Cambrionix
CHARGE • CONNECT • MANAGE

Cambrionix Hub API

翻訳されたユーザーマニュアル

Cambrionix
Hub API

JA

1.目 次

2.序章

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.

Python ラッパーには、JSON にあまり精通していなくてもスクリプトを記述できるパブリックドメインの JSON-RPC ライブラリが用意されています。または、選択したプログラミング言語を使用して、標準の TCP/IP ソケットを介してデーモンに直接接続し、JSON 形式のデータを送受信することもできます。API を使用してリモート ネットワークに接続されたハブと通信する場合、これは SSH トンネルを介して行われます。

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.Cambrionix ユニットのプロパティを API コマンドにマッピングします。

このマニュアルの最新版は、次のリンクの当社 Web サイトからダウンロードできます。
https://www.cambrionix.com/cambrionix-api

JSON-RPC ライブラリ

API は、JSON-RPC over TCP を使用します。JSON-RPC は、データ形式としての JSON への接続です。JSON-RPC は「JavaScript Object Notation Remote Procedure Call」の略です。つまり、JSON はデータ交換用の軽量フォーマットを表しています。転送しやすい構造化データのフォーマットです。

JSON-RPC をサポートする任意のプログラミング言語を使用できます。ライブラリは、他の言語で広く利用できます。

2.1.インストール

macOS®のインストール

macOS® の場合、CambrionixApiService をデーモン プロセスとして実行するようにセットアップするインストーラーが提供されます。サービスは、構成時にオンデマンドでロードするように構成され、リッスン ポートは launchd によって接続されます。

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.

インストールでは、必要なすべての手順を実行して、インストールされている Python 2 および 3 のバージョンを構成し、さまざまなサンプル スクリプトに誘導することもできます。

Windows のインストール

Windows の場合、Windows サービスとして実行するように CambrionixApiService をセットアップする自己解凍型インストーラーが提供されます。

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.

インストールでは、Python 2 および 3 をインストールして構成するために必要なすべての手順を実行し、さまざまなサンプル スクリプトに誘導することもできます。

Linux のインストール

Linux® パッケージは、GUI または apt を使用してコマンド ラインからインストールできる Debian パッケージとして提供されます。

```
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
```

armhf バージョンは、oDroid と Raspberry Pi でもテストされています。

現在、rpm ファイルはサポートされていません。サポートが必要な場合は、以下のページから詳細情報を入手できます。

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

USB ドライバー

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

これらのドライバーはインストーラーに含まれており、インストール中にオプションのコンポーネントを選択することでインストールできます。必要なドライバーが既に存在する場合、オプションは表示されません。ただし、Cambrionix 充電器がホスト マシンに接続されるまで、これらのドライバーのインストールは完了しません。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.前提条件

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

USB ハードウェアへの直接アクセス

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.

Windows での Thunderbolt™

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

USB情報用の同期可能な充電器

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.これは、同期対応の製品にのみ存在します。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.

ユニバーサルファームウェア製品のバージョン

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.

ファームウェア	部品番号	商品名
ユニバーサル	PP15S	PowerPad15S
ユニバーサル	PP15C	PowerPad15C
ユニバーサル	PP8S	PowerPad8S
ユニバーサル	SS15	SuperSync15
ユニバーサル	TS3-16	ThunderSync3-16
頭いい	TS3-C10	ThunderSync3-C10
ユニバーサル	U16S スペード	U16S スペード
ユニバーサル	U8S	U8S
PDシンク	PDSync-C4	PDSync-C4
ユニバーサル	ModIT-Max	ModIT-Max
モーター制御	モーター制御盤	ModIT-Max

USBドライバー

The Cambrionix Hub API daemon (CambrionixApiService) must be able to communicate with the local hub.ハブがUSBデバイスとして表示されます。USBデバイスには、仮想通信ポート (VCP) が付属しています。仮想通信は、標準のシリアル通信ポート、またはよく呼ばれるCOMポートのように動作します。オペレーティングシステムには、適切なVCP (仮想 COM ポート) ドライバーがインストールされている必要があります。

