

# API Client

## Overview

cnMaestro supports a RESTful API as part of its On-Premises deployment. This API allows customers to read data and perform operations programmatically using their own client applications. The API is supported over HTTPS, and messages are exchanged in JSON format. Modern programming languages have rich support for RESTful interfaces.

## API Clients

API Clients are external applications that access the RESTful API over HTTPS using OAuth 2.0 Authentication. They require a **Client ID** and **Client Secret** for access, both of which are detailed later in this section. They are configured by navigating to **Services > API Clients**.



Application Name	Application Description	Client Id	Swagger	Actions
Test API 20 Jan	Test NBAPI	rNcWbrNkGsp2lg	Swagger	  

# RESTful API Specification

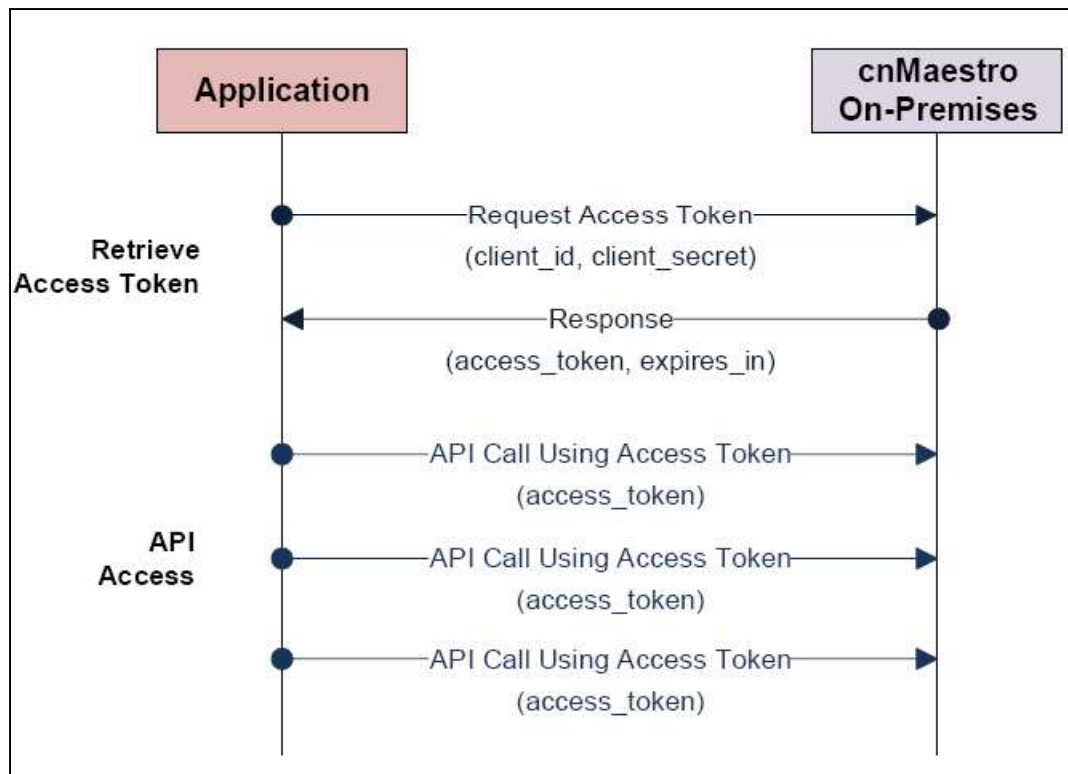


## NOTE:

The cnMaestro API is changing to v2 in the 3.0 Release. v1 continues to be supported through 3.1.x. Cambium Networks recommends using v2 on any new API applications and updating from v1 as soon as possible. The changes to the v2 API are limited and described later in this chapter.

## Authentication

API Authentication uses OAuth2. The client retrieves an Access Token to start the session. It then sends API requests until the Access Token times out, at which point the token can be regenerated.



## Establishing Session

A session is created by sending the Client ID and Client Secret to the cnMaestro server. These are generated in the cnMaestro UI and stored with the application. The Client ID defines the cnMaestro account and application, and the Client Secret is a private string mapped to the specific application. The Client Secret should be stored securely.

If the session is established successfully, an Access Token is returned along with an expiration string. The Access Token is used to authenticate the session. The expiration is the interval, in seconds, in which the Access Token remains valid. If the Access Token expires, a new session needs to be created.

## API Access

With the Access Token, the application can make cnMaestro API calls. The token is sent in an Authentication header on each API request. Details are provided later in this document.

## Session Expiration

If a token expires, an expiration error message is returned to the client. The client can then generate a new token using the Client ID and Client Secret. Tokens will expire immediately if the Client API account that created them is deleted. The default expiration time for a token is 3600 seconds (1 hour). This is configurable in the UI.

## Concurrent Access

Each client supports a single Access Token or multiple Access Tokens. Multiple Access Tokens allows concurrent access.

### Single Access Token

If only one Access Token is enabled at a time, whenever a new Access Token is generated from the Client ID and Client Secret, the previous Token will immediately expire.

### Multiple Access Tokens

If multiple access tokens are supported, then many clients can concurrently access the API. If another Access Token is created, the previous will remain valid until their original expiration.

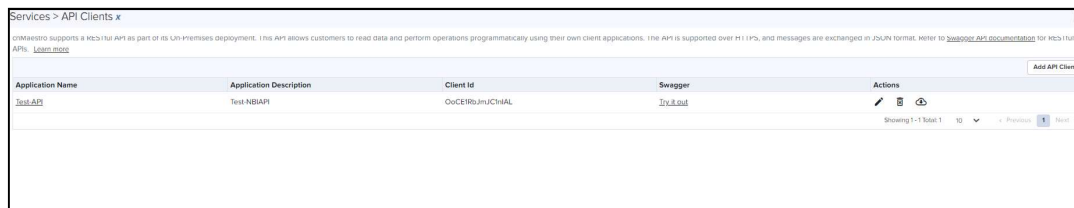
## Rate Limiting

Request Rate Limiting is not enabled in the On-Premises version of cnMaestro. It is up to the application owner to make sure requests do not overtax the system.

# Swagger API

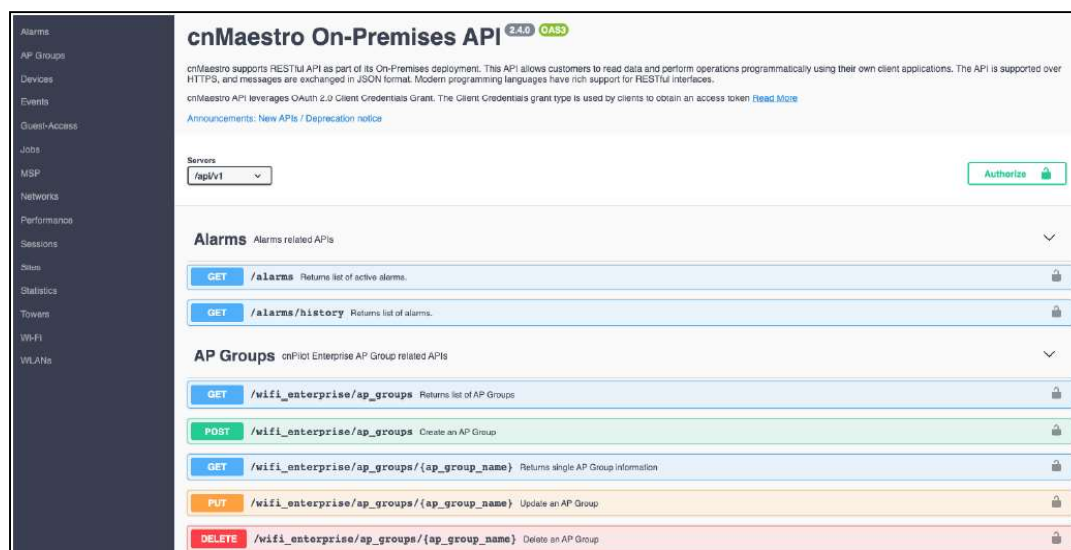
## Introduction

The RESTful API documentation is now supported through Swagger. Swagger UI allows visualization and interaction with the API resources. You can access Swagger by navigating to **Services > API Clients** grid and clicking on **Swagger API documentation**.



