Cambrionix
CHARGE • CONNECT • MANAGE

Connect Hub API

User Manual

Connect Hub API

EN

2. Table of Contents

2. Table of Contents **2**

2. Introduction **5**

2.1. Prerequisites 6

3. Quick Start **7**

3.1. Call Structure 7

3.2. JSON-RPC Error Object 8

4. API Endpoints **10**

4.1. Authentication 11

4.1.1 Add Authentication 12

4.1.2 Get Authentication 13

4.1.3 Remove Authentication 14

4.2. Certificate 15

4.2.1 Add Certificate 16

4.2.2 Get Certificate 18

4.2.3 Remove Certificate 22

4.3. Utility 23

4.3.1 Get API Configuration 24

4.3.2 Set API Configuration 26

4.3.3 Get API Details 28

4.3.4 Exit the API Service 32

4.4. Devices 33

4.4.1 Get Devices 34

4.4.2 Get Device 37

4.5. Hubs 40
4.5.1 Get Hubs 41
4.5.2 Get Hub 45
4.5.3 Send Hub Command 49
4.5.4 Get Hub Endpoints 51
4.5.5 Get Hub Firmware 55
4.5.6 Update Hub Firmware 57
4.5.7 Get Hub Firmware Update 59
4.5.8 Get Default Power Limits 62
4.5.9 Load Power Configuration 64
4.5.10 Save the Power Configuration 65
4.5.11 Set Hub Settings 66
4.6. Ports 67
4.6.1 Get Hub Ports State 68
4.6.2 Set Hub Ports LED 72
4.6.3 Get Hub Ports LED 74
4.6.4 Set Hub Port Mode 77
4.6.5 Get Hub Ports Supported Modes 79
4.6.6 Get Ports Power Limits 81
4.6.7 Set Ports Power Limits 85
4.6.8 Get Hub Port State 87
4.6.9 Set Hub Port LED 93
4.6.10 Get Hub Port LED 95
4.6.11 Set Hub Port Mode 98
4.6.12 Get Port Power Limits 100

4.6.13 Set Port Power Limits 102
4.7. Firmware 104
4.7.1 Get Firmwares 105
4.7.2 Get Firmware 108
4.7.3 Add Firmware 110
4.7.4 Delete Firmware 112
4.8. Endpoints 114
4.8.1 Get Computer Endpoints 115
4.9. Apple Devices 118
4.9.1 Put Devices into DFU 119
4.9.2 Reboot Devices 122
4.9.3 Put Device into DFU 125
4.9.4 Reboot Device 127
5. Token Exchange 129

2. Introduction

This manual provides a comprehensive overview of the Connect Hub API. It is a local RESTful API designed for managing Cambrionix hubs, connected devices, and hardware groups within on-site environments. The API allows direct integration with hub hardware for real-time control, monitoring, and configuration, without the need for cloud connectivity.

It supports a wide range of hardware management tasks, including querying device and hub status, setting port modes, controlling LEDs, updating firmware, and organising devices into groups. Its structured and efficient design makes it suitable for both small-scale installations and more complex environments.

By enabling local hub control, device grouping, and firmware management, the Connect Hub API helps simplify hardware operations and provides a flexible foundation for developing reliable and scalable device management workflows within the Cambrionix ecosystem.

How the API Works

The Connect Hub API operates over HTTPS, enabling secure, local management of hubs, connected devices, and hardware groups through a RESTful interface. This platform-independent API works with a variety of tools, including web browsers, command-line utilities, and custom-built applications, making it adaptable to different environments.

To support practical implementation, this manual includes examples using cURL (Client URL), a commonly used command-line tool for sending HTTPS requests. Its flexibility and scripting features make it well suited for automating tasks such as hub configuration, device control, hardware grouping, and firmware updates when using the Connect Hub API.

What is cURL?

The Connect Hub API uses HTTPS to provide secure communication, ensuring data integrity and confidentiality when managing hubs, devices, and hardware groups within a local network.

cURL is a widely used command-line tool for interacting with RESTful APIs. It allows users to send and receive data securely over HTTPS and is well suited for scripting and automating tasks with the Connect Hub API.

Using cURL, users can perform standard HTTP operations such as GET to retrieve device information, POST to register hardware, PUT to update configurations, and DELETE to remove devices or hubs. The API provides structured responses that are straightforward to process, helping to simplify integration and automation.

This makes cURL a valuable tool for creating efficient, repeatable workflows in local hardware management and for incorporating hub control into broader software solutions.

Using cURL with the Connect Hub API

When interacting with the Connect Hub API, which follows a RESTful architecture, cURL is a key tool for composing and sending requests. This method supports efficient and structured communication with the API over HTTPS. cURL helps streamline integration by enabling automation and simplifying the process of retrieving structured responses from the API.

Request type	Description
GET	Fetches data from the server
POST	Sends data to the server to create a new resource
DELETE	Removes a resource from the server

Each request is sent to the API endpoint and the response is returned in a structured JSON format.

Automating Interactions with cURL

Because cURL can be integrated into shell scripts and other automation tools, it is highly effective for automating routine tasks, such as setting location information for new hardware. By incorporating cURL into your workflows, you can reduce manual interaction and create automated processes that streamline assinging and editing location information.

2.1. Prerequisites

Before using the Connect Hub API, certain requirements must be met to ensure proper operation.

- Cambrionix Connect Licence: A valid Cambrionix Connect Licence is required to access the full functionality of the API.

While the API operates locally, a stable network connection within the local environment is essential. Internet access is not required for most operations, but it may be necessary for certain licensed features that require validation or communication with external services.

3. Quick Start

3.1. Call Structure

When sending requests using cURL, each command must begin with the curl command followed by the -X option, which specifies the HTTPS request method (such as GET, POST, or DELETE). This option tells the server what action to perform on the specified resource. For example:

```
curl -X 'request-type'
```

Request type	Description
GET	Fetches data from the server
POST	Sends data to the server to create a new resource
DELETE	Removes a resource from the server
PATCH	Sends data to partially update an existing resource
PUT	Sends data to update or replace an existing resource, or create it if it does not exist

The next piece of information that will be required is the API endpoint, this information for each action can be found in the API Endpoints section of the manual

```
curl -X 'request-type' \
API-endpoint/
```

Data

The -d option (short for --data) is used to send data in the body of an HTTP request when using 'POST' or 'PATCH' requests. It allows you to include data in your request, such as form fields, JSON objects, or other data types, that the server expects to process.

```
curl -X 'POST' \
API-method/
-d '{data}'
```

Headers and Authentication

The Connect Hub API requires certain headers to be included in the request, such as an authentication token. cURL allows you to add these headers using the `-H` option.

```
curl -X 'request-type' \
API-method/
-H "Authorization: Bearer token"
```

A token exchange will need to take place using custom services within the Cambrionix Connect platform, for more information on that please see the Token Exchange information.

Handling API Responses

The API will return responses in JSON format. cURL outputs the response directly to the terminal by default.

3.2. JSON-RPC Error Object

When a call to the API encounters an error, the response object will include an error member. This error member holds an object that provides detailed information about the error. The format of the object is shown below:

```
{
  "code": error-code,
  "message": "error-message",
  "detail": "error-message-detail"
}
```

error-code

A Number that indicates the error type that occurred. This is an integer.

error-message

A String providing a short description of the error.

error-message-detail

A Primitive or Structured value that contains additional information about the error.

example

```
{
  "code": 401,
  "message": "Unauthorized",
  "detail": "Invalid token or missing authentication header."
}
```

4. API Endpoints

The Connect Hub API is organised into endpoint groups, each serving a specific purpose in managing Cambrionix hubs and connected devices. These groups cover a wide range of functionality—from authentication and certificate management to querying devices, configuring ports, handling firmware, and supporting Apple device control. This structure enables developers to efficiently access and control different aspects of the system as needed.

Endpoints	Description
Authentication	Authentication-related queries involve processes that verify user or system identity, manage access tokens, enforce security measures like multi-factor authentication, and control permissions to ensure secure access
Certificate	Configure or query the certificate that the API uses for secure communication and authentication.
Utility	General-purpose endpoints for retrieving or setting API configuration, viewing service details, and exiting the service.
Devices	These endpoints enable listing all devices on the system or accessing details for a specific device, supporting efficient device identification and management.
Hubs	Provides access to hub information and controls, including listing hubs, running commands, checking firmware, and updating settings.
Ports	Manage and query port states, modes, and LEDs on hubs—both individually and in bulk. Supports control over port behaviour and status monitoring.
Firmware	Endpoints for managing hub firmware—upload, delete, retrieve, and apply firmware files, as well as view hardware details and perform updates.
Endpoints	Provides information on USB endpoint memory usage, either system-wide or for a specific hub.
Apple Devices	Provides DFU mode and reboot control for Apple devices connected to supported hubs, either hub-wide or per port (Connect licence required).

4.1. Authentication

To use the Connect Hub API, it is essential to obtain a valid and appropriate license from Cambrionix. This license grants access to the API's full range of features and ensures that your organisation complies with the terms of service. When making requests to the Connect Hub API, your organisation must be authenticated to validate the license and confirm that it has the necessary permissions to interact with the API.

Authentication is a crucial step in ensuring secure and authorised access to the API, and Cambrionix offers several methods to manage this process effectively. Proper authentication not only verifies your organisation's license but also protects your data and devices from unauthorised access.

There are several ways to manage and implement authentication for the Connect Hub API, ensuring that your organisation's API requests are valid and secure:

Endpoint	Description
Add Authentication	This endpoint is used to add or register an organisation’s authentication details to the Connect Hub API. When an organisation is onboarded, this step links the organisation’s credentials to their Cambrionix account. This process ensures that all future API requests made by the organisation are properly authenticated and validated against the assigned license.
Remove Authentication	This method allows the removal of an organisation’s authentication details from the Connect Hub API. If an organisation no longer requires access or if credentials need to be revoked (due to security concerns, expired licenses, or other factors), this method will remove the associated authentication information. Once removed, the organisation will no longer be able to authenticate API requests until new credentials are added.
Get Authentication	This method retrieves the current authentication configuration for an organisation. It provides details about the active authentication mechanisms that are linked to the organisation's account. This method is useful for reviewing the authentication setup, ensuring credentials are up to date, and troubleshooting any authentication issues that may arise during API interactions.

4.1.1 Add Authentication

Add an organisation for authentication with the Connect Hub API

Syntax: see Call Structure

Type = 'POST'

```
'http://localhost:43424/api/v1/auth?id=GUID'
```

Parameter	Description
GUID	The organisations id which is a guid. This can be found in Cambrionix connect by navigating to the organisations page and selecting the organisation you wish to find the id for. Please see https://connect.cambrionix.com for more information.

Returns

```
  true
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/auth?id=00000000-0000-0000-
0000-000000000000' \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": "00000000-0000-0000-0000-000000000000",
  "jwks_uri": "string"
```

Returns

```
  true
```

Errors

If there is an error in the API method then a JSON-RPC Error Object will be returned

4.1.2 Get Authentication

Get the information on the organisations that are authenticated with the Connect Hub API.

