Cambrionix
CHARGE • CONNECT • MANAGE

Cambrionix Hub API

User Manual

Cambrionix
Hub API

EN

1. Table of Contents

2. Introduction

The Cambrionix Hub API is designed to control Cambrionix products through a locally installed service called 'CambrionixApiService'. This service provides a programming interface to control Cambrionix units.

A Python wrapper is provided with a public domain JSON-RPC library that will allow scripts to be written without needing to be overly familiar with JSON. Alternatively, you may use the programming language of your choice to connect directly to the daemon over a standard TCP/IP socket and send and receive JSON formatted data. When the API is used to communicate with a remote network attached hub, this is done over an SSH tunnel.

The Cambrionix Hub API supports multiple simultaneous client connections to itself and supports concurrent access to numerous hubs.

The Cambrionix Hub API is implemented in CambrionixApiService, which sits between the application and the Cambrionix units. It maps the properties of the Cambrionix units into API commands.

You can download the latest version of this manual from our website at the following link.
https://www.cambrionix.com/cambrionix-api

JSON-RPC library

The API uses JSON-RPC over TCP. JSON-RPC is a connection to JSON as a data format. JSON-RPC stands for 'JavaScript Object Notation Remote Procedure Call'. In short, JSON represents a lightweight format for data interchange. It is a format for structuring data that is easy to transfer.

Any programming language that supports JSON-RPC can be used; libraries are widely available for other languages.

2.1. Installation

macOS® Installation

For macOS®, an installer is provided that will set up CambrionixApiService to run as a daemon process. The service is configured to load on-demand when it's configured, the listening port is attached to by launchd.

Due to how the Cambrionix Hub API runs, it requires User Account Control (UAC) interaction to be installed. This means that the installer cannot be run silently.

The installation can also perform all necessary steps to configure any installed versions of Python 2 and 3 and direct you to various example scripts.

Windows Installation

For Windows, a self-extracting installer is provided that will set up CambrionixApiService to run as a Windows service.

Due to how the Cambrionix Hub API runs, it requires User Account Control (UAC) interaction to be installed. This means that the installer cannot be run silently.

The installation can also perform all necessary steps to install and configure Python 2 and 3, and direct you to various example scripts.

Linux Installation

The Linux® package comes as a Debian package which you can install either via the GUI or from the command line using apt:

```
sudo dpkg -i /Downloads/cambrionix-api-setup-????????.deb
```

You can also set the API to run in a persistent mode which will then run when you reboot your system this can be achieved by using the following command

```
sudo /usr/bin/CambrionixApiService --install --persistent
```

An armhf version has also been tested on oDroid and Raspberry Pi.

We currently do not have support for rpm files; if you require assistance, then you can obtain more information from the page below.

https://fedingo.com/how-to-convert-deb-to-rpm-files-in-linux/

USB Drivers

The USB drivers that are required to access Cambrionix hubs are included with most OS's by default.

These drivers are included in the installer and can be installed by selecting the optional component during installation. The option will not appear if the necessary drivers are already present. However ,installation of these drivers does not complete until a Cambrionix charger is attached to the host machine. If you install the API and the USB drivers before the first time you connect a Cambrionix charger, then the API will not start, and you will need to reboot the host machine after connecting a Cambrionix hub that will trigger the completion of the USB driver installation, to ensure that the API service is correctly started.

2.2. Prerequisites

Before using the Cambrionix Hub API, a few steps and checks need to be completed.

Direct access to USB hardware

For the API to retrieve USB information from connected devices, it must have direct access to the hardware. This means that running in a Virtual machines (VM) such as Parallels, VirtualBox and Microsoft Hyper-V are not supported as the virtualisation prevents the API from determining which USB device is connected to which physical port. Also, it is not unusual that such a virtual environment will not have access to serial devices necessary to communicate with the hub to query information.

Thunderbolt™ with Windows

You may need to update your Thunderbolt™ Bus Drivers and possibly the BIOS on Windows. Once the Thunderbolt™ device has been accepted to connect, you may need to turn it off and on again for Windows to connect physically

Sync capable charger for USB information

For the API to return USB device information such as the VID, PID, Manufacturer, Description or Serial Number, there must be a USB connection from the host machine to the connected device. This is only present on sync-capable products. Charge-only products have a USB connection to the charger but not to connected devices. The API is functional with charge-only chargers but cannot return the USB device information.

Version for universal firmware products

When used with this API, products using the universal firmware must have firmware version 1.52 or later installed. We recommend that the latest version is installed available from our website or through Cambrionix Connect; a table of all products and the firmware used is below.

Firmware	Part Number	Product Name
Universal	PP15S	PowerPad15S
Universal	PP15C	PowerPad15C
Universal	PP8S	PowerPad8S
Universal	SS15	SuperSync15
Universal	TS3-16	ThunderSync3-16
SMART	TS3-C10	ThunderSync3-C10
Universal	U16S Spade	U16S Spade
Universal	U8S	U8S
PDSync	PDSync-C4	PDSync-C4
Universal	ModIT-Max	ModIT-Max
Motor Control	Motor control board	ModIT-Max

USB drivers

The Cambrionix Hub API daemon (CambrionixApiService) must be able to communicate with the local hub. The hub will appear as a USB device. The USB device will be accompanied by a virtual communications port (VCP). The virtual communications operates like a standard serial communications port, or COM port as it is often called. The operating system must have the appropriate VCP (Virtual COM Port) driver installed.

Linux®	macOS	Windows
The default support in the kernel is sufficient. Do not install the D2XX drivers as this conflicts with the required VCP drivers.	The default support in OS is sufficient. Do not install the D2XX drivers as this conflicts with the required VCP drivers.	The D2XX support can coexist with the VCP support. These drivers are automatically installed on newer versions of Windows 10.

Operating System

We have tested the Cambrionix Hub API and can confirm that the following operating systems work with the Cambrionix Hub API. There may be other OS that will work but may not have been tested:

- Windows 10
- Windows 11

- macOS 11 (Big Sur)
- macOS 12 (Monterey)
- macOS 13 (Ventura)
- macOS 14 (Sonoma)
- macOS 15 (Sequoia)
- Linux Ubuntu
- Linux Debian

With Linux, we only perform testing using the OS mentioned above. An ARM hard float (armhf) version has also been tested on oDroid and Raspberry Pi.

3. Quick start

Some example scripts in Node.JS, C#, VB.Net, and Python are included with the installed files. For Python, we recommend using the newer asyncio example rather than the older synchronous examples.

- Windows: %ProgramFiles%\Cambrionix\CambrionixAPI/examples
- Linux: /usr/local/share/cambrionix/apiservice/examples.
- macOS: /Library/Cambrionix/ApiService/examples.

For each code type in the examples folder you will need to install the necessary programs.

- Python 3.4 for the python examples, as well as Python 3.4, you will need the jsonrpc-websocket module. Reference information on using Python can be found here.
- Node.JS for the nodejs example, will also require either NPM or Yarn. More information can be found here.
- Visual Studio for the C# or VB.Net examples

To install the Python async api package go to examples/python/asyncio and run

```
pip install .
```

3.1. Python Example

The jsonrpc-websocket module automatically converts python script to a JSON-RPC request. The below example is how the websocket will convert a simple script.

Python.

```
cbrxapi.cbrx_connection_open("DJ000102")
```

JSON-RPC Translation.

```
{
  "id": 0,
  "jsonrpc": "2.0",
  "method": "cbrx_connection_open",
  "params": [
    "DJ000102"
  ]
}
```

Replies are automatically converted back from JSON into a Python dictionary, or list or value as appropriate.

Here is an example of using the API, the code is written in Python 3.6: The example code doesn't check for errors so the Python script will simply stop on an error. Robust code error handling should be incorporated in your own software.

```
# Import the cbrxapi library.
import sys
from cbrxapi import cbrxapi

print("Querying API Version...")
try:
    result = cbrxapi.cbrx_apiversion()
except Exception as e:
    print(f"Could not communicate with API : {e}")
    result = None

if result:
    print(f"API Version {result[0]}.{result[1]}")


# Call cbrx_discover with "local" to find any locally attached Cambrionix
units.
# This will return a list of local Cambrionix units.
print("Discovering local devices..")
result = cbrxapi.cbrx_discover("local")
if not result or len(result) == 0:
    print("No Cambrionix unit found.")
    sys.exit(0)

print(f"Discovered {len(result)} units")

for unit_id in result:
    serial_port = cbrxapi.cbrx_discover_id_to_os_reference(unit_id)

    try:
        # Open a connection to the Cambrionix unit, which will return a handle
for
        # the connection.
        handle = cbrxapi.cbrx_connection_open(unit_id)
    except Exception as e:
        print(f"Could not open connection to {unit_id} : {e}")
        handle = None

    if handle:
        # Using the handle, get the "Hardware" and "nrOfPorts" properties
        hardware = cbrxapi.cbrx_connection_get(handle, "Hardware")
        n_ports = cbrxapi.cbrx_connection_get(handle, "nrOfPorts")

        # Done using the Cambrionix unit, close the handle.
        cbrxapi.cbrx_connection_close(handle)

        # Finally, print out the information retrieved from the Cambrionix
unit.
        print(f"* {hardware} on {serial_port} has {n_ports} ports")
```

3.2. TypeScript Example

This is a simple example of using Typescript to use the API to obtain information on the hubs and devices. Further information on TypeScript can be found here.

```
import React from 'react';
import Websocket from 'react-websocket';

class MyApiInterface extends React.Component {
  lastId = 0;

  render() {
    return (
      <Websocket ref={r => this.websocket = r} reconnect
                 url="ws://localhost:43424" protocol="jsonrpc"
                 onMessage={this.onDataReceived.bind(this)}
                 onOpen={this.onApiConnection.bind(this)}
                 onClose={this.onApiDisconnection.bind(this)} />
    );
  }

  requests = {};

  onDataReceived(json) {
    const data = JSON.parse(json);
    const id = data.id;
    if (id) {
      const request = this.requests[id];
      if (request && request.callback) {
        request.callback(data);
      }
      delete this.requests[id];
    }
    else
    {
      //Could get a notification here if you enable them on active connection
      //Notifications have no id and can arrive at any time
    }
  }

  makeRequest(method, params, callback) {
    var packet = {
      jsonrpc: "2.0",
      id:      ++this.lastId,
      method:  method,
      params:  params,
    };

    this.requests[packet.id] = {packet: packet, callback: callback};

    this.websocket.sendMessage(JSON.stringify(packet));
  }
```

```
  onApiConnection() {
    console.log("Connected");
    this.makeRequest("cbrx_discover", ["local"], console.log);
  }

  onApiDisconnection() {
    console.log("Disconnected");
    this.requests = {}
  }
}
```

3.3. HTTP GET Example

Connections can be made directly to an http prefixed URI, in which case the json is extracted from either the address itself, or the body content of the GET request. You can try this example in your browser or from the command line, using curl:

```
curl -get http://localhost:43424/?{\"id\":0,\"json-
rpc\":\"2.0\",\"method\":\"cbrx_discover\",\"params\":[\"all\"]}
```

Socket connections can be simple binary data, http GET requests or Web-sockets (such as from Node.js). For example, pasting the following into the address bar of your browser should allow you to see quick results:

```
http://localhost:43424/?{"jsonrpc":"2.0","id":1,"method":"cbrx_dis-
cover","params":["all"]}
```

Please be aware that on some Terminal/Command Prompt windows, you may find that you need to encode the URL to prevent error's from occuring.

Once encoded, the above URL should look like:

```
http://-
localhost:43424/%7B%22jsonrpc%22:%222.0%22,%22id%22:0,%22method%22:%22cbrx_
apidetails%22%7D
```

3.4. Error Handling

A JSON-RPC error will return an error member containing the following members:

- code (mandatory) – an integer indicating either a pre-defined JSON-RPC error code in the range -32768 to -32000 or a CBRXAPI error code as documented in the section CBRXAPI

specific errors section.

- message (optional) – a message string explaining the error code
- data (optional) – extra information about the error like debug messages or handles.

The Python JSON-RPC used causes an exception for an error response with the following mapping:

- member code is returned in e.error_code
- member message is returned in e.error_message
- member data is returned in e.error_data.

You can catch an error response with:

```
try:
    handle = cbrxapi.cbrx_connection_open(id)
except jsonrpc.RPCFault as e:
    gotException = True
    errorCode = e.error_code
    errorMessage = e.error_message
    errorData = e.error_data
```

Example of how to create an error and the response it will give:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_connection_open",
  "params": [
    "0"
  ]
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "error": {
    "code": -10001,
    "message": "ID not found"
  }
}
```

4. API Call Structure

The descriptions of the API calls contain JSON-RPC requests / responses as you would see them on the wire.

4.1. JSON-RPC request object

JSON-RPC is a stateless, light-weight remote procedure call (RPC) protocol. A JSON-RPC is represented by sending a Request object. The Request object has the following members:

jsonrpc

A String specifying the version of the JSON-RPC protocol. MUST be exactly "2.0".

id

An identifier established by the Client that MUST contain a String, Number, or NULL value if included. If it is not included it is assumed to be a notification.

method

A String containing the name of the method to be invoked.

params

A Structured value that holds the parameter values to be used during the invocation of the method. This is not required for every method

Grouping this all together will give the complete JSON-RPC request:

```
{
    "jsonrpc": "version",
    "id": 0,
    "method": "method-name",
    "params": [
        "structured-params"
    ]
}
```

4.2. JSON-RPC response object

When a rpc call is made, there will be a Response, except for in the case of Notifications. The Response is expressed as a single JSON Object, with the following members:

jsonrpc

A String specifying the version of the JSON-RPC protocol. MUST be exactly "2.0".

id

This member is the same as the value of the id member in the Request Object.

result

The value of this member is determined by the method in the Request object.

error

This member is returned only on error.

Grouping this all together will give the complete JSON-RPC response:

```
{
    "jsonrpc": "version",
    "id": 0,
    "result": "method-result"
}
```

4.3. JSON-RPC Error Object

When a call encounters an error, the Response Object will contain the error member with a value that is an Object with the following members:

code

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

message

A String providing a short description of the error.

data

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

Grouping this all together will give the complete JSON-RPC response:

```
{ "jsonrpc": "version", "id": 0, "error": { "code": "error-code",
        "message": "error-message" } }
```

4.4. Call Structure

This part has been removed from the Syntax and Return sections throughout this manual to simplify the documentation.

Two further key-value pairs need to be passed to complete the JSON-request; One indicating the version of JSON-RPC being used, in this case 2.0 and an id identifying this request:

The id is mandatory but only relevant if multiple requests can be outstanding simultaneously over the same connection. It helps to match responses to (asynchronous) requests. The response for a request will be given the matching id by CambrionixApiService.

```
{
  "jsonrpc": "2.0",
  "id": 0
}
```

5. API Methods

There are 3 groups of calls in the API:

- Version - Obtain details about the API
- Discovery - Obtain details about what is connected to the API
- Connection - Manage connections and devices connected

Version

API Call	Description
cbrx_apiversion	Obtain the version of the API
cbrx_apidetails	Obtain an enhanced version of the details of the API

Discovery

API Call	Description
cbrx_discover	Discover Cambrionix units
cbrx_discover_id_to_os_reference	Map a unit ID from unit to device as used by the OS
cbrx_find	Search for devices attached to local Cambrionix units
cbrx_get_usb (tree)	Return the entire USB tree that has been discovered
cbrx_get_usb (descriptors)	Request entire dump of a USB device's descriptor information
cbrx_config_set	Set configuration options

Connection

API Call	Description
cbrx_certificate	Manage certificates and private keys to the API
cbrx_connection_open	Open a connection to a hub

API Call	Description
cbrx_connection_close	Close an open connection to a hub
cbrx_connection_getdictionary	Get all keys on a hub specified by the connectionHandle.
cbrx_connection_get	Get a key from a hub specified by connectionHandle
cbrx_hub_get	Get a key from a hub specified by hubs serial number
cbrx_device_get	Get a key from a hub specified by USB device's serial number
cbrx_connection_setdictionary	List all writeable and command keys for the hub specified by connectionHandle
cbrx_connection_set	Set a key to the value specified on a hub specified by connectionHandle
cbrx_hub_set	Set a key to the value specified on a hub specified by hubs serial number
cbrx_notifications	Send notifications
cbrx_firmware	Add or remove firmware files
cbrx_connection_closeandlock	Close all connections to a hub and lock it
cbrx_connection_unlock	Unlock a hub that was previously locked.
cbrx_connection_cli	Perform command line interface operation
cbrx_pair_device	Initiaite pairing of an iOS device
cbrx_exit	Restart the API

5.1. cbrx_apidetails

Returns an enhanced version of the details of the API. This information can also be gained by passing an optional true parameter to cbrx_apiversion.

Syntax: see API Call Structure

```
{
  "method": "cbrx_apidetails"
}
```

Returns:

```
{
  "result": {
    "version": [version-number],
    "semver": "semver-variant",
    "commitid": commitid-number,
    "branch": "branch-name",
    "capability": [API-capability],
    "notifications": [possible-notification],
    "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	The full name of the API version
commitid-variant	The number value of the Commit ID
branch-name	The branch of API installed
API-capability	available with version of API
possible-notification	Array of strings to show possible notification. see API Notifications
install-location	The location of install files
logs-location	The location of where logs are stored
settings-location	The location of API settings
documentation-location	The web address of API documentation
brand-information	The brand of the CPU
CPU-architecture	The architecture of the CPU
CPU-features	Features available on CPU
cores-value	How many cores the CPU has
os-information	Operating system running on local machine

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_apidetails"
}
```

Example Successful response

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": {
    "version": [
      3,
      7,
      0,
      34
    ],
    "semver": "3.7.0+34",
    "guid": {
      "id": "d0dc3cac-e165-4e38-88bb-39064431bdc9",
      "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
    },
    "host": [
      {
        "ip": "10.167.111.81",
        "port": 0,
        "nameServer": "10.167.111.241",
        "domainName": "CBRX.LOCAL",
        "hostName": "CBRXPC-011",
        "adapterName": "Intel(R) Ethernet Controller (3) I225-V",
        "adapterType": "Ethernet"
      }
    ],
    "commitid": 4287981321,
    "branch": "release",
    "capability": [
      "protobuf",
      "crash-report",
      "notification"
    ],
    "notifications": [
      "usb-changed",
      "usb-device-attached",
      "usb-device-detached",
      "discover-changed",
      "dead-hub-changed",
      "firmware-progress",
      "rfid-received",
      "rfid-removed",
      "over-temperature",
      "over-voltage",
      "under-voltage",
      "certificate-changed"
    ],
    "install": "C:\\Program Files\\Cambrionix\\API",
    "logging": "C:\\ProgramData\\Cambrionix\\Log",
    "settings": "C:\\ProgramData\\Cambrionix",
    "documentation": "C:\\Program Files\\Cambrionix\\API\\Cambrionix Hub API
Reference.html",
    "cpu": {
      "brand": "12th Gen Intel(R) Core(TM) i9-12900K",
      "arch": "x64",
```

```
        "features": [
          "aes",
          "avx",
          "avx2",
          "bmi1",
          "bmi2",
          "clflushopt",
          "clfsh",
          "clwb",
          "cx16",
          "cx8",
          "erms",
          "f16c",
          "fma3",
          "fpu",
          "mmx",
          "movbe",
          "pclmulqdq",
          "popcnt",
          "rdrnd",
          "rdseed",
          "sha",
          "smx",
          "ss",
          "sse",
          "sse2",
          "sse3",
          "sse4_1",
          "sse4_2",
          "ssse3",
          "tsc",
          "vaes",
          "vpclmulqdq"
        ],
        "cores": 24
      },
      "os": "Windows 10 Pro 21H2 Build 19044.1889 64-bit"
    }
  }
```

5.2. cbrx_apiversion

Return the version of the API running.