Application Name	Application Description	Client Id	Swagger	Actions
<a href="#">Test-NB-API</a>	Test-NB-API	00CE16bJnUChAL	<a href="#">Try it out</a>	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Add API Client</a>

## Sample Swagger UI Screenshot

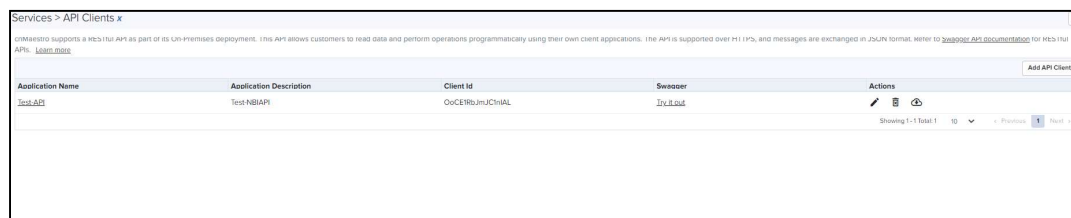


## Client ID and Client Secret Generation

### cnMaestro User Interface

The Client Id and Client Secret are created in the cnMaestro UI by navigating to **Services > API Client**. Each client application should be added as an API Client.

#### Step 1: Navigate to Services > API Clients

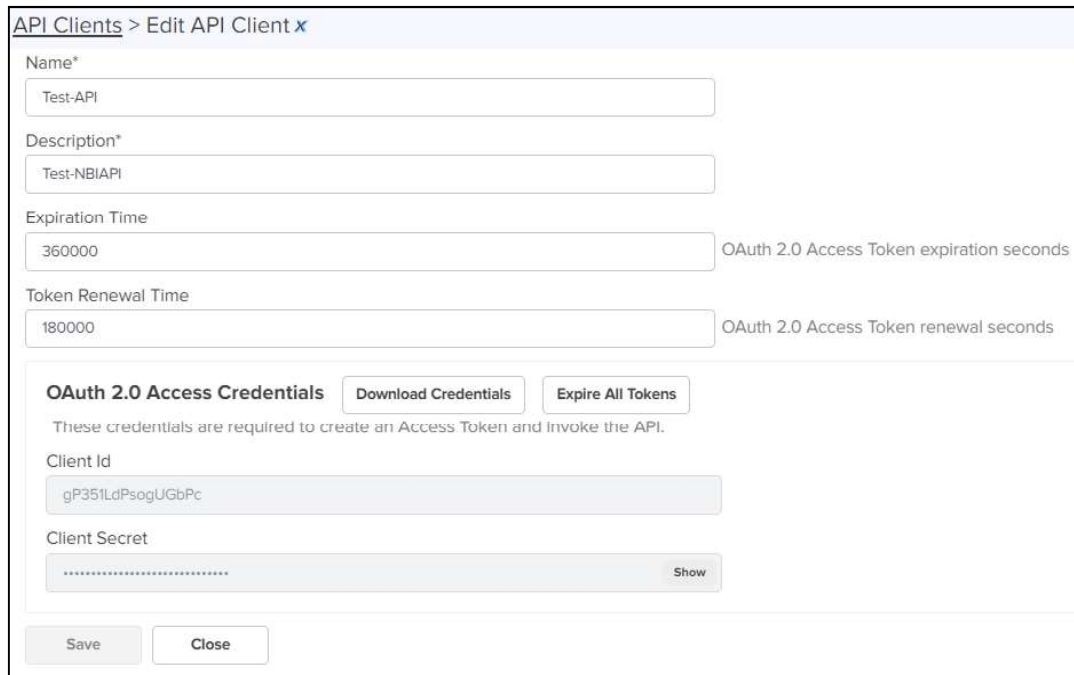


#### Step 2: Create a New API Client

Select **Add API Client** to add a client, then fill in the requested details, and click **Save**.

### Step 3: Download the Client ID and Client Secret

Download and store the Client ID and Client Secret by clicking **Download Credentials**. Both are required to create an API session.



API Clients > Edit API Client x

Name\*  
Test-API

Description\*  
Test-NBIAPI

Expiration Time  
360000 OAuth 2.0 Access Token expiration seconds

Token Renewal Time  
180000 OAuth 2.0 Access Token renewal seconds

**OAuth 2.0 Access Credentials** Download Credentials Expire All Tokens

These credentials are required to create an Access Token and invoke the API.

Client Id  
gP351LdPsogUGbPc

Client Secret  
\*\*\*\*\* Show

Save Close

## API Session

### Introduction

The cnMaestro API leverages the Client Credentials section of the [OAuth 2.0 Authorization Framework \(RFC 6749\)](#). An API session can be created using any modern programming language. The examples below highlight how messages are encoded and responses returned.

### Retrieve Access Token



**NOTE:**

The steps below are for the On-Premises release of cnMaestro.

#### Access Token Request (RFC 6749, section 4.4.2)

In order to generate a session, the client should retrieve an access token from cnMaestro. This is done by base64 encoding the **client\_id** and **client\_secret** downloaded from the cnMaestro Web UI and sending them to the cnMaestro server. The **Authorization** header is created by base64 encoding these fields as defined below. Note the fields are separated by a colon (:):

```
Authorization: Basic BASE64(<client_id>:<client_password>)
```

In the body of the **POST** the parameter **grant\_type** must be set to **client\_credentials**.

```
POST /api/v2/access/token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
```

Alternatively, instead of using the **Authorization** header, the credentials can be passed within the body of the **POST**:

```
POST /api/v2/access/token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_id=s6BhdRkqt3&client_secret=7Fjfp0ZBr1KtDRbnfVdmIw
```

### Access Token Response (RFC 6749, section 4.4.3)

The response returned from cnMaestro includes the `access_token` that should be used in subsequent requests. The `expires_in` field defines how many seconds the token is valid.

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "access_token": "2YotnFZFEjrlzCsicMWpAA",
  "token_type": "bearer",
  "expires_in": 3600
}
```

### Error Response (RFC 6749, section 5.2)

If there is an error, an HTTP 400 (Bad Request) error code is returned along with one of the following error messages:

Message	Details
invalid_request	Required parameter is missing from the request.
invalid_client	Client authentication failed.
unauthorized_client	The client is not authorized to use the grant type sent.
unsupported_grant_type	The grant type is not supported.

An example error response is below:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "error": "invalid_request"
}
```

## Access Resources

Once the **access\_token** is retrieved, API requests are sent to cnMaestro server using the format below. The **access\_token** is sent within the HTTP **Authorization** header.

```
GET /api/v2/devices
Accept: application/json
Authorization: Bearer ACCESS_TOKEN
```

## API Details

### HTTP Protocol

#### HTTP Response Codes

The following response codes are supported in cnMaestro and may be returned through the HTTP protocol.

Code	Description	Use in cnMaestro
400	Bad Request	Status field in request validation related errors.
502	Bad Gateway	Internal server error that may require a reboot.
403	Forbidden	An authenticated user tries to access a non-permitted resource.
500	Internal Server Error	A server-side error happened during processing the request.
405	Method Not Allowed	A method (GET, PUT, POST) is not supported for the resource.
404	Not Found	Server could not locate the requested resource.
501	Not Implemented	The request method is not recognized.
200	OK	Standard response for successful HTTP requests.
413	Payload Too Large	The request is larger than the server is willing to handle
431	Request Header Fields Too Large	The header fields are too large to be processed.
503	Service Unavailable	Internal server error that may require a reboot.
429	Too Many Requests	The client has sent too many requests in a given interval.
401	Unauthorized	User tried to access a resource without authentication.
422	Unprocessable Entity	The server understands the request but cannot process it.