Linux®	マックOS	ウィンドウズ
カーネルのデフォルトのサポートで十分です。必要なVCPドライバーと競合するため、D2XXドライバーをインストールしないでください。	OSのデフォルトのサポートで十分です。必要なVCPドライバーと競合するため、D2XXドライバーをインストールしないでください。	D2XXサポートは、VCPサポートと共存できます。これらのドライバーは、新しいバージョンのWindows 10に自動的にインストールされます。

オペレーティング·システム

私たちはテストしましたCambrionix Hub API以下のオペレーティングシステムが動作することを確認できます。Cambrionix Hub API。動作するがテストされていない他のOSもあるかもしれません:

- ウィンドウズ10
- ウィンドウズ11

- macOS 11 (ビッグサー)
- macOS 12 (モントレー)
- macOS 13 (ベンチュラ)
- macOS 14 (ソノマ)
- macOS 15 (Sequoia)
- リナックス ウブントゥ
- リナックス デビアン

Linux の場合、上記の OS を使用してのみテストを実行します。ARM ハード フロート (armhf) バージョンも oDroid および Raspberry Pi でテストされています。

3.クイックスタート

インストールされたファイルには、Node.JS、C#、VB.Net、および Python のサンプル スクリプト が含まれています。Python の場合、古い同期の例ではなく、新しい asyncio の例を使用することをお勧めします。

- Windows: %ProgramFiles%\Cambrionix\CambrionixAPI/examples
- Linux: /usr/local/share/cambrionix/apiservice/examples.
- macOS: /ライブラリ/Cambrionix/ApiService/examples.

サンプル フォルダー内のコード タイプごとに、必要なプログラムをインストールする必要があります。

- Python サンプル用の Python 3.4 と Python 3.4 には、jsonrpc-websocket モジュールが必要です。Python の使用に関する参照情報は、ここにあります。
- nodejs の例の Node.JS には、NPM または Yarn も必要です。詳細については、こちらをご覧ください。
- C# または VB.Net のサンプル用の Visual Studio

Python async api パッケージをインストールするには、examples/python/asyncio に移動して実行します

```
pip install .
```

3.1.Python の例

jsonrpc-websocket モジュールは、python スクリプトを JSON-RPC リクエストに自動的に変換します。以下の例は、websocket が単純なスクリプトを変換する方法です。

パイソン。

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

JSON-RPC 変換。

```
{
  "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.

API の使用例を次に示します。コードは Python 3.6 で記述されています。サンプルコードはエラーをチェックしないため、Python スクリプトは単にエラーで停止します。堅牢なコード エラー処理を独自のソフトウェアに組み込む必要があります。

```
# 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 の例

これは、Typescript を使用して API を使用し、ハブとデバイスに関する情報を取得する簡単な例です。TypeScript の詳細については、こちらを参照してください。

```
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 の例

http プレフィックス付き URI に直接接続できます。この場合、json はアドレス自体または GET 要求の本文コンテンツから抽出されます。curl を使用して、ブラウザーまたはコマンド ラインからこの例を試すことができます。

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

ソケット接続は、単純なバイナリ データ、http GET 要求、または Web ソケット (Node.js など) にすることができます。たとえば、ブラウザのアドレス バーに次のコードを貼り付けると、結果がすぐに表示されます。

```
http://localhost:43424/?{"jsonrpc":"2.0","id":1,"method":"cbrx_
discover","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:%22c-brx_apidetails%22%7D
```

3.4.エラー処理

JSON-RPC エラーは、次のメンバーを含むエラー メンバーを返します。

- code (必須) - -32768 から -32000 の範囲の事前定義された JSON-RPC エラー コード、または セクション CBRXAPI 固有のエラー セクションに記載されている CBRXAPI エラー コードのいずれか を示す整数。
- message (オプション) - エラー コードを説明するメッセージ文字列
- data (オプション) - デバッグ メッセージやハンドルなどのエラーに関する追加情報。

使用される Python JSON-RPC は、次のマッピングでエラー応答の例外を引き起こします。

- メンバーコードは e.error_code で返されます
- メンバー メッセージは e.error_message で返されます
- メンバー データは e.error_data に返されます。

次の方法でエラー応答をキャッチできます。