Syntax: See API Call Structure

```
{
  "method": "cbrx_apiversion"
}
```

There is another method which can be used, see cbrx_apidetails for more information.

Returns:

```
{
  "result": [Version-number]
}
```

Version-number consists of two numbers separated by commas. The leftmost number is called the major, The rightmost number is called the minor.

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_apiversion"
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": [
    3,
    7
  ]
}
```

5.3. cbrx_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 as this could violate security if they are in limited access folders. 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 in the usual way, 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 there are guides that can be found online.

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

5.4. cbrx_certificate (set)

Supply a certificate and private key to the API.

Syntax: see Call Structure

```
{
  "method": "cbrx_certificate",
  "params": [
    "set",{
      "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:

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

Errors

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

Notes

- The files will need to be stored in a location that is accessible by the system and API.

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_certificate",
  "params": [
    "set",
    {
      "private-key": "C:\\git\\capi\\cbrxjson\\certificate\\key.pem",
      "certificate": "C:\\git\\capi\\cbrxjson\\certificate\\cert.pem"
    }
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": true
}
```

5.5. cbrx_certificate (remove)

Remove the certificate and private key from the API:

Syntax: see Call Structure

```
{
  "method": "cbrx_certificate",
  "params": ["remove"]
}
```

Returns:

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

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_certificate",
  "params": [
    "remove"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": true
}
```

5.6. cbrx_certificate (get)

Get the supplied certificate and private key information from the API.

Syntax: see Call Structure

```
{
  "method": "cbrx_certificate",
  "params": "get"
}
```

Returns:

```
"result": {
  "certificate": "Certificate",
  "subject": {
    "C": "Country",
    "L": "Location",
    "O": "Organisation",
    "CN": "Common name"
  },
  "issuer": {
    "C": "Country",
    "O": "Organisation",
    "CN": "Common name"
  },
  "serial_number": "Serial number",
  "algorithm": "Algorithm",
  "extensions": {
    "subjectAltName": [Alternative names]
```

```
  },
  "validity": {
  "not_after": Vaild until,
  "not_before": Valid from
}
```

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

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_certificate",
  "params": "get"
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": {
    "certificate": "-----BEGIN CERTIFICATE-----\r\D.....CF7ig==\r\n-----END
CERTIFICATE-----\r\n",
    "subject": {
      "C": "GB",
      "L": "Cambridge",
      "O": "Cambrionix Limited",
      "CN": "*.api.cambrionix.com"
    },
    "issuer": {
      "C": "US",
      "O": "DigiCert Inc",
      "CN": "DigiCert TLS RSA SHA256 2020 CA1"
    },
    "serial_number": "096...9BFBE",
    "algorithm": "sha256WithRSAEncryption",
    "extensions": {
      "subjectAltName": [
        "*.api.cambrionix.com",
        "api.cambrionix.com"
      ]
    },
    "validity": {
      "not_after": 169...99,
      "not_before": 16...400
    }
  }
}
```

5.7. cbrx_config_set

This function allows setting persistent configuration options. Multiple configuration keys can be set at once as the Example below shows.

Syntax: see API Call Structure

```
{
  "method": "cbrx_config_set",
  "params": {
    "configuration-key:configuration-value
  }
}
```

Parameter	Description
configuration-key	The configuration key you wish to change, as per table below.
configuration-value	The value you wish to change the configuration to.

Configuration-key	Desription
adb_path	The full pathname to the ADB executable from Android™ Developer Tools.
battery-update-enabled	Will the battery update be performed at all.
battery-updated-concurrency	How many concurrent battery updates will be run simultaneously.
battery-update-frequence-seconds	How many seconds between battery updates.

Returns:

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

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_config_set",
  "params": {
    "battery-update-enabled": true,
    "battery-update-concurrency": 2,
    "battery-update-frequency-seconds": 60
  }
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": true
}
```

5.8. cbrx_connection_cli

Perform command line interface operation on the connected hub and return the complete result. This allows you to run commands directly on the hub's command line without stopping the API service. This method is only for using the CLI commands to obtain information and not update settings, if you wish to change the internal hub settings please use the Settings command.

Syntax: see API Call Structure

```
{
  "method": "cbrx_connection_cli",
  "params": [
    connection-handle
    cli-command
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
cli-command	The CLI command you wish to send. For all CLI commands see the CLI Documentation www.cambrionix.com/cambrionix-cli

Returns:

```
{
  "result": [cli-response]
}
```

cli-response is an array of strings containing all the lines of output returned from the command. For more information see www.cambrionix.com/cli

Examples

Example JSON-RPC request

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_connection_cli",
  "params": [
    7654,
    "id"
  ]
}
```

Example successful response:

```
{
"jsonrpc": "2.0",
"id": 0,
"result": ["mfr:cambrionix,mode:main,hw:PP15S,hwid:0x13,fw:1.83,bl: 0.12,s-
n:000000,group:-,fc:un"]
 }
```

5.9. cbrx_connection_close

Close a connection to a hub previously opened, as specified by the connection handle.

Syntax: see API Call Structure

```
{
  "method": "cbrx_connection_close",
  "params": [connection-handle]
}
```

connection-handle is the The Connection Handles as an integer

Returns:

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

Errors

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

Example

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_connection_close",
  "params": [
    7654
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": true
}
```

5.10. cbrx_connection_closeandlock

Close all connections to a hub and lock it against further use until released by

cbrx_connection_unlock. Other processes that were using these connections will get errors returned if trying to access this hub.

Syntax: see API Call Structure

```
{
  "method": "cbrx_connection_closeandlock",
  "params": [hub-serial]
}
```

hub-serial is a string which is the serial number of the hub, each string is guaranteed to be unique.

Returns:

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

Errors

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

Example

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_connection_closeandlock",
  "params": [
    "DB0074F5"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": true
}
```

5.11. cbrx_connection_get

From the hub specified by the connection handle, get the key value.

Syntax: see API Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
     "dictionary-key"
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
dictionary-key	as returned by a call to cbrx_connection_getdictionary see Get Dictionary for more information

Returns:

```
{
  "result": [dictionary-value]
}
```

dictionary-value is the value of the dictionary key, see Get Dictionary for more information.

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_connection_get",
  "params": [
    569,
    "nrOfPorts"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": 10
}
```

5.12. cbrx_connection_getdictionary

Get all keys that can return information on the hub specified.

Syntax: see API Call Structure

```
{
  "method": "cbrx_connection_getdictionary",
  "params": [connection-handle]
}
```

connection-handle is the Connection Handles as an integer

Returns:

```
{
  "result": [dictionary]
}
```

*dictionary*is an array of strings containing the names of the keys and values for the device. Please see Get Dictionary section.

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_connection_getdictionary",
  "params": [
    7654
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": [dictionary]
}
```

*dictionary*is an array of strings containing the names of the keys and values for the device. Please see Get Dictionary section.

5.13. cbrx_connection_open

Open a connection to the hub specified. A successful open results in a connection handle that can be used for further calls.

Syntax: see API Call Structure

```
{
  "method": "cbrx_connection_open",
  "params": [
    hub-serial ,
    location
  ]
}
```

Parameters	Description
hub-serial	This is the serial number of the hub returned from cbrx_discover
location	see below table, if not included will default to local

Location parameter	Description
local	connect to the local hub
docks	connect to a hub in a Docks the id must be specified which can be found from cbrx_discover

Returns:

```
{
  "result": [connection-handle]
}
```

connection-handle is the Connection Handles as an integer

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_connection_open",
  "params": [
    "DB0074F5"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": 579
}
```

5.14. cbrx_connection_set

On the hub specified by the connection handle, set the key value. Calls to a dock will result in the relevant key being set on both the chargers except for the port specific keys which will be directed to the appropriate charger only.

Syntax: see API Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "dictionary-key",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
dictionary-key	as returned by a call to cbrx_connection_setdictionary see Set Dictionary for more information
Value	The value you wish to set the key too

Returns:

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

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_connection_set",
  "params": [
    5313,
    "TwelveVoltRail.OverVoltage",
    true
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": true
}
```

5.15. cbrx_connection_setdictionary

List all writeable value keys and command keys for the hub specified by Connection Handles.

Syntax: see API Call Structure

```
{
  "method": "cbrx_connection_setdictionary",
  "params": [connection-handle]
}
```

connection-handle is the Connection Handles as an integer

Returns:

```
{
  "result": [dictionary]
}
```

*dictionary*is an array of strings containing the names of the writeable keys and command keys for the device. Please see Set Dictionary section.

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_connection_setdictionary",
  "params": [
    7654
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": [dictionary]
}
```

For *dictionary*please see Set Dictionary section

5.16. cbrx_connection_unlock

Unlock a hub that was previously locked.

Syntax: see API Call Structure

```
{
  "method": "cbrx_connection_unlock",
  "params": [hub-serial]
}
```

hub-serial is a string which is the serial number of the hub, each string is guaranteed to be unique.

Returns:

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

Errors

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

Example

Example JSON-RPC request

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_connection_unlock",
  "params": [
    "DB0074F5"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": true
}
```

5.17. cbrx_device_get

From the hub specified by the serial number, get the key value. similar to cbrx_connection_get Only get values that are relevant to ports are accepted.

Note that this is slower then other methods if you need to do multiple operations on the same device.

Syntax: see API Call Structure

```
{
  "method": "cbrx_device_get",
  "params": [
usb-serial,
    dictionary-key
  ]
}
```

Parameter	Description
usb-serial	USB serial number
dictionary-key	as returned by a call to cbrx_connection_getdictionary see Get Dictionary for more information

Returns:

```
{
  "result": [dictionary-value]
}
```

dictionary-value is the key value that is specified see Get Dictionary for more information.

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_device_get",
  "params": [
    "0000802000184C390CD2002E",
    "USBSpeed"
  ]
}
```

Example successful response:

```
{
"jsonrpc": "2.0",
"id": 0,
"result": "High"
}
```

5.18. cbrx_discover

Discover Cambrionix units, obtain hub serial number.

Syntax: see API Call Structure

```
{
  "method": "cbrx_discover",
  "params": [unit]
}
```

Unit parameter	Description
local	Unit ID for hub attached
docks	Unit IDs for multiple hubs connected together and attached

Returns:

```
{
  "result": [hub-serial]
}
```

hub-serial is a string which is the serial number of the hub, each string is guaranteed to be unique.

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_discover",
  "params": [
    "local"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": [
    "000000897FD0505A"
  ]
}
```

5.19. cbrx_discover ("all")

Discover all units and return detailed information about the hubs and their connected devices. Note that only devices that show up in a USB scan will be included. Unlike the other discovery methods, instead of an array of serial numbers, it will be an object of serial numbers with contents.

Syntax: see Call Structure

```
{
  "method": "cbrx_discover",
  "params": ["all"]
}
```

Returns:

```
{
  "result": {
    "hub-serial":
    {
      "Status": "status",
      "Manufacturer": "manufacturer-name",
      "Firmware": "firmware-version",
      "Bootloader": "bootloader-version",
      "SerialNumber": "hub-serial",
      "Group": "group-order",
      "FormFactor": "firmware-type",
      "PanelID": hardware-id,
      "Hardware": "product-name",
      "HostSerialPort": "serial-port",
      "USBVersion": usb-version,
      "LocationID": location-ID,
      "nrOfPorts": port-quantity,
      "ExtPSU": external-PSU,
      "Uptime_sec": runtime,
      "Rebooted": reboot-flag,
      "SyncSupported": sync-possible,
      "FiveVolt": 5V-present,
      "TwelveVolt": 12V-present,
      "TemperatureMonitoring": temp-possible,
```

```
      "HardwareFlags": "hardware-flags",
      "Devices": {device-string}

    }

  }

}
```

Output	Description
hub-serial	This is the serial number of the hub returned from cbrx_discover
status	whether there is a serial port open, see Status
manufacturer-name	Defined name of manufacturer, Default is 'Cambrionix'
firmware-version	Version number of the firmware. Format 'N.nn'
bootloader-version	Version number of the bootloader. Format 'N.nn'
group-order	Used to order hubs which is useful when updating connected products so that down-stream products are updated and rebooted first.
firmware-type	Used to denote which firmware the product accepts
hardware-id	hardware ID number of front panel product
product-name	Hardware name of product
serial-port	The Serial port the product is connected to.
usb-version	The USB version number of the connection to the hub. Format 'N.nn'
location-ID	The Location IDs as an Integer
port-quantity	How many ports the product has
external-psu	Whether the product has an external power supply unit
runtime	How long the product has been powered (in ms). No limit

Output	Description
reboot-flag	Whether the reboot flag is true or false
sync-possible	Whether the product is capable of sync. true or false
5V-present	Whether the product has 5V supplied. true or false
12V-present	Whether the product has 12V supplied. true or false
temp-possible	Whether the product is can of monitor temperature. true or false
hardware-flags	Hardware flags as detailed in Get Dictionary
device-string	Information from connected devices, see Device string

Example

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_discover",
  "params": [
    "all"
  ]
}
```

Example successful response:


```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": {
    "000000897FD0505A": {
      "Status": "active",
      "Manufacturer": "cambrionix",
      "Firmware": "1.88.0",
      "Bootloader": "0.21",
      "SerialNumber": "000000897FD0505A",
      "Group": "-",
      "FormFactor": "un",
      "PanelID": 48,
      "Hardware": "SuperSync15",
      "HostSerialPort": "COM3",
      "USBVersion": 2.1,
      "LocationID": 574750720,
```

```
        "USB3CompanionLocationID": 2723151872,
        "HostPortLocationID": 574771200,
        "nrOfPorts": 15,
        "ExtPSU": true,
        "Uptime_sec": 167551,
        "Rebooted": true,
        "SyncSupported": true,
        "FiveVolt": true,
        "TwelveVolt": true,
        "TemperatureMonitoring": true,
        "HardwareFlags": "SLET",
        "Devices": {}
      }
    }
  }
```

5.20. cbrx_discover_id_to_os_reference

Map a unit ID for a discovered hub to a device name as used by the OS. This can only be used for locally attached Cambrionix products.

Syntax: see API Call Structure

```
{
  "method": "cbrx_discover_id_to_os_reference",
  "params": "[hub-serial]"
}
```

hub-serial is a string which is the serial number of the hub, each string is guaranteed to be unique.

Returns:

```
{
  "result": ["device-name"]
}
```

device-name is what the OS uses for the connection that the hub which is identified by the *hub-serial*. For more information please see Serial port

Example:

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_discover_id_to_os_reference",
  "params": [
    "DB0074F5"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": "COM9"
}
```

5.21. cbrx_exit

Restart the API.