#### HTTP Response Codes

Request Headers

Header	Details
Authorization	Used in every API request to send the Access Token. Example: Authorization: Bearer <Access-Token>
Accept	Set to application/json
Content-Type	Set to application/json

## REST Protocol

### Resource URLs

The format for cnMaestro path and parameters are the following:

Access a collection of resources:

```
/api/{version}/{resource}?{parameter}={value}&{parameter}={value}
```

Access a single resource:

```
/api/{version}/{resource}/{resource_id}?{parameter}={value}&{parameter}={value}
```

Access a sub-resource on a collection (this is also possible on single resources):

```
/api/{version}/{resource}/{sub-resource}?{parameter}={value}&{parameter}={value}
```

For example - read the statistics for MAC, Type, and IP on all devices:

```
/api/v2/devices/statistics?fields=mac,type,ip_wan
```

### Version

The version is equal to v2 in this release.

### Resource

Resources are the basic objects in the system

Context	Details
alarms	Current active alarms.
alarms/history	Historical alarms, including active alarms.
devices	Devices, including ePMP, PMP, and WiFi.
events	Historical events.
msp	MSP managed services.



Context	Details
networks	Configured networks.
sites	Configured WiFi sites.
towers	Configured Fixed Wireless towers.

## Sub-Resources

Sub-Resources apply to top-level resources. They provide a different view of the resource data, or a filtered collection based upon the resource. They include:

Context	Details
alarms	Alarms mapped to the top-level resource.
alarms/history	Historical alarms mapped to the top-level resource.
clients	Wireless LAN clients mapped to the top-level resource.
devices	Devices mapped to the top-level resource.
events	Events mapped to the top-level resource.
mesh/peers	Wireless LAN mesh peers mapped to the top-level resource.
operations	Operations available to the top-level resource
performance	Performance data for the top-level resource.
statistics	Statistics for the top-level resource.

## Responses

### Successful Response

In a successful HTTP 200 response, data is returned using the following structure. The actual payload is presented in JSON format. The request URL is:

```
/api/v2/devices?fields=mac,type&limit=5
```

```
{
  "paging": {
    "offset": 0,
    "limit": 5,
    "total": 540
  },
  "data": [
    {
      "mac": "C1:00:0C:00:00:21",
      "type": "wifi-home"
    },
    {
      "mac": "C1:00:0C:00:00:18",
      "type": "wifi-home"
    },
    {
      "mac": "C1:00:0C:00:00:12",
      "type": "wifi-home"
    },
    {
      "mac": "C1:00:0C:00:00:15",
      "type": "wifi-home"
    },
    {
      "mac": "C1:00:0C:00:00:06",
      "type": "wifi-home"
    }
  ]
}
```

## Error Response

Error responses return a message and an error cause. If the `start_time` and `stop_time` are mandatory query parameters and someone missed to provide them in the url gives the following error response with message and cause.

```
{
  "error": {
    "message": "Missing required property: stop_time \n Missing required property: start_time",
    "cause": "InvalidInputError"
  }
}
```

## Parameters

Most APIs can be modified to filter the data and limit the number of entries returned. The parameter options are listed below. The specific fields, and the appropriate values, vary for each API.

### Field Selection

Field selection is supported through the optional “fields” parameter, which can specify the specific data to return from the server. If this parameter is missing, all available fields will be returned.

Parameter	Details
fields	Define exactly what fields should be returned in a request. The names are provided as a comma-separated list.

Fields can limit which JSON parameters are returned.

Example: To retrieve name, type and location information for all devices.

Request:

```
/api/v2/devices?fields=mac,type
```

Response:

```
{
  "paging": {
    "total": 3,
    "limit": 100,
    "offset": 0
  },
  "data": [
    {
      "mac": "00:44:E6:34:89:48",
      "type": "wifi-enterprise"
    },
    {
      "mac": "00:44:16:E5:33:E4",
      "type": "wifi-enterprise"
    },
    {
      "mac": "00:44:26:46:32:22",
      "type": "wifi-enterprise"
    }
  ]
}
```

## Filtering

A subset of fields support filtering. These are defined as query parameters for a particular resource, and they are listed along with the API specification. Some of the standard filtering parameters are below:

Field	Details
network	(Devices) Configured Network name.
severity	(Alarms, Events) Alarm or Event severity (critical, major, minor, notice).
site	(Devices) Configured Site name.
state	(Alarms) Alarm state (active, cleared).

Field	Details
status	(Devices) Device status [online, offline, onboarding]
tower	(Devices) Configured Tower name.
type	(Devices) Device type [60ghz-cnwave, cnreach, cnmatrix, epmp, pmp, wifi-enterprise, wifi-home, wifi, ptp] (wifi includes wifi-home and wifi-enterprise).

Filters can be used simultaneously for Resources and Sub-Resources.

Example: Retrieve all WiFi devices that are online.

Request:

```
/api/v2/devices?type=wifi&status=online
```

Response:

```
{
  "paging": {
    "total": 1,
    "limit": 100,
    "offset": 0
  },
  "data": [
    {
      "ip": "233.187.212.38",
      "location": {
        "type": "Point",
        "coordinates": [
          77.55310127974755,
          12.952351523837196
        ]
      },
      "mac": "C1:00:0C:00:00:24",
      "msn": "SN-C1:00:0C:00:00:24",
      "name": "Hattie",
      "network": "Bangalore",
      "product": "cnPilot R201",
      "registration_date": "2017-05-23T21:28:37+05:30",
      "status": "online",
      "site": "Bangalore_Industrial",
      "type": "wifi-home",
      "hardware_version": "V1.1",
      "software_version": "2.4.4",
      "status_time": 1495560086
    }
  ]
}
```

## Time Filtering

Events, Alarms, and Performance data can be filtered by date and time using ISO 8601 format.

Example: January 12, 2015 UTC would be encoded as **2015-01-12**.

Example: January 12, 2015 1:00 PM UTC would be encoded as **2015-01-12T13:00:00Z**.

The parameters are below. If they are not specified, then the start or stop times will be open-ended.

Parameter	Details
start_time	Inclusive start time of interval.
stop_time	Inclusive stop time of interval.

## Sorting

Sorting is supported on a selected subset of fields within certain requests. sort is used to specify sorting columns. The sort order is ascending unless the path name is prefixed with a '-', in which case it would be descending.

Parameter	Details
sort	Used to get the records in the order of the given attribute.

Example: To retrieve devices in sorted (ascending) order by name.

Request:

```
/api/v2/devices?sort=name
```

Example: To retrieve devices in sorted (descending) order by mac.

Request:

```
/api/v2/devices?sort=-mac
```

## Pagination

The limit and offset query parameters are used to paginate responses.

Parameter	Details
limit	Maximum number of records to be returned from the server.
offset	Starting index to retrieve the data.

Example: To retrieve the first 10 ePMP devices

Request:

```
/api/v2/devices?offset=3&limit=1
```

Response:

```
{
  "paging": {
    "total": 6,
    "limit": 1,
    "offset": 3
  },
  "data": [
    {
      "status": "online",
      "product": "cnPilot E400",
      "network": "Mumbai",
      "software_version": "3.3-b14",
      "registration_date": "2017-04-28T08:57:33+00:00",
      "site": "Central",
      "hardware_version": "Force 200",
      "status_time": "3498",
      "msn": "Z834275ABCDH",
      "mac": "00:04:36:46:34:AA",
      "location": {
        "type": "Point",
        "coordinates": [
          0,
          0
        ]
      },
      "type": "wifi-enterprise",
      "name": "E400-4634AA"
    }
  ]
}
```

## Internal Response Limits

When clients try to access a resource type without pagination, the server will return the first 100 entries that match the filter criteria. The response will always carry a metadata to convey total count and current offset and limit. Maximum number of results at any point is 100 even though limit provided is more than 100.