```
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
```

エラーの作成方法とその応答の例:

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

応答:

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

4.API Call Structure

API 呼び出しの説明には、回線上で見られるように、JSON-RPC 要求/応答が含まれています。

4.1.JSON-RPC リクエスト オブジェクト

JSON-RPC は、ステートレスで軽量なリモート プロシージャコール(RPC) プロトコルです。JSON-RPC は Request オブジェクトを送信することで表されます。Request オブジェクトには次のメンバーがあります。

jsonrpc

JSON-RPC プロトコルのバージョンを指定する文字列。正確に「2.0」でなければなりません。

id

含まれている場合は文字列、数値、または NULL 値を含まなければならない、クライアントによって確立された識別子。含まれていない場合は、通知と見なされます。

方法

呼び出されるメソッドの名前を含む文字列。

パラメータ

メソッドの呼び出し中に使用されるパラメーター値を保持する構造化された値。これはすべての方法で必要なわけではありません

これをすべてグループ化すると、完全な JSON-RPC リクエストが得られます。

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

4.2.JSON-RPC 応答オブジェクト

rpc 呼び出しが行われると、通知の場合を除いて、応答があります。応答は、次のメンバーを持つ単一の JSON オブジェクトとして表現されます。

jsonrpc

JSON-RPC プロトコルのバージョンを指定する文字列。正確に「2.0」でなければなりません。

id

このメンバーは、Request オブジェクトの id メンバーの値と同じです。

結果

このメンバーの値は、Request オブジェクトのメソッドによって決定されます。

エラー

このメンバーは、エラーが発生した場合にのみ返されます。

これをすべてグループ化すると、完全な JSON-RPC 応答が得られます。

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

4.3.JSON-RPC Error Object

呼び出しでエラーが発生すると、応答オブジェクトには、次のメンバーを持つオブジェクトである値を持つエラー メンバーが含まれます。

コード

発生したエラーの種類を示す数値。これは整数です。

メッセージ

エラーの簡単な説明を提供する文字列。

データ

エラーに関する追加情報を含むプリミティブまたは構造化された値。

これをすべてグループ化すると、完全な JSON-RPC 応答が得られます。

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

4.4.Call Structure

この部分は、ドキュメントを簡素化するために、このマニュアル全体で構文とリターンのセクションから削除されました。

JSON 要求を完了するには、さらに 2 つのキーと値のペアを渡す必要があります。使用されている JSON-RPC のバージョンを示すもの (この場合は 2.0) と、このリクエストを識別する ID:

ID は必須ですが、同じ接続を介して複数のリクエストが同時に未処理になる可能性がある場合にのみ関連します。(非同期) 要求への応答を一致させるのに役立ちます。要求に対する応答には、CambrionixApiService によって一致する ID が与えられます。

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

5.API メソッド

API には 3 つの呼び出しグループがあります。

- バージョン - API に関する詳細を取得する
- 検出 - API に接続されているものに関する詳細を取得します
- 接続 - 接続と接続されているデバイスを管理します

バージョン

API 呼び出し	説明
cbrx_apiversion	API のバージョンを取得する
cbrx_apidetails	API の詳細の拡張バージョンを取得する

発見

API 呼び出し	説明
cbrx_discover	Cambrionix ユニットを発見
cbrx_discover_id_to_os_reference	OS で使用されるように、ユニット ID をユニットからデバイスにマップします
cbrx_find	ローカルの Cambrionix ユニットに接続されているデバイスを検索する
cbrx_get_usb (tree)	発見された USB ツリー全体を返す
cbrx_get_usb (descriptors)	Request entire dump of a USB device's descriptor information
cbrx_config_set	構成オプションの設定

繋がり

API 呼び出し	説明
cbrx_証明書	API への証明書と秘密鍵を管理する

API 呼び出し	説明
cbrx_connection_open	ハブへの接続を開く
cbrx_connection_close	開いているハブへの接続を閉じる
cbrx_connection_getdictionary	によって指定されたハブ上のすべてのキーを取得します 接続ハンドル。
cbrx_connection_get	で指定されたハブからキーを取得します 接続ハンドル
cbrx_hub_get	ハブのシリアル番号で指定されたハブからキーを取得します
cbrx_device_get	USB デバイスのシリアル番号で指定されたハブからキーを取得します
cbrx_connection_setdictionary	connectionHandle で指定されたハブのすべての書き込み可能キーとコマンド キーを一覧表示します
cbrx_connection_set	ハブで指定された値にキーを設定します connectionHandle で指定
cbrx_hub_set	ハブで指定された値にキーを設定します ハブのシリアル番号で指定
cbrx_notifications	通知を送信する
cbrx_ファームウェア	ファームウェア ファイルの追加または削除
cbrx_connection_closeandlock	ハブへのすべての接続を閉じてロックする
cbrx_connection_unlock	以前にロックされていたハブのロックを解除します。
cbrx_connection_cli	コマンド ライン インターフェイス操作を実行する
cbrx_pair_device	Initiaite pairing of an iOS device
cbrx_exit	API を再起動する

5.1.cbrx_apidetails

API の詳細の拡張バージョンを返します。この情報は、オプションの true パラメータを cbrx_apiversion に渡すことによっても取得できます。

構文: 参照API Call Structure