Syntax: See API Call Structure

```
{
  "method": "cbrx_exit"
}
```

Returns:

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

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_exit"
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": true
}
```

5.22. cbrx_find

Search for devices attached to local Cambrionix units.

Syntax: see API Call Structure

```
{
  "method": "cbrx_find",
  "params": [ID]
}
```

ID can be any of the following forms:

ID Paramater	Description
VID	Search for any devices matching the vendor ID. Displayed as an Integer
VID & PID	Search for devices exactly matching the vendor and product IDs. Displayed as an Integer
NAME	Search for anything that matches the provided regex. The regex is run against a string made up of the manufacturer, product name, USB serial number and DeviePath (For an iPhone this is the UDID). "<manufacturer-name>\x1D<product-name>\x1D<serial-number>". The regex is performed as a search, rather than a match, so you do not need to do things like ".*iPhone.*" to match substrings; "iPhone" is sufficient. You can be as strict as you like. Additionally, if the phone's identity or internal serial number have been detected, then these will also be matched.

Returns:

```
{
  "result": {
    "usb-serial": {
      "HostDevice": "hub-serial",
      "HostPort": device-port,
      "HostDescription": "product-name",
```

```
      "HostSerial": "serial-port",
      "Device": ["device-string"]
      }

    }

  }

}
```

Output	Description
usb-serial	The serial number of the device
hub-serial	This is the serial number of the hub returned from cbrx_discover
device-port	Port number of the hub the device is attached too
product-name	Hardware name of product
serial-port	The Serial port the product is connected to.
device-string	Information from connected devices, see Device string

The returned data is keyed on the serial number of any devices matching the search criteria. The value of each node holds details of the location and the exact device details.

The entire USB tree is searched for the specified items, and if found anywhere beneath a Cambrionix hub, then the connection details will be returned. This would be especially useful for devices that are plugged into an intermediate hub device rather than being directly connected to the Cambrionix hub, such as a phone with battery extended and extra USB slots.

For any search results that do not have their own USB serial number, there will be an additional entry of NoSerial that is an array of such results, see information in below example.

Example

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_find",
  "params": [
    "i(Phone|Pad)"
  ]
}
```

Example successful response

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": {
    "974a9d1e6848316264a8a9d8b094b7d5e63a7ae5": {
      "HostDevice": "60003",
      "HostPort": 2,
      "HostDescription": "TS3-C10",
      "Device": {
        "VID": 1452,
        "PID": 4779,
        "Manufacturer": "Apple Inc.",
        "Description": "iPad",
        "SerialNumber": "974a9d1e6848316264a8a9d8b094b7d5e63a7ae5",
        "DeviceType": "Apple",
        "LocationID": 856686592,
        "DevicePath": "\\\\?\\usb#vid_1234&pid_5678##{a5dcbf10-653004fb951ed}",
        "USBVersion": 2,
        "USBPower": {
          "State": "D0",
          "Description": "On"
        },
        "USBSpeed": {
          "Speed": "480Mbps",
          "Description": "High"
        },
        "Endpoints": {
          "Active": 6,
          "Maximum": 8,
          "Memory": 32768
        },
        "Battery": {
          "DataSource": "imobiledevice",
          "TrustLevel": "paired",
          "PairingSupported": true,
          "CurrentLevel": 100,
          "CurrentTime": 1663145986,
          "StartingLevel": 100,
          "StartingTime": 1663145986,
          "CapacityNew": 11560,
          "Capacity": 11441,
          "ChargingStatus": "full",
          "HealthPercent": 98
        },
        "PhoneSerialNumber": "DLXKJ4QAF182",
        "PhoneIdentity": "iPad",
        "MacAddress": "60:fe:c5:b1:98:8c",
        "PhoneSoftwareVersion": "10.3.3"
      }
    },
    "8b2f4103b3b74117c5bc7ca57829cb0daedcad19": {
      "HostDevice": "60003",
      "HostPort": 1,
      "HostDescription": "TS3-C10",
      "Device": {
```

```
          "VID": 1452,
          "PID": 4779,
          "Manufacturer": "Apple Inc.",
          "Description": "iPad",
          "SerialNumber": "8b2f4103b3b74117c5bc7ca57829cb0daedcad19",
          "DeviceType": "Apple",
          "LocationID": 857735168,
          "USBVersion": 2,
          "USBPower": {
            "State": "D0",
            "Description": "On"
          },
          "USBSpeed": {
            "Speed": "480Mbps",
            "Description": "High"
          },
          "Endpoints": {
            "Active": 6,
            "Maximum": 8,
            "Memory": 32768
          },
          "Battery": {
            "DataSource": "imobiledevice",
            "LastError": "ideviceinfo returned PASSWORD_PROTECTED",
            "TrustLevel": "error",
            "PairingSupported": true
          }
        }
      }
    }
  }
```

5.23. cbrx_firmware

The firmware methods can control all aspects of updating the firmware on Cambrionix hubs. There are several sub-commands that allow you to add or remove firmware files from the API's local storage, list the currently available firmware files, update firmware from provided files and check the status of existing firmware updates.

The firmware type for can be one of "un" for Universal, "pd" for PDSync, "st" for the TS3-C10 or "mc" for motor control board. For information on your hubs requirements you can use theFirmware Requirements API method.

Firmware	Part Number	Product Name
Universal	PP15S	PowerPad15S
Universal	PP15C	PowerPad15C
Universal	PP8S	PowerPad8S
Universal	SS15	SuperSync15
Universal	TS3-16	ThunderSync3-16
SMART	TS3-C10	ThunderSync3-C10
Universal	U16S Spade	U16S Spade
Universal	U8S	U8S
PDSync	PDSync-C4	PDSync-C4
Universal	ModIT-Max	ModIT-Max
Motor Control	Motor control board	ModIT-Max

Syntax: see API Call Structure

```
{
  "method": "cbrx_firmware",
  "params": [firmware-call]
}
```

firmware-call	Description
add	Provides firmware to the API as an available update source. see cbrx_firmware (add)
remove	Removes firmware from being available to the API. see cbrx_firmware (remove)
list	List all available firmware. see cbrx_firmware (list)
update	Start the firmware update. see cbrx_firmware (update)
status	Get the status of the firmware update. see cbrx_firmware (status)

5.24. cbrx_firmware (add)

Adds the firmware file to the API local storage on the host running the API. Adding firmware files is done by providing a Base64 encoded zip of the file. For more information on Base64 encoding please see the following link.

Syntax: see Call Structure

```
{
  "method": "cbrx_firmware",
  "params": [
    "add",
    "filename",
    encoded-Bytes
  ]
}
```

Parameters	Description
filename	The name of the firmware file
encoded-Bytes	The encoded zip of the file in Base64

Returns:

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

Examples

Example JSON-RPC request: (note in this example the base64 encoded text has been reduced for simplification and clarity within the manual)

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_firmware",
  "params": [
    "add",
    "CambrionixFirmware-v1.87-un.enfir",
    "eJwsnduOLTFSbd9Lqn/8f8BywVzYg=="
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": true
}
```

5.25. cbrx_firmware (list)

Obtain a list of all available firmware versions

Syntax: see Call Structure

```
{
  "method": "cbrx_firmware",
  "params": ["list"]
}
```

Returns:

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

Parameter	Description
filename	Name of the file
firmware-version	Version number of the firmware

Example

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_firmware",
  "params": [
    "list"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": [
    {
      "filename": "CambrionixFirmware-1.0.3+39-00-st.enfir",
      "version": "1.0.3+39"
    },
    {
      "filename": "CambrionixFirmware-v1.86-un.enfir",
      "version": "1.86"
    },
    {
      "filename": "CambrionixFirmware-v1.87-un.enfir",
      "version": "1.87"
    }
  ]
}
```

5.26. cbrx_firmware (remove)

Remove the firmware file from the API local storage on the host running the API.

Syntax: see Call Structure

```
{
  "method": "cbrx_firmware",
  "params": [
    "remove","filename"
  ]
}
```

filename is the name of the firmware file.

Returns:

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

Example

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": "0",
  "method": "cbrx_firmware",
  "params": [
    "remove",
    "CambrionixFirmware-v1.86-un.enfir"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": "0",
  "result": true
}
```

5.27. cbrx_firmware (status)

This method can be used to obtain the status of a firmware update.

Syntax: see Call Structure

```
{
  "method": "cbrx_firmware",
  "params": [
    "status",
    "connection-handle",
  ]
}
```

connection-handle is the Connection Handles as an integer

Returns:

```
{
  "result": [
    "Version": "firmware-version",
    "Type": "firmware-type",
    "Progress": progress-percentage,
    "Stage": "stage-value"
  ]
}
```

Parameter	Description
firmware-version	Version number of the firmware
firmware-type	Used to denote the type of firmware
progress-percentage	The update progress as a percentage
stage-value	The "stage" the firmware update is currently in

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

Errors

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-failes	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

Example

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_firmware",
  "params": [
    "status",
    "7654"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": [
    "Version": "1.79",
    "Type": "un",
    "Progress": 60,
    "Stage": "verifying"
  ]
}
```

5.28. cbrx_firmware (update)

Use this method to start firmware updates, some products can have multiple firmware such as display, motor control firmware etc. You can start the update of multiple firmware with a single command as documented in the example.

Syntax: see Call Structure

```
{
  "method": "cbrx_firmware",
  "params": [
    "update",
    "connection-handle",
    "firmware-type""filename"
  ]
}
```

Parameter	Description
firmware-type	This is a two letter code to define the specific firmware type on the device, more information on the firmware type can be found in the cbrx_firmware section
filename	Name of the file. A particular firmware file can be used from those available, to obtain available files use cbrx_firmware (list)
connection-handle	The Connection Handles as an integer

From version 3.9 onwards with the Cambrionix Hub API you can substitute the connection-handle with the hubs serial number.

Returns:

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

Errors

If there is an error in the API method then a JSON-error object will be returned.

Example

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": "0",
  "method": "cbrx_firmware",
  "params": [
    "update",
    "7654",
    {
      "un":"CambrionixFirmware-v1.86-un.enfir",
      "mc":"CambrionixFirmware-v1.0.0-mc.enfir"
    }
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": "0",
  "result": true
}
```

5.29. cbrx_get_usb (tree)

Return the entire USB tree that has been discovered.

Syntax: see Call Structure

```
{
  "method": "cbrx_get_usb,
  "params": ["tree"]
}
```

Returns

```
{
  "result": [
    {
      "VID": vendor-id,
      "PID": product-id,
      "Description": "description",
      "LocationID": location-id,
      "USBVersion": USB-version,
      "USBPower": {
        "State": "power-state",
        "Description": "power-description"
        },
      "HostController": {
        "Type": "host-controller-type",
        "EndpointTotal": active-endpoints,
        "EndpointPeakTotal": peak-endpoints,
        "EndpointMemoryUsed": endoint-memory,
        "EndpointPeakMemoryUsed": peak-endpoint-memory
        },
      "children": [
        {
```

```
            "VID": vendor-id,
            "PID": product-id,
            "LocationID": location-id,
            "USBVersion": USB-version,
            "USBPower":

              {

              "State": "power-state",
              "Description": "power-description"
              },

            "USBSpeed":

              {

              "Speed": "USB-speed",
              "Description": "speed-name"
              },

            "Endpoints":

              {

              "Active": active-endpoints,
              "Maximum": maximum-endpoints,
              "Memory": endpoint-memory,

              "TotalInTree":
                {

                "Endpoints": tree-endpoints

                "Memory": tree-memory

              }

            }

          }

        ]

      }

  ]

}
```

Output	Description
Vendor-ID	Device Vendor ID, or VID. Displayed as an Integer

Output	Description
product-ID	Product ID, PID. Displayed as an Integer
description	Name of hardware
location-id	The Location IDs as an Integer
usb-version	The USB version number of the connection to the hub. Format 'N.nn'
power-state	USB Power States code
power-description	USB power turned on/off
host-controller-type	The type of USB host controller
peak-endpoints	Peak endpoint usage on the USB host controller. see Endpoints
peak-endpoint-memory	Peak endpoint memory usage. see Endpoints
USB-speed	Maximum speed USB connection capable of
speed-name	Name of USB connection i.e. SuperSpeed USB 5Gbps
active-endpoints	How many endpoints the device is using
maximum-endpoints	How many endpoints the device is capable of using
endpoint-memory	Amount of memory being used by endpoints
tree-endpoints	How many endpoints the tree is using
tree-memory	Amount of memory being used by endpoints in the tree

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_get_usb",
  "params": ["tree"]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": [
    {
      "VID": 32902,
      "PID": 40429,
      "Description": "Intel(R) USB 3.1 eXtensible HostController - 1.10
        (Microsoft)", "LocationID": 553648128, "USBVersion": 3.1, "USBPower": {
        "State": "D0",
      "Description": "On"
    },
    "HostController": {
      "Type": "XHCI",
      "EndpointTotal": 9,
      "EndpointPeakTotal": 60,
      "EndpointMemoryUsed": 57344,
      "EndpointPeakMemoryUsed": 331776
    },
    "children": [
      {
        "VID": 3141,
        "PID": 26403,
        "LocationID": 558891008,
        "USBVersion": 2.01,
        "USBPower": {
          "State": "D0",
          "Description": "On"
        },
        "USBSpeed": {
          "Speed": "480Mbps",
          "Description": "High"
        },
        "Endpoints": {
          "Active": 2,
          "Maximum": 3,
          "Memory": 12288
        }
      },
      {
        "VID": 1161,
        "PID": 57506,
        "LocationID": 560988160,
        "USBVersion": 1.1,
        "USBPower": {
          "State": "D2",
          "Description": "Low power"
        },
        "USBSpeed": {
          "Speed": "12Mbps",
          "Description": "Full"
        },
        "Endpoints": {
          "Active": 6,
          "Memory": 24576
```

```
          }
        },
        {
          "VID": 0,
          "PID": 0,
          "LocationID": 563085312,
          "USBVersion": 0,
          "USBSpeed": {
            "Speed": "1.5Mbps",
            "Description": "Low"
          },
          "Endpoints": {
            "Active": 1,
            "Memory": 4096
          }
        ]
      }
    ]
  }
```

5.30. cbrx_get_usb (descriptors)

Request entire dump of a USB device's descriptor information. This can be a lot of data for some devices (especially phones and tablets).

Syntax: see Call Structure

```
{
  "method": "cbrx_get_usb,
  "params": ["descriptors, locationID | hub-serial "]
}
```

Variable	Description
locationID	The Location IDs as an Integer
hub-serial	This is the serial number of the hub returned from cbrx_discover

Returns

All variable names in the returned data match the names in Chapter 9 of the USB 3.2 specification for ease of reference. Each descriptors raw fields (as taken from the USB 3.2 spec) are represented first, and where appropriate an additional "Derived" member will be present that shows bitfields or resolved string descriptors.

For example, on the device's main descriptor there is an iManufacturer field, which is the index of the string descriptor used for that name. This will also be present in Derived.Manufacturer.

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "method": "cbrx_get_usb",
  "params": ["descriptors", "123456789abcdef"],
  "id": 0
}
```

Example successful response from a standard USB flash drive:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": {
    "RawBytes": "120120030000000951092b17010001020301",
    "bLength": 18,
    "bDescriptorType": 1,
    "bNumConfigurations": 1,
    "bcdUSB": 800,
    "bDeviceClass": 0,
    "bDeviceSubClass": 0,
    "bDeviceProtocol": 0,
    "bMaxPacketSize0": 9,
    "idVendor": 2385,
    "idProduct": 5931,
    "bcdDevice": 1,
    "iManufacturer": 1,
    "iProduct": 2,
    "iSerialNumber": 3,
    "Derived": {
      "DescriptorType": "Device",
      "CurrentConfiguration": 1,
      "DeviceClass": "Reserved"
    },
    "Configurations": {
      "1": [
        {
          "RawBytes": "09022c00010100ff8025",
          "bLength": 9,
          "bDescriptorType": 2,
          "wTotalLength": 44,
          "bConfigurationValue": 1,
          "bmAttributes": 128,
          "bNumDescriptors": 1,
          "iConfiguration": 0,
          "reserved1": 0,
          "reserved2": 1,
          "SelfPowered": 0,
          "Derived": {
            "DescriptorType": "Configuration",
            "MaxPower": 37,
            "RemoteWakeUp": 0,
            "BusPowered": 0
          }
        },
        {
          "RawBytes": "090400000208065000",
          "bLength": 9,
          "bDescriptorType": 4,
          "iInterface": 0,
          "bInterfaceNumber": 0,
          "bAlternateSetting": 0,
          "bNumEndpoints": 2,
          "bInterfaceClass": 8,
          "bInterfaceSubClass": 6,
```

```
      "bInterfaceProtocol": 80,
      "Derived": {
        "DescriptorType": "Interface"
      }
    },
    {
      "RawBytes": "0705ff8102000400",
      "bLength": 7,
      "bDescriptorType": 5,
      "bInterval": 0,
      "bEndpointAddress": 129,
      "bmAttributes": 2,
      "wMaxPacketSize": 1024,
      "Derived": {
        "DescriptorType": "Endpoint",
        "EndpointAddress": 1,
        "Direction": "In",
        "Type": "Bulk"
      }
    },
    {
      "RawBytes": "06300f000000",
      "bLength": 6,
      "bDescriptorType": 48,
      "wBytesPerInterval": 0,
      "bMaxBurst": 15,
      "bmAttributes": 0,
      "Derived": {
        "DescriptorType": "SuperSpeedEndpointCompanion",
        "MaxStreams": 0,
        "Mult": 0,
        "SspCompanion": 0
      }
    },
    {
      "RawBytes": "07050202000400",
      "bLength": 7,
      "bDescriptorType": 5,
      "bInterval": 0,
      "bEndpointAddress": 2,
      "bmAttributes": 2,
      "wMaxPacketSize": 1024,
      "Derived": {
        "DescriptorType": "Endpoint",
        "EndpointAddress": 2,
        "Direction": "Out",
        "Type": "Bulk"
      }
    },
    {
      "RawBytes": "06300f000000",
      "bLength": 6,
      "bDescriptorType": 48,
      "wBytesPerInterval": 0,
      "bMaxBurst": 15,
```

```
            "bmAttributes": 0,
            "Derived": {
              "DescriptorType": "SuperSpeedEndpointCompanion",
              "MaxStreams": 0,
              "Mult": 0,
              "SspCompanion": 0
            }
          }
        ]
      },
      "Strings": {
        "1": "Kingston",
        "2": "DataTraveler 70",
        "3": "1831BFBD3065F551C96001E7"
      },
      "BOS": {
        "RawBytes": "050f160002",
        "bLength": 5,
        "bDescriptorType": 15,
        "Derived": {
          "DescriptorType": "BOS"
        },
        "wTotalLength": 22,
        "bNumDescriptors": 2,
        "Capabilities": [
          {
            "RawBytes": "07100206000000",
            "bLength": 7,
            "bDescriptorType": 16,
            "bDevCapabilityType": 2,
            "bmAttributes": 6,
            "Derived": {
              "DescriptorType": "DeviceCapability",
              "CapabilityType": "USB20Extension",
              "LPMCapable": 1,
              "BESLAndAlternateHIRDSupported": 1,
              "BaselineBESLValid": 0,
              "DeepBESLValid": 0,
              "BaselineBESL": 0,
              "DeepBESL": 0
            }
          },
          {
            "RawBytes": "0a1003000e00020affff07",
            "bLength": 10,
            "bDescriptorType": 16,
            "bDevCapabilityType": 3,
            "wU2DevExitLat": 2047,
            "bmAttributes": 0,
            "wSpeedsSupported": 14,
            "bFunctionalitySupport": 2,
            "bU1DevExitLat": 10,
            "Derived": {
              "DescriptorType": "DeviceCapability",
              "CapabilityType": "SuperSpeedUSB",
```

```
                "LTMCapable": 0,
                "SpeedsSupported": [
                  "Full",
                  "High",
                  "SuperSpeed"
                ]
              }
            }
          ]
        }
      }
    }```
```

5.31. cbrx_hub_get

From the hub specified by the hubs serial number, get the key value. similar to cbrx_connection_get.

Note that this is function will be slower if you need to do multiple operations on the same hub.

Syntax: see API Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    hub-serial,
    dictionary-key
  ]
}
```

Parameter	Description
hub-serial	This is the serial number of the hub returned from cbrx_discover
dictionary-key	See Get Dictionary for more information

Returns:

```
{
  "result": [dictionary-value]
}
```

dictionary-value is the key value that is specified see Get Dictionary for more information.