Example: To retrieve all devices.

Request:

```
/api/v2/devices
```

Response:

```
{
  data: {devices: [ {name: 'ePMP_5566', type:'ePMP', location:'blr'} , {...}... ] },
  paging:{
    "limit":25,
    "offset":50,
    "total":100
  }
}
```

The response returns the following values in the paging section:

Parameter	Details
limit	Current setting for the limit.
offset	Starting index for the records returned in the response (begins at 0).
total	Total number of records that can be retrieved.

## Access API

### Token (basic request)

```
POST
/api/v2/access/token
```

The access API generates token using the Client ID and Client Password created in the cnMaestro UI. The token can be leveraged for API calls through the expiration time. Only one token is supported for each Client ID at any given time.

### Request

#### Headers

Header	Value
Accept (optional)	application/json
Authorization	Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type	application/x-www-form-urlencoded

The `client_id` and `client_secret` are encoded and sent in the Authorization header. The encoding is:

```
BASE64(client_id:client_secret)
```

#### Body

The body needs to have the `grant_type`.

```
grant_type=client_credentials
```

### Response

The response returns credentials for API access.

#### Body

Name	Details
access_token	Access token to return with each API request.
expires_in	Time in seconds that the API session will remain active.
token_type	This will always be bearer.
<pre>{   "access_token": "2YotnFZFEjrlzCsicMWpAA",   "token_type": "bearer",   "expires_in": 3600 }</pre>	

## Example

<b>Request</b>
<pre>curl https://10.110.134.12/api/v2/access/token \ -X POST -k \ -u 8YKCxq72qpjnYmXQ:pcX5BmdJ2f4QLM5RfgsS4jOtxAdTRF \ -d grant_type=client_credentials</pre>
<b>Response</b>
<pre>{"access_token": "d587538f445d30eb2d48e1b7f7a6c9657d32068e", "token_type": "bearer", "expires_in": 86400}</pre>

## Token (alternate request)

POST
/api/v2/access/token

An alternative form is supported in which the client\_ID and client\_secret are sent in the body, rather than the Authorization header.

## Request

### Headers

Header	Value
Accept (optional)	application/json
Content-Type	application/x-www-form-urlencoded

### Body

grant_type=client_credentials&client_id=s6BhdRkqt3&client_secret=7Fjfp0ZBr1KtDRbnfVdmIw
-----------------------------------------------------------------------------------------



## Response

The response to both forms is the same.

### Body

Name	Details
access_token	Access token to return with each API request.
expires_in	Time in seconds that the API session will remain active.
token_type	This will always be bearer.
<pre>{   "access_token": "2YotnFZFEjrlzCsicMWpAA",   "token_type": "bearer",   "expires_in": 3600 }</pre>	

## Example

<b>Request</b>
<pre>curl https://10.110.134.12/api/v2/access/token \ -X POST -k \ -d grant_type=client_credentials \ -d client_id=8YKCxq72qpjnYmXQ \ -d client_secret=pcX5BmdJ2f4QLM5RfgsS4jOtxAdTRF</pre>
<b>Response</b>
<pre>{"access_token": "ee4e077cf457196eb4d27cf6f02686dc07763059", "token_type": "bearer", "expires_in": 86400}</pre>

## Validate Token

<b>GET</b>
<code>/api/v2/access/validate_token</code>

Verify an Access Token is valid and return the time remaining before it expires.

## Request

### HTTP Headers

Header	Value
Accept (optional)	application/json
Authorization	Bearer <ACCESS_TOKEN>

## Response

Body

Name	Details
expires_in	Time in seconds that the API session will remain active.
<pre>{   'expires_in': 86399 }</pre>	

## Example

<b>Request</b>
<pre>curl https://10.110.134.12/api/v2/access/validate_token \ -X GET -k \ -H "Authorization: Bearere4e077cf457196eb4d27cf6f02686dc07763059"</pre>
<b>Response</b>
<pre>{"expires_in":85643}</pre>

## Selected APIs

### Overview

cnMaestro APIs are defined within the Swagger specification, accessed here

<https://docs.cloud.cambiumnetworks.com/api/3.0.0/index.html#/>. This section only presents additional details for the Device, Statistics and Performance APIs, which have unique responses based upon Device Type, and are difficult to present within Swagger.

## cnMaestro v2 API

Beginning with cnMaestro 3.0.0, the API version changes from **v1** to **v2**. The **v1** version will be supported through 3.1.0, but Cambium recommends updating existing API code to use **v2**. For most commands, swapping v1 in the URL with v2 should be sufficient. However, the following APIs may need to be rewritten while moving to v2.

- AP Groups
- Devices
- Statistics
- Performance
- Mesh Peers
- Operations

There are Unique API responses such as:

- [Device API Response \(v2 Format\)](#)
- [Statistics API Response \(v2 Format\)](#)

- Performance API Response (v2 Format)

## Devices API Response (v2 Format)

Name	Details	ePMP	PMP	Wi-Fi	cnReac h	cnVisio n	PTP	cnMatri x	60 GHz cnWave
ap_group	AP Group			X					
cbrs_state	CBRS state		X						
cbrs_status	CBRS status		X						
config.sync_ reason	Configuration synchronization reason	X	X	X	X	X	X	X	
config.sync_ status	Configuration synchronization status	X	X	X	X	X	X	X	
config.variable s	Device is mapped to configuration variables	X	X	X	X	X	X	X	
config.version	Current configuration version	X	X	X	X	X	X	X	
country	Country	X	X	X		X			
country_code	Regulatory band						X		
description	Description	X	X	X	X	X	X	X	X
hardware_ version	Hardware version	X	X	X	X	X	X	X	X
inactive_ software_ version	Inactive software version	X	X	X	X	X	X	X	
ip	IP address	X	X	X	X	X	X	X	X
ipv6	IPv6	X		X		X			X
last_reboot_ reason	Reason for the last reboot (see 24.1)	X	X	X	X	X	X	X	

Name	Details	ePMP	PMP	Wi-Fi	cnReac h	cnVisio n	PTP	cnMatri x	60 GHz cnWave
link_symmetry	Link symmetry						X		
location	Location	X	X	X	X	X	X	X	X
mac	MAC address	X	X	X	X	X	X	X	X
managed_account	Managed account name	X	X	X	X	X	X	X	X
maximum_range	Maximum range (KM)	X	X			X	X		
msn	Manufacturer serial number	X	X	X	X	X	X	X	X
name	Device name	X	X	X	X	X	X	X	X
network	Network	X	X	X	X	X	X	X	X
product	Product name	X	X	X	X	X	X	X	X
registration_date	Registration date	X	X	X	X	X	X	X	X
role							X		
site	Site			X				X	X
site_id	Site unique identifier			X				X	X
software_version	Active Software version	X	X	X	X	X	X	X	
status	Status (online, offline, onboarding)	X	X	X	X	X	X	X	X

Name	Details	ePMP	PMP	Wi-Fi	cnReach	cnVision	PTP	cnMatrix	60 GHz cnWave
status_time	Uptime/downtime time interval (sec)	X	X	X	X	X	X	X	X
tower	Tower	X	X		X	X	X	X	
type	Device type (epmp, pmp, wifi-home, wifi-enterprise, cnreach, ptp, cnmatrix, 60ghz-cnwave)	X	X	X	X	X	X	X	X

