {
          "radioMac": "70:88:6b:c7:00:00",
          "status": "COMPLETE",
          "token": 1,
          "topoResps": {
            "0": {
              "local_radio": {
                "bestSnr": 4.75,
                "rxBeamAzimuth": 45,
                "rxBeamElevation": 0,
                "txBeamAzimuth": 1,
                "txBeamElevation": 0
              },
              "remote_radio": {
                "bestSnr": 4,
                "rxBeamAzimuth": -43,
                "rxBeamElevation": 0,
                "txBeamAzimuth": -18,
                "txBeamElevation": 0
              },
              "respMac": "70:88:6b:c7:00:01"
            },
            "1": {
              "local_radio": {
                "bestSnr": 8.25,
                "rxBeamAzimuth": 0,
                "rxBeamElevation": 0,
                "txBeamAzimuth": 0,
                "txBeamElevation": 0
              },
              "remote_radio": {
                "bestSnr": 3,
                "rxBeamAzimuth": 45,
                "rxBeamElevation": 0,
                "txBeamAzimuth": -30,
                "txBeamElevation": 0
              },
              "respMac": "70:88:6b:c7:00:02"
            }
          }
        }
      },
      "scanType": "TOPO",
      "txNode": "70:88:6b:c7:00:00"
    }
  }
}
```

6 Performance Management

This category measures node performance.

6.1 Get Node Statistics

Returns the interface statistics for the unit.

6.1.1 Request

statistics Request

Name	Description
URL	`https://<server_name>/rest/v002/performance/stats`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP method	GET
Request data	None.
content type	Not applicable.
Authentication	Not required.

6.1.2 Response

Returns an array "`interface_stats`" for the interfaces which includes the elements shown in the table.

statistics Response

Name	Description
Content type	Application/JSON.
interface	The name of the interface.
traffic counters	JSON object with the following items.
rx_bytes	Count of received octets.
rx_crc_errors	Count of received packets with CRC errors.
rx_dropped	Count of received packets dropped.
rx_packets	Count of received ethernet frames.
tx_bytes	Count of transmitted octets.
tx_dropped	Count of transmitted packets dropped.
tx_errors	Count of transmitted packet errors.
tx_packets	Count of transmitted ethernet frames.

Kb_name	Hostname of the radio (KB-XX-XX-XX).

7 Security Management

This category manages security settings, including changing passwords and setting authorized organization name for security certificate.

Note: NoFor firmware version 2.7.3 and later, the default username is **admin** and the default password is **admin**. For firmware version 1.8.1 and earlier, the default username and password are **kwikbit**. (During a mitigation period, the K60DN and K60CN1 devices can be reached with either username/password.)

7.1 Change the user password

Changes the password for the “admin” user.

Allowed characters are the same as configuration data (see Allowed characters), except:

- Additionally, the following characters are allowed:

```
< > &
```

- Whitespace is not allowed.
- Only the following characters are not allowed:

```
' " \
```

7.1.1 Request

change password Request

Name	Description
URL	`https://<server_name>/rest/v002/security/password`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	PATCH
Request data	`{"new_password": "entered_new_password"}`
Content type	Application/JSON.
Authentication	Required.

7.1.2 Response

password Response

Name	Description
Content type	Not applicable.
Data	None.

7.2 Install a Client CA certificate

Installs a CA certificate on the device that is used to authenticate incoming client sessions. Once a client CA Certificate is installed, REST API calls can be authenticated by passing a client certificate signed by that CA.

The client CA certificate file must be in PEM format and must contain the following, in this order:

1. The client CA certificate.
2. Any associated intermediate CA certificates.
    1. **Note:** A maximum of two intermediate CA certificates are allowed.
3. The Root CA certificate.

The filename for the client CA certificate should:

- Begin with an alphanumeric character or an underscore.
- Contain only alphanumeric characters, underscores, hyphens, and periods.

If the filename does not conform to these requirements, this API will silently change the filename to `custom_certificate`.

To remove custom client CA certificates and reset the device to the factory default certificate, submit with the URL parameter `reset=true`.

Important note about the device data and time

By default, when the device is rebooted (or is booted for the first time), it will use the date and time of **January 1, 2022, 00:00:00 GMT**. Once the device is online and connects to an upstream NTP server, the date and time will be updated to reflect the current date and time.

However, for a client to be authenticated prior to the device being set to the current date and time, the CA certificate must have its notBefore value set to a date and time prior to January 1, 2022, 00:00:00 GMT. If not, you will receive an error message similar to `certificate is not yet valid`.

7.2.1 Request

install_client_ca_certificate Request

Name	Description
URL	`https://<server_name>/rest/v002/security/install_client_ca_certificate`
URL parameters	`reset=true` (When used to remove custom client CA certificates) `kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	POST (used to install or reset), GET
Request header	When installing a certificate: • Content-Type: application/octet-stream • X-File-Name: <CA_cert_filename>
Request data	When installing a certificate: • Client CA certificate as octet stream with filename. When removing a certificate: • Requires an empty string. For example, in curl, include the option `-d ""`.
Content type	application/octet-stream
Authentication	Required.
https://<server_name>/rest/v002/security/install_client_ca_certificate \ -X POST \ -H "Content-Type:application/octet-stream" \ -H "X-File-Name: <CA_cert_filename>" \ --data-binary @"<CA_cert_filename>"	

7.2.2 Response

install_client_ca_certificate Response

Name	Description
content-type	text/plain
cache-control	The caching policy associated with the certificate.
accept-ranges	Bytes
content-length	The number of bytes of data in the returned document.
date	The server date and time.
server	The name and version of the web server.
Data	Output displaying the parameters contained in the certificate.

7.2.3 Examples

Install a new client CA certificate:

```
$ curl -k -u admin:admin  \
https://<server_name>/rest/v002/security/install_client_ca_certificate \
-X POST \
-H "Content-Type:application/octet-stream" \
-H "X-File-Name: <CA_cert_filename>" \
--data-binary @"<CA_cert_filename>"

content-type: text/plain
cache-control: public, must-revalidate, proxy-revalidate
accept-ranges: bytes
content-length: 447
date: Fri, 04 Oct 2024 21:10:08 GMT
server: lighttpd/1.4.73
```

View information about the installed client CA certificate, using a client certificate for authentication:

```
curl -k -u admin:admin  \
https://<server_name>/rest/v002/security/install_client_ca_certificate \
HTTP/2 200

kb_certificate: in-use
content-type: text/plain
cache-control: public, must-revalidate, proxy-revalidate
accept-ranges: bytes
content-length: 447
date: Sat, 05 Oct 2024 01:59:54 GMT
server: lighttpd/1.4.73

using custom client CA file: <CA_cert_filename>
subject=CN=<Common_name>, O=<Org_name>, OU=<Org_Unit>
issuer=CN=<Common_issuer_name>, O=<Org_name>, ST=<State>, C=<Country>
notBefore=Jan  1 06:01:01 2017 GMT
notAfter=Jan  1 06:01:01 2057 GMT
---
subject=CN=<Common_name>, O=<Org_name>, OU=<Org_Unit>
issuer=CN=<Common_issuer_name>, O=<Org_name>, ST=<State>, C=<Country>
notBefore=Jan  1 06:01:01 2017 GMT
notAfter=Jan  1 06:01:01 2057 GMT
---
```

Remove custom client CA certificates and revert to the factory-installed certificate (named `Root_CA.cert` with `OU=Altowav_device_<device_name>`):

```
$ curl -k -u admin:admin  \
https:// <server_name>/rest/v002/security/install_client_ca_certificate?reset=true \
-X POST \
-d ""

HTTP/2 200
kb_certificate: in-use
content-type: text/plain
cache-control: public, must-revalidate, proxy-revalidate
content-length: 1
date: Sat, 05 Oct 2024 18:28:07 GMT
server: lighttpd/1.4.73
```

7.3 Download a certificate signing request from the device

Obtains a certificate signing request (CSR) from the device, which can be forwarded to a Certificate Authority (CA) to create a server certificate for the device.

The CSR contains the following information:

- The subject line:

Subject: CN = <server_name>, O = Altowav, OU = Altowav_device

- Requested Extensions:

X509v3 Subject Alternative Name:

DNS: <sever_name>, DNS: <server_name>.local

X509v3 Basic Constraints:

CA:FALSE

X509v3 Key Usage:

Digital Signature, Key Encipherment, Data Encipherment

In these examples, `<server_name>` is the hostname of the Altowav device (KB-XX-XX-XX).

Important: Because the CSR uses the hostname of the device for the DNS entries in the Subject Alternative Name (SAN), a certificate based on this CSR will authenticate by using the device's hostname. To establish an authenticated connection to the device, users will have to connect to the device by using its hostname, and your network will need to be configured so that the hostname resolves to the correct IP address.

7.3.1 Request

get_csr Request

Name	Description
URL	`https://<server_name>/rest/v002/security/get_csr`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	GET
Request data	None
Content type	Application/JSON.
Authentication	Required.

7.3.2 Response

get_csr Request

get_csr ResponseName	Description
content-type	text/plain
cache-control	public, must-revalidate, proxy-revalidate
accept-ranges	bytes
content-length	The number of bytes of data in the returned document.
date	The server date and time.
server	The name and version of the web server.
Data	The text of the CSR.

7.4 Install a server CA certificate

Installs a device server certificate along with the associated CA certificate chain required for the device to act as an HTTPS server. Once the server certificate and associated CA certificate chain are uploaded, the device will be recognized by any TLS client associated with the same CA.

The server certificate file must be in PEM format and must contain the following, in this order:

1. The server certificate.
2. The signing CA certificate.
3. Any associated intermediate CA certificates.
    2. **Note:** A maximum of two intermediate CA certificates are allowed.
4. The Root CA certificate.

The filename for the server certificate should:

- Begin with an alphanumeric character or an underscore.
- Contain only alphanumeric characters, underscores, hyphens, and periods.

If the filename does not conform to these requirements, this API will silently change the filename to `custom_certificate`.

7.4.1 Obtain a server CA certificate

To obtain a server CA certificate, download a certificate signing request (CSR) from the device and forward the CSR to a Certificate Authority (CA) to create a server certificate for the device.

Important note about the device data and time

By default, when the device is rebooted (or is booted for the first time), it will use the date and time of **January 1, 2022, 00:00:00 GMT**. Once the device is online and connects to an upstream NTP server, the date and time will be updated to reflect the current date and time.

However, for server authentication to work prior to the device being set to the current date and time, the Certificate Authority (CA) that creates the server certificate must have its notBefore value set to a date and time prior to January 1, 2022, 00:00:00 GMT. If not, you will receive an error message similar to `certificate is not yet valid`.

Once the certificate has been obtained from the CA, install it on the device by using the `install_certificate` API. To remove the server CA certificate and reset the device to the factory default certificate, submit with the URL parameter `reset=true`.

7.4.2 Request

install_certificate Request

Name	Description
URL	`https://<server_name>/rest/v002/security/install_certificate`
URL parameters	`reset=true` (when used to delete the installed client CA certificate) `kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	POST (used to install or reset), GET

Request header	When installing a certificate: • Content-Type: application/octet-stream • X-File-Name: < server_cert_filename>
Request data	When installing a certificate: • Server and CA certificate document as octet stream with filename. When removing a certificate: • Requires an empty string. For example, in curl, include the option `-d ""`.
Content type	application/octet-stream
Authentication	Required.
Example: https://<server_name>/rest/v002/security/install_certificate \ -X POST \ -H "Content-Type:application/octet-stream" \ -H "X-File-Name: <server_cert_filename>" \ --data-binary @"<server_cert_filename>"	

7.4.3 Response

install_certificate Response

Name	Description
content-type	text/plain
cache-control	The caching policy associated with the certificate.
accept-ranges	bytes
content-length	The number of bytes of data in the body of the request.
date	The server date and time.
server	The name and version of the web server.
Data	Output displaying the parameters contained in the certificate.

7.4.1 Examples

Install a new server certificate:

```
$ curl -k -u admin:admin  \
https://<server_name>/rest/v002/security/install_certificate \
-X POST \
-H "Content-Type:application/octet-stream" \
-H "X-File-Name: <filename>" \
--data-binary @"<filename>"

content-type: text/plain
cache-control: public, must-revalidate, proxy-revalidate
accept-ranges: bytes
content-length: 595
date: Fri, 04 Oct 2024 21:10:08 GMT
```

```
server: lighttpd/1.4.73
```

View information about the installed server certificate:

```
curl -k -u admin:admin  \
https://<server_name>/rest/v002/security/install_certificate

HTTP/2 200
content-type: text/plain
cache-control: public, must-revalidate, proxy-revalidate
accept-ranges: bytes
content-length: 595
date: Sat, 05 Oct 2024 01:59:54 GMT
server: lighttpd/1.4.73

using custom certificate file
---
subject=CN=<device_hostname>, O=<Org_name>,
issuer=CN=<Common_issuer_name>, O=<Org_name>, ST=<State>, C=<Country>
notBefore=Jan  1 00:01:01 2017 GMT
notAfter=Oct  3 21:45:22 2049 GMT
---
subject=CN=<Common_name>, O=<Org_name>, OU=<Org_Unit>
issuer=CN=<Common_issuer_name>, O=<Org_name>, ST=<State>, C=<Country>
notBefore=Jan  1 00:01:01 2017 GMT
notAfter=Sep 26 13:51:35 2049 GMT
---
subject=CN=<Common_name>, O=<Org_name>, OU=<Org_Unit>
issuer=CN=<Common_issuer_name>, O=<Org_name>, ST=<State>, C=<Country>
notBefore=Jan  1 06:01:01 2017 GMT
notAfter=Jan  1 06:01:01 2057 GMT
---
```

Remove custom server certificates and revert to the factory-installed certificate (named `<device_name>`, for example, `KB-C0-00-01`):

```
$ curl -k -u admin:admin  \
https://<server_name>/rest/v002/security/install_client_ca_certificate?reset=true \
-X POST \
-d ""

HTTP/2 200
kb_certificate: in-use
content-type: text/plain
cache-control: public, must-revalidate, proxy-revalidate
content-length: 1
date: Sat, 05 Oct 2024 18:28:07 GMT
server: lighttpd/1.4.73
```

7.5 Test client and server authentication

Use the `altowav` endpoint as a lightweight mechanism to verify client and server authentication. The endpoint returns a text document containing only the text `ALTOWAV`.

7.5.1 Request

altowav Request

Name	Description
URL	`https://<server_name>/rest/v002/security/altowav`
URL parameters	`kb_name=KB-XX-XX-XX` (see Global query parameters)
HTTP methods	GET
Request data	`None`
Content type	Application/JSON.
Authentication	Required.

7.5.1 Response

altowav Response

Name	Description
Kb_certificate	in-use
content-type	application/plain
cache-control	The caching policy associated with the certificate.
accept-ranges	bytes
content-length	1067
date	The server date and time.
server	The name and version of the web server.
Data	Text file containing only the text `ALTOWAV`.