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_hub_get",
  "params": [
    "000000897FD0505A",
    "nrOfPorts"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": 15
}
```

5.32. cbrx_hub_set

On the hub specified by the connection handle, set the key value. Similar to cbrx_connection_set

Note that this function will be slower if you need to do multiple operations on the same hub.

Syntax: see API Call Structure

```
{
  "method": "cbrx_hub_set",
  "params": [
    hub-serial,
    dictionary-key,
    Value
  ]
}
```

Parameter	Description
hub-serial	This is the serial number of the hub returned from cbrx_discover
dictionary-key	See Set Dictionary for more information
Value	The value you wish to apply to the key

Returns:

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

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_hub_set",
  "params": [
    "7FD0505A",
    "TwelveVoltRail.OverVoltage",
    true
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": "0",
  "result": true
}
```

5.33. cbrx_notifications

The API supports sending of notifications for certain events. Notification packets are the same as other JSON-RPC Response objects, except that they do not have an "id" field.

Note: Notifications are only sent to active socket connections that have requested them. Closing a socket and opening another one will mean you need to re-request notifications. See the Quick start for examples

Syntax: see API Call Structure

```
{
  "method": "cbrx_notifications",
  "params": ["notification"],
}
```

notification is from the array of strings to shown when sending cbrx_apidetails a full list of possible notifications can be found in the section API Notifications

Returns:

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

Errors

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_notifications",
```

```
  "params": [
    "usb-device-attached"
  ]
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": true
}
```

5.34. cbrx_pair_device

Initiate pairing of an iOS device. This is not usually necessary as it will occur automatically when the API attempts to query battery information.

Syntax: see API Call Structure

```
{
  "method": "cbrx_pair_device,
  "params": ["UDID"]
}
```

Variable	Description
UDID	The device USB serial number

Returns

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

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_pair_device",
  "params": ["1234567"]
}
```

Example successful response:

```
{
```

```
    "jsonrpc": "2.0",
    "id": 0,
    "result": true
  }
```

6. API Notifications

API notification packets are the same as other JSON-RPC packets, except that they do not have an "id" field. Most of these notifications do not supply anything in the "params" field. These notifications will only be sent if they are enabled using the cbrx_notifications method.

A list of possible notitifcations available using our API can be found in the table below. Most of these notifications do not supply anything in the "params" field.

Notification	Description
all	Request all notifications
discover-changed	The API detected a change in the available hubs. You should re-run cbrx_discover at this point.
dead-hub-changed	The API detected that a hub has either become unresponsive or cannot be connected to
firmware-progress	Request updates on firmware update progress
over-temperature	Hub is over temperature
over-voltage	Hub is over voltage
rfid-received	Receive notification when a RFID card is presented to a sensor
rfid-removed	Receive notification when an RFID card is removed from a sensor
usb-device-attached	More detailed than usb-changed, this will tell you of specific devices that have attached.
usb-device-detached	More detailed than usb-changed, this will tell you of specific devices that have detached.
under-voltage	Hub is under voltage
usb-changed	The API detected a change in the USB Tree.

discover-changed

A change in the hubs that are available to the API has been detected.

Example notification packet:

```
 {
  "jsonrpc": "2.0",
  "method": "discover-changed"
}
```

dead-hub-changed

The API detected that a hub has either become unresponsive or cannot be connected to, for example because another program has it's serial port opened.

Example notification packet:

```
 {
  "jsonrpc": "2.0",
  "method": "dead-hub-changed",
  "params": {
    "IsDead":True
  }
}
```

firmware-progress

Information on the firmware update progress. For more information on the output see cbrx_firmware (status)

Example notification packet:

```
 {
  "jsonrpc": "2.0",
  "method": "firmware-progress",
  "params": {
    "Progress": 60,
    "Stage": "flashing",
    "Type": "charger",
    "HostDevice": "1212343456567878",
    "HostSerial": "/dev/tty.usbmodem1421502",
    "HostDescription": "PS15-USB3"
  }
}
```

over-temperature

The hub is over operating temperature, see product user manuals for more details.

Example notification packet:

```
{
  "jsonrpc": "2.0",
  "method": "over-temperature"
}
```

over-voltage

The hub is over recomended voltage, see product user manuals for more details.

Example notification packet:

```
 {
  "jsonrpc": "2.0",
  "method": "over-voltage"
}
```

rfid-received

An RFID sensor has detected that a RFID card is present.

Example notification packet:

```
 {
  "jsonrpc": "2.0",
  "method": "rfid-received",
  "params": "12784556655489628"
}
```

rfid-removed

An RFID sensor has detected that a RFID card has been removed.

Example notification packet:

```
{
  "jsonrpc": "2.0",
  "method": "rfid-removed",
  "params": "12784556655489628"
}
```

usb-device-attached

A device has become available to the API and a detailed information output is presented about the device.

Example notification packet:

```
 {
  "jsonrpc": "2.0",
  "method": "usb-device-attached",
  "params": {
    "HostDevice": "1212343456567878",
    "HostSerial": "/dev/tty.usbmodem1421502",
    "HostPort": 7,
    "HostDescription": "PS15-USB3",
    "USB2": {
      "Description": "iPhone",
      "LocationID": 573710336,
      "Manufacturer": "Apple Inc.",
      "PID": 4776,
      "SerialNumber": "012a37d1fa07617ad7ef0430ba49f479ab9fb6b8",
      "USBVersion": 2,
      "VID": 1452
    }
  }
}
```

usb-device-detached

A device is no longer available to the API and a detailed information output is presented about the device.

Example notification packet:


```
{
  "jsonrpc": "2.0",
  "method": "usb-device-detached",
  "params": {
    "HostDevice": "1212343456567878",
    "HostSerial": "/dev/tty.usbmodem1421502",
    "HostPort": 7,
    "HostDescription": "PS15-USB3",
    "USB2": {
      "Description": "iPhone",
      "LocationID": 573710336,
      "Manufacturer": "Apple Inc.",
      "PID": 4776,
      "SerialNumber": "012a37d1fa07617ad7ef0430ba49f479ab9fb6b8",
      "USBVersion": 2,
      "VID": 1452
    }
  }
}
```


under-voltage

The hub is under recomended voltage, see product user manuals for more details.

Example notification packet:


```
 {
  "jsonrpc": "2.0",
  "method": "under-voltage"
}
```

usb-changed

There has been a change in the USB-tree.

Example notification packet:

```
{
  "jsonrpc": "2.0",
  "method": "usb-changed"
}
```

7. Deprecated Methods

These methods exist to support backwards compatibility only and should not be used. These methods may be removed in future versions.

API Call	Description
cbrx_apiversion (true)	Obtain a detailed version of the API

7.1. cbrx_apiversion (true)

!This method was deprecated in API version 3.0 please use 'cbrx_apidetails!

Return a detailed version of the API running.

Syntax: see API Call Structure

```
{
  "method": "cbrx_apiversion",
  "params": [true]
}
```

Returns:

```
{
  "result": {
    "version": [version-number],
    "semver": "semver-variant",
    "commitid": commitid-number,
    "branch": "branch-name",
    "capability": [API-capability],
    "notifications": [possible-notification],
    "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	The full name of the API version
commitid-variant	The number value of the Commit ID
branch-name	The branch of API installed
API-capability	API information for Cambrionix internal use
possible-notification	Array of strings to show possible notifications. see API Notifications
install-location	The location of install files
logs-location	The location of where logs are stored
settings-location	The location of API settings
documentation-location	The web adress of API documentation
brand-information	The brand of the CPU
CPU-architecture	The architecture of the CPU
CPU-features	Features available on CPU
cores-value	How many cores the CPU has
os-information	Operating system running on local machine

Errors

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

Examples

Example JSON-RPC request:

```
{
```

```
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_apiversion",
  "params": [
    True
  ]
}
```

Example Successful response

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": {
    "version": [
      3,
      7,
      0,
      34
    ],
    "semver": "3.7.0+34",
    "guid": {
      "id": "d0dc3cac-e165-4e38-88bb-39064431bdc9",
      "computerId": "35aea4bc-44d3-4e9e-9b3c-c33b965c5639"
    },
    "host": [
      {
        "ip": "10.167.111.81",
        "port": 0,
        "nameServer": "10.167.111.241",
        "domainName": "CBRX.LOCAL",
        "hostName": "CBRXPC-011",
        "adapterName": "Intel(R) Ethernet Controller (3) I225-V",
        "adapterType": "Ethernet"
      }
    ],
    "commitid": 4287981321,
    "branch": "release",
    "capability": [
      "protobuf",
      "crash-report",
      "notification"
    ],
    "notifications": [
      "usb-changed",
      "usb-device-attached",
      "usb-device-detached",
      "discover-changed",
      "dead-hub-changed",
      "firmware-progress",
      "rfid-received",
      "rfid-removed",
```

```
        "over-temperature",
        "over-voltage",
        "under-voltage",
        "certificate-changed"
      ],
      "install": "C:\\Program Files\\Cambrionix\\API",
      "logging": "C:\\ProgramData\\Cambrionix\\Log",
      "settings": "C:\\ProgramData\\Cambrionix",
      "documentation": "C:\\Program Files\\Cambrionix\\API\\Cambrionix Hub API
  Reference.html",
      "cpu": {
        "brand": "12th Gen Intel(R) Core(TM) i9-12900K",
        "arch": "x64",
        "features": [
          "aes",
          "avx",
          "avx2",
          "bmi1",
          "bmi2",
          "clflushopt",
          "clfsh",
          "clwb",
          "cx16",
          "cx8",
          "erms",
          "f16c",
          "fma3",
          "fpu",
          "mmx",
          "movbe",
          "pclmulqdq",
          "popcnt",
          "rdrnd",
          "rdseed",
          "sha",
          "smx",
          "ss",
          "sse",
          "sse2",
          "sse3",
          "sse4_1",
          "sse4_2",
          "ssse3",
          "tsc",
          "vaes",
          "vpclmulqdq"
        ],
        "cores": 24
      },
      "os": "Windows 10 Pro 21H2 Build 19044.1889 64-bit"
    }
  }
```

7.2. cbrx_get_usbtree

!This method was deprecated in API version 3.5 please use 'cbrx_get_usb (tree)'!

Return the entire USB tree that has been discovered.

Syntax: see API Call Structure

```
{
  "method": "cbrx_get_usbtree",
}
```

Returns

```
{
  "result": [
    {
      "VID": vendor-id,
      "PID": product-id,
      "Description": "description",
      "LocationID": location-id,
      "USBVersion": USB-version,
      "USBPower": {
        "State": "power-state",
        "Description": "power-description"
        },
      "HostController": {
        "Type": "host-controller-type",
        "EndpointTotal": active-endpoints,
        "EndpointPeakTotal": peak-endpoints,
        "EndpointMemoryUsed": endoint-memory,
        "EndpointPeakMemoryUsed": peak-endpoint-memory
        },
      "children": [
        {
```

```
            "VID": vendor-id,
            "PID": product-id,
            "LocationID": location-id,
            "USBVersion": USB-version,
            "USBPower":

              {

              "State": "power-state",
              "Description": "power-description"
              },

            "USBSpeed":

              {

              "Speed": "USB-speed",
              "Description": "speed-name"
              },

            "Endpoints":

              {

              "Active": active-endpoints,
              "Maximum": maximum-endpoints,
              "Memory": endpoint-memory

            }

          }

        ]

      }

  ]

}
```

Output	Description
Vendor-ID	Device Vendor ID, VID. Displayed as an Integer
product-ID	Product ID, PID. Displayed as an Integer
description	Name of hardware
location-id	The Location IDs as an Integer

Output	Description
usb-version	The USB version number of the connection to the hub. Format 'N.nn'
power-state	USB Power States code
power-description	USB power turned on/off
host-controller-type	The type of USB host controller
peak-endpoints	Peak endpoint usage on the USB host controller. see Endpoints
peak-endpoint-memory	Peak endpoint memory usage. see Endpoints
USB-speed	Maximum speed USB connection capable of
speed-name	Name of USB connection i.e. SuperSpeed USB 5Gbps
active-endpoints	How many endpoints the device is using
maximum-endpoints	How many endpoints the device is capable of using
endpoint-memory	Amount of memory being used by endpoints

Examples

Example JSON-RPC request:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_get_usbtree"
}
```

Example successful response:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": [
    {
      "VID": 32902,
      "PID": 40429,
```

```
      "Description": "Intel(R) USB 3.1 eXtensible HostController - 1.10
        (Microsoft)", "LocationID": 553648128, "USBVersion": 3.1, "USBPower": {
        "State": "D0",
      "Description": "On"
    },
    "HostController": {
      "Type": "XHCI",
      "EndpointTotal": 9,
      "EndpointPeakTotal": 60,
      "EndpointMemoryUsed": 57344,
      "EndpointPeakMemoryUsed": 331776
    },
    "children": [
      {
        "VID": 3141,
        "PID": 26403,
        "LocationID": 558891008,
        "USBVersion": 2.01,
        "USBPower": {
          "State": "D0",
          "Description": "On"
        },
        "USBSpeed": {
          "Speed": "480Mbps",
          "Description": "High"
        },
        "Endpoints": {
          "Active": 2,
          "Maximum": 3,
          "Memory": 12288
        }
      },
      {
        "VID": 1161,
        "PID": 57506,
        "LocationID": 560988160,
        "USBVersion": 1.1,
        "USBPower": {
          "State": "D2",
          "Description": "Low power"
        },
        "USBSpeed": {
          "Speed": "12Mbps",
          "Description": "Full"
        },
        "Endpoints": {
          "Active": 6,
          "Memory": 24576
        }
      },
      {
        "VID": 0,
        "PID": 0,
        "LocationID": 563085312,
        "USBVersion": 0,
```

```
          "USBSpeed": {
            "Speed": "1.5Mbps",
            "Description": "Low"
          },
          "Endpoints": {
            "Active": 1,
            "Memory": 4096
          }
        ]
      }
    ]
  }
```

8. Device string

When the API queries a device it can return the below string.

```
"Device": {
  "VID": vendor-ID,
  "PID": product-ID,
  "Manufacturer": "device-manufacturer",
  "Description": "description",
  "SerialNumber": "usb-serial",
  "LocationID": location-id,
  "DevicePath": device-path
  "USBVersion": usb-version,
  "USBPower": {
    "State": "power-state",
    "Description": "power-description"
    },
  "USBSpeed": {
    "Speed": "USB-speed",
    "Description": "USB-description"
    "Capability": {
      "Speed": "capable-speed",
      "Description": "capable-description"
      }
    },
  "Endpoints": {
    "Active": active-endpoints,
    "Maximum": maximum-endpoints,
    "Memory": endpoint-memory
```

```
    },
  "Battery": {
    "DataSource": "battery-data-source",
    "TrustLevel": "trust-level",
    "PairingSupported": support-pairing,
    "CurrentLevel": battery-current-level,
    "CurrentTime": current-hub-time,
    "StartingLevel": charge-start-level,
    "StartingTime": charge-start-time,
    "CapacityNew": new-battery-capacity,
    "Capacity": current-battery-capacity,
    "ChargingStatus": "charge-status",
    "HealthPercent": battery-health,
    "Temperature": device-temperature
    },
  "PhoneSerialNumber": "phone-serial",
  "PhoneIdentity": "phone-name",
  "PhoneModel": "phone-model",
  "IMEI": "IMEI-number",
  "MacAddress": "MacAddress",
  "PhoneSoftwareVersion": "phone-OS-version",
  "PhoneECID": ECID,
  "PhoneProductType": "phone-product",
  "PhoneOSType": "phone-OS",
  "PhoneColour": "phone-colour"
  }
}
```

Output	Description
vendor-ID	Device Vedor ID number, VID. Displayed as an Integer
product-ID	Product ID number, PID. Displayed as an Integer
device-manufacturer	The name of the device manufacturer
description	Name of the hardware
usb-serial	USB serial number
location-id	The Location IDs as an Integer
device-path	The platform specific path for the device
usb-version	The USB version number of the connection to the hub
power-state	USB Power States code
power-description	USB power turned on/off
USB-speed	Maximum speed USB connection capable of
USB-description	Name of USB connection i.e. SuperSpeed USB 5Gbps
capable-speed	Maximum data speed device capable of
capable-description	Name of maximum data speed device capable of
active-endpoints	How many endpoints the device is using, displayed as a integer
maximum-endpoints	How many endpoints the device is capable of using, displayed as a integer
endpoint-memory	Amount of memory being used by endpoints, displayed as a integer
battery-data-source	The source of the device battery information

Output	Description
trust-level	Whether the device is trusted/ paired
support-pairing	Whether the device supports being trusted/ paired
battery-current-level	Current battery percent level of device, displayed as a integer
current-hub-time	The hub time in ms, shown as an integer in ms
charge-start-level	Battery percentage level when device connected, displayed as a integer
charge-start-time	The hub time in ms charging started, shown as an integer in ms
new-battery-capacity	The battery capacity of device from new, displayed as a integer
current-battery-capacity	The battery capacity of the device now, displayed as a integer
charge-status	Charging status of the battery i.e. full
battery-health	Battery health percentage, displayed as a integer
device-temperature	The temperature the device is reporting
phone-serial	Phone serial number
phone-name	Name of the phone
phone-model	The model of the phone i.e. "iPhone 12"
IMEI-number	The IMEI number of the phone
MacAdress	A unique address assigned to the mobile device. It is a 48 bit value, consisting of twelve hexadecimal characters

Output	Description
phone-os-version	Version number of the OS on the phone
ECID	Exclusive Chip Identification also referred to as Unique Chip ID
phone-product	Mobile device code i.e. "iPhone13,2"
phone-OS	The OS version of the phone i.e. "iPhone OS"
phone-colour	The colour of the phone

9. API Management

9.1. Stopping the API service

To stop the Cambrionix Hub API service, the process varies slightly depending on your operating system. Below are detailed instructions for Windows, macOS and Linux users.

Windows

If you wish to stop the Cambrionix Hub API service on a Windows machine, you can easily do so through the Task Manager:

1. Open Task Manager:

   Right-click on the Taskbar and select Task Manager, or press Ctrl + Shift + Esc to open it directly.

2. Navigate to Services:

   In Task Manager, click on the Services tab to view all the services running on your system.

3. Locate the 'CambrionixApiService'Service:

   Scroll through the list of services until you find 'CambrionixApiService'.

4. Stop the Service:

   Right-click on the 'CambrionixApiService'service and select Stop from the context menu. This will immediately stop the service, halting all Cambrionix Hub API related functionality until the service is restarted.