Syntax: see Call Structure

Type = 'GET'

```
'http://localhost:43424/api/v1/auth'
```

Returns:

A list of organisations stored on the host displayed as an array.

Example

```
curl -X 'GET' \
'http://localhost:43424/api/v1/auth?id=fe783079-434f-412d-b312-
d783c76a83e2' \
  -H 'accept: application/json'
```

Returns

```
{
  "id": "fe783079-434f-412d-b312-d783c76a83e2"
}
```

Errors

If there is an error in the API method then a JSON-RPC Error Object will be returned

4.1.3 Remove Authentication

Remove an organisation from being authenticated with the Connect Hub API

Syntax: see Call Structure

Type = 'DELETE'

```
'http://localhost:43424/api/v1/auth?id=GUID'
```

Parameter	Description
GUID	The organisations id, this can be found in Cambrionix connect by navigating to the organisations page and selecting the organisation you wish to find the id for. Please see https://connect.cambrionix.com for more information.

Returns:

```
  true
```

Example

```
curl -X 'DELETE' \
  'http://localhost:43424/api/v1/auth?id=00000000-0000-0000-
0000-000000000000' \
  -H 'accept: text/plain'
```

Returns

```
  true
```

Errors

If there is an error in the API method then a JSON-RPC Error Object will be returned

4.2. Certificate

Supply (or remove) a certificate and private key to the API to allow SSL connections from outside of localhost (the machine the API is running on). Without this certificate, the API will only listen for connections on localhost:43424. Once a valid certificate and private key are provided, this will change to 0.0.0.0:43424. External connections (not from localhost) will only be allowed if they are SSL connections (HTTPS or Secure WebSockets).

The API does not make a copy of the certificate or private key. The user that the API is running as will need access to the files to be able to use them. This is all tested when the "set" command is issued and should provide sufficient error information if it does not work.

It is up to the user to supply a certificate that is suitable for their usage. For example, if it is not signed by a certificate authority, then you will need to deal with this, such as signing your certificate with your own certificate authority and adding that to your application or browser.

- With Google Chrome you can use this guide.
- With Firefox you can use this guide.
- With Safari you can use this guide.

for other browsers refer to the browser supplier for more information.

Only a single certificate configuration is supported. If a password is supplied, it is stored and obfuscated for security.

Method	Description
Add Certificate	Get the certificates assigned to the API
Remove Certificate	Set the certificate for the API
Get Certificate	Remove the certificate from the API

4.2.1 Add Certificate

Add the certificate to the API

Syntax: see Call Structure

Type = 'POST'

```
'http://localhost:43424/api/v1/certificate' \
  -d '{
  "private-key": "Key-filename",
  "certificate": "certificate-filename",
  "password": "password"
}'
```

Parameter	Description
key-filename	The filename including the path of the private key
certificate-filename	The filename including the path of the certificate
password	optional password if required by private key

Returns

```
   true
```

Notes

- The files will need to be stored in a location that is accessible by the account under which the API runs

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/certificate' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
```

```
  "private-key": "C:\\Cambrionix\\certificate\\key.pem",
  "certificate": "C:\\Cambrionix\\certificate\\cert.pem"
}'
```

Returns

```
   true
```

Errors

If there is an error in the API method then a JSON-RPC Error Object will be returned

4.2.2 Get Certificate

Get the information on the certificate associated with the API

Syntax: see Call structure

Type = 'GET'

```
'http://localhost:43424/api/v1/certificates' \
```

Returns:

```
{
  "certificate": "Certificate",
  "subject": {
    "C": "Country",
    "L": "Location",
    "O": "Organisation",
     "CN": "Common name"
  },
  "issuer": {
    "C": "Country",
    "O": "Organisation",
    "CN": "Common name"
  },
  "serial_number": "Serial number",
  "validity": {
  "not_after": Vaild until,
  "not_before": Valid from
  },
  "algorithm": "Algorithm",
  "extensions": {
    "subjectAltName": ["Alternative names"]
  }
}
```

Variable	Description
Certificate	The public certificate in its entirety
Country	Country code

Variable	Description
Location	Specific Location company is registered
Organisation	The organisations name
Common name	The name the organisation is referred to in the certificate
Serial number	Used to uniquely identify the certificate within a CA's systems
Valid until	The time and date past which the certificate is no longer valid
Valid from	The earliest time and date on which the certificate is valid
Algorithm	This contain a hashing algorithm and a digital signature algorithm. For example "sha256RSA" where sha256 is the hashing algorithm and RSA is the signature algorithm
Alternative names	All name associated with the certificate

Example

```
curl -X 'GET'
  \'http://localhost:43424/api/v1/certificates' \
  -H 'accept: application/json'
```

Returns


```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIIGRDCCBSy-
gAwIBAgIQbeI1QvLxUbOlpVErFYf3jjANBgkqhkiG9w0BAQsFADCB\njzELMAkGA1UEBhMC
R0IxGzAZBgNVBAgTEk-
dyZWF0ZXIgTWFuY2hl-
c3RlcjEQMA4G\nA1UEBxMHU2FsZm9yZDEYMBYGA1UEChMPU2VjdGlnbyBMaW1pdGVkMTcwNQYDVQQD\
nEy5TZWN0aWd-
vIFJTQSBEb21haW4gVmFsaWRh-
dGlvbiBTZWN1cmUgU2VydmVyIENB\nMB4XDTI0MDcxNTAwMDAwMFoXDTI1MDcxNTIzNTk1OVowHzEdM
BsGA1UEAwwUKi5h\ncGkuY2FtYnJp-
b25peC5jb20wg-
gEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB\nAQC3WBp12hG0pFSmoWmjwmI7Y7soLHTQ
msPtMZ1q8P+i-
h+SCwKIZDTlI6GENm51c\nF0D2Pr-
```

```
wtANhSV1YGiveb4lZXfDhUuk2n+PLm/dTIT8lE0dtTpLz+gDxzF5HZ7cQR\nVw36nCRTo30gve+b
lWPtvVsk46gVMeQ6lYf7OOHAMpX0/t-
gwZ81ZExzSnFnN2tas\n-
q4P9rVGSXZQSuy54E7NOOOyUbZQ0MojVNlOMLOZ1ojW5RqBLokTnmqD6ob0kh+uC\n
6CTFD+o0JxgYc-
sfK+huxLd-
dSiKmSb-
izFwjXwLm-
lJP1ayzb/cdRlPrtA3FWVWahxe\ngQ8EfkBO2g+TzPLxFPTxr6zVAgMBAAGjggMJMIIDBTAfBgNVHSM
EGDAWgBSNjF7E\nVK2K4Xfp-
m/m-
bBeG4AY1h4TAdBgNVHQ4EFgQUt3v/N/ZPmg4ZSGHNUuFYWj2oREsw\nDgYDVR0PAQH/BAQDAgWgMAw-
GA1UdEwEB/wQC
MAAwHQYDVR0lBBYwFAYIKwYBBQUH\nAwEGCCsGAQUFBwMCMEkGA1UdIARCMEAwNAYLKwYBBAGyMQEC-
AgcwJTAjBggrBgEF\nBQcCARYXaHR0cHM6Ly9zZ
WN0aWd-
vLmNvbS9DUFMwCAYGZ4EMAQIBMIGEBg-
grBgEF\nBQcBAQR4MHYwTwYIKwYBBQUHMAKGQ2h0dHA6Ly9jcnQuc2VjdGlnby5jb20vU2Vj\ndGln
b1JTQURvbWFpblZh-
bGlkYXRp-
b25TZWN1cmVTZXJ2ZXJDQS5jcnQwIwYIKwYB\nBQUHMAGGF2h0dHA6Ly9vY3NwLnNlY3RpZ28uY29t-
MDMGA1UdEQQsMC
qCFCouYXBp\nLmNh-
bWJy-
aW9uaXguY29t-
ghJh-
cGkuY2FtYnJpb25peC5jb20wggF9BgorBgEEAdZ5\nAgQCBIIBbQSCAWkBZwB2AN3cyjSV1+EWBeeVM
vrHn/g9HFDf2wA6FBJ2Ciysu8gq\nAAABkLYpiZYAAAQDAEcwRQIgaVZ0j2uGG1785bQl4q6fAOoGB-
fFQOgYZ/LyQaIsy\n3UgCIQD1JsrjrUJfMIeV
T3+Yozq9zhFM1SkxLIn-
pBLj2/mIh-
nQB1AA3h8jAr0w3B\nQGISCepVLvxHdHyx1+k-
w7w5CHrR+Tqo0AAABkLYpiXkAAAQDAEYwRAIgbT1Nc3J0\nBfo
MuEeB/38b-
vc7zrIgYmtLc9SkNA2rlCEwCIC8jYOF0jC2J45oOFh1kDxWq-
duDC\nEJ4/jnhr/EN7OfkHAHYAEvFONL1TckyEBhnDjz96E/jntWKHiJxtM
AWE6+WGJ-
joA\nAAGQ-
timJdgAABAMARzBFAiEAon-
r/D5//hgp0xNpxw/TxEoZJpNdIaLY90McL5UAj\njFUCIHB/HrsQbgMr4EScfpFD4upUNUrJHes2Q
SKfisExFmtXMA0GCSqGSIb3DQEB\nCwUAA4IBAQA6B1JeTNLWw8oCBAGRIW0N7CMjvQQ033/xKK-
coaAMrkzYMJ/znOJVp\nsAqLM5rB0Xfly+qCDH9q5
VoyUi38IOJSi1RbDNlV4gM1+H/rPhO/ymUr0DitEENv\nIQk77kyCoZicWdtpSZ6l7yp7I2/F0kQJR-
QCiRbsk3TfEFmPaoNcVIlaFzFdSjlLx\n+PC7H
ALGzcY25pQZP56PRh-
pD763iJAyU1JOTOAx6l/kklnnZdAH7yiqDViMt4FO6\n-
bizq0SxIT8mXoMnDm+Rg0lU7uqQplaEOEkn4Elb83cKgPca/vt8d48Mb
fuWkdaEv\nQzFYgVXZtfgs0zW5beKzvs3DQZoG/gEl\n-----END CERTIFICATE-----\n",
    "subject": {
      "CN": "*.api.cambrionix.com"
    },
    "issuer": {
      "C": "GB",
      "ST": "Greater Manchester",
      "L": "Salford",
      "O": "Sectigo Limited",
      "CN": "Sectigo RSA Domain Validation Secure Server CA"
```

```
      },
      "serial_number": "6DE23542F2F151B3A5A5512B1587F78E",
      "algorithm": "sha256WithRSAEncryption",
      "extensions": {
        "subjectAltName": [
          "*.api.cambrionix.com",
          "api.cambrionix.com"
        ]
      },
      "validity": {
        "not_after": 1752623999,
        "not_before": 1721001600
      }
    }
  }
```


Errors

If there is an error in the API method then a JSON-RPC Error Object will be returned

4.2.3 Remove Certificate

Remove the certificate and private key from the API:

Syntax: see Call Structure

Type = 'DELETE'