## Statistics API Response (v2 format)

Statistics API Response v2 format are shown for the following devices:

- 60 GHz cnWave
- cnMatrix
- cnReach
- Fixed Wireless
- PTP
- Wi-Fi

### 60 GHz cnWave

#### General

Name	Details	Mode
cpu	CPU utilization	All
last_sync	Last synchronization (UTC Unix time milliseconds)	All
mac	MAC address	All
managed_account	Managed account name	All
memory	Available memory	All
mode	Device mode	All
name	Device name	All
network	Network	All

Name	Details	Mode
site	Site name	All
status	Status [online, offline, claimed, waiting, onboarding]	All
status_time	Uptime/downtime interval (seconds)	All
sync_mode	Radio Sync mode [RF, GPS, None]	All
type	Device type	All

## Networks

Name	Details	Mode
ipv6	IPv6 address	All

## Radios (Array format)

Name	Details	Mode
radios[].channel	Channel	All
radios[].id	Radio Id	All
radios[].mac	Radio MAC	All
radios[].rx_bps	Receive bits/second	All
radios[].tx_bps	Transmit bits/second	All
radios[].sync_mode	Radio Sync mode [RF, GPS, None]	All

## Ethernet Ports (Array format)

Name	Details	Mode
ethports[].port	Port name	All
ethports[].rx_pkts	Received packets	All
ethports[].tx_pkts	Transmitted packets	All
ethports[].speed	Port speed and duplex	All

**General**

Name	Details
cpu	CPU utilization
config_version	Configuration version
last_sync	Last synchronization (UTC Unix time milliseconds)
mac	MAC address
managed_account	Managed account name
memory	Available memory
mode	Device mode
name	Device name
network	Network
site	Site name
site_id	Site unique identifier
status	Status [online, offline, claimed, waiting, onboarding]
status_time	Uptime/downtime interval (seconds)
tower	Tower name
type	Device type

**Networks**

Name	Details
ip	IP address

## General

Name	Details	Mode
config_version	Configuration version	All
connected_sms	Connected SM count	All
last_sync	Last synchronization (UTC Unix time milliseconds)	All
mac	MAC address	All
managed_account	Managed account name	All
mode	Device mode	All
name	Device name	All
network	Network	All
status	Status [online, offline, claimed, waiting, onboarding]	All
status_time	Uptime/downtime interval (seconds)	All
tower	Tower name	All
type	Device type	All

## Networks

Name	Details	Mode
ip	IP address	All

## Radios (Array format)

Name	Details	Mode
radios[].device_id	Device ID	Radios
radios[].id	Radio Id	Radios
radios[].linked_with	Linked with	Radios
radios[].mac	Radio MAC	Radios
radios[].margin	Margin	Radios
radios[].mode	Radio mode [ap, ep, rep]	Radios



Name	Details	Mode
radios[].neighbors	Radio neighbors	Radios
radios[].network_address	Network address	Radios
radios[].noise	Average noise (dB)	Radios
radios[].power	Transmit power	Radios
radios[].rssi	RSSI value (dB)	Radios
radios[].rx_bytes	Receive bytes	Radios
radios[].software_version	Current software version	Radios
radios[].temperature	Radio temperature	Radios
radios[].type	Radio type [ptp, ptmp]	Radios
radios[].tx_bytes	Transmit bytes	Radios

## Fixed Wireless (ePMP and PMP)

### General

Name	Details	ePMP	PMP
ap_mac	AP MAC	SM	SM
config_version	Configuration version	AP/SM	AP/SM
connected_sms	Connected SM count	AP	AP
cpu	CPU utilization		AP/SM
distance	SM distance (KM)	SM	SM
gain	Antenna gain (dBi)	AP/SM	AP/SM
gps_sync_state	GPS synchronization state	AP/SM	
last_sync	Last synchronization (UTC Unix time milliseconds)	AP/SM	AP/SM
mac	MAC address	AP/SM	AP/SM
managed_account	Managed account name	AP/SM	AP/SM
mode	Device mode	AP/SM	AP/SM
name	Device name	AP/SM	AP/SM

Name	Details	ePMP	PMP
network	Network	AP/SM	AP/SM
reboots	Reboot count	AP/SM	
status	Status [online, offline, claimed, waiting, onboarding]	AP/SM	AP/SM
status_time	Uptime/downtime interval (seconds)	AP/SM	AP/SM
temperature	Temperature		AP/SM
tower	Tower name	AP	AP
vlan	VLAN		AP/SM

## Networks

Name	Details	ePMP	PMP
default_gateway	Default gateway	AP/SM	AP/SM
ip	IP address	AP/SM	AP/SM
ipv6	IPv6 address	AP/SM	
ip_dns	DNS	AP/SM	AP/SM
ip_dns_secondary	Secondary DNS		AP/SM
ip_wan	WAN IP	AP/SM	
lan_mode_status	LAN mode status [no-data, half, full]	AP/SM	
lan_mtu	MTU size	SM	
lan_speed_status	LAN speed status	AP/SM	
lan_status	LAN status [down, up]	AP/SM	AP/SM
netmask	Network mask	AP/SM	AP/SM

## Radios

Name	Details	ePMP	PMP
radio.auth_mode	Authentication mode	SM	
radio.auth_type	Authentication type ePMP [open, wpa1, eap- ttls] PMP [disabled, enabled]	AP/SM	AP/SM
radio.channel_width	Channel width ePMP [5 MHz, 10 MHz, 20 MHz, 40 MHz] PMP [...]	AP/SM	AP/SM
radio.color_code	Color code		AP/SM
radio.dfs_status	DFS status ePMP: [not-applicable, channel- availability-check, in- service, radar- signal-detected, alternate- channel- monitoring, not-in- service] PMP: [Status String]	AP/SM	AP/SM
radio.dl_err_drop_pkts	Downlink error drop packets	SM	
radio.dl_err_drop_pkts_ percentage	Downlink error drop packets percentage	SM	
radio.dl_frame_utilization	Downlink frame utilization		AP
radio.dl_lqi	Downlink Link Quality Indicator		SM
radio.dl_mcs	Downlink MCS	SM	
radio.dl_modulation	Downlink Modulation		SM
radio.dl_pkts	Downlink packet count	AP/SM	AP/SM
radio.dl_pkts_loss	Downlink packet loss		AP/SM
radio.dl_retransmits	Downlink Retransmission	AP/SM	
radio.dl_retransmits_pct	Downlink Retransmission percentage	AP/SM	
radio.dl_rssi	Downlink RSSI	SM	SM
radio.dl_rssi_imbalance	Downlink RSSI imbalance		AP
radio.dl_snr	Downlink SNR (dB)	SM	
radio.dl_snr_h	Downlink SNR (dB) horizontal		SM

Name	Details	ePMP	PMP
radio.dl_snr_v	Downlink SNR (dB) vertical		SM
radio.dl_throughput	Downlink throughput	AP/SM	AP/SM
radio.frequency	RF frequency	AP/SM	AP/SM
radio.frame_period	Frame period		AP
radio.mac	Wireless MAC	AP/SM	
radio.mode	Radio mode [eftp-master, eftp-slave, tdd, tdd-ptp, ap/sm]	AP/SM	
radio.sessions_dropped	Session drops	AP	AP/SM
radio.software_key_throughput	Software key - max throughput		SM
radio.ssid	SSID	AP/SM	
radio.sync_source	Synchronization source		AP
radio.sync_state	Synchronization state		AP
radio.tdd_ratio	TDD ratio ePMP [75/25, 50/50, 30/70, flexible] PMP [...]	AP	AP
radio.tx_capacity	SM transmit capacity	SM	
radio.tx_power	Radio transmit power	AP/SM	AP/SM
radio.tx_quality	SM transmit quality	SM	
radio.ul_err_drop_pkts	Uplink error drop packets	SM	
radio.ul_err_drop_pkts_percentage	Uplink error drop packets percentage	SM	
radio.ul_frame_utilization	Uplink frame utilization		AP
radio.ul_mcs	Uplink MCS	AP/SM	
radio.ul_modulation	Uplink Modulation example [2X MIMO-B]		SM
radio.ul_lqi	Uplink Link Quality Indicator		SM
radio.ul_pkts	Uplink packet count	AP/SM	AP/SM
radio.ul_pkts_loss	Uplink packet loss		AP/SM