Linux

For Linux, the Cambrionix Hub API service can be stopped from the command line. Most modern Linux distributions, such as Ubuntu, Fedora, and Debian, use systemd for service management.

To stop the service, use the following command:

```
sudo systemctl stop CambrionixApiService
```

macOS

On macOS, services like the Cambrionix Hub API are managed by launchd, which handles system-wide and user-level services.

To stop the Cambrionix Hub API service, use the following command:

```
sudo /usr/bin/CambrionixApiService--remove
```

9.2. Starting the API Service

To start the Cambrionix Hub API service, the process will vary depending on your operating system. Below are the instructions for Windows, Linux, and macOS users.

Windows

To start the Cambrionix Hub API service on a Windows machine, follow these steps:

1. Open Task Manager:

   Right-click on the Taskbar and select Task Manager, or press Ctrl + Shift + Esc to open it directly.

2. Navigate to Services:

   In Task Manager, click on the Services tab to view all the services running on your system.

3. Locate the 'CambrionixApiService'Service:

   Scroll through the list of services until you find 'CambrionixApiService'.

4. Start the Service:

   Right-click on the 'CambrionixApiService'service and select Start from the context menu. This will immediately start the service, enabling the API to handle requests again.

Linux

For Linux, where the Cambrionix Hub API service is managed by systemd, you can start the service using the following command:

```
sudo systemctl start CambrionixApiService
```

To check that the service is running, you can use:

```
sudo systemctl status CambrionixApiService
```

macOS

On macOS, the Cambrionix Hub API service is managed by launchd. To start the service, use the following command:

```
sudo /usr/bin/CambrionixApiService--install
```

To check if the service is running, you can run:

```
sudo launchctl list | grep cambrionix
```

10. Additional information

Cambrionix Connect Recorder Service

The Recorder service is an optional installation component, which can record events such as device health, charging history and connection events. These can subsequently be viewed in client software.

Stopping the API service

- Windows

If you wish to stop the Cambrionix Hub API service from running then open Task Manager, then click through to services and right click the 'CambrionixAPIService' and click 'Stop'

- Linux and macOS

If you wish to stop the Cambrionix Hub API service from running then issue the following command

```
sudo /usr/bin/CambrionixApiService --remove
```

Starting the API Service

- Windows

To start the Cambrionix Hub API service from running then open Task Manager, then click through to services and right click the 'CambrionixAPIService' and click 'Start'

- Linux and macOS

If you wish to start the Cambrionix Hub API service from running then issue the following command

```
sudo /usr/bin/CambrionixApiService --install
```

Limitations

The API provides a means of controlling most of the features of Cambrionix products, however there are some limitations. The API can only be used with the following products.

Firmware	Part Number	Product Name
Universal	PP15S	PowerPad15S
Universal	PP15C	PowerPad15C
Universal	PP8S	PowerPad8S
Universal	SS15	SuperSync15
Universal	TS3-16	ThunderSync3-16
SMART	TS3-C10	ThunderSync3-C10
Universal	U16S Spade	U16S Spade
Universal	U8S	U8S
PDSync	PDSync-C4	PDSync-C4
Universal	ModIT-Max	ModIT-Max
Motor Control	Motor control board	ModIT-Max

Preventing Windows from Assigning New COM Ports to Identical USB Devices

When you connect multiple USB devices with the same hardware to a Windows PC, the operating system typically identifies each device by its unique hardware serial number. Windows uses this serial number to assign a new COM port for each device, even if they are identical in terms of hardware. Over time, this can result in a long list of COM ports being assigned, which clutters the device manager and can make managing devices more difficult and confusing.

To prevent this, you can instruct Windows to ignore the hardware serial number for specific USB devices. By doing so, Windows will treat all devices with the same Vendor ID (VID) and Product ID (PID) as a single device and allocate only one COM port, no matter how many of these identical devices are connected over time. This stops the COM port list from unnecessarily filling up.

The following registry entry ensures that Windows ignores the hardware serial number for a USB device with the specific VID 0403 and PID 6015:

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\usbflags]
"IgnoreHWSerNum04036015"=hex:01
```

- IgnoreHWSerNum04036015: This entry instructs Windows to ignore the hardware serial number for devices with a Vendor ID (0403) and Product ID (6015). These IDs refer to a specific USB device model.
- hex:01: This value enables the behaviour, telling Windows to treat all devices with this VID and PID as if they have the same serial number.

With this registry setting applied, Windows will no longer assign a new COM port each time an identical device (with the same VID and PID) is connected. Instead, it will reuse the same COM port, preventing the list of COM ports from unnecessarily expanding. This makes it easier to manage USB devices that connect through COM ports, particularly in environments where multiple identical devices are frequently connected and disconnected. It prevents the clutter and confusion caused by having a large number of assigned but unused COM ports.

This solution is particularly useful in scenarios such as:

- Development environments where multiple USB hubs are connected and disconnected.
- Production setups where identical devices are frequently used, and preventing an accumulation of redundant COM ports is necessary.
- Device testing environments where serial communication is crucial, but the aim is to avoid an ever-expanding list of COM ports

Connection Handles

A connection handle is a representation of a physical connection. The connection handle defines not only which driver to use but which data source to use with that driver. Within a segment of code the connection handle identifies a structure that contains connection information.

You can obtain a connection handle by using cbrx_connection_open

If there is no activity on an open handle for more than 30s, the handle will become invalid. Subsequent calls attempting to use a deleted handle will fail with CBRXAPI_ERRORCODE_ INVALIDHANDLE. Software using the API must be able to cope with this situation and respond accordingly. Software may simply call cbrx_connection_open again in order to obtain a fresh handle, or if you register for notifications they will be left open indefinitley.

Location IDs

A Location ID is a 32-bit unsigned integer that represents the location of the device in the USB tree. For example, if the driver creates a location string for the device of the form a&b&c, the Location ID will be 0x00000abc.

Serial port

The serial port the device is connected to will vary in its description depending on the OS the API is running on. For macOS and Linux this will be shown as the location, for Windows this will be shown as a COM port, please see examples below.

Windows

```
COM5
```

macOS

```
dev/tty.usbmodem141502
```

Linux

```
/dev/ttyUSB0
```

Endpoints

If you experience the endpoint limit, you are likely to see an error indicating you have “run out of USB resources” on your host computer, although you may not see any error message at all and (any of) your USB devices may randomly fail to operate correctly or become intermittent.

The USB endpoint limitation applies to xHCI (USB3) host controllers on recent motherboards only. These USB host controllers have limited memory and typically offer between 64 and 128 USB endpoints.

The problem with the limited number of endpoints provided by xHCI host controllers is that most USB devices, especially mobile phones and tablets, enumerate as more than one endpoint. A typical mobile device might enumerate as 5 endpoints. Consequently, if your host controller has 64 available endpoints you would only be able to connect 64/5 devices to that host controller, equivalent to 12 devices. Please be aware that all USB devices, including USB hubs, require endpoints, so when you are setting up your work flow it is important to factor in this limitation.

Power States

Device power states are named D0, D1, D2, and D3. D0 is the fully on state, and D1, D2, and D3 are low-power states. The state number is inversely related to power consumption: higher numbered states use less power.

Status

The API can return the following options on the status of the hub connection.

status response	Description
idle	Hub is connected but not talking
active	Hub is connected and in use
missing	Hub is no longer on the USBtree
unresponsive	Hub is no longer responding
locked	Hub connection is locked

11. Logging

The API is able to generate logging information for all USB events, and store information on what has happened and specific hardware information. This is useful to see what is happening with the API and can capture any faults or issues.

To enable logging you will need to create a config file ending in .log.cfg . You can use the following command to create the logging cfg file manually:

```
echo*=DEBUG>/etc/opt/cambrionix/cambrionix.log.cfg
```

Then after re-producing the problem, you can zip the logs from the folder

```
/var/log/cambrionix
```

You may delete the file below when you are finished with it.

```
/etc/opt/cambrionix/cambrionix.log.cfg
```

Default locations

Log messages generated by the CambrionixApiService go to syslog.

Using Windows the logs will default to the below location

```
C:\ProgramData\Cambrionix
```

Using macOS the logs will default to the below location

```
Library>Logs>Cambrionix
```

Using Linux the logs will default to the below location

```
/var/log/cambrionix
```

Logging to investigate behaviour

If you are experiencing a bug or an issue, you can obtain logs of the behaviour, to see in more detail what is happening.

1. Enable logging on the API
2. Use the hub in a way that causes the issue you are seeing.
3. Wait for the issue to occur
4. Make a note of the time that the issue occurs then zip the folder the logs are stored.

Once you have this information you can either review the logs yourself or if you are in contact with Cambrionix regarding support you can send the logs via the support ticket system.

List of logging options.

There is a list of different options for logs to capture within the Cambrionix Hub API below is a list of all logs that can be enabled directly from the .log.cfg file and what needs to entered in the file to enable the specific logs.

We would advise to have all logging enabled with the API so if any issue occurs the event is captured in the logs and investigation can take place.

api.battery=DEBUG
api.client=DEBUG
api.client.getset=DEBUG
api.core=DEBUG
api.core.discovery=DEBUG
api.daemon=DEBUG
api.daemon.handle.manager=DEBUG
api.dictionary=DEBUG
api.encoder=DEBUG
api.hub=DEBUG
api.hub.state=DEBUG
api.json=DEBUG
api.json.socket=DEBUG
api.json.websocket=DEBUG
api.serial=DEBUG
api.usbtree=DEBUG
lib.console=DEBUG
lib.filesystem.watcher=DEBUG
lib.network=DEBUG
lib.service=DEBUG
lib.service.user=DEBUG
lib.settings=DEBUG
lib.task=DEBUG
lib.thread=DEBUG
lib.timer=DEBUG
lib.watchdog=DEBUG

12. Docks

When multiple products have been connected, with the second charger connected to an expansion port of the first charger and so on, this is known as a Dock. For some operations it may be convenient to treat these two chargers as a single unit, that combines the ports of both chargers.

To access the dock as a single unit, first call cbrx_discover with the parameter docks to obtain the list of docks available. Then call cbrx_connection_open with the ID in question and also specifying "docks".

The dock unit will return the combined total for keys such as nrOfPorts and TotalCurrent_mA. The range of ports is expanded to cover the combined total number of ports for connected products. The product directly connected to the computer will have its ports referenced first followed by those of the charger connected to first charger's expansion port.

Some keys such as Hardware or Firmware do not combine and so these keys will return the value for the parent charger. If it is desired to get the values of these keys from the downstream charger then it is possible to open and retrieve them from that charger in the usual manner. Opening a charger does not interfere with access to the dock except as to when settings are changed.

13. Dynamic Hubs

It is possible to open a dynamic hub that is a combination of various other hubs. This behaves in the same way that Docks do. To open a dynamic hub, simple combine the serial numbers of all the hubs you wish to open into a special "Dynamic:" prefixed name as shown in this example.

```
# Given three Cambrionix hubs with serial numbers of 'AAAAAAAA',
        'BBBBBBBB' and 'CCCCCCCC'   handleA =
        cbrxapi.cbrx_connection_open("AAAAAAAA") handleB =
        cbrxapi.cbrx_connection_open("BBBBBBBB") handleC =
        cbrxapi.cbrx_connection_open("CCCCCCCC") handleABC =
        cbrxapi.cbrx_connection_open("Dynamic:AAAAAAAA:BBBBBBBB:CCCCCCCC")
        print(cbrxapi.cbrx_connection_get(handleA, "nrOfPorts")) # 15
        print(cbrxapi.cbrx_connection_get(handleB, "nrOfPorts")) # 8
        print(cbrxapi.cbrx_connection_get(handleC, "nrOfPorts")) # 8
        print(cbrxapi.cbrx_connection_get(handleABC, "nrOfPorts")) # 31
```

This dynamic hub is treated as a single entity with it's ports being numbered from 1 to N, where N is the total number of ports across all hubs included.

14. Dictionaries

For each hub, the API can return two dictionaries:

- The Get dictionary, containing keys that can be read.
- The Set dictionary, containing keys which can be set.

The key-value pairs returned depend on the feature set(s) supported by the unit.

14.1. Feature Sets

The following feature sets are available:

Feature set	Description
base	Base level functionality supported by all Cambrionix units
sync	Syncing capability
5V	The unit has a fixed 5V power supply
12V	The unit has a 12v power supply
temperature	The unit has a temperature sensor
PD	The unit implements the USB Power Delivery Specification
gate	The unit has a motor control product to control locking gates to secure devices connected to ports.

All products support the base feature set.

The range of possible values for a key in the base feature set can be extended if an additional feature set is also available.

The Hardware key returns a value for the type of hub.

These are the extra feature sets CambrionixApiService supports for the various types of hub:

hub type returned by “Hardware”	sync	5V	12V	Temperature	PD	gate
PP8C		yes	yes	yes		
PP8S	yes	yes	yes	yes		
PP15C		yes	yes	yes		

hub type returned by “Hardware”	sync	5V	12V	Temperature	PD	gate
PP15S	yes	yes	yes	yes		
SS15	yes	yes	yes	yes		
Series8		yes				
U8C-EXT		yes	yes	yes		
U8C		yes				
U8RA	yes	yes				
U8S-EXT	yes	yes	yes	yes		
U8S	yes	yes				
U10C		yes				
U10S	yes	yes				
U12S	yes	yes				
U16S Spade-NL	yes	yes				
PD-Sync 4	yes*1			yes	yes	
ThunderSync2-16	yes	yes		yes		
ThunderSync3-16	yes	yes		yes		
ModIT Max*2	yes	yes		yes		yes

*1 It is to be noted that while the PDSync-C4 does not implement the “sync” feature set as such, nevertheless it does have sync capabilities and these are always available. This means that there is no need to switch between charge mode and sync mode.

*2 The ModIT Max will identify itself as a ThunderSync3-16, but it has additional hardware for gate control.

14.2. Get Dictionary

Key	Feature set
Attached	5V
Compiled	base
EnabledProfiles	5V
Firmware	base
Firmware Requirements	base
FiveVoltRail_flags	5V
FiveVoltRail_Limit_Max_V	5V
FiveVoltRail_Limit_Min_V	5V
FiveVoltRail_V	5V
FiveVoltRailMax_V	5V
FiveVoltRailMin_V	5V
Gates	gate
Group	base
Hardware	base
HardwareFlags	base
HardwareInformation	base
Health	base
HostPresent	PD
InputRail_flags	PD
InputRail_Limit_Max_V	PD
InputRail_Limit_Min_V	PD
InputRail_V	PD
InputRailMax_V	PD

Key	Feature set
InputRailMin_V	PD
Key.N	5V
ModeChangeAuto	sync
nrOfPorts	base
PanelID	base
Port.N.Battery	sync
Port.N.Current_mA	base
Port.N.Description	PD
Port.N.Energy_Wh	base
Port.N.Flags	base
Port.N.FlashDrive	sync
Port.N.LocationID	sync
Port.N.Manufacturer	PD
Port.N.Mode	base
Port.N.PID	PD
Port.N.ProfileID	5V
Port.N.Profiles	5V
Port.N.SerialNumber	PD
Port.N.TimeCharged_sec	base
Port.N.TimeCharging_sec	base
Port.N.USBStrings	PD
Port.N.VID	PD
Port.N.Voltage_10mV	PD
PortInfo.N	base
PortsInfo	base

Key	Feature set
Profile.N.enabled	5V
pwm_percent	temperature
Rebooted	base
SecurityArmed	5V
Settings	base
SystemTitle	base
Temperature_C	temperature
Temperature_flags	temperature
Temperature_Limit_Max_C	temperature
TemperatureMax_C	temperature
TotalCurrent_mA	5V
TotalPower_W	5V
TwelveVoltRail_flags	12V
TwelveVoltRail_Limit_Max_V	12v
TwelveVoltRail_Limit_Min_V	12v
TwelveVoltRail_V	12V
TwelveVoltRailMax_V	12V
TwelveVoltRailMin_V	12V
Uptime_sec	base

Attached

A bit-field with one bit set for each port with a device attached, port 1 in bit 0, port 2 in bit 1 and so on.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Attached"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": attached-bit
}
```

attached-bit is an integer value.

Example, of three devices connected to ports 1,2,3

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 7
}
```

Compiled

Timestamp of firmware version.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Compiled"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "compiled-date"
}
```

compiled-date is the timestamp of when the Firmware was compiled format "MMM DD YYYY HH mm SS"

Timestamp	Description
MMM	Month first 3 letters in english
DD	Date of the month as an integer
YYYY	Year as an integer
HH	Hour of build, 0-23
mm	Minute of build, 0-59
SS	Second of build, 0-59

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "Jul 08 2015 10:43:20"
}
```

EnabledProfiles

List of global profiles currently enabled

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "EnabledProfiles"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "profiles"
}
```

profiles is list of all the charging profiles which are applied to the hub, profiles are displayed as a single number with a space between each profile.

Example

Return value where charging profiles 1,2,3 and 4 are enabled.

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "1 2 3 4"
}
```

Firmware

Firmware version string.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Firmware"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "firmware-version"
}
```

*firmware-version*Is the version number of the firmware. Format 'N.nn'

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "1.55"
}
```

Firmware Requirements

Get the types of firmware applicable to this hub, returned as an array containing information for all types of firmware that the hub accepts. For each entry, the form factor field indicates the firmware type for that part, which can be one of “un” for Universal, “pd" for PDSync, "st" for the TS3-C10 or “mc” for motor control board.

Firmware	Part Number	Product Name
Universal	PP15S	PowerPad15S
Universal	PP15C	PowerPad15C
Universal	PP8S	PowerPad8S
Universal	SS15	SuperSync15
Universal	TS3-16	ThunderSync3-16
SMART	TS3-C10	ThunderSync3-C10
Universal	U16S Spade	U16S Spade
Universal	U8S	U8S
PDSync	PDSync-C4	PDSync-C4
Universal	ModIT-Max	ModIT-Max
Motor Control	Motor control board	ModIT-Max

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "FirmwareRequirements"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

Returns an array of information about each type of firmware currently installed, with details of the firmware types required to update them.