```
'http://localhost:43424/api/v1/certificate'
```

Returns

```
true
```

Example

```
curl -X 'DELETE' \
  'http://localhost:43424/api/v1/certificate' \
  -H 'accept: application/json'
```

Returns

```
true
```

Errors

If there is an error in the API method then a JSON-RPC Error Object will be returned

4.3. Utility

Endpoints for general API operations, including retrieving and setting configuration, viewing API details, and exiting the service. These functions support overall API management and maintenance.

Method	Description
Get API Configuration	Get the API configuration
Set API Configuration	Set the API configuration
Get API Details	Get the details of the running API
Exit the API Service	Triggers a service restart when running as a service

4.3.1 Get API Configuration

This endpoint retrieves the current API configuration settings in place on the system. It provides a concise overview of key operational parameters, including battery update concurrency and frequency, whether battery updates are enabled, and console dimensions.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/config'
```

Returns:

```
{
  "result": {
    "battery-update-concurrency": battery-update-concurrency,
    "battery-update-enabled": battery-update-enabled,
    "battery-update-frequency-seconds": battery-update-
frequency,
    "console-height": console-height,
    "console-width": console-width
  }
}
```

Output	Description
battery-update-concurrency	How many devices will update battery information at the same time
battery-update-enabled	Boolean value as to whether the API updates battery information
battery-update-frequency	How often the battery information updates
console-height	The console height
console-width	The console width

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/config' \
  -H 'accept: application/json'
```

Returns

```
{
  "result": {
    "battery-update-concurrency": 22,
    "battery-update-enabled": true,
    "battery-update-frequency-seconds": 1,
    "console-height": 60,
    "console-width": 125
  }
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.3.2 Set API Configuration

This endpoint is used to set the API configuration options currently in effect on the system. It allows you to customise key operational parameters such as battery update behaviour (including concurrency, frequency, and whether updates are enabled)

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/config'
```

In the data section of the request, include the information you wish to add, formatted as shown below.

```
{
  "battery-update-concurrency": battery-update-concurrency,
  "battery-update-enabled": battery-update-enabled,
  "battery-update-frequency-seconds": battery-update-frequency,
}
```

Variable	Description
battery-update-concurrency	How many devices will update battery information at the same time
battery-update-enabled	Boolean value as to whether the API updates battery information
battery-update-frequency	How often the battery information updates

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/config' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "battery-update-concurrency": 6,
  "battery-update-enabled": true,
  "battery-update-frequency-seconds": 10
}'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.3.3 Get API Details

This endpoint provides detailed information about the current state and configuration of the API.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/details'
```

Returns:

```
{
  "version": [version-number],
  "semver": "semver-variant",
  "guid": {
    "id": "service-id",
    "computerId": "computer-id"
  },
  "host":
  [
    {
      "ip": "ip-address",
      "port": API-port,
      "domainName": "domain-name",
      "hostName": "host-name",
      "adapterName": "adapter-name",
      "adapterType": "adapter-type"
    },
    {
      "ip": "ip-address",
      "port": API-port,
      "domainName": "domain-name",
      "adapterName": "adapter-name",
      "adapterType": "adapter-type"
    }
  ],
  "commitid": commitid-number,
  "branch": "branch-name",
  "install": "install-location",
  "logging": "logs-location",
```

```
  "settings": "settings-location",
  "documentation": "documentation-location",
  "cpu": {
    "brand": "brand-information",
    "arch": "CPU-architecture",
    "features": [CPU-features],
    "cores": cores-value
  },
  "os": "OS-information"
}
```

Output	Description
version-number	Version number of API as an integer (Major,Minor,Revision,Build)
semver-variant	Version number of API as an string (Major,Minor,Revision,Build)
service-id	This is the id of the service as a unique 128-bit number
computer-id	This is the id of the computer as a unique 128-bit number
ip-address	The digital address for your internet-connected devices as a string
API-port	The port number the API uses as an integer
domain-name	The name of the network domain as a string
host-name	The name of the host computer as a string
adapter-name	The name of the network adaptor as a string
adapter-type	The type of network adaptor as a string
commitid-variant	The number value of the Commit ID as an integer
branch-name	The branch of API installed (release / beta) as a string
install-location	The file location of where the API is installed on the computer as a string
logs-location	The file location of where logs are stored as a string
settings-location	The file location of API settings as a string

Output	Description
documentation-location	The web address of API documentation
brand-information	The brand of the CPU as a string
CPU-architecture	The architecture of the CPU as a string
CPU-features	Features available on CPU displayed as an array
cores-value	How many cores the CPU has as an integer
os-information	Operating system running on local machine as a string

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/details' \
  -H 'accept: application/json'
```

Returns


```
{
  "result": {
    "version": [
      3,
      24,
      0,
      74
    ],
    "semver": "3.24.0",
    "guid": {
      "id": "d0dc3cac-e165-4e38-88bb-39064431bdc9",
      "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
    },
    "instance": {
      "started": "2025-04-08T23:19:22.536",
      "upTime": {
        "seconds": 641732,
        "description": "7d 10h 15m"
      }
    },
    "host": [
      {
        "ip": "10.167.111.95",
        "port": 43424,
        "domainName": "CBRX.LOCAL",
        "hostName": "CBRXPC-011",
```

```
        "adapterName": "Intel(R) Ethernet Controller (3) I225-V",
        "adapterType": "Ethernet"
      },
      {
        "ip": "172.27.128.1",
        "port": 43424,
        "hostName": "CBRXPC-011",
        "adapterName": "Hyper-V Virtual Ethernet Adapter",
        "adapterType": "Ethernet"
      }
    ],
    "commitid": 1798096184,
    "branch": "release",
    "install": "C:\\Program Files\\Cambrionix\\API",
    "logging": "C:\\ProgramData\\Cambrionix\\Log",
    "settings": "C:\\ProgramData\\Cambrionix",
    "documentation": "https://-
downloads.cambrionix.com/documentation/en/Cambrionix-Hub-API-User-Manual.pdf",
    "cpu": {
      "brand": "12th Gen Intel(R) Core(TM) i9-12900K",
      "arch": "x64",
      "features": [
        "aes",
        "avx",
        "avx2",
        "vpclmulqdq"
      ],
      "cores": 24
    },
    "os": "Windows 11 Pro 24H2 Build 26100.3775 64-bit"
  }
}
```


Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.3.4 Exit the API Service

When the API is running as a service, calling this endpoint will trigger a service restart. As a result, a response may not be returned, since the restart process can interrupt the connection before the server has a chance to reply.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/exit'
```

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/exit' \
  -H 'accept: text/plain'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.4. Devices

Endpoints for retrieving information about devices connected to Cambrionix hubs. These endpoints allow users to list all connected devices or access details for a specific device, supporting effective device monitoring and management within local workflows.

Method	Description
Get Devices	Get information on all connected devices
Get Device	Get information on a specific device

4.4.1 Get Devices

This endpoint returns a list of USB devices currently connected to the computer where the API is running. For each device, it provides detailed identification and classification data, including vendor and product IDs, product name, USB specifications (such as speed, version, and device class), and unique identifiers like the USB serial number.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/devices'
```

Returns:

```
[
  {
    "vendor": {
      "id": vendor-id,
      "name": "vendor-name"
    },
    "product": {
      "id": product-id,
      "name": "product-name"
    },
    "usb": {
      "speed": {
        "current": {
          "value": "usb-speed",
          "description": "usb-speed-description"
        }
      },
      "version": usb-version,
      "deviceClass": {
        "id": device-id,
        "description": "device-description"
      }
    },
    "identifiers": {
      "usbSerialNumber": "usb-serial",
      "vid": vendor-id,
      "pid": product-id
```

```
    },
    "location": {
      "id": location-id,
      "computerId": "computer-id"
    },
    "type": "device-type"
  }
]
```

Output	Description
vendor-id	Device Vendor ID number, VID. Displayed as an Integer
vendor-name	This is the name of the Vendor
product-id	Device Product ID number, PID. Displayed as an Integer
product-name	This is the name of the Product
usb-speed	Maximum speed USB connection capable of
usb-speed-description	Name of USB connection i.e. SuperSpeed USB 5Gbps
usb-version	The USB version number of the connection to the hub
device-id	This is the id of the device as a unique 128-bit number
device-description	This is a description of the device
usb-serial	The serial number of the device
location-id	This is the id of the location as a unique 128-bit number
computer-id	This is the id of the computer as a unique 128-bit number
device-type	This is the type of device i.e. mobile

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/devices' \
  -H 'accept: application/json'
```

Returns


```
{
    "result": [
      {
        "vendor": {
          "id": 1060,
          "name": ""
        },
        "product": {
          "id": 10060,
          "name": "USB Input Device"
        },
        "usb": {
          "speed": {
            "current": {
              "value": "480Mbps",
              "description": "High"
            }
          },
          "version": 2.01,
          "deviceClass": {
            "id": 0,
            "description": "device"
          }
        },
        "identifiers": {
          "usbSerialNumber": "7&cfb515b&0&5",
          "vid": 1060,
          "pid": 10060
        },
        "location": {
          "id": 625168384,
          "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
        },
        "type": "other"
      }
    ]
  }
```


Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.4.2 Get Device

This endpoint retrieves detailed information about a specific USB device connected to the computer where the API is running. It returns key identification data such as vendor and product IDs, the device name, USB specifications including speed, version, and class, as well as unique identifiers like the USB serial number. This allows you to accurately locate and inspect a single device from the list of connected hardware.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/devices/usb-serial'
```

Variable	Description
usb-serial	The serial number of the device

Returns:

```
[
  {
    "vendor": {
      "id": vendor-id,
      "name": "vendor-name"
    },
    "product": {
      "id": product-id,
      "name": "product-name"
    },
    "usb": {
      "speed": {
        "current": {
          "value": "usb-speed",
          "description": "usb-speed-description"
        }
      },
      "version": usb-version,
      "deviceClass": {
        "id": device-id,
        "description": "device-description"
```

```
      }
    },
    "identifiers": {
      "usbSerialNumber": "usb-serial",
      "vid": vendor-id,
      "pid": product-id
    },
    "location": {
      "id": location-id,
      "computerId": "computer-id"
    },
    "type": "device-type"
  }
]
```

Output	Description
vendor-id	Device Vendor ID number, VID. Displayed as an Integer
vendor-name	This is the name of the Vendor
product-id	Device Product ID number, PID. Displayed as an Integer
product-name	This is the name of the Product
usb-speed	Maximum speed USB connection capable of
usb-speed-description	Name of USB connection i.e. SuperSpeed USB 5Gbps
usb-version	The USB version number of the connection to the hub
device-id	This is the id of the device as a unique 128-bit number
device-description	This is a description of the device
usb-serial	The serial number of the device
location-id	This is the id of the location as a unique 128-bit number
computer-id	This is the id of the computer as a unique 128-bit number
device-type	This is the type of device i.e. mobile

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/devices/7&cfb515b&0&5' \
  -H 'accept: application/json'
```

Returns


```
{
    "result": [
      {
        "vendor": {
          "id": 1060,
          "name": ""
        },
        "product": {
          "id": 10060,
          "name": "USB Input Device"
        },
        "usb": {
          "speed": {
            "current": {
              "value": "480Mbps",
              "description": "High"
            }
          },
          "version": 2.01,
          "deviceClass": {
            "id": 0,
            "description": "device"
          }
        },
        "identifiers": {
          "usbSerialNumber": "7&cfb515b&0&5",
          "vid": 1060,
          "pid": 10060
        },
        "location": {
          "id": 625168384,
          "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
        },
        "type": "other"
      }
    ]
  }
```


Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.5. Hubs

Endpoints for accessing and managing individual hubs within your environment. These allow users to retrieve hub details, run ad-hoc commands, check firmware and hardware information, update NVRAM settings, and monitor endpoint usage. They support efficient hub configuration and diagnostics as part of broader hardware management workflows.