Name	Details	ePMP	PMP
radio.ul_retransmits	Uplink Retransmission	SM	
radio.ul_retransmits_pct	Uplink Retransmission percentage	AP/SM	
radio.ul_rssi	Uplink RSSI	SM	SM
radio.ul_snr	Uplink SNR (dB)	SM	
radio.ul_snr_h	Uplink SNR (dB) horizontal		SM
radio.ul_snr_v	Uplink SNR (dB) vertical		SM
radio.ul_throughput	Uplink throughput	AP/SM	AP/SM
radio.wlan_status	WLAN status [down, up]	AP/SM	

## PTP

### General

Name	Details	Mode
config_version	Configuration version	All
connected_sms	Connected SM count	Master
gain	Antenna gain (dBi)	All
last_sync	Last synchronization (UTC Unix time milliseconds)	All
mac	MAC address	All
managed_account	Managed account name	All
mode	Device mode [AP, SM]	All
name	Device name	All
network	Network	All
status	Status [online, offline, claimed, waiting, onboarding]	All
status_time	Uptime/downtime interval (seconds)	All
temperature	Temperature	All
tower	Tower name	All
type	Device type	All
vlan	VLAN	All

## Networks

Name	Details	Mode
default_gateway	Default gateway	All
ip	IP address	All
ip_dns	DNS	All
ip_dns_secondary	Secondary DNS	All
lan_status	LAN status [down, up]	All
netmask	Network mask	All

## Ethernet Ports (Array format)

Name	Details	Mode
ethports[].port	Port name	All
ethports[].rx_frames	Ports receive frames oversize	All
ethports[].rx_util	Ports receive bandwidth utilization	All
ethports[].speed	Ports speed and duplex	All
ethports[].tx_util	Ports transmit bandwidth utilization	All

## Radios

Name	Details	Mode
radio.byte_error_ratio	Byte Error Ratio	All
radio.channel_width	Channel width ePMP: [5 MHz, 10 MHz, 20 MHz, 40 MHz] PMP: [...]	All
radio.color_code	Color code	All
radio.rx_frequency	Receive frequency	All
radio.tx_frequency	Transmit frequency	All
radio.tx_power	Radio transmit power	All

## Wi-Fi

**NOTE:**

Mode is Enterprise, Home, or All.

### General

Name	Details	Mode
config_version	Configuration version	All
cpu	CPU utilization	All
mac	MAC address	All
managed_account	Managed account name	All
memory	Available memory	All
mode	Device mode	All
name	Device name	All
network	Network	All
parent_mac	Parent MAC	All
site	Site name	All
site_id	Site unique identifier	All
status	Status [online, offline, claimed, waiting, onboarding]	All
status_time	Uptime/downtime interval (seconds)	All
type	Device type	All

### Networks

Name	Details	Mode
ip	IP address	All
ipv6	IPv6 address	All
ip_dns	DNS	All
ip_dns_secondary	Secondary DNS	All
ip_wan	WAN IP	All

Name	Details	Mode
lan_mode_status	LAN mode status [no-data, half, full]	Enterprise
lan_speed_status	LAN speed status	All
lan_status	LAN status [down, up]	Home
netmask	Network mask	All

### Radios (Array format)

Name	Details	Mode
radios[].airtime	Airtime	All
radios[].band	Radio band	All
radios[].bssid	Radio mac	Enterprise
radios[].channel	Channel	All
radios[].id	Radio Id	All
radios[].multicast_rate	Multicast rate	Enterprise
radios[].noise_floor	Noise floor	Enterprise
radios[].num_clients	Number of clients	All
radios[].num_wlans	Number of WLANs	Enterprise
radios[].power	Transmit power	All
radios[].quality	RF Quality description	Enterprise
radios[].radio_state	Radio state	Enterprise
radios[].rx_bps	Receive bits/second	All
radios[].rx_bytes	Receive bytes	All
radios[].tx_bps	Transmit bits/second	All
radios[].tx_bytes	Transmit bytes	All
radios[].unicast_rates	Unicast rates	Enterprise
radios[].utilization	Radio utilization	Enterprise



## Performance API Response (v2 format)

Performance API Response v2 Format are shown for following devices:

- 60 GHz cnWave
- cnMatrix
- cnReach
- Fixed Wireless
- PTP
- Wi-Fi

### 60 GHz cnWave

#### General

Name	Details	Mode
mac	MAC address	All
managed_account	Managed account name	All
mode	Device mode	All
name	Device name	All
network	Network	All
online_duration	Duration of device connection with server (seconds)	All
site	Site	All
timestamp	Timestamp	All
type	Device type	All
uptime	Device online time (seconds)	All

#### Radios (Array format)

Name	Details	Mode
radios[].id	Radio ID	All
radios[].rx_bps	Receive bits/second	All
radios[].tx_bps	Transmit bits/second	All

#### Ethernet Ports (Array format)

Name	Details	Mode
ethports[].max_rx	Ports maximum receive bytes	All
ethports[].max_tx	Ports maximum transmit bytes	All
ethports[].min_rx	Ports minimum receive bytes	All
ethports[].min_tx	Ports minimum transmit bytes	All

Name	Details	Mode
ethports[].port	Port name	All
ethports[].rx	Ports receive bytes	All
ethports[].tx	Ports transmit bytes	All

## cnMatrix

### General

Name	Details
mac	MAC address
managed_account	Managed account name
mode	Device mode
name	Device name
network	Network
site	Site
timestamp	Timestamp
tower	Tower
type	Device type

### Switch

Name	Details
switch.rx.broadcast_pkts	Receive broadcast packets
switch.dl_kbits	Downlink usage (in kbits on hour or minute basis)
switch.dl_throughput	Downlink throughput (Kbps)
switch.ul_kbits	Uplink (in Kbits on hour or minute basis)
switch.rx.multicast_pkts	Receive multicast packets
switch.ul_throughput	Uplink throughput (Kbps)
switch.rx.pkts_err	Receive Packet error
switch.rx.unicast_pkts	Receive unicast packets

Name	Details
switch.tx.broadcast_pkts	Transmit broadcast packets
switch.tx.multicast_pkts	Transmit multicast packets
switch.tx.pkts_err	Transmit packet error
switch.tx.unicast_pkts	Transmit unicast packets

## cnReach

### General

Name	Details	Mode
mac	MAC address	All
managed_account	Managed account name	All
mode	Device mode	All
name	Device name	All
network	Network	All
online_duration	Duration of device connection with server (seconds)	All
sm_count	Connected SM count	All
timestamp	Timestamp	All
tower	Tower	All
type	Device type	All
uptime	Device online time (seconds)	All

### Radios (Array format)

Name	Details	Mode
radios[].id	Radio ID	Radios
radios[].neighbors	Radio neighbors	Radios
radios[].noise	Average noise	Radios
radios[].power	Transmit power	Radios
radios[].rssi	RSSI value	Radios

Name	Details	Mode
radios[].rx_bytes	Receive bytes	Radios
radios[].throughput	Total throughput	Radios
radios[].tx_bytes	Transmit bytes	Radios