```
{
  "result": [
    {
      "Manufacturer": "manufacturer-name",
      "Hardware": "product-name",
      "Firmware": "firmware-version",
      "Bootloader": "bootloader-version",
      "Group": "group-order",
      "FormFactor": "firmware-type",
      "HardwareID":hardware-id,
      "Group": "group-order"
      "SerialNumber": "hub-serial",
    }
  ]
}
```

Output	Description
hub-serial	This is the serial number of the hub returned from cbrx_discover
manufacturer-name	Defined name of manufacturer, Default is 'Cambrionix'
firmware-version	Version number of the firmware. Format 'N.nn'
bootloader-version	Version number of the bootloader. Format 'N.nn'
group-order	Used to order hubs which is useful when updating connected products so that down-stream products are updated and rebooted first.

Output	Description
firmware-type	Used to denote which firmware the product accepts
hardware-id	hardware ID number of front panel product
product-name	Hardware name of product

Example


```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": [
        {
            "Manufacturer": "cambrionix",
            "Hardware": "ThunderSync3-16",
            "Firmware": "1.87",
            "Bootloader": "0.21",
            "FormFactor": "un",
            "HardwareID": 50,
            "Group": "-",
            "SerialNumber": "DJ00ASBK"
        },
        {
            "Manufacturer": "cambrionix",
            "Hardware": "Motor Board",
            "Firmware": "0.08",
            "Bootloader": "0.05",
            "FormFactor": "mc",
            "HardwareID": 1,
            "Group": "+",
            "SerialNumber": "DJ00ASBK"
        }
    ]
}
```

FiveVoltRail_flags

Returns list of 5V supply rail error flags that have been detected, if any.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "FiveVoltRail_flags"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "flag"
}
```

Flag	Description
UV	Under voltage occurred
OV	Over voltage occurred
UV OV	Both under and over voltage occurred.

Example

```
{
    "jsonrpc": "2.0",
```

```
        "id": 0,
        "result": "UV OV"
    }
```

FiveVoltRail_Limit_Max_V

Upper limit of the 5V rail that will trigger the error flag.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "FiveVoltRail_Limit_Max_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": FiveVoltRail-Limit-Max
}
```

FiveVoltRail-Limit-Max is a decimal number in Volts, format n.nn

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 5.58
}
```

FiveVoltRail_Limit_Min_V

Lower limit of the 5V rail that will trigger the error flag

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "FiveVoltRail_Limit_Min_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": FiveVoltRail-Limit-Min
}
```

FiveVoltRail-Limit-Min is a decimal number in Volts, format n.nn

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 3.50
}
```

FiveVoltRail_V

Current 5V supply voltage in Volt (V)

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "FiveVoltRail_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": FiveVoltRail_V
}
```

FiveVoltRail_V is a decimal number in Volts, format n.nn

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 5.25
}
```

FiveVoltRailMax_V

Highest 5V supply voltage measured in Volt (V)

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "FiveVoltRailMax_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":FiveVoltRailMax_V
}
```

FiveVoltRailMax_V is a decimal number in Volts, format n.nn

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 5.25
}
```

FiveVoltRailMin_V

Lowest 5V supply voltage measured in Volt (V)

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "FiveVoltRailMin_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":FiveVoltRailMin_V
}
```

FiveVoltRailMin_V is a decimal number in Volts, format n.nn

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 5.20
}
```

Gates

Returns an object describing the states of all gates on the expansion product if present. Currently, this is only available on the ModIT range.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Gates"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":
  {
    "N":"state"
  }
}
```

N is the port number

state	Description
open	The gate is open
closed	The gate is closed
opening	The gate is opening

state	Description
closing	The gate is closing
stalled	The gate has stopped moving
timeout	The gate has taken too long to respond
unknown	Neither the top nor bottom end switches are engaged
open-stalled	The gate has stalled in the open position
closed-stalled	The gate has stalled in the closed position
opening-stalled	The gate has stalled whilst opening
closing-stalled	The gate has stalled whilst closing
disabled	Disabled flag has been set which prevents the gate from opening/ closing

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": {
        "1": "open",
        "2": "closed",
        "3": "closed",
        "4": "opening"
    }
}
```

Group

Group letter read from PCB jumpers.

Syntax: see Call Structure

```
{
 "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Group"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "group-order"
}
```

group-order Is used to order hubs which is useful when updating connected products so that down-stream products are updated and rebooted first., or “-” if no group jumper was fitted.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "-"
}
```

Hardware

Part number of the hub.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Hardware"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "product-name"
}
```

product-name is the hardware name of product

Example

```
{
  "result": "ThunderSync3-16"
}
```

HardwareFlags

Flags indicating whether features are present

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "HardwareFlags"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "flag"
}
```

Flag	Description
S	Sync feature set
L	5V feature set
E	12V feature set
T	Temperature feature set
P	PD feature set

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "SLET"
}
```

HardwareInformation

Information on the hub

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "HardwareInformation"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": {
    "ProductName": “product-name”,
    "ProductWebPage": "product-webpage",
    "TemperatureRangeC": {
      "Min": min-temperature,
      "Max": max-temperature
    },
    "HumidityRange": {
      "Min": min-humidity,
      "Max": max-humidity
    },
    "DimensionsMillimetres": {
```

```
      "Width": product-width,
      "Length": product-length,
      "Height": product-height
    },
    "HostPortType": "host-port",
    "HostPortBandwidth": "host-port-bandwidth",
    "HubMaxPowerOutputWatts": hub-max-power-output,
    "Ports": {
      "1": {
        "HardwareInformation": {
          "Type": "port-type",
          "Bandwidth": "port-bandwidth",
          "VoltageMax": port-max-volts,
          "MilliampsMax": port-max-current
        }
      },
      // etc all ports.
    }
    }
  }
}
```

Variable	Description
product-name	Product name of the hub
product-webpage	The webpage for the hub
min-temperature	The minimum recommended temperature for the hubs enviroment (°C)

Variable	Description
max-temperature	The maximum recommended temperature for the hubs enviroment
min-humidity	The minimum ambient humidty % for the hubs enviroment
max-humidity	The maximum ambient humidity % for the hubs enviroment
product-width	The width of the hub (mm)
product-length	The length of the hub (mm)
product-height	The height of the hub (mm)
host-port	The USB type of the host port
host-port-bandwidth	The maximum bandwidth for the host port (Gbps)
hub-max-power-output	The maximum power output for the whole hub
port-type	The USB type of the downstream ports
port-bandwidth	The maximum bandwidth for the downstream ports (Gbps)
port-max-volts	The maximum Voltage output for the downstream ports (V)
port-max-current	The maximum Current output fo the downstream ports (mA)

Example


```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "ProductName": "ThunderSync3-C10",
    "ProductWebPage": "https://www.cambrionix.com/products/thundersync3-c10",
    "TemperatureRangeC": {
      "Min": 10,
      "Max": 35
    },
    "HumidityRange": {
      "Min": 5,
      "Max": 95
```

```
      },
      "DimensionsMillimetres": {
        "Width": 193,
        "Length": 136,
        "Height": 34
      },
      "HostPortType": "Thunderbolt 3",
      "HostPortBandwidth": "40Gbps",
      "HubMaxPowerOutputWatts": 150,
      "Ports": {
        "1": {
          "HardwareInformation": {
            "Type": "USB Type-C",
            "Bandwidth": "5Gbps",
            "VoltageMax": 5.2,
            "MilliampsMax": 3000
          }
        },
        // etc all ports.
      }
    }
  }
```

Health

All available keys that are not port specific and change dynamically, as a dictionary.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Health"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": {
    "Uptime_sec": Uptime_sec,
    "FiveVoltRail_V": FiveVoltRail_V,
    "FiveVoltRailMin_V":FiveVoltRailMin_V,
    "FiveVoltRailMax_V":FiveVoltRailMax_V,
    "FiveVoltRail_flags": "FiveVoltRail_flags",
    "TwelveVoltRail_V":TwelveVoltRail_V,
    "TwelveVoltRailMin_V":TwelveVoltRailMin_V,
    "TwelveVoltRailMax_V":TwelveVoltRailMax_V,
    "InputRail_V":InputRail_V
    "InputRailMin_V":InputRailMin_V,
    "InputRailMax_V":InputRailMax_V,
```

```
    "TwelveVoltRail_flags": "TwelveVoltRail_flags",
    "InputRail_flags": "InputRail_flags",
    "Temperature_C":Temperature_C,
    "TemperatureMax_C":TemperatureMax_C,
    "Temperature_flags": "Temperature_flags",
    "Rebooted":Rebooted
  }
}
```

Output	Description
Uptime_sec	see Uptime_sec
FiveVoltRail_V	see FiveVoltRail_V
FiveVoltRailMin_V	see FiveVoltRailMin_V
FiveVoltRailMax_V	see FiveVoltRailMax_V
FiveVoltRail_flags	see FiveVoltRail_flags
TwelveVoltRail_V	see TwelveVoltRail_V
TwelveVoltRailMin_V	see TwelveVoltRailMin_V
TwelveVoltRailMax_V	see TwelveVoltRailMax_V
InputRail_V	see InputRail_V
InputRailMin_V	see InputRail_Limit_Min_V
InputRailMax_V	see InputRailMax_V
TwelveVoltRail_flags	see TwelveVoltRail_flags
InputRail_flags	see InputRail_flags
Temperature_C	see Temperature_C
TemperatureMax_C	see TemperatureMax_C
Temperature_flags	see Temperature_flags
Rebooted	see Rebooted

Example


```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": {
        "Uptime_sec": 528422,
        "FiveVoltRail_V": 5.23,
        "FiveVoltRailMin_V": 5.14,
        "FiveVoltRailMax_V": 5.25,
        "FiveVoltRail_flags": "",
        "TwelveVoltRail_V": 12.12,
        "TwelveVoltRailMin_V": 11.99,
        "TwelveVoltRailMax_V": 12.2,
        "InputRail_V": 12.12,
        "InputRailMin_V": 11.99,
        "InputRailMax_V": 12.2,
        "TwelveVoltRail_flags": "",
        "InputRail_flags": "",
        "Temperature_C": 37.3,
        "TemperatureMax_C": 41.1,
        "Temperature_flags": "",
        "Rebooted": true
    }
}
```

HostPresent

The hub monitors the host USB socket for an attached host computer.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "HostPresent"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":host
}
```

host	description
true	Host is detected
false	No host is detected

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": true
}
```

InputRail_flags

List of input rail error flags if any are set.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "InputRail_flags"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":flag
}
```

Flag	Description
UV	under voltage occurred
OV	over voltage occurred
no flags	voltage is acceptable

Example

```
{
    "jsonrpc": "2.0",
```

```
        "id": 0,
        "result": "OV UV"
    }
```

InputRail_Limit_Max_V

Upper limit of the input rail that will trigger the error flag

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "InputRail_Limit_Max_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": InputRail-Limit-Max
}
```

InputRail-Limit-Max is a decimal number in Volts, format n.nn

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 24.7
}
```

InputRail_Limit_Min_V

Lower limit of the input rail that will trigger the error flag.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "InputRail_Limit_Min_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": InputRail-Limit-Min
}
```

InputRail-Limit-Min is a decimal number in Volts, format n.nn

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 9.59
}
```

InputRail_V

Current input rail supply in Volts (V).

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "InputRail_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":InputRail_V
}
```

InputRail_V is a decimal number in Volts, format n.nn

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 24.03
}
```

InputRailMax_V

Highest input voltage measured in Volts (V).

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "InputRailMax_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":InputRailMax_V
}
```

InputRailMax_V is a decimal number in Volts, format n.nn

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 24.14
}
```

InputRailMin_V

Lowest input voltage measured in Volts (V).

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "InputRailMin_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":InputRailMin_V
}
```

is a decimal number in Volts, format n.nn

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 23.82
}
```

Key.N

Get information if a button has been pressed, double-clicks cannot be detected.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Key.N"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": button
}
```

button	Description
0	button n has not been pressed since the last time this entry was read
1	button n has been pressed since the last time this entry was read

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 0
}
```

ModeChangeAuto

Mode change from Charge to Sync is automatic.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "ModeChangeAuto"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":auto-change
}
```

auto-change	Description
true	Mode change from Charge to Sync is automatic.
false	Mode change from Charge to Sync is manual.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": true
}
```

nrOfPorts

Number of USB ports on the hub.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "nrOfPorts"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":portsnumber
}
```

portsnumber is an integer of the amount of ports available

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 8
}
```

Note

- On the PDSync-C4 you will have an additional port 0, which is the information on the host port.

PanelID

PanelID number of front panel board, if fitted.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "PanelID"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "Panel-ID"
}
```

Panel-ID is the ID number of front panel product, if not fitted will return Absent/None

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "Absent"
}
```

Port.N.Battery

If possible, retrieve the current battery level of the connected device. See notes about battery information collection. Depending on the device type (Android™, iOS etc.) and the host OS, different data may be returned.

For Apple, "Apple Mobile Device Support" must be installed (included with iTunes)

For Android, adb must be installed and running.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
   connection-handle
   "Port.N.Battery"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": {
    "CurrentLevel":battery-current-level,
    "CurrentTime":current-hub-time,
    "StartingLevel":charge-start-level,
    "StartingTime":charge-start-time,
  }
}
```

Output	Description
battery-current-level	Current battery level of device displayed as a percentage
current-hub-time	The hub time, shown as an integer in ms
charge-start-level	Battery percentage level when device connected
charge-start-time	The hub time charging started, shown as an integer in ms

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": {
    "CurrentLevel": 78,
    "CurrentTime": 15234254346,
    "StartLevel": 23,
    "StartTime": 15124151512,
    }
}
```

Port.N.Current_mA

Current being delivered to the USB device connected to this USB port in milli-Amperes (mA).

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.Current_mA"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": Current_mA
}
```

Current_mA is the current being delivered to the device, in mA (milliamperes)

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 0
}
```

Port.N.Description

Description as reported by the USB device attached to this USB port if it could be detected. Empty string is returned if it could not be detected.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.Description"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "Description"
}
```

Description is the name of the hardware.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "SuperPhone6"
}
```

Port.N.Energy_Wh

Energy the USB device on this USB port has consumed.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.Energy_Wh"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":energy-Wh
}
```

energy-Wh is the power in Watt-hours (calculated every second), format of n.n

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 0.0
}
```

Port.N.Flags

Get a list of all flags on a specific port

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.Flags"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "flag"
}
```

List of case-sensitive flag characters, separated by spaces. O, S, B, I, P, C, F are mutually exclusive. A, D are mutually exclusive.	
Flag	**Description**
O	Port is in OFF mode
S	Port is in SYNC mode
B	Port is in Biased mode
I	Port is in charge mode, and is IDLE
P	Port is in charge mode, and is PROFILING

C	Port is in charge mode, and is CHARGING
F	Port is in charge mode, and is has FINISHED charging
A	Device is ATTACHED to this port
D	No device is attached to this port. Port is DETACHED
T	Device has been stolen from port: THEFT
E	ERRORs are present. See health command
R	System has REBOOTED. See crf command
r	Vbus is being reset during mode change

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "R D S"
}
```

Port.N.FlashDrive

If detected, returns the mount point of a USB flash drive. For Windows this will be a drive letter, otherwise it will be a volume mount point.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.FlashDrive"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "Path:": "location",
  "Capacity":total-memory,
  "Available":available-memory
}
```

Output	Description
location	Location the disk is in the host system
total-memory	Total memory on disk
available-memory	Memory on disk not being used

Examples:

Windows:

```
{
    "json": "2.0",
    "id": 0,
    "Path:": "H:",
    "Capacity": 123123123,
    "Available": 123123
}
```

macOS®:

```
{
    "json": "2.0,
    "id": 0,
    "Path:": "/Volumes/SanDisk1",
    "Capacity": 123123123,
    "Available": 123123
}
```

Linux:

```
{
    "json": "2.0",
    "id": 0,
    "Path:": "/media/bob/SanDisk1",
    "Capacity": 123123123,
    "Available": 123123
}
```

If there is no flash drive, the return value will simply be false.

This same information will also be provided in PortsInfo, PortInfo.N or cbrx_discover('all') in a "FlashDrive" field where applicable and present. If not applicable or preset, this field will be absent.