Method	Description
Get Hubs	This endpoint returns key details and configuration information for hubs connected to the API host.
Get Hub	This endpoint returns key details and configuration information for a specific hub connected to the API host.
Send Hub Command	This endpoint sends a command to a specific hub, enabling direct interaction and real-time control.
Get Hub Endpoints	This endpoint returns detailed USB endpoint usage for a specific hub, with an option to combine USB 2.0 and 3.0 data for non-Thunderbolt hubs.
Get Hub Firmware	This endpoint returns all firmware details for a specific hub, including versions and hardware information.
Update Hub Firmware	This endpoint updates the firmware on a specified hub
Get Hub Firmware Update	This endpoint returns the current status and progress of a firmware update for a specific hub.
Get Default Power Limits	Retrieves the default power limits for each port, overall hub power usage, and the range of allowable configuration wattages per port
Load Power Configuration	This endpoint restores the hub's power configuration to the most recently saved settings
Save the Power Configuration	This endpoint saves the current power configuration on the hub
Set Hub Settings	This endpoint updates a hub's internal settings in real time using configuration data sent with the request.

4.5.1 Get Hubs

This endpoint returns information about all hubs connected to the computer running the API, including each hub's serial number, product name, connection status, firmware version, and basic configuration details.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs'
```

Returns:

```
[
  {
    "serialNumber": "hub-serial",
    "product": {
      "id": "product-id",
      "name": "hub-name"
    },
    "host": {
      "connection": {
        "status": "connection-status",
        "updated": time,
          "speed": {
            "current": {
              "value": "usb-speed",
              "description": "usb-speed-description"
            }
          }
        }
      },
      "firmware": {
        "version": "firmware-version",
        "type": "firmware-type"
      },
      "settings": {hub-settings},
      "serialPort": {
        "status": "connection-status",
        "location": {
          "computerId": "computer-id",
```

```
          "locationId": location-id
        },
        "connection": "serial-port"
      },
      "limits": [hub-limits]
    }
  }
]
```

Output	Description
hub-serial	This is the serial number of the hub
product-id	Device Product ID number, PID. Displayed as an Integer
hub-name	Name of the hub
connection-status	The connection status i.e. connected
time	The hub time, shown as an integer in ms
usb-speed	Maximum speed USB connection capable of
usb-speed-description	Name of USB connection i.e. SuperSpeed USB 5Gbps
firmware-version	Version number of the firmware. Format 'N.nn'
firmware-type	Used to denote which firmware the product accepts
hub-settings	The hub internal settings
location-id	This is the id of the location as a unique 128-bit number
computer-id	This is the id of the computer as a unique 128-bit number
serial-port	The Serial port the product is connected to.
hub-limits	This is a list of limits on the hub that will set error flags e.g. under and over current limits

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs' \
  -H 'accept: application/json'
```

Returns


```
{
  "result": [
    {
      "serialNumber": "string",
      "product": {
        "id": "SuperSync15",
        "name": "SuperSync15"
      },
      "host": {
        "connection": {
          "status": "connected",
          "updated": 1729757954795,
          "speed": {
            "current": {
              "value": "5Gbps",
              "description": "SuperSpeed USB 5Gbps"
            }
          }
        }
      },
      "firmware": {
        "version": "1.2.0",
        "type": "st"
      },
      "settings": {
        "stagger": 50,
        "stagger_offset": 50
      },
      "serialPort": {
        "status": "connected",
        "location": {
          "computerId": "01234567-89ab-cdef-0123-456789abcdef",
          "locationId": 627113984
        },
        "connection": "COM3"
      },
      "limits": [
        {
          "type": "usb",
          "unit": "volts",
          "min": 4.5,
          "max": 5.5
```

```
            }
          ]
        }
      ]
    }
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.5.2 Get Hub

You can use this endpoint to retrieve detailed information about a specific hub by including its serial number in the request. The response provides key details such as the hub's serial number, product name, connection status, firmware version, and core configuration settings. Additionally, it includes information about the device's serial port and any configured operating limits.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial'
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
[
  {
    "serialNumber": "hub-serial",
    "product": {
      "id": "product-id",
      "name": "hub-name"
    },
    "host": {
      "connection": {
        "status": "connection-status",
        "updated": time,
          "speed": {
            "current": {
              "value": "usb-speed",
              "description": "usb-speed-description"
            }
          }
        }
      },
      "firmware": {
        "version": "firmware-version",
```

```
        "type": "firmware-type"
      },
      "settings": {hub-settings},
      "serialPort": {
        "status": "connection-status",
        "location": {
          "computerId": "computer-id",
          "locationId": location-id
        },
        "connection": "serial-port"
      },
      "limits": [hub-limits]
    }
  }
]
```

Output	Description
hub-serial	This is the serial number of the hub
product-id	Device Product ID number, PID. Displayed as an Integer
hub-name	Name of the hub
connection-status	The connection status i.e. connected
time	The hub time, shown as an integer in ms
usb-speed	Maximum speed USB connection capable of
usb-speed-description	Name of USB connection i.e. SuperSpeed USB 5Gbps
firmware-version	Version number of the firmware. Format 'N.nn'
firmware-type	Used to denote which firmware the product accepts
hub-settings	The hub internal settings
location-id	This is the id of the location as a unique 128-bit number
computer-id	This is the id of the computer as a unique 128-bit number
serial-port	The Serial port the product is connected to.

Output	Description
hub-limits	This is a list of limits on the hub that will set error flags e.g. under and over current limits

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/47663017' \
  -H 'accept: application/json'
```

Returns


```
{
  "result": [
    {
      "serialNumber": "47663017",
      "product": {
        "id": "SuperSync15",
        "name": "SuperSync15"
      },
      "host": {
        "connection": {
          "status": "connected",
          "updated": 1729757954795,
          "speed": {
            "current": {
              "value": "5Gbps",
              "description": "SuperSpeed USB 5Gbps"
            }
          }
        }
      },
      "firmware": {
        "version": "1.2.0",
        "type": "st"
      },
      "settings": {
        "stagger": 50,
        "stagger_offset": 50
      },
      "serialPort": {
        "status": "connected",
        "location": {
          "computerId": "01234567-89ab-cdef-0123-456789abcdef",
          "locationId": 627113984
        },
        "connection": "COM3"
      },
      "limits": [
```

```
            {
              "type": "usb",
              "unit": "volts",
              "min": 4.5,
              "max": 5.5
            }
          ]
        }
      ]
    }
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.5.3 Send Hub Command

This endpoint allows you to send a specific command to a hub by including the hub's serial number in the request. It enables direct communication with the device, allowing you to trigger specific actions or configure settings as needed. By specifying the command in the request body, you can interact with the hub's functions, such as adjusting configurations, querying status, or controlling behaviour in real time.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/command' \
```

Variable	Description
hub-serial	The serial number of the hub

In the data section of the request, put each CLI command you wish to send on a separate line, for information on the different CLI commands available please see the user manual.

https://www.cambrionix.com/pages/cambrionix-cli

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/command' \
  -H 'accept: text/plain' \
  -H 'Content-Type: text/plain' \
  -d 'id
state
'
```

Returns

```
>> id
mfr:cambrionix,mode:main,hw:SuperSync15,hwid:0x30,fw:1.89.4, bl:0.21,s-
n:0000007B6BD6C7A6,group:-,fc:un

>> state
 1, 0000, R D S, 0, 0, x, 0.00
```

```
 2, 0000, R D S, 0, 0, x, 0.00
 3, 0000, R D S, 0, 0, x, 0.00
 4, 0000, R D S, 0, 0, x, 0.00
 5, 0000, R D S, 0, 0, x, 0.00
 6, 0000, R D S, 0, 0, x, 0.00
 7, 0000, R D S, 0, 0, x, 0.00
 8, 0000, R D S, 0, 0, x, 0.00
 9, 0000, R D S, 0, 0, x, 0.00
10, 0000, R D S, 0, 0, x, 0.00
11, 0000, R D S, 0, 0, x, 0.00
12, 0000, R D S, 0, 0, x, 0.00
13, 0000, R D S, 0, 0, x, 0.00
14, 0000, R D S, 0, 0, x, 0.00
15, 0000, R D S, 0, 0, x, 0.00
```

4.5.4 Get Hub Endpoints

You can use this endpoint to get detailed USB endpoint usage information for a specific hub by including the hub's serial number in the request. The response includes key data such as memory usage, the number of endpoints in use, and configuration details across the hub and its connected ports.

The endpoint also supports an optional combined parameter. When set to true, it merges USB 2.0 and USB 3.0 data for non-Thunderbolt hubs. This is useful for viewing a simplified overview of endpoint usage. By default, this parameter is set to false.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/endpoints?combined=combined'
```

Variable	Description
hub-serial	The serial number of the hub
combined	A Boolean value if you want the USB2 and USB3 values combined

Returns:

```
{
  "hubs": [
    {
      "usage": {
        "self": {
          "memory": endpoint-memory,
          "endpoints": endpoints
        },
        "ports-only": {
          "memory": endpoint-memory,
          "endpoints": endpoints
        },
        "self-with-ports": {
          "memory": endpoint-memory,
          "endpoints": endpoints
```

```
        },
        "self-with-ports-recursive": {
          "memory": endpoint-memory,
          "endpoints": endpoints
        }
      },
      "info": {
        "hardware": "hub-name",
        "node-type": "node-type",
        "locationId": location-id
      },
      "ports": [
        {
          "id": -1,
          "name": "device-name",
          "serial": "device-serial",
          "usb": {
            "version": usb-version,
            "speed": "usb-speed"
          },
          "locationId": location-id,
          "self": {
            "memory": endpoint-memory,
            "endpoints": endpoints
          },
          "all": {
            "memory": endpoint-memory,
            "endpoints": endpoints
          }
        }
      ]
    }
  ]
}
```

Output	Description
endpoint-memory	This is the amount of endpoint memory consumed in bytes
endpoints	This the the amount of endpoints
hub-name	Name of the hub
node-type	This is the USB node

Output	Description
location-id	This is the id of the location as a unique 128-bit number
device-name	The is the name of the device
device-serial	This is the serial number of the device
usb-version	The USB version number of the connection to the hub
usb-speed	Maximum speed USB connection capable of

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/endpoints' \
  -H 'accept: application/json'
```