## Fixed Wireless (ePMP and PMP)

### General

Name	Details	ePMP	PMP
mac	MAC address	AP/SM	AP/SM
managed_account	Managed account name	AP/SM	AP/SM
mode	Device mode	AP/SM	AP/SM
name	Device name	AP/SM	AP/SM
network	Network	AP/SM	AP/SM
online_duration	Duration of device connection with server (seconds)	AP/SM	AP/SM
sm_count	Connected SM count	AP	AP
sm_drops	Session drops	AP/SM	AP
timestamp	Timestamp	AP/SM	AP/SM
tower	Tower	AP/SM	AP/SM
type	Device type	AP/SM	AP/SM
uptime	Device online time (seconds)	AP/SM	AP/SM

### Radios

Name	Details	ePMP	PMP
radio.dl_frame_utilization	Downlink frame utilization		AP
radio.dl_kbits	Downlink usage (in Kbits on hour or minute basis)	AP/SM	
radio.dl_mcs	Downlink MCS	SM	

Name	Details	ePMP	PMP
radio.dl_modulation	Downlink modulation		SM
radio.dl_pkts	Downlink packet count	AP/SM	
radio.dl_pkts_loss	Downlink packet loss		AP/SM
radio.dl_retransmits_pct	Downlink retransmission percentage	AP/SM	
radio.dl_rssi	Downlink RSSI	SM	SM
radio.dl_rssi_imbalance	Downlink RSSI imbalance		SM
radio.dl_snr	Downlink SNR	SM	
radio.dl_snr_h	Downlink SNR (dB) horizontal		SM
radio.dl_snr_v	Downlink SNR (dB) vertical		SM
radio.dl_throughput	Downlink Throughput (Kbps)	AP/SM	AP/SM
radio.ul_frame_utilization	Uplink frame utilization		AP
radio.ul.kbits	Uplink usage (in Kbits on hour or minute basis)	AP/SM	
radio.ul_mcs	Uplink MCS	SM	
radio.ul_modulation	Uplink modulation		SM
radio.ul_pkts	Uplink packet count	AP/SM	
radio.ul_pkts_loss	Uplink packet loss		AP/SM
radio.ul_retransmits_pct	Uplink Retransmission percentage	AP/SM	
radio.ul_rssi	Uplink RSSI	SM	SM
radio.ul_snr	Uplink SNR	SM	
radio.ul_snr_h	Uplink SNR (dB) horizontal		SM
radio.ul_snr_v	Uplink SNR (dB) vertical		SM
radio.ul_throughput	Uplink Throughput (Kbps)	AP/SM	AP/SM

## PTP

### General

Name	Details	Mode
mac	MAC address	All
managed_account	Managed account name	All
mode	Device mode	All
name	Device name	All
network	Network	All
sm_count	Connected SM count	Master
timestamp	Timestamp	All
tower	Tower	All
type	Device type	All

### Ethernet Ports (Array format)

Name	Details	Mode
ethports[].max_rx	Ports maximum receive bytes	All
ethports[].max_tx	Ports maximum transmit bytes	All
ethports[].min_rx	Ports minimum receive bytes	All
ethports[].min_tx	Ports minimum transmit bytes	All
ethports[].pkt_error	Ports packet error	All
ethports[].port	Port name	All
ethports[].rx	Ports receive bytes	All
ethports[].tx	Ports transmit bytes	All

## Ethernet

Name	Details	Mode
ethernet.link_loss	Link loss	All
ethernet.pcb_temperature	PCB temperature	All
ethernet.rx_channel_util	Receive channel utilization	All
ethernet.rx_capacity	Receive capacity	All
ethernet.ssr	Signal strength ratio	All
ethernet.rx_power	Receive power	All
ethernet.sfp_interface.tx	SFP transmit bytes	All
ethernet.rx_throughput	Receive throughput	All
ethernet.tx_channel_util	Transmit channel utilization	All
ethernet.tx_capacity	Transmit capacity	All
ethernet.tx_power	Transmit power	All
ethernet.tx_throughput	Transmit throughput	All
ethernet.vector_error	Vector error	All

## Wi-Fi

### General

Name	Details	Mode
mac	MAC address	All
managed_account	Managed account name	All
mode	Device mode	All
name	Device name	All
network	Network	All
online_duration	Duration of device connection with server ( seconds)	All
site	Site	All

Name	Details	Mode
timestamp	Timestamp	All
type	Device type	All
uptime	Device online time ( seconds)	All

#### Radios (Array format)

Name	Details	Mode
radios[].clients	Number of clients	All
radios[].id	Radio ID	All
radios[].rx_bps	Receive bits/second	All
radios[].throughput	Total throughput	All
radios[].tx_bps	Transmit bits/second	All



#### NOTE:

The specification for the equivalent v1 APIs is available in the Appendix.

- [Statistics API Response \(v1 Format\)](#)
- [Performance API Response \(v1 Format\)](#)



## External Guest Access Login API

Integrates an external captive portal with the Cambium Networks AP while posting directly to cnMaestro. This API provides the support for the external captive portal to make login requests.

POST /api/v2/ext-portals/login

### Request:

curl -X

```
/api/v2/ext-portals/login" -H "accept: */*" -H "Authorization: Bearer  
e88916f5b663c1ea966af835c8a0a19c20d17686" -H "Content-Type: application/json"-d
```

### Body

```
{"ga_ap_mac\":\"11-22-33-44-55-66\", \"ga_cmac\":\"11-22-33-44-55-65\", \"ga_  
Qv\":\"eUROBR86HBgAGDEEVgQAGw4UWRUCACYVMgFPMV5ZWVfUVdGX1ZFJXxZR1dLBhMUMww\", \"ga_  
user\":\"test-user\", \"ga_pass\":\"test-pass\"}
```

### Response:

```
{  
  "data": {  
    "mType": 3,  
    "msgId": 28,  
    "status": <integer values>,  
    "prefixQs": <true/false>,  
    "expiry": <integer values>,  
    "action": <integer values>,  
    "cmac": <client mac>,  
    "msg": <Radius Returned Message>,  
    "extURL": <external url string>  
  }  
}
```

The status value description is provided in the table below.

Status	Description
0	Login is successful.
1	Invalid login request, the client is not currently associated to the AP which is being requested for login here.
2	RADIUS reject due to invalid username/password.
3	RADIUS timeout, AP didn't received the RADIUS response.
4	Missing RADIUS server config on the WLAN config of the AP.
5	If LDAP configured on the AP for authentication then LDAP server responded back with reject.
6	LDAP timeout happened on the AP for the request.
7	Missing LDAP configuration on the WLAN configuration of the AP.
8	Logout is successful.
9	Logout failed due to missing session on the AP. Most likely client session is already deleted from this AP.

The response parameter name and details is shown below.

Name	Details
prefixQs	True: Add query strings to landing URL on success. False: Remove query strings from landing URL on success. <code>prefixQs</code> and <code>action</code> values are driven based on WLAN configuration.
action	0: On success action redirects the user to AP onboard logout page. 1: On success redirects user to an external URL. 2: On success redirects user to its original URL.
msg	Message is based on RADIUS attribute reply message (18) in the RADIUS Access Accept or Reject message.
expiry	Displays the session time for the given guest session.
cmac	MAC address of the client.