```
{
"メソッド": "cbrx_apidetails"
}
```

戻り値:

```
{
"結果": {
"version": [バージョン番号],
"semver": " semver-variant "、"commitid": commitid-number 、
"branch": " branch-name "、"capability": [ API-capability ]、
"notifications": [ possible-notification ]、"install ": "インストール場所", "ロギング": "ログの場所", "設定": "設定の場所", "ドキュメント": "ドキュメントの場所", "cpu": {
"brand": " brand-information ", "arch": " CPU-architecture ",
"features": [ CPU-features ], "cores": cores-value
      },
"os": " OS情報"
  }
}
```

出力	説明
バージョンナンバー	整数としてのAPIのバージョン番号 (メジャー、マイナー、リビジョン、ビルド)
セミバリアント	API バージョンの完全な名前

出力	説明
commitid バリアント	コミット ID の数値
支店名	インストールされた API のブランチ
API機能	APIのバージョンで利用可能
可能な通知	可能な通知を表示する文字列の配列。見るAPI 通知
インストール場所	インストールファイルの場所
ログの場所	ログが保存される場所
設定場所	API 設定の場所
ドキュメントの場所	API ドキュメントの Web アドレス
ブランド情報	CPUのブランド
CPU アーキテクチャ	CPUのアーキテクチャ
CPU 機能	CPUで利用できる機能
コア値	CPUのコア数
OS情報	ローカル マシンで実行されているオペレーティング システム

例

JSON-RPC リクエストの例:

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

例 成功応答

```
{
  "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

実行中の API のバージョンを返します。

構文: 参照 API Call Structure

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

使用できる別の方法があります。cbrx_apidetails詳細については。

戻り値:

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

*バージョン番号は、コンマで区切られた 2 つの*数字で構成されます。左端の数字をメジャー、右端の数字をマイナーと呼びます。

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.3.cbrx_証明書

API に証明書と秘密鍵を提供 (または削除) して、localhost (API が実行されているマシン) の外部からの SSL 接続を許可します。この証明書がないと、API は localhost:43424 での接続のみをリッスンします。有効な証明書と秘密鍵が提供されると、これは 0.0.0.0:43424 に変更されます。外部接続 (localhost からではない) は、SSL 接続 (HTTPS または Secure WebSockets) である場合にのみ許可されます。

アクセスが制限されたフォルダにある場合、セキュリティに違反する可能性があるため、API は証明書または秘密キーのコピーを作成しません。API を実行しているユーザーは、ファイルを使用できるようにファイルにアクセスする必要があります。これはすべて「set」コマンドが発行されたときにテストされ、