Returns


```
{
  "result": {
    "hubs": [
      {
        "usage": {
          "self": {
            "memory": 57344,
            "endpoints": 14
          },
          "ports-only": {
            "memory": 0,
            "endpoints": 0
          },
          "self-with-ports": {
            "memory": 57344,
            "endpoints": 14
          },
          "self-with-ports-recursive": {
            "memory": 57344,
            "endpoints": 14
          }
        },
        "info": {
          "hardware": "SuperSync15",
          "node-type": "USB2 Root",
          "locationId": 621805568
        },
        "ports": []
      },
```

```
        {
          "usage": {
            "self": {
              "memory": 81920,
              "endpoints": 10
            },
            "ports-only": {
              "memory": 90112,
              "endpoints": 11
            },
            "self-with-ports": {
              "memory": 172032,
              "endpoints": 21
            },
            "self-with-ports-recursive": {
              "memory": 172032,
              "endpoints": 21
            }
          },
          "info": {
            "hardware": "SuperSync15",
            "node-type": "USB3 Root",
            "locationId": 2769289216
          },
          "ports": [
            {
              "id": -1,
              "name": "SAMSUNG SAMSUNG_Android",
              "serial": "RFCQ8LJM",
              "usb": {
                "version": 3.2,
                "speed": "5Gbps"
              },
              "locationId": 2769424384,
              "self": {
                "memory": 90112,
                "endpoints": 11
              },
              "all": {
                "memory": 90112,
                "endpoints": 11
              }
            }
          ]
        }
      ]
    }
  }
```


Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.5.5 Get Hub Firmware

You can use this endpoint to retrieve the firmware details for a specific hub by including the hub's serial number in the request. The response returns all types of firmware required for that hub, including the main firmware version, the bootloader version, hardware ID, and product description.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/hardware'
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
[
  {
    "description": "hardware-description",
    "firmware": {
      "version": "firmware-version",
      "type": "firmware-type",
      "hardwareId": hardware-Id
    },
    "bootloader": {
      "version": "bootloader-version"
    },
    "serialNumber": "hub-serial"
  }
]
```

Output	Description
hardware-description	This is a description of the hardware, defaults to the hub name
firmware-version	Version number of the firmware. Format 'N.nn'
firmware-type	Used to denote which firmware the product accepts

Output	Description
hardware-Id	This is a hardware Id for the hub
bootloader-version	Version number of the bootloader. Format 'N.nn'
hub-serial	The serial number of the hub

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/hardware' \
  -H 'accept: application/json'
```

Returns

```
{
  "result": [
    {
      "description": "SuperSync15",
      "firmware": {
        "version": "1.89.4",
        "type": "un",
        "hardwareId": 48
      },
      "bootloader": {
        "version": "0.21"
      },
      "serialNumber": "0000007B6BD6C7A6"
    }
  ]
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.5.6 Update Hub Firmware

This endpoint updates the firmware on a specified hub using a previously uploaded firmware file. You must supply the hub's serial number and the details of the firmware file to apply the update. The firmware file must have been previously added to the API using theAdd Firmware endpoint. This operation ensures that hubs can be maintained with the latest supported firmware versions.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/firm-
ware/update'
```

Variable	Description
hub-serial	The serial number of the hub

In the data section of the request, include the information you wish to add, formatted as shown below.

```
[
  {
    "filename": "firmware-filename",
    "type": "firmware-type"
  }
]
```

Output	Description
firmware-filename	The filename of the firmware
firmware-type	Used to denote which firmware the product accepts

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/47663017/firmware/update' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '[
  {
    "filename": "CambrionixFirmware-v1.88.7-un.enfir",
    "type": "un"
  }
]'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.5.7 Get Hub Firmware Update

You can use this endpoint to check the current state of a firmware update for a specific hub by including the hub's serial number in the request. The response provides information about the progress and status of the update, including which firmware components are being updated, their current versions, and any relevant status messages or errors.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/firmware/update'
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
{
  "type": "firmware-type",
  "fromVersion": "firmware-version",
  "version": "firmware-version",
  "startTime": time,
  "endTime": time,
  "stage": "stage-value"
  "progress": progress-percentage
}
```

Output	Description
firmware-type	Used to denote the type of firmware
firmware-version	Version number of the firmware
time	The hub time, shown as an integer in ms
stage-value	The "stage" the firmware update is currently in
progress-percentage	The update progress as a percentage

stage-value	Description
none	Firmware is not being updated
connecting	Connecting to the hub to update the firmware
init	The update is initialising
erasing	Erasing current firmware
erased	Current firmware has been erased
updating	New firmware is being installed
updated	New firmware has finished being installed
verifying	Checking that the firmware has installed correctly
complete	The check has completed
rebooting	Rebooting the hub after all checks and installation complete
rebooted	Hub has been rebooted and is ready for use
skipped	Update skipped as hub already updated

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/firmware/update' \
  -H 'accept: application/json'
```

Returns


```
{
  "result": {
    "progress": 43,
    "type": "main",
    "version": {
      "old": "1.89.4",
      "new": "1.89.3"
    },
    "time": {
      "start": 17453237918083308,
      "end": 0
    },
    "stage": "updating"
  }
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

If there is an error in the stage then one of the below errors will appear in the stage values. Any of these stage errors mean that the hub's firmware is in an invalid state and would need to be re-done.

stage-error	Description
crypt-init-failed	The wrong type of firmware was used for the selected device
init-failed	The initialisation stage failed
erase-failed	The current firmware could not be erased
flash-failed	The new firmware could not be installed onto the hub
check-failed	The installation checks failed
reboot-failed	The hub was unable to be rebooted

4.5.8 Get Default Power Limits

Retrieves the default power limits for each port, overall hub power usage, and the range of allowable configuration wattages per port. This endpoint is only applicable to Smart Hubs that support Power Delivery (e.g., ThunderSync5-C16 PD).

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/power/limits'
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
{
  "port": {
    "maximum": power,
    "allowed": [power]
  },
  "hub": {
    "total": power,
    "used": power
  },
  "unit": "power-unit"
}
```

Output	Description
power	This is the power value
power-unit	This is the unit of the power value

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/47663017/power/limits' \
  -H 'accept: application/json'
```

Returns


```
{
  "result": {
    "port": {
      "maximum": 60000,
      "allowed": [
        4500,
        15000,
        27000,
        36000,
        45000,
        60000
      ]
    },
    "hub": {
      "total": 300000,
      "used": 0
    },
    "unit": "milliwatts"
  }
}
```


Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.5.9 Load Power Configuration

This endpoint restores the hub's power configuration to the most recently saved settings. Applicable only to Smart Hubs with Power Delivery support (e.g., ThunderSync5-C16 PD).

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/power/setup/revert'
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/47663017/power/setup/revert' \
  -H 'accept: application/json' \
  -d ''
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.5.10 Save the Power Configuration

This endpoint saves the current power configuration on the hub, ensuring your settings are retained after a reboot. Applicable only to Smart Hubs with Power Delivery support (e.g., ThunderSync5-C16 PD).

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/power/setup/save'
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/47663017/power/setup/save' \
  -H 'accept: application/json' \
  -d ''
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.5.11 Set Hub Settings

You can use this endpoint to configure the internal settings of a hub by including the hub's serial number in the request. It allows you to send specific configuration data directly to the device, enabling real-time updates to its internal setup. The settings are defined in the request body and applied upon execution.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/settings' \
```

Variable	Description
hub-serial	The serial number of the hub

In the data section of the request, put the internal settings you wish to update and the value you wish to update, please see individual hub user manuals for the settings that are applicable and information on the setting.

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/settings' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "attach_threshold": 0
}'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6. Ports

Endpoints for managing and monitoring the ports on a hub. These allow users to view port states, control LEDs, set port modes, and check supported configurations. Functionality is available for both individual ports and all ports on a hub, enabling detailed port-level control and customisation across devices.

Method	Description
Get Hub Ports State	This endpoint returns the current state of each port
Set Hub Ports LED	This endpoint allows you to set the LED colour for all port
Get Hub Ports LED	Retrieves the LED state for all ports
Set Hub Ports Mode	This endpoint is used to set the port modes for all port
Get Hub Ports Supported Modes	Returns the supported port modes for the hub
Get Ports Power Limits	This endpoint returns the current state of each port
Set Ports Power Limits	This endpoint sets the maximum power limit that each port
Get Hub Port State	This endpoint retrieves the current state of a specific port
Set Hub Port LED	This endpoint allows you to set the LED colour for a specific port
Get Hub Port LED	This endpoint retrieves the LED state for a specific port
Set Hub Port Mode	This endpoint sets the mode for a specified port
Get Port Power Limits	This endpoint retrieves the current power limit configured for a specific port
Set Port Power Limits	This endpoint allows you to set the maximum power limit for an individual port

4.6.1 Get Hub Ports State

This endpoint returns the current state of each port on a specific hub by including the hub's serial number in the request. The response provides details for each port, such as its ID, connection location, and the computer it is associated with. It also includes the port's state, indicating whether a device is attached and the current mode (e.g., "on"). This allows for real-time monitoring of port activity and operational status across the hub.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports'
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
[
  {
    "id": port-number,
    "location": {
      "port": {
        "hub": {
          "serialNumber": "hub-serial"
        },
        "id": port-number
      },
      "id": location-id,
      "computerId": "computer-id"
    },
    "state": {
      "attached": attached,
      "mode": "port-mode"
    }
  }
]
```

Output	Description
hub-serial	This is the serial number of the hub
port-number	The number of the port
location-id	This is the id of the location as a unique 128-bit number
computer-id	This is the id of the computer as a unique 128-bit number
attached	Boolean value if the port has a device attached
port-mode	The mode the port is in, see individual product user manual for information on available port modes

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/ports' \
  -H 'accept: application/json'
```

Returns

This response has been truncated for clarity within the manual.


```
{
  "result": [
    {
      "id": 1,
      "location": {
        "port": {
          "hub": {
            "serialNumber": "0000007B6BD6C7A6"
          },
          "id": 1
        },
        "id": 621875200,
        "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
      },
      "state": {
        "attached": false,
        "mode": "on"
      }
    },
```

```
      {
        "id": 2,
        "location": {
          "port": {
            "hub": {
              "serialNumber": "0000007B6BD6C7A6"
            },
            "id": 2
          },
          "id": 621879296,
          "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
        },
        "state": {
          "attached": false,
          "mode": "on"
        }
      },
      ............
      {
        "id": 14,
        "location": {
          "port": {
            "hub": {
              "serialNumber": "0000007B6BD6C7A6"
            },
            "id": 14
          },
          "id": 621953024,
          "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
        },
        "state": {
          "attached": false,
          "mode": "on"
        }
      },
      {
        "id": 15,
        "location": {
          "port": {
            "hub": {
              "serialNumber": "0000007B6BD6C7A6"
            },
            "id": 15
          },
          "id": 621948928,
          "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
        },
        "state": {
          "attached": false,
          "mode": "on"
        }
      }
    ]
  }
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.2 Set Hub Ports LED

This endpoint allows you to set the LED colour for all ports on a specified hub simultaneously. By providing the hub's serial number and the desired LED colour configuration in the request payload, you can update the LED status across every port at once.

This operation is applicable to Smart hubs that support RGB per-port LED management (e.g., ThunderSync5-C16 PD).

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/leds'
```

Variable	Description
hub-serial	The serial number of the hub

In the data section of the request, include the information you wish to add, formatted as shown below.

```
{
  "color": "colour"
}
```

Variable	Description
colour	The colour you want the LED to be from the list below

- red
- darkred
- green
- darkgreen
- blue
- darkblue
- magenta
- darkmagenta
- yellow
- brown
- cyan
- darkcyan

- black
- white
- grey

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/47663017/ports/leds' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "color": "red"
}'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.3 Get Hub Ports LED

Retrieves the LED state for all ports on the specified hub. A response is returned only if the hub supports this feature (e.g., ThunderSync5-C16 PD).