## 60 GHz cnWave RESTful API

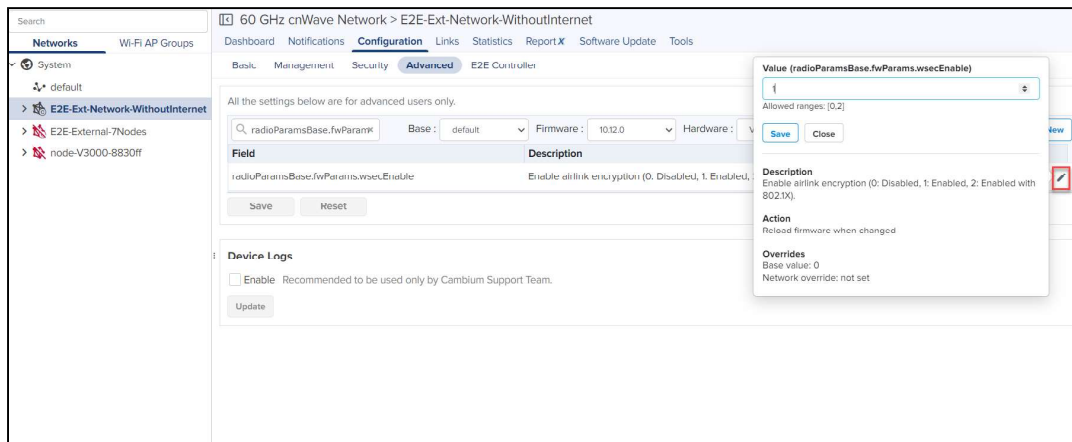
cnMaestro supports configuration overrides for 60 GHz cnWave E2E Network, E2E Controller, and Node(s) using the RESTful API.

### E2E Network

To determine the configuration parameters available in an E2E Network, navigate to **E2E Network > Configuration > Advanced**. Search for the desired **Field**, and review its **Description**, allowed **Values**, and **Override status**. Use the RESTful API to override single or multiple fields.

GET /api/v2/cnwave60/networks/{network\_id}/configuration

PUT /api/v2/cnwave60/networks/{network\_id}/configuration



Field names are separated by dots. Each substring between dots will be converted to objects and the last substring will be the key and value.

### Example

In case of field name `radioParamsBase.fwParams.wsecEnable`, payload will be:

```
{
  "radioParamsBase": {
    "fwParams": {
      "wsecEnable": 1
    }
  }
}
```



#### WARNING:

Partial update is not allowed. Always send full configuration that needs to be pushed to E2E Network.

### Device (Node) Configuration

To update Device configuration, navigate to **Node > Configuration > Advanced**. Search for the **Field**, and review its **Description**, allowed **Values**, and **Overrides status**. Use the RESTful API to override those fields.

GET /api/v2/cnwave60/networks/{mac}/configuration

PUT /api/v2/cnwave60/networks/{mac}/configuration

Field names are separated by dots. Each substring between dots will be converted to objects and the last substring will be the key and value.

## Example

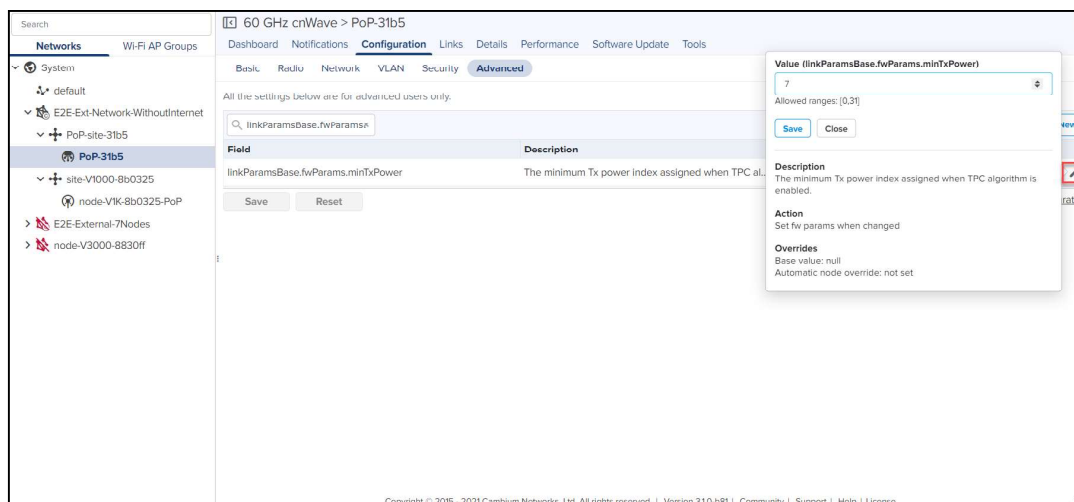
In case of field name `linkParamsBase.fwParams.minTxPower`, object to send in the API payload will be:

```
{
  "linkParamsBase": {
    "fwParams": {
      "minTxPower": 6
      "maxTxPower": 8
    }
  }
}
```

The below two APIs are introduced in Release 3.1.0 to update multiple device configurations overrides.

GET `/api/v2/cnwave60/networks/{network_id}/devices/overrides`

PUT `/api/v2/cnwave60/networks/{network_id}/devices/overrides`



### WARNING:

Partial update is not allowed. Always send full configuration that needs to be pushed to 60 GHz cnWave Devices.

The example payload for PUT request is seen from cnMaestro UI.

## Example

```
{
  "device1_name": {
    "radioParamsBase": {
      "fwParams": {
        "txPower": 6
      }
    }
  },
  "device2_name": {
    "popParams": {
      "POP_IFACE": "nic2"
    }
  }
}
```

**NOTE:**

You can download the full config of the node by clicking on the **Show Full Configuration** as well and then get the JSON key and pass in RESTful API.

## E2E Controller

To update E2E Controller configuration, navigate to **E2E Network > Configuration > E2E Controller**. Search for the desired **Field**, and review its **Description**, allowed **Values**, and **Override status**. Use the RESTful API to override those fields.

GET /api/v2/cnwave60/networks/{network\_id}/controller/configuration

PUT /api/v2/cnwave60/networks/{network\_id}/controller/configuration

Field names are separated by dots. Each substring between dots will be converted to objects and the last substring will be the key and value.

### Example

In case of field name `prefixAllocParams.seedPrefix`, payload will be:

```
{
  "prefixAllocParams": {
    "seedPrefix": "fd00:ceed:1992:1400::/56"
  }
}
```

**WARNING:**

Partial update is not allowed. Always send full configuration that needs to be pushed to the E2E Controller.

# cnPilot Guest Access

This section describes how to configure Guest Access using cnMaestro. This feature allows the clients to connect through Free Tier, Buying Vouchers or Paid Access types.

The Guest Access feature creates a separate network for guests by providing Internet access to guest wireless devices (mobiles, laptops, etc).



## NOTE:

The Guest Access feature is supported on cnPilot E-series Enterprise devices.

## Configuration

- Create the Guest Access Portal in cnMaestro
- Map the device to cnMaestro

## Create the Guest Access Portal in cnMaestro

1. Basic details
2. Access Portal
3. Splash page
4. Sessions

## Procedure for creating Guest Access

### Prerequisites

1. Navigate to **Services > Guest Access Portal**.

Guest Portal Name	Description	Managed Account	Event Logging	Free Access	Paid Access	Voucher Access
SAIL_GATE	SDR testing	Base Infrastructure	Yes	Yes	No	Yes



## NOTE:

The Floating Management IP should be used to access the Guest Access Portal. This means DNS should be mapped to the Floating Management IP, and not to one of the unique IP addresses of the cnMaestro instances.

2. Click **Add Portal**. A maximum of four portals can be created per account.
3. Enter **Name** and **Description**.

Add Guest Portal

Managed Account

Base Infrastructure

Name\*

Description

☐ Client Login Event Logging

Save

Cancel

4. Click **Save**.

## Basic Details

The **Basic** Details page contains the Managed Account Type Name and Description.

Guest Access Portal > test

Basic

Access

Splash

Sessions

Managed Account

Base Infrastructure

Name\*

test

Description

☒ Client Login Event Logging

Save



### NOTE:

A name once created for the Portal cannot be changed.

## Access Portal

The Access Portal tab has three different access types:

- Free
- Paid
- Vouchers