Port.N.LocationID

Return the location ID for a specific port. This does not require a device to be attached and so may be used to uniquely identify a USB slot. Location IDs indicate the bus number that a USB host controller is on in the first byte, then the port numbers down the tree for child devices.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.LocationID"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":location-id
}
```

location-id Is the Location IDs as an Integer

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 563154944
}
```

Note

- for USB3 hubs, this location ID will be different when a USB3 device is plugged in compared to a USB2 device

Port.N.Manufacturer

Manufacturer as reported by the USB device attached to this USB port, if it could be detected. Empty string is returned if it could not be detected.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.Manufacturer"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "device-manufacturer"
}
```

device-manufacturer is the name of the device manufacturer

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "SuperPhone Makers Inc."
}
```

Port.N.Mode

Current port mode.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle
    "Port.N.Mode"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "mode"
}
```

For Standard USB hubs, the mode can be any of:

mode character	Description
s	Sync mode
c	Charge mode
b	Biased mode
o	Off

For Type-C hubs, the mode can be:

mode character	Description
c	On
o	Off

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "s"
}
```

Port.N.PID

Product ID of the USB device attached to this USB port, if it could be detected.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.PID"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":product-id
}
```

product-id is the product ID number or PID. Displayed as an Integer. 0 (zero) is returned if it could not be detected.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 0
}
```

Port.N.ProfileID

Profile ID number.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.PID"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":profile-ID
}
```

profile-ID is the profile number, or 0 if not charging.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 0
}
```

Port.N.Profiles

List of enabled profiles for this port.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.Profiles"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "profiles"
}
```

profiles is list of all the charging profiles which are applied to the port, profiles are displayed as a single number with a space between each profile.

Example

charging profiles 1,2,3 and 4 are enabled.

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "1 2 3 4"
}
```

Port.N.SerialNumber

Serial number as reported by the USB device attached to this USB port, if it could be detected. Empty string is returned if it could not be detected.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.SerialNumber"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": "usb-serial"
}
```

usb-serial is the USB serial number

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "1127dfa9037s1a8cb1"
}
```

Port.N.TimeCharged_sec

Time in seconds since this USB port detected the device has completed charging.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.TimeCharged_sec"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":charge-complete
}
```

charge-complete is the time in seconds that has passed since the device completed charging, -1 will be returned if this port has not detected completed charging.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 0
}
```

Port.N.TimeCharging_sec

Time in seconds since this USB port started charging an attached device. 0 will be returned if the USB port has not started charging an attached device.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.TimeCharging_sec"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":charging-time
}
```

charging-time is the time passed in seconds that a port has been charging a device.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 0
}
```

Port.N.USBStrings

A dictionary containing the values for “Manufacturer”, “Description” and “SerialNumber” for this USB port.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.USBStrings"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "SerialNumber": "usb-serial",
  "Description": "description",
  "Manufacturer": "device-manufacturer"
}
```

Output	Description
usb-serial	USB serial number
description	Name of the hardware
device-manufacturer	The name of the device manufacturer

Example

```
{
    "json": 2.0,
    "id": 0,
    "SerialNumber": "23213dfe12e2412",
    "Description": "SuperPhone6",
    "Manufacturer": "SuperPhone Makers Inc."
}
```

Port.N.VID

Vendor ID of the USB device attached to this USB port.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.VID"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": Vendor-ID
}
```

Vendor-ID is the vendor ID, VID. Displayed as an Integer. 0 (zero) is returned if it could not be detected.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 0
}
```

Port.N.Voltage_10mV

Voltage being supplied to the port in 10mV.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Port.N.Voltage_10mV"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":voltage-10mV
}
```

voltage-10mV is the voltage supplied to the port as an integer in increments of 10mV.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 520
}
```

PortInfo.N

Get all port information for specified port. All available keys and values for this port as a dictionary.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "PortInfo.N"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":{
    "Port":port-number,
    "Current_mA": Current_mA,
    "LocationID":location-id,
    "Flags": flags,
    "USBVersion":usb-version,
    "VID":Vendor-ID,
    "PID":product-ID,
    "Manufacturer": "device-manufacturer",
    "Description": "description",
    "SerialNumber": "usb-serial",
```

```
    "USBTree": {
      "LocationID":location-id,
      "USBVersion":usb-version,
      "USBPower": {
        "State": "power-state",
        "Description": "power-description"
      },
      "USBSpeed": {
        "Speed": "USB-speed",
        "Description": "USB-description"
        "Capability": {
          "Speed": "capable-speed",
          "Description": "capable-description"
        }
      },
      "Endpoints": {
        "Active":active-endpoints,
        "Memory":endpoint-memory
      }
    }
  }
}
```

Output	Description
port-number	The number of the port on the hub
Current_mA	Current being delivered to the device, in mA (milliamperes)
location-id	The Location IDs as an Integer

Output	Description
flags	Flags on the port, see Port.N.Flags
usb-version	The USB version number of the connection to the hub. Format 'N.nn'
Vendor-ID	Device Vedor ID number or VID. Displayed as an Integer
product-ID	Product ID number or PID. Displayed as an Integer
device-manufacturer	The name of the device manufacturer
description	Name of the hardware
usb-serial	USB serial number
power-state	USB Power States code
power-description	USB power turned on/off
USB-speed	Maximum speed USB connection capable of
USB-description	Name of USB connection i.e. SuperSpeed USB 5Gbps
capable-speed	Maximum data speed device capable of
capable-description	Name of maximum data speed device capable of
active-endpoints	How many endpoints the device is using
endpoint-memory	Amount of memory being used by endpoints

Example


```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": {
        "Port": 1,
        "Current_mA": 1084,
        "LocationID": 563154944,
        "Flags": "R A S",
```

```
            "USBVersion": 2.1,
            "VID": 1256,
            "PID": 26720,
            "Manufacturer": "SAMSUNG",
            "Description": "SAMSUNG_Android",
            "SerialNumber": "RFCN20Q8LJM",
            "USBTree": {
                "LocationID": 563154944,
                "USBVersion": 2.1,
                "USBPower": {
                    "State": "D0",
                    "Description": "On"
                },
                "USBSpeed": {
                    "Speed": "480Mbps",
                    "Description": "High",
                    "Capability": {
                        "Speed": "10Gbps",
                        "Description": "SuperSpeed USB 10Gbps"
                    }
                },
                "Endpoints": {
                    "Active": 9,
                    "Memory": 36864
                }
            }
        }
    }
```

PortsInfo

Get all port information for all ports. All available information for all ports as a dictionary of dictionaries. Most of these values can be queried individually

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "PortsInfo"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns

```
{
  "Port.1": {
    "Port":port-number,
    "Current_mA": Current_mA,
    "Flags": flags,
    "ProfileID":profile-ID,
    "TimeCharging_sec":charging-time,
    "TimeCharged_sec":charge-complete,
    "Energy_Wh":energy-Wh,
    "VID":Vendor-ID,
    "PID":product-ID,
    "Manufacturer": "device-manufacturer",
```

```
    "Description": "description",
    "SerialNumber": "usb-serial",
    "PhoneSerialNumber": "phone-serial",
    "PhoneIdentity": "phone-name",
    "IMEI": "IMEI-number",
    "MacAddress": "MacAdress",
    "PhoneSoftwareVersion": "phone-OS-version"
    "USBTree": {
      "USB2": {
        "LocationID":location-id,
        "VID":Vendor-ID,
        "PID":product-ID,
        "Manufacturer": "device-manufacturer",
        "Description": "description",
        "SerialNumber": "usb-serial",
        "USBVersion":usb-version,
        "Battery": {
          "DataSource": "battery-data-source",
          "TrustLevel": "trust-level",
          "PairingSupported":support-pairing,
          "HealthPercent":battery-health,
          "CurrentLevel":battery-current-level,
          "CurrentTime":current-hub-time,
          "StartingLevel":charge-start-level,
          "StartingTime":charge-start-time,
          "CapacityNew":new-battery-capacity,
          "Capacity":current-battery-capacity,
          "ChargingStatus": "charge-status",
```

```
        },
        "PhoneSerialNumber": "phone-serial",
        "PhoneIdentity": "phone-name",
        "IMEI": "IMEI-number",
        "MacAddress": "MacAdress",
        "PhoneSoftwareVersion": "phone-OS-version"
      }
    },
    "Battery": {
      "DataSource": "battery-data-source",
      "TrustLevel": "trust-level",
      "PairingSupported":support-pairing,
      "CurrentLevel":battery-current-level,
      "CurrentTime":current-hub-time,
      "StartingLevel":charge-start-level,
      "StartingTime":charge-start-time,
      "CapacityNew":new-battery-capacity,
      "Capacity":current-battery-capacity,
      "ChargingStatus": "charge-status",
      "HealthPercent":battery-health
    }
  }
}
```

Output	Description
port-number	The number of the port on the hub
Current_mA	Current being delivered to the mobile device, in mA (milliamperes)
flags	Flags on the port, see Port.N.Flags

Output	Description
profile-ID	The profile number, or 0 if not charging.
charging-time	Time passed in seconds that a port has been charging a device.
charge-complete	Time in seconds since the device completed charging
energy-Wh	Power in Watt-hours (calculated every second), format of n.n
Vendor-ID	Device Vedor ID number or VID. Displayed as an Integer
product-ID	Product ID number or PID. Displayed as an Integer
device-manufacturer	The name of the device manufacturer
description	Name of the hardware
usb-serial	USB serial number
phone-serial	Phone serial number
phone-name	Name of the phone
IMEI-number	The IMEI number of the phone
MacAdress	A unique address assigned to the mobile device. It is a 48 bit value, consisting of twelve hexadecimal characters
phone-OS-version	Version number of the OS on the phone
location-id	The Location IDs as an Integer
usb-version	The USB version number of the connection to the hub. Format 'N.nn'
battery-data-source	The source of the device battery information
trust-level	Whether the device is trusted/ paired
support-pairing	Whather the device supports being trusted/ paired

Output	Description
battery-health	Battery health shown as a percentage
battery-current-level	Current battery level of device displayed as a percentage
current-hub-time	The hub time, shown as an integer in ms
charge-start-level	Battery percentage level when device connected
charge-start-time	The hub time charging started, shown as an integer in ms
new-battery-capacity	The battery capacity of device from new
current-battery-capacity	The battery capacity of the device now
charge-status	Charging status of the battery i.e. full
support-pairing	Whather the device supports being trusted/ paired
battery-health	Battery health shown as a percentage

Example

Trimmed example of information returned.


```
{
    "json": "2.0",
    "id": 0,
    "Port.1": {
        "Port": 1,
        "Current_mA": 126,
        "Flags": "R A S",
        "ProfileID": 0,
        "TimeCharging_sec": 0,
        "TimeCharged_sec": 0,
```

```
            "Energy_Wh": 0.0,
            "VID": 1452,
            "PID": 4776,
            "Manufacturer": "SuperPhone Makers Inc.",
            "Description": "SuperPhone",
            "SerialNumber": "012a37d1fa07617ad7ef0430ba49f479ab9fb6b8",
            "PhoneSerialNumber": "ZCZCZCZCZCZC",
            "PhoneIdentity": "My Old SuperPhone",
            "IMEI": "354430099009999",
            "MacAddress": "aa:bb:cc:ff:ee:ff",
            "PhoneSoftwareVersion": "12.4.8",
            "USBTree": {
                "USB2": {
                    "LocationID": 589570048,
                    "VID": 1452,
                    "PID": 4776,
                    "Manufacturer": "SuperPhone Makers Inc.",
                    "Description": "SuperPhone",
                    "SerialNumber": "012a37d1fa07617ad7ef0430ba49f479ab9fb6b8",
                    "USBVersion": 2.0,
                    "Battery": {
                        "DataSource": "imobiledevice",
                        "TrustLevel": "paired",
                        "PairingSupported": true,
                        "HealthPercent": 95,
                        "CurrentLevel": 100,
                        "CurrentTime": 1613056296,
                        "StartingLevel": 100,
                        "StartingTime": 1613056293,
                        "CapacityNew": 1751,
                        "Capacity": 1678,
                        "ChargingStatus": "full"
                    },
                    "PhoneSerialNumber": "ZCZCZCZCZCZC",
                    "PhoneIdentity": "My Old SuperPhone",
                    "IMEI": "354430099009999",
                    "MacAddress": "aa:bb:cc:ff:ee:ff",
                    "PhoneSoftwareVersion": "12.4.8"
                }
            },
            "Battery": {
                "DataSource": "imobiledevice",
                "TrustLevel": "paired",
                "PairingSupported": true,
                "HealthPercent": 95,
                "CurrentLevel": 100,
                "CurrentTime": 1613056296,
                "StartingLevel": 100,
                "StartingTime": 1613056293,
                "CapacityNew": 1751,
                "Capacity": 1678,
                "ChargingStatus": "full"
            }
        },
        "Port.2": {
```

```
            "Port": 2,
            "Current_mA": 0,
            "Flags": "R D S",
            "ProfileID": 0,
            "TimeCharging_sec": 0,
            "TimeCharged_sec": 0,
            "Energy_Wh": 0.0,
            "VID": 0,
            "PID": 0,
            "Manufacturer": "",
            "Description": "",
            "SerialNumber": ""
        },
        "Port.3": ...
    }
```

Profile.N.enabled

Get information if a specific profile is enabled. See specific product user manuals on profiles available on your hub.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Profile.N.enabled"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":profile-enabled
}
```

profile-enabled	Description
true	Specific profile is enabled
false	Specific profile is not enabled

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": false
}
```

pwm_percent

Fan speed.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "pwm_percent"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":fan-percent
}
```

fan-percent is the fan speed as a percentage, displayed as a number 0-100

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 100
}
```

Rebooted

A flag indicating if the system has been rebooted since power up.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Rebooted"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":rebooted
}
```

Rebooted	Description
true	system has been rebooted
false	no reboot has occurred.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": true
}
```

SecurityArmed

Is security armed?

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "SecurityArmed"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":security-armed
}
```

security-armed	Description
true	Security has been armed
false	Security has not been armed

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": false
}
```

Settings

Obtain current hub Internal hub settings.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Settings",
    true
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result": {
    "company_name": "manufacturer-name",
    "product_name": "product-name",
    "local_name": "local-name",
    "attach_threshold": "attach-threshold",
    "default_profile": [default-profile],
    "remap_ports": [port-order],
    "ports_on": [ports-on],
    "sync_chrg": [sync-charge],
    "alt_sync_chrg": [alt-sync-charge],
    "misc_flags":Internal hub-flags,
```

```
    "display_mode": "display-mode",
    "charged_threshold": "charged-threshold",
    "temperature_max": "shutdown-temperature",
    "stagger": "stagger"
  }
}
```

Variables	Description
manufacturer-name	Defined name of manufacturer, Default is 'Cambrionix'
product-name	Hardware name of product
local-name	Local name set by the user, "-" if not set
attach-threshold	Current drawn in mA that the hub detects a device is connected, "d" means factory default is set
default-profile	Default profile for each port, comma seperated list
port-order	Order the ports are by port number, comma seperated list
ports-on	Whether each port is default on, 0 is default off 1 is default on, comma seperated list
sync-charge	Whether CDP* on each port on, 0 is off, 1 is on, comma seperated list
alt-sync-charge	Whether alternative CDP* on each port on, 0 is off, 1 is on, comma seperated list
Internal hub-flags	If any Internal hub Misc flags are active
display-mode	Change the display mode for logs, "d" means factory default is set
charged-threshold	Current drawn in mA that the hub detects a device is fully charged, "d" means factory default is set

Variables	Description
shutdown-temperature	Temperature that will shutdown the hub if reached in Celsius, "d" means factory default is set
stagger	A delay between ports turning on in ms, "d" means factory default is set

*Charging Downstream Port (CDP) Being enabled means that a port is capable of transferring data and charging the device at the same time with a higher current than just data syncing alone. With CDP enabled the hub can supply up to 1.5 A

If you disable CDP you will receive the notification “This Hub has the Charge Downstream Port UCS mode disabled. This could limit the maximum current seen on some ports.” This notification is there to ensure you haven’t turned this off by accident and can still have the highest charge available.

Example


```
{
    "jsonrpc": "2.0",
    "id": 5,
    "result": {
        "company_name": "cambrionix",
        "product_name": "SuperSync15",
        "local_name": "-",
        "attach_threshold": "d",
        "default_profile": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        "remap_ports": [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],
        "ports_on": [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
        "sync_chrg": [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
        "alt_sync_chrg": [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
        "misc_flags": 0,
        "display_mode": "d",
        "charged_threshold": "d",
        "temperature_max": "d",
        "stagger": "d"
    }
}
```

SystemTitle

The system identification text.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "SystemTitle"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":system-title
}
```

system-title is the full descriptive name of the hub

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "cambrionix U8S-EXT 8 Port USB Charge+Sync"
}
```

Temperature_C

Present PCB temperature in degrees Celsius.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Temperature_C"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":temperature
}
```

temperature is a measured temperatures as a decimal. ≤0 °C will return 0. Measured temperatures ≥100 °C will return 100.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 37.7
}
```

Temperature_flags

Temperature error flags:

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Temperature_flags"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":temperature-flags
}
```

temperature-flags	Description
OT	over temperature event has occurred.
empty	temperature is acceptable

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": "OT"
}
```

Temperature_Limit_Max_C

Upper limit of the acceptable temperature range that will trigger the error flag.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Temperature_Limit_Max_C"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":temperature-limit-max
}
```

temperature-limit-max the upper limit in Celsius displayed as a decimal.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 65.0
}
```

TemperatureMax_C

Highest PCB temperature recorded in degrees Celsius.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "TemperatureMax_C"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":temperature-max
}
```

temperature is a measured temperatures displayed as a decimal. ≤0 °C will return 0. Measured temperatures ≥100 °C will return 100.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 39.9
}
```

TotalCurrent_mA

Total current in mA for all USB ports.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "TotalCurrent_mA"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":total-current
}
```

total-current is the total current across all ports in mA displayed as a decimal

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 0
}
```

TotalPower_W

Total power being consumed on all USB ports in Watts (W).

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "TotalPower_W"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns

```
{
"result": "total-power"
}
```

total-power is the total power across all ports in watts displayed as an decimal.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 3.4
}
```

TwelveVoltRail_flags

List of 12V supply rail error flags.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "TwelveVoltRail_flags"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":twelve-volt-flags
}
```

twelve-volt-flags	Description
“UV”	under voltage occurred
“OV”	over voltage occurred
“OV UV”	both over and under voltage occurred
" "	voltage is acceptable

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": " "
}
```

TwelveVoltRail_Limit_Max_V

Upper limit of the 12V rail that will trigger the error flag.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "TwelveVoltRail_Limit_Max_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":twelve-volt-limit-max
}
```

twelve-volt-limit-max is the maximum amount of volts before error is flagged in Volts displayed as a decimal.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 14.5
}
```

TwelveVoltRail_Limit_Min_V

Lower limit of the 12V rail that will trigger the error flag.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "TwelveVoltRail_Limit_Min_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":twelve-volt-limit-min
}
```

twelve-volt-limit-min is the minimum amount of volts before error is flagged in Volts displayed as a decimal.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 9.59
}
```

TwelveVoltRail_V

Current 12V supply voltage in Volts (V).

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "TwelveVoltRail_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":twelve-volt
}
```

twelve-volt is the amount of volts being supplied in Volts displayed as a decimal.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 12.43
}
```

TwelveVoltRailMax_V

Highest 12V supply voltage measured.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "TwelveVoltRailMax_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":twelve-volt-max
}
```

twelve-volt-max is the highest amount of volts seen in Volts displayed as a decimal.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 12.52
}
```

TwelveVoltRailMin_V

Lowest 12V supply voltage measured in Volts (V).