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/leds'
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
[
  {
    "id": port-number,
    "mode": "led-mode",
    "led": {
      "pattern": pattern-code,
      "color": {
        "red": rgb-value,
        "green": rgb-value,
        "blue": rgb-value
      },
      "brightness": {
        "low": led-intensity,
        "high": led-intensity
      }
    }
  }
]
```

Output	Description
port-number	The number of the port
led-mode	This is the LED mode i.e. on / off

Output	Description
pattern-code	A bit pattern describing LED flashing between low and high brightness.
rgb-value	This is the value of the colour component
led-intensity	This is how bright the LED is

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/47663017/ports/leds' \
  -H 'accept: application/json'
```

Returns

```
{
  "result": [
    {
      "id": 1,
      "mode": "off"
    },
    {
      "id": 2,
      "mode": "off"
    },
    {
      "id": 3,
      "mode": "on",
      "led": {
        "pattern": 32768,
        "color": {
          "red": 0,
          "green": 255,
          "blue": 0
        },
        "brightness": {
          "low": 192,
          "high": 255
        }
      }
    },
    {
      "id": 4,
      "mode": "off"
    },
    {
      "id": 5,
      "mode": "off"
```

```
      },
      {
        "id": 6,
        "mode": "off"
      },
      {
        "id": 7,
        "mode": "off"
      },
      {
        "id": 8,
        "mode": "off"
      },
      {
        "id": 9,
        "mode": "off"
      },
      {
        "id": 10,
        "mode": "off"
      },
      {
        "id": 11,
        "mode": "off"
      },
      {
        "id": 12,
        "mode": "off"
      },
      {
        "id": 13,
        "mode": "off"
      },
      {
        "id": 14,
        "mode": "off"
      },
      {
        "id": 15,
        "mode": "off"
      },
      {
        "id": 16,
        "mode": "off"
      }
    ]
  }
```


Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.4 Set Hub Port Mode

This endpoint sets the mode for a specified port on the connected hub.

You must provide the hub's serial number and the target port number along with the desired mode in the request. Only the specified port's mode will be updated without affecting the modes of other ports on the hub.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/port-number/mode'
```

Variable	Description
hub-serial	The serial number of the hub
port-number	The number of the port

In the data section of the request, include the information you wish to add, formatted as shown below.

```
{
  "mode": "port-mode"
}
```

Variable	Description
port-mode	This is the mode of the port see individual product manuals for the applicable modes

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/47663017/ports/1/mode' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "mode": "on"
}'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.5 Get Hub Ports Supported Modes

Returns the supported port modes for the hub, along with descriptive information for each mode. This allows clients to understand the valid configuration options available for managing port behaviour.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/-modes/supported'
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
{
  "mode": "port-mode",
  "description": "port-mode"
}
```

Output	Description
port-mode	This is the mode of the port see individual product manuals for the applicable modes
mode-description	This is a description of the port mode

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/ports/modes/supported' \
  -H 'accept: application/json'
```

Returns

```
{
  "result": [
    {
      "mode": "on",
      "description": "Enable the port and allow USB data transfer"
    },
    {
      "mode": "off",
      "description": "Disable the port"
    },
    {
      "mode": "charge",
      "description": "Enable the port for charging only, with no USB data trans-
fer."
    },
    {
      "mode": "detect",
      "description": "Enable the port for device detection only, with no USB
data transfer or charging."
    }
  ]
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.6 Get Ports Power Limits

This endpoint returns the current state of each port on a specific hub by including the hub's serial number in the request. The response provides details for each port, such as its ID, connection location, and the computer it is associated with. It also includes the port's state, indicating whether a device is attached and the current mode (e.g., "on"). This allows for real-time monitoring of port activity and operational status across the hub.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/-power/limit'
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
{
  "id": port-number
  "maximum": power,
  "allowed": [power]
  },
  "hub": {
    "total": power,
    "used": power
  },
  "unit": "power-unit"
}
```

Output	Description
port-number	The number of the port
power	This is the power value
power-unit	This is the unit of the power value

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/47663017/ports/power/limit' \
  -H 'accept: application/json'
```

Returns


```
{
  "result": [
    {
      "id": 1,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 2,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 3,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 4,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 5,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 6,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
```

```
      }
    },
    {
      "id": 7,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 8,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 9,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 10,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 11,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 12,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 13,
      "maximum": {
        "value": 60000,
        "unit": "milliwatts"
      }
    },
    {
      "id": 14,
      "maximum": {
        "value": 60000,
```

```
          "unit": "milliwatts"
        }
      },
      {
        "id": 15,
        "maximum": {
          "value": 60000,
          "unit": "milliwatts"
        }
      },
      {
        "id": 16,
        "maximum": {
          "value": 60000,
          "unit": "milliwatts"
        }
      }
    ]
  }
```


Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.7 Set Ports Power Limits

This endpoint sets the maximum power limit that each port on a specified hub can deliver. It is only applicable to Smart hubs that support Power Delivery (e.g., ThunderSync4-C16 PD). The power limit is applied across all ports on the hub as defined in the request payload. Ports that already have a device connected when this setting is applied will not be affected.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/-power/limit'
```

Variable	Description
hub-serial	The serial number of the hub

In the data section of the request, include the information you wish to add, formatted as shown below.

```
{
  "limit": power
}
```

Variable	Description
power	This is the power value in mA

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
```

```
  'http://localhost:43424/api/v1/hubs/DJ00V8A/ports/power/limit' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "limit": 60000
}'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.8 Get Hub Port State

This endpoint retrieves the current state of a specific port on a hub by including the hub's serial number and the port ID in the request. The response includes the port's status, such as whether a device is attached and the current mode, along with sensor readings like voltage and current. If a device is connected, additional information is provided, including USB speed, version, vendor and product details, serial number, and device type.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/port-number'
```

Variable	Description
hub-serial	The serial number of the hub
port-number	This is the number of the port

Returns:

```
[
  {
    "id": port-number,
    "location": {
      "port": {
        "hub": {
          "serialNumber": "hub-serial"
        },
        "id": port-number
      },
      "id": location-id,
      "computerId": "computer-id"
    },
    "state": {
      "attached": attached,
      "mode": "port-mode"
```

```
    },
  },
  "sensors": [
    {
      "type": "sensor-type",
      "value": sensor-value
    },
  ],
  "device": {
    "vendor": {
      "id": vendor-id,
      "name": "vendor-name"
    },
    "product": {
      "id": product-id,
      "name": "product-name"
    },
    "usb": {
      "speed": {
        "current": {
          "value": "usb-speed",
          "description": "usb-speed-description"
        },
        "supported": {
          "value": "usb-speed",
          "description": "usb-speed-description"
        }
      },
      "version": usb-version,
      "deviceClass": {
        "id": device-id,
        "description": "device-description"
      }
    },
    "identifiers": {
      "usbSerialNumber": "usb-serial",
      "commonId": "common-id",
      "vid": vendor-id,
      "pid": product-id
    },
    "location": {
      "port": {
        "hub": {
          "serialNumber": "hub-serial"
          },
```

```
        "id": port-number
      },
      "id": location-id,
      "computerId": "computer-id"
    },
    "type": "device-type",
    "name": "device-name"
  }
}
```

Output	Description
hub-serial	This is the serial number of the hub
port-number	The number of the port
location-id	This is the id of the location as a unique 128-bit number
computer-id	This is the id of the computer as a unique 128-bit number
attached	Boolean value if the port has a device attached
port-mode	The mode the port is in, see individual product user manual for information on available port modes
sensor-type	This is the sensor type e.g. miliamps
sensor-value	The is the reported value of the sensor e.g. amount of amps
vendor-id	Device Vendor ID number, VID. Displayed as an Integer
vendor-name	This is the name of the Vendor
product-id	Device Product ID number, PID. Displayed as an Integer
product-name	This is the name of the Product
usb-speed	Maximum speed USB connection capable of
usb-speed-description	Name of USB connection i.e. SuperSpeed USB 5Gbps
usb-version	The USB version number of the connection to the hub
device-id	This is the id of the device as a unique 128-bit number

Output	Description
device-description	This is a description of the device
usb-serial	The serial number of the device
common-id	This is the common id of the device as a unique 128-bit number
hub-serial	The serial number of the hub
port-number	This is the number of the port
location-id	This is the id of the location as a unique 128-bit number
computer-id	This is the id of the computer as a unique 128-bit number
device-type	This is the type of device i.e. mobile

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/ports/6' \
  -H 'accept: application/json'
```

Returns

```
{
  "result": {
    "id": 6,
    "location": {
      "port": {
        "hub": {
          "serialNumber": "0000007B6BD6C7A6"
        },
        "id": 6
      },
      "id": 2769424384,
      "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
    },
    "state": {
      "attached": true,
      "mode": "on"
    },
    "sensors": [
      {
        "type": "milliamps",
```

```
          "value": 1386
        },
        {
          "type": "volts",
          "value": 5.2
        }
      ],
      "device": {
        "vendor": {
          "id": 1256,
          "name": "SAMSUNG"
        },
        "product": {
          "id": 26720,
          "name": "SAMSUNG_Android"
        },
        "usb": {
          "speed": {
            "current": {
              "value": "5Gbps",
              "description": "SuperSpeed USB 5Gbps"
            },
            "supported": {
              "value": "10Gbps",
              "description": "SuperSpeed USB 10Gbps"
            }
          },
          "version": 3.2,
          "deviceClass": {
            "id": 0,
            "description": "device"
          }
        },
        "identifiers": {
          "usbSerialNumber": "RFCN20Q8LJM",
          "commonId": "fc622830-f5c6-5fec-864d-784bb11ea81b",
          "vid": 1256,
          "pid": 26720
        },
        "location": {
          "port": {
            "hub": {
              "serialNumber": "0000007B6BD6C7A6"
            },
            "id": 6
          },
          "id": 2769424384,
          "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
        },
        "type": "mobile",
        "name": "SAMSUNG SAMSUNG_Android"
      }
    }
  }
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.9 Set Hub Port LED

This endpoint allows you to set the LED colour for a specific port on a designated hub. By providing the hub's serial number and the desired port number in the request payload, you can control the LED status individually.

It is applicable to Smart hubs that support RGB per-port LED management (e.g., ThunderSync5-C16 PD).

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/port-number/led'
```

Variable	Description
hub-serial	The serial number of the hub
port-number	The number of the port

In the data section of the request, include the information you wish to add, formatted as shown below.

```
{
  "color": "colour"
}
```

Variable	Description
colour	The colour you want the LED to be from the list below

- red
- darkred
- green
- darkgreen
- blue
- darkblue
- magenta
- darkmagenta
- yellow

- brown
- cyan
- darkcyan
- black
- white
- grey

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/47663017/ports/1/led' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "color": "red"
}'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.10 Get Hub Port LED

This endpoint retrieves the LED state for a specific port on a designated hub.

You must provide the hub's serial number and the port number in the request. A response will be returned only if the hub supports per-port RGB LED state retrieval (e.g., ThunderSync5-C16 PD).

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/port-number/led'
```

Variable	Description
hub-serial	The serial number of the hub
port-number	The number of the port

Returns:

```
[
  {
    "id": port-number,
    "mode": "led-mode",
    "led": {
      "pattern": pattern-code,
      "color": {
        "red": rgb-value,
        "green": rgb-value,
        "blue": rgb-value
      },
      "brightness": {
        "low": led-intensity,
        "high": led-intensity
      }
    }
  }
]
```

Output	Description
port-number	The number of the port
led-mode	This is the LED mode i.e. on / off
pattern-code	A bit pattern describing LED flashing between low and high brightness.
rgb-value	This is the value of the colour component
led-intensity	This is how bright the LED is

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/47663017/ports/3/led' \
  -H 'accept: application/json'
```

Returns

```
{
  "result": [
    {
      "id": 3,
      "mode": "on",
      "led": {
        "pattern": 32768,
        "color": {
          "red": 0,
          "green": 255,
          "blue": 0
        },
        "brightness": {
          "low": 192,
          "high": 255
        }
      }
    }
  ]
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/47663017/ports/1/led' \
  -H 'accept: application/json'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.11 Set Hub Port Mode

This endpoint sets the mode for a specified port on the connected hub.

You must provide the hub's serial number and the target port number along with the desired mode in the request. Only the specified port's mode will be updated without affecting the modes of other ports on the hub.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/port-number/mode'
```

Variable	Description
hub-serial	The serial number of the hub
port-number	The number of the port

In the data section of the request, include the information you wish to add, formatted as shown below.

```
{
  "mode": "port-mode"
}
```

Variable	Description
port-mode	This is the mode of the port see individual product manuals for the applicable modes

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/47663017/ports/1/mode' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "mode": "on"
}'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.12 Get Port Power Limits

This endpoint retrieves the current power limit configured for a specific port on a designated hub.

You must provide the hub's serial number and the port number in the request. The response returns the configured maximum power limit for the specified port, allowing for real-time monitoring and management of per-port power settings on the hub.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/port-number/power/limit'
```

Variable	Description
hub-serial	The serial number of the hub
port-number	The number of the port

Returns:

```
{
  "id": port-number
  "maximum": power,
  "allowed": [power]
  },
  "hub": {
    "total": power,
    "used": power
  },
  "unit": "power-unit"
}
```

Output	Description
port-number	The number of the port
power	This is the power value
power-unit	This is the unit of the power value

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/hubs/47663017/ports/1/power/limit' \
  -H 'accept: application/json'
```

Returns

```
{
  "result": {
    "id": 1,
    "maximum": {
      "value": 60000,
      "unit": "milliwatts"
    }
  }
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.6.13 Set Port Power Limits

This endpoint allows you to set the maximum power limit for an individual port on a specified hub.

You must provide the hub's serial number, the target port number, and the desired power limit in the request payload. This operation is only applicable to Smart hubs that support Power Delivery (e.g., ThunderSync5-C16 PD).

Ports that already have a device connected when this setting is applied will not have their current power configuration affected.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/port-number/power/limit'
```

Variable	Description
hub-serial	The serial number of the hub
port-number	The number of the port

In the data section of the request, include the information you wish to add, formatted as shown below.

```
{
  "limit": power
}
```

Variable	Description
power	This is the power value in mA

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/47663017/ports/1/power/limit' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "limit": 60000
}'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.7. Firmware

Endpoints for managing firmware files. These endpoints allow users to upload, retrieve and delete firmwares, as well as view firmware and hardware details. They support efficient firmware administration.

Method	Description
Get Firmwares	This endpoint retrieves a list of firmware files currently known to the API
Get Firmware	This endpoint retrieves the full content of a specified firmware file
Add Firmware	This endpoint allows you to upload a new firmware file to the API
Delete Firmware	This endpoint allows you to delete a specific firmware file from the API

4.7.1 Get Firmwares

This endpoint retrieves a list of firmware files currently known to the API. When called, it returns an array of firmware entries, each containing the filename and associated version details.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/firmware/files'
```

Returns:

```
[
  {
    "filename": "firmware-filename",
    "versions": [
      {
        "type": "firmware-type",
        "version": "firmware-version"
      }
    ]
  }
]
```

Output	Description
firmware-filename	The filename of the firmware
firmware-version	Version number of the firmware. Format 'N.nn'
firmware-type	Used to denote which firmware the product accepts

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/firmware/files' \
  -H 'accept: application/json'
```

Returns


```
{
  "result": [
    {
      "filename": "CambrionixFirmware-v1.88.7-un.enfir",
      "versions": [
        {
          "type": "un",
          "version": "1.88.7"
        }
      ]
    },
    {
      "filename": "Firmware-00-ps-1.0.4.enfir",
      "versions": [
        {
          "type": "ps",
          "version": "1.0.4"
        }
      ]
    },
    {
      "filename": "Firmware-st-2.1.0+144.enfir",
      "versions": [
        {
          "type": "st",
          "version": "2.1.0+144"
        }
      ]
    },
    {
      "filename": "Firmware-un-1.84.0.enfir",
      "versions": [
        {
          "type": "un",
          "version": "1.84"
        }
      ]
    },
    {
      "filename": "Firmware-un-1.88.4.enfir",
      "versions": [
        {
          "type": "un",
          "version": "1.88.4"
        }
      ]
    },
    {
      "filename": "Firmware-un-1.89.2.enfir",
      "versions": [
        {
```

```
            "type": "un",
            "version": "1.89.2"
          }
        ]
      },
      {
        "filename": "Firmware-un-1.89.3.enfir",
        "versions": [
          {
            "type": "un",
            "version": "1.89.3"
          }
        ]
      },
      {
        "filename": "Firmware-un-1.89.4.enfir",
        "versions": [
          {
            "type": "un",
            "version": "1.89.4"
          }
        ]
      }
    ]
  }
```


Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.7.2 Get Firmware

This endpoint retrieves the full content of a specified firmware file known to the API.

You must provide the filename as part of the request. In response, the API returns the raw data of the requested firmware file, allowing it to be downloaded or processed as needed.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/firmware/files/firmware-file-name'
```

Variable	Description
firmware-filename	The filename of the firmware

Returns:

This will return the content of the firmware file

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/firmware/files/CambrionixFirmware-v1.88.7-un.enfir' \
  -H 'accept: text/plain'
```

Returns

For clarity in this manual the response has been truncated

```
*format enfir1
*firmware_version 1.88.7
*firmware_type un
*ktvect B2E8773D
@E4DC37191102097BFFABF39F
 ......
*flash_crc 0158022A
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.7.3 Add Firmware

This endpoint allows you to upload a new firmware file to the API.

By providing the firmware file content and the filename, the firmware is added to the API's internal repository, making it available for future operations such as device updates.

This functionality ensures the system remains up-to-date with the latest supported firmware versions.

Syntax: see 'Call Structure'

Type = 'POST''

```
'http://localhost:43424/api/v1/firmware/files/firmware-file-name'
```

Variable	Description
firmware-filename	The filename of the firmware

In the data section of the request, include the content from the .enfir file provided by Cambrionix

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/firmware/files/CambrionixFirmware-v1.88.7-un.enfir' \
  -H 'accept: application/json' \
  -H 'Content-Type: text/plain' \
  -d '*format enfir1
*firmware_version 1.88.7
*firmware_type un
*ktvect B2E8773D
@E4DC37191102097BFFABF39F
 ......
*flash_crc 0158022A'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.7.4 Delete Firmware

This endpoint allows you to delete a specific firmware file from the API's internal repository.

By providing the filename in the request payload, you can permanently remove the firmware entry, making it unavailable for future device updates or operations.

This ensures that outdated or unnecessary firmware files can be managed and kept up-to-date within the system.

Syntax: see 'Call Structure'

Type = 'DELETE''

```
'http://localhost:43424/api/v1/firmware/files/firmware-file-name'
```

Variable	Description
firmware-filename	The filename of the firmware

Returns:

```
{
  "result": true
}
```

Example

```
curl -X 'DELETE' \
  'http://localhost:43424/api/v1/firmware/files/CambrionixFirmware-v1.88.7-un.enfir' \
  -H 'accept: text/plain'
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.8. Endpoints

Endpoints for retrieving information about USB endpoint memory usage. These allow users to view system-wide usage, helping monitor resource availability and performance across connected devices.

Method	Description
Get Computer Endpoints	This endpoint retrieves detailed endpoint usage information for the USB controllers

4.8.1 Get Computer Endpoints

This endpoint retrieves detailed endpoint usage information for the USB controllers associated with the connected computer.

The response includes key data for each controller, such as its description, location ID, and usage statistics for USB 2.0 and USB 3.0 separately, as well as a combined total.

Syntax: see 'Call Structure'

Type = 'GET'

```
'http://localhost:43424/api/v1/endpoints'
```

Returns:

```
[
  {
    "info": {
      "description": "description)",
      "locationId": location-id
    },
    "usage": {
      "usb2": {
        "endpoints": endpoints,
        "memory": endpoint-memory
      },
      "usb3": {
        "endpoints": endpoints,
        "memory": endpoint-memory
      },
      "total": {
        "endpoints": endpoints,
        "memory": endpoint-memory
      }
    }
  }
]
```

Output	Description
description	This the name of hardware

Output	Description
location-id	This is the id of the location as a unique 128-bit number
endpoint-memory	This is the amount of endpoint memory consumed in bytes
endpoints	This the the amount of endpoints

Example

```
curl -X 'GET' \
  'http://localhost:43424/api/v1/endpoints' \
  -H 'accept: application/json'
```

Returns

```
{
  "result": [
    {
      "info": {
        "description": "NVIDIA USB 3.10 eXtensible Host Controller - 1.10
(Microsoft)",
        "locationId": 553648128
      },
      "usage": {
        "usb2": {
          "endpoints": 0,
          "memory": 0
        },
        "usb3": {
          "endpoints": 0,
          "memory": 0
        },
        "total": {
          "endpoints": 0,
          "memory": 0
        }
      }
    },
    {
      "info": {
        "description": "Intel(R) USB 3.10 eXtensible Host Controller - 1.10
(Microsoft)",
        "locationId": 570425344,
        "endpoint-memory": {
          "peak-usage": 40960
        }
      },
      "usage": {
        "usb2": {
```