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "TwelveVoltRailMin_V"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":twelve-volt-min
}
```

twelve-volt-min is the smallest amount of volts seen in Volts displayed as a decimal.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 12.31
}
```

Uptime_sec

Time in seconds the hub has been running since the last reset.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Uptime_sec"
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "result":uptime
}
```

uptime is the amount of time in seconds of continuos running, diplayed as an integer.

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 151304
}
```

14.3. Set Dictionary

Key	Feature set
Beep	5V
ClearErrorFlags	base
ClearLCD	5V
ClearRebootFlag	base
FiveVoltRail.OverVoltage	5V
FiveVoltRail.UnderVoltage	5V
InputRail.OverVoltage	PD
InputRail.UnderVoltage	PD
LCDText.ROW.COL	5V
Mode	base
Port.N.gate	gate
Port.N.led1	base
Port.N.led2	base
Port.N.led3	base
Port.N.leds	base
Port.N.mode	base
Port.N.profiles	sync
Port.N.RGB	gate
ProfileEnable.n	5V
Reboot	base
RemoteControl	base
RGBControl	gate
SecurityArmed	5V

Key	Feature set
Settings	base
Temperature.OverTemperature	temperature
TwelveVoltRail.OverVoltage	12V
TwelveVoltRail.UnderVoltage	12V

Beep

Beep for the number of milliseconds passed in.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "beep",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
time period	An integer for the amount of time required for the beep in ms

Returns:

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

Errors

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

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "beep",
        250
    ]
}
```

ClearErrorFlags

Clear all error flags

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "ClearErrorFlags",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Clear the error flags

Returns:

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

Errors

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

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "ClearErrorFlags",
        true
    ]
}
```

ClearLCD

Clear the LCD.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "ClearLCD",
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer

Returns:

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

Errors

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

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
```

```
        "params": [
            7654,
            "ClearLCD",
        ]
    }
```

ClearRebootFlag

Clear the reboot flag.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "ClearRebootFlag",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Clear the Rebooted flag

Returns:

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

Errors

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

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "ClearRebootFlag",
        true
    ]
}
```

FiveVoltRail.OverVoltage

Force the behaviour of a 5V over voltage condition.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "FiveVoltRail.OverVoltage",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Set 5V over voltage flag

Returns:

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

Errors

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

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        ""FiveVoltRail.OverVoltage"",
        true
    ]
}
```

FiveVoltRail.UnderVoltage

Force the behaviour of a 5V under voltage condition.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "FiveVoltRail.UnderVoltage",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Set 5V under voltage flag

Returns:

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

Errors

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

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "FiveVoltRail.UnderVoltage",
        true
    ]
}
```

InputRail.OverVoltage

Force the behaviour of an input rail over voltage condition.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "InputRail.OverVoltage",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Set voltage input over voltage flag

Returns:

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

Errors

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

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "InputRail.OverVoltage",
        true
    ]
}
```

InputRail.UnderVoltage

Force the behaviour of an input rail under voltage condition.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "InputRail.UnderVoltage",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Set voltage input under voltage flag

Returns:

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

Errors

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

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "LCDText.4.5",
        "hello"
    ]
}
```

LCDText.ROW.COL

Write the string on the LCD at (row, column). Row and column are zero based.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "LCDText.ROW.COL",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
ROW	LCD Row you wish to start writing
COL	LCD Collum you wish to start writing
Value	The value you wish to set for the key

Value	Description
String	A Text string you wish to display on the LCD

Returns:

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

Errors

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

Example

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "InputRail.UnderVoltage",
        true
    ]
}
```

Mode

Set same mode to all USB Ports. Please see product user manuals for details on the modes supported by each hub.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "mode",
    "Value"
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
c	Charge mode
s	Sync and Charge mode
b	biased mode
o	off

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Mode",
        "s"
    ]
}
```

Port.N.gate

Open or close specified gate. You should monitor the state of the required gate via cbrx_connection_get(handle, “Gates”) to ensure it completes.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Port.N.gate",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
N	Port number
Value	The value you wish to set for the key

Value	Description
open	Open gate
close	Close gate
stop	Stop current gate action

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.gate",
        "open"
    ]
}
```

Port.N.led1

Set the status of the first LED

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Port.N.led1",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
N	Port number
Value	The value you wish to set for the key

Value	Description
Flash pattern	0-255 with the LEDs flashing according to the bit pattern represented by the value.

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.led1",
        170
    ]
}
```

Port.N.led2

Set the status of the second LED

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Port.N.led2",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
N	Port number
Value	The value you wish to set for the key

Value	Description
Flash pattern	0-255 with the LEDs flashing according to the bit pattern represented by the value.

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.led2",
        170
    ]
}
```

Port.N.led3

Set the status of the third LED

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Port.N.led3",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
N	Port number
Value	The value you wish to set for the key

Value	Description
Flash pattern	0-255 with the LEDs flashing according to the bit pattern represented by the value.

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.led3",
        170
    ]
}
```

Port.N.leds

Set the status of all three LEDs

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Port.N.leds",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
N	Port Number
Value	The value you wish to set for the key

Value

A 24 bit numeric value consisting of the individual LED settings as 8 bit values shifted and OR'ed together. i.e. led1 | (led2 << 8) | (led3 << 16), so with led1 and led2 as zero, and led3 being 0b10101010 (decimal 170), the result should be 11,141,120 decimal.

On a ThunderSync3, 255 is Green, 65,280 is red, 16,711,680 is Yellow.

On a ModIT, Blue is used instead of Yellow, but you can of course mix colours into any RGB mix.

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.leds",
        11193404
    ]
}
```

Port.N.mode

Set mode of a single USB port.Sync mode can only be set on device that implement the sync feature set. Biased mode can only be set on devices that implement the 5V feature set.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Port.N.mode",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
N	Port number
Value	The value you wish to set for the key

Value	Description
c	Charge mode
s	Sync and Charge mode
b	Biased mode
o	Off

Returns:

```
{
```

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.Mode",
        "c"
    ]
}
```

Port.N.profiles

Set the list of enabled profiles.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Port.N.profiles",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
N	Port number
Value	The value you wish to set for the key

Value	Description
profile	A comma separated list of profiles to enable, see product user manuals for details on profiles applicable to your hub.

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.profiles",
        "1,2,3"
    ]
}
```

Port.N.RGB

Set RGB colour of ModIT LEDs.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Port.N.RGB",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
N	Port number
Value	The value you wish to set for the key

Value

Colour value can either be an integer (where you must supply full RGBA), or a string. For a string, you can specify it as RGB, RGBA, RRGGBB or RRGGBBAA. Much like you can with an HTML colour. For example, use “FF0000” or “F00” for red, “FFFFFF” for white and so on. Optionally supply the alpha (intensity) digits, so “FFFFFF80” for half bright white.

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.RGB",
        "ff08"
    ]
}
```

ProfileEnable.n

Enable or disable the global profile n

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "ProfileEnable.n"
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
n	Profile number

Returns:

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

Errors

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

Example

```
{
    "id": 0,
```

```
        "jsonrpc": "2.0",
        "method": "cbrx_connection_set",
        "params": [
            7654,
            "ProfileEnable.n"
        ]
    }
```

Reboot

Reboot the hub now. The API will attempt to re-establish connection automatically, but you should not expected to receive updated results for several seconds.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Reboot",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Reboot the hub.

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Reboot",
        "true"
    ]
}
```

RemoteControl

Enable / disable controlling of the unit controls. This will allow the LEDs or LCD to be updated or panel button pushes to be detected.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "RemoteControl",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Enable manual remote control mode
false	Disable manual remote control mode
"auto"	Enable auto control mode via the API

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "RemoteControl",
        "true"
    ]
}
```

RGBControl

Enable / disable ModIT RGB LED control for ports. This does not require RemoteControl to be enabled.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "RGBControl",
    {
      "port": N,
      "enable":
      value
    }
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
N	port number
value	The value you wish to set for the key

Value	Description
true	Enable control of the RGB LED's
false	Disbale control of the RGB LED's

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "RGBControl",
        {
            "port": 8,
            "enable": true
        }
    ]
}
```

Multiple ports

If you wish to set control on a range of ports then the params would change. You would need to enter two values the 'start' value of the port to start with and the 'end' value of the port to finish with.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "RGBControl",
```

```
    {
      "start": N,
      "end": N,
      "enable":
      value

    }

  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
N	port number
value	The value you wish to set for the key

Value	Description
true	Enable control of the RGB LED's
false	Disbale control of the RGB LED's

Returns:

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

Errors

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

Example

```
  {
```

```
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "RGBControl",
        {
            "start": 1,
            "end": 8,
            "enable": true
        }
    ]
}
```

SecurityArmed

Enable / disable security feature. If the security is enabled, removal of a device from a port will sound an alarm (if installed) and flash lights (if installed).

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "SecurityArmed",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	enable security
false	disable security

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "SecurityArmed",
        "true"
    ]
}
```

Settings

Perform command line interface operation on the connected hub and return the complete result. This allows you to run commands directly on the hub's command line without stopping the API service. In order to update the settings the CLI setting will require a settings_unlock\n prefix to the command for more information on using CLI commands please see the CLI documentation https://www.cambrionix.com/cambrionix-cli.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Settings",
    "settings_unlock\nCommand"
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Command	The Command you wish to send to the hub For all CLI commands see the CLI Documentation https://www.cambrionix.com/cambrionix-cli

Returns:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": "Unlocked \nSetting updated"
}
```

Errors

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

If you attempt to update a setting that is already set you will receive a message stating that the setting is already set such as the example of sending the ports on below

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "result": "Unlocked \nForcing ports on has already been defined."
}
```

Example

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_hub_set",
  "params": [
    "DM01K2A8",
    "settings",
    "settings_unlock\nsettings_set ports_on 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1"
  ]
}
```

Temperature.OverTemperature

Force the behaviour of an over temperature condition.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "Temperature.OverTemperature",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Set the over temperature flag

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Temperature.OverTemperature",
        "true"
    ]
}
```

TwelveVoltRail.OverVoltage

Force the behaviour of a 12V over voltage condition. TwelveVoltRail is the same as InputRail.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "TwelveVoltRail.OverVoltage",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Set the 12V over voltage flag

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "TwelveVoltRail.OverVoltage",
        "true"
    ]
}
```

TwelveVoltRail.UnderVoltage

Force the behaviour of a 12V under voltage condition.

TwelveVoltRail is the same as InputRail.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_set",
  "params": [
    connection-handle,
    "TwelveVoltRail.UnderVoltage",
    Value
  ]
}
```

Parameter	Description
connection-handle	The Connection Handles as an integer
Value	The value you wish to set for the key

Value	Description
true	Set the 12V under voltage flag

Returns:

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

Errors

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

Example

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "TwelveVoltRail.UnderVoltage",
        "true"
    ]
}
```

14.4. Deprecated Dictionaries

These dictionaries exist to support backwards compatibility only and should not be used. These key-values may be removed in future versions.

API Call	Description
Settings	Obtain current hub Internal hub settings.

Settings

Obtain current hub Internal hub settings, returns as text.

Syntax: see Call Structure

```
{
  "method": "cbrx_connection_get",
  "params": [
    connection-handle,
    "Settings",
  ]
}
```

connection-handle is the Connection Handles as an integer.

Returns:

```
{
  "jsonrpc": "2.0",
  "id": 5,
  "result": [
    "Current memory Settings :",
    "",
    "settings_set company_name manufacturer-name",
    "settings_set product_name product-name",
    "settings_set local_name local-name",
    "settings_set attach_threshold attach-threshold",
    "settings_set default_profile default-profile",
    "settings_set remap_ports port-order",
    "settings_set ports_on ports-on",
```

```
    "settings_set sync_chrg sync-charge",
    "settings_set alt_sync_chrg alt-sync-charge",
    "settings_set misc_flags Internal hub-flags",
    "settings_set display_mode display-mode",
    "settings_set charged_threshold charged-threshold",
    "settings_set temperature_max shutdown-temperature",
    "settings_set stagger stagger"
  ]
}
```

Variables	Description
manufacturer-name	Defined name of manufacturer, Default is 'Cambrionix'
product-name	Hardware name of product
local-name	Local name set by the user, "-" if not set
attach-threshold	Current drawn in mA that the hub detects a device is connected, "d" means factory default is set
default-profile	Default profile for each port
port-order	Order the ports are by port number
ports-on	Whether each port is default on, 0 is default off 1 is default on
sync-charge	Whether CDP* on each port on, 0 is off, 1 is on, comma seperated list
alt-sync-charge	Whether alternative CDP* on each port on, 0 is off, 1 is on
Internal hub-flags	If any Internal hub Misc flags are active
display-mode	Change the display mode for logs, "d" means factory default is set
charged-threshold	Current drawn in mA that the hub detects a device is fully charged, "d" means factory default is set

Variables	Description
shutdown-temperature	Temperature that will shutdown the hub if reached in Celsius, "d" means factory default is set
stagger	A delay between ports turning on in ms, "d" means factory default is set

*Charging Downstream Port (CDP) Being enabled means that a port is capable of transferring data and charging the device at the same time with a higher current than just data syncing alone. With CDP enabled the hub can supply up to 1.5 A

If you disable CDP you will receive the notification “This Hub has the Charge Downstream Port UCS mode disabled. This could limit the maximum current seen on some ports.” This notification is there to ensure you haven’t turned this off by accident and can still have the highest charge available.

Example


```
{
    "jsonrpc": "2.0",
    "id": 5,
    "result": [
        "Current memory Settings :",
        "",
        "settings_set company_name cambrionix",
        "settings_set product_name SuperSync15",
        "settings_set local_name -",
        "settings_set attach_threshold d",
        "settings_set default_profile 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ",
        "settings_set remap_ports 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 ",
        "settings_set ports_on 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0",
        "settings_set sync_chrg 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1",
        "settings_set alt_sync_chrg 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ",
        "settings_set misc_flags 0000",
        "settings_set display_mode d",
        "settings_set charged_threshold d",
        "settings_set temperature_max d",
        "settings_set stagger d"
    ]
}
```

15. Socket Connections

When using the Python wrapper that provides the cbrxapi module, each time a call is made to the API, a socket is created. This socket is then used to send the command and receive the response before being closed.

If you are writing your own program, you may wish to consider creating a single socket at the start of your communication with the API and keeping this socket open until you wish to stop using the API. Keeping the socket open for the lifetime of your communication with the API will reduce the load on the system and lead to shorter communication cycles with the API.

If you do choose to manage your own socket connections to the API, it is important that you do not close the socket before receiving the response from the final command. Closing the socket without waiting to receive the response may lead to the requested operation not being completed, this is especially important on set and close operations.

16. Controlling the LEDs

The API can control product LEDs. By default these LEDs are controlled automatically by the product to indicate the state that a port is in.

In order for the LEDs to be controlled by the API this automatic control must be disabled and this is done by setting the “RemoteControl” key to be ‘True’. If you wish to return control of the LEDs to the automatic control you set “RemoteControl” to be ‘False’. See cbrx_connection_set for more information on using this Method.

Control of an LED is achieved by providing an 8 bit value which is interpreted in binary as a pattern that is continuously cycled through. So by setting the value 11110000b, the LED will flash slowly. The LED will be lit where there is a ‘1’ and unlit where there is a ‘0’. Alternatively setting the value 10101010b will make the LED flash fast. The pattern need not be symmetrical so 10010000b will produce two short flashes close together with a longer pause before the cycle repeats.

Any value set for an LED while RemoteControl is False will be overwritten and so have no effect.

A special argument of "auto" in place of True allows the hub to override the user set LED pattern when a device attached to that port is removed.

17. Battery Information

Battery information can be retrieved for connected devices. For Android™ devices using Android Debug Bridge (ADB), and for iOS devices an in-built build of libimobile.

ADB can be used to query the battery level on any Android™ devices providing a few conditions are met.

- The Android platform tools are installed, these can be downloaded from here.
- The ADB binary is in the path, or it's path is provided to the API via cbrx_config_set.
- The device has USB debugging enabled.
- You have trusted the computer from the phone if the phone requires it.

See this page for details on enabling debug mode on Android™ devices. The only options that are required are to enable developer mode and USB debugging.

```
# Install Android platform tools on Linux sudo apt install
        Android-platform-tools#   # Install Android platform tools on macOS
        brew cask install Android-platform-tools   # Install Android
        platform tools on Windows # Goto
        https://developer.Android.com/studio/releases/platform-tools # Download
        SDK Platform-Tools for Windows # Extract and add the folder to your
path
        or use # cbrx_config_set("adb_path" <pathname>) to add to API
        settings.
```

Finding adb without the path

Alternatively from setting the path, we can tell the API where to find these programs.

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_config_set",
  "params": {
    "adb_path": "/usr/local/bin"
  }
}
```

Mobile-device battery trust-levels

To obtain the battery information on mobile devices (phones / tablets) the device must be paired with the host system. To pair a device you will need to trust the host system on the

mobile device when first connecting. There are various trust levels which are documented below.

Trust level	Description
"not-paired"	Device not paired
"paired"	Device paired
"pending"	Battery information pending
"failed"	Failed to obtain battery information
"prohibited"	Prohibited from obtaining battery information
"error"	Error on obtaining battery information

18. API Error codes

Code	Value	Description
CBRXAPI_ERRORCODE_IDNOTFOUND	-10001	ID not found. The unit ID passed in does not represent a hub or it has been disconnected since discovery was last run. Note that there is an internal timeout that will close unused handles after a minute.
CBRXAPI_ERRORCODE_NOHANDLINGTHREAD	-10002	Unable to start handling thread. This error is not applicable past 2.1.
CBRXAPI_ERRORCODE_KEYNOTFOUND	-10003	Key not found. A key that is passed in cannot be found. It may be misspelled or not exist in the dictionary for this unit.
CBRXAPI_ERRORCODE_ERRORSETTINGVALUE	-10004	Could not set value. The (key value) pair was not acceptable. This could mean the key does not exist or is misspelled the value is of the wrong type or the value passed is invalid or out of range.
CBRXAPI_ERRORCODE_INVALIDHANDLE	-10005	Invalid handle. The handle passed in to a function is not valid or no longer valid. This could happen either by passing in an incorrect value or if the handle has already been closed (i.e. by cbrx_closeandlock being called) or the unit has been disconnected from the computer.
CBRXAPI_ERRORCODE_TIMEOUT	-10006	Timeout on communication. An operation towards a hub took too long to complete. It may have been disconnected or just slow to respond. It is worth retrying the operation.
CBRXAPI_ERRORCODE_DROPPED	-10007	Socket connection to remote has been dropped.
CBRXAPI_ERRORCODE_METHOD_REMOVED	-10008	The method has been removed.
CBRXAPI_ERRORCODE_AGAIN	-10009	System not ready. Try again. This is likely caused by a very prompt call to an API function and the system has not progressed through startup enough to service it.
CBRXAPI_ERRORCODE_FIRMWARE_UPDATE	-10010	Error performing firmware update.

Code	Value	Description
CBRXAPI_ERRORCODE_FIRMWARE_FILE	-10011	Firmware file error. This would usually be due to file format errors.
CBRXAPI_ERRORCODE_DEVICE_NOT_FOUND	-10012	Device not found.
CBRXAPI_ERRORCODE_CONNECTION_ERROR	-10014	Could not open the serial port connection to the hub.
CBRXAPI_ERRORCODE_HUB_NOT_FOUND	-10013	Hub not found.

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 Cambrionix 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 Cambrionix 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