```
            "endpoints": 0,
            "memory": 0
          },
          "usb3": {
            "endpoints": 0,
            "memory": 0
          },
          "total": {
            "endpoints": 0,
            "memory": 0
          }
        }
      },
      {
        "info": {
          "description": "Intel(R) USB 3.20 eXtensible Host Controller - 1.20
(Microsoft)",
          "locationId": 587202560,
          "endpoint-memory": {
            "peak-usage": 720896
          }
        },
        "usage": {
          "usb2": {
            "endpoints": 77,
            "memory": 409600
          },
          "usb3": {
            "endpoints": 23,
            "memory": 188416
          },
          "total": {
            "endpoints": 100,
            "memory": 598016
          }
        }
      }
    ]
}
```


Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.9. Apple Devices

Endpoints for managing Apple devices connected to supported hubs. These allow users to reboot devices or place them into DFU mode, either across the entire hub or on specific ports. A valid Connect licence and sign-in are required to access this functionality. These commands also require a Cambrionix Type-C hub such as a ThunderSync3-C10

Method	Description
Put Devices into DFU	This endpoint places all connected Apple devices into DFU mode
Reboot Devices	This endpoint reboots all connected Apple devices on a hub
Put Device into DFU	This endpoint puts an Apple device on a specific port into DFU mode
Reboot Device	This endpoint reboots an Apple device on a specific port

4.9.1 Put Devices into DFU

This endpoint allows you to send a command to place all connected Apple devices into DFU mode by specifying the hub's serial number in the request. It enables direct communication with the hub to perform this action. A valid licence with the required permissions is needed to use this feature refer to the Token Exchange section for details on how to obtain a token once the appropriate licence is in place.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ap-
ple/dfu/enter' \
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
{
  "ports": [
    {
      "id": port-number,
      "result": result,
      "reason": description
    }
  ]
}
```

Output	Description
port-number	The number of the port
result	The result of the command this will be boolean
description	This provides the reason why the command failed when the result is returned as false

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/apple/dfu/enter' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiBoDTbNbIrDPF3foDtZLvqnSLmVvPFw' \
  -d ''
```

Returns


```
{
  "result": {
    "ports": [
      {
        "id": 1,
        "result": true
      },
      {
        "id": 2,
        "result": false,
        "reason": "No device attached"
      },
      {
        "id": 3,
        "result": true
      },
      {
        "id": 4,
        "result": false,
        "reason": "No device attached"
      },
      {
        "id": 5,
        "result": false,
        "reason": "No device attached"
      },
      {
        "id": 6,
        "result": false,
        "reason": "No device attached"
      },
      {
        "id": 7,
        "result": false,
        "reason": "No device attached"
      },
      {
        "id": 8,
        "result": false,
        "reason": "No device attached"
```

```
        },
        {
          "id": 9,
          "result": false,
          "reason": "No device attached"
        },
        {
          "id": 10,
          "result": false,
          "reason": "No device attached"
        }
      ]
    }
  }
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.9.2 Reboot Devices

This endpoint allows you to send a command to reboot all connected Apple devices by specifying the hub's serial number in the request, enabling direct communication with the hub to carry out the action. A valid licence with the required permissions is needed to use this feature refer to the Token Exchange section for details on how to obtain a token once the appropriate licence is in place.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ap-
ple/reboot' \
```

Variable	Description
hub-serial	The serial number of the hub

Returns:

```
{
  "ports": [
    {
      "id": port-number,
      "result": result,
      "reason": description
    }
  ]
}
```

Output	Description
port-number	The number of the port
result	The result of the command this will be boolean
description	This provides the reason why the command failed when the result is returned as false

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/apple/reboot' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer e21SrJxqiE1BJZLvqnSLmVvPFw' \
  -d ''
```

Returns


```
{
  "result": {
    "ports": [
      {
        "id": 1,
        "result": true
      },
      {
        "id": 2,
        "result": false,
        "reason": "No device attached"
      },
      {
        "id": 3,
        "result": true
      },
      {
        "id": 4,
        "result": false,
        "reason": "No device attached"
      },
      {
        "id": 5,
        "result": false,
        "reason": "No device attached"
      },
      {
        "id": 6,
        "result": false,
        "reason": "No device attached"
      },
      {
        "id": 7,
        "result": false,
        "reason": "No device attached"
      },
      {
        "id": 8,
        "result": false,
        "reason": "No device attached"
```

```
        },
        {
          "id": 9,
          "result": false,
          "reason": "No device attached"
        },
        {
          "id": 10,
          "result": false,
          "reason": "No device attached"
        }
      ]
    }
  }
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.9.3 Put Device into DFU

This endpoint allows you to send a command to put an Apple device on a specific port into DFU mode by specifying the hub's serial number and port number in the request. It enables direct communication with the hub to perform this action. A valid licence with the required permissions is needed to use this feature refer to the Token Exchange section for details on how to obtain a token once the appropriate licence is in place.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ports/port-number/apple/dfu/enter' \
```

Variable	Description
hub-serial	The serial number of the hub
port-number	The number of the port

Returns:

```
{
  "result": result,
  "reason": description
}
```

Output	Description
result	The result of the command this will be boolean
description	This provides the reason why the command failed when the result is returned as false

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/ports/6/apple/dfu/enter'
\
```

```
    -H 'accept: application/json' \
    -H 'Authorization: Bearer eyJhbGciOiJKvPFw' \
    -d ''
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

4.9.4 Reboot Device

This endpoint allows you to send a command to reboot an Apple device on a specific port by specifying the hub's serial number and port number in the request, enabling direct communication with the hub to perform the action. A valid licence with the required permissions is needed to use this feature refer to the Token Exchange section for details on how to obtain a token once the appropriate licence is in place.

Syntax: see 'Call Structure'

Type = 'POST'

```
'http://localhost:43424/api/v1/hubs/hub-serial/ap-
ple/reboot' \
```

Variable	Description
hub-serial	The serial number of the hub
port-number	The number of the port

Returns:

```
{
  "result": result,
  "reason": description
}
```

Output	Description
result	The result of the command this will be boolean
description	This provides the reason why the command failed when the result is returned as false

Example

```
curl -X 'POST' \
  'http://localhost:43424/api/v1/hubs/0000007B6BD6C7A6/ports/6/apple/reboot' \
```

```
    -H 'accept: application/json' \
    -H 'Authorization: Bearer eyJhbGciOiJSUzI1NtZLvqnSLmVvPFw' \
    -d ''
```

Returns

```
{
  "result": true
}
```

Errors

If there is an error in the API method then JSON-RPC Error Object a will be returned

5. Token Exchange

In order to communicate with the API you will require a token to be sent which is authenticated with the Cambrionix Connect platform. The token must be included with the Syntax in order for the endpoint to be accessed, if you do not have a valid token an error response will be received.

The token is provided by the Cambrionix Connect Authentication API within the Cambrionix Connect platform, more information on this can be found in the relevant user manual which can be downloaded from https://www.cambrionix.com/software.

Create a service

In order to get a token you will need to create a service within Cambrionix Connect with the relevant permissions for the API, information on the permissions can be found in the Cambrionix Connect user manual which can be downloaded from the following the browser application https://connect.cambrionix.com

Requesting a token

To request a token you will need to use the below HTTPS request and add in the parameters applicable to what you want the token to have permissions for, once you have a token you should save it as a variable to then add into the header as explained in the Call Structure section of this manual.

```
https://connect.cambrionix.com/api/v1/organizations/
{organisation-id}/oauth2/token?audience={service-id}&grant_
type=client_credentials&subject_token_
type=urn:ietf:params:oauth:token-type:access_token&client_id=
{client-id}&client_secret={service-secret}&scope=permissions
```

Parameter	Description
organisation-id	The ID of the organisation you are using
service-id	The service ID obtained from API Details
client-id	The Id of the custom service
service-secret	The secret obtained from the custom service

Parameter	Description
permissions	The permissions required for the action, see the Cambrionix Connect user manual which can be downloaded from the following link cambrionix.com/pages/cambrionix-connect/ for example API.DFU for all apple device related controls

Use of Trademarks, Registered Trademarks, and other Protected Names and Symbols

This manual may make reference to trademarks, registered trademarks, and other protected names and or symbols of third-party companies not related in any way to Cambrionix. Where they occur these references are for illustrative purposes only and do not represent an endorsement of a product or service by Cambrionix, or an endorsement of the product(s) to which this manual applies by the third-party company in question.

Cambrionix hereby acknowledges that all trademarks, registered trademarks, service marks, and other protected names and /or symbols contained in this manual and related documents are the property of their respective holders

"Mac® and macOS® are trademarks of Apple Inc., registered in the U.S. and other countries and regions."

"Intel® and the Intel logo are trademarks of Intel Corporation or its subsidiaries."

"Thunderbolt™ and the Thunderbolt logo are trademarks of Intel Corporation or its subsidiaries."

"Android™ is a trademark of Google LLC"

"Chromebook™ is a trademark of Google LLC."

"iOS™ is a trademark or registered trademark of Apple Inc,in the US and other countries and is used under license."

"Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries"

“ Microsoft™ and Microsoft Windows™ are trademarks of the Microsoft group of companies.”

"Cambrionix® and the logo are trademarks of Cambrionix Limited."

All trademarks and registered trademarks mentioned are acknowledged and respected as the property of their respective holders.

Important Notice on Protected Information

Please note that certain components of Cambrionix technology are considered protected intellectual property (IP) of Cambrionix. Specifically:

- Source Code: The source code of our software is proprietary and cannot be provided.
- Proprietary Methods: Detailed descriptions and implementations of our proprietary methods are also protected.

As such, requests for access to the source code or other protected information will be respectfully declined. We appreciate your understanding and cooperation.

Cambrionix Patents

Title	Link	Application Number	Grant Number
Syncing and Charging Port	GB2489429	1105081.2	2489429
CAMBRIONIX	UK00002646615	2646615	00002646615
CAMBRIONIX VERY INTELLIGENT...	UK00002646617	2646617	00002646617

Licensing

The use of Connect Hub API is subject to the Cambrionix Connect SaaS conditions, the document can be downloaded and viewed using the following link.

https://downloads.cambrionix.com/documentation/en/Cambrionix-Connect-SaaS-Conditions.pdf

The use of Connect Hub API is subject to the Cambrionix Licence agreement, the document can be downloaded and viewed using the following link.

https://downloads.cambrionix.com/documentation/en/Cambrionix-Licence-Agreement.pdf

Cambrionix

CHARGE • CONNECT • MANAGE

Cambrionix Limited

The Maurice Wilkes Building

Cowley Road

Cambridge CB4 0DS

United Kingdom

+44 (0) 1223 755520

https://www.cambrionix.com

Cambrionix Ltd is a company registered in England and Wales

with the company number 06210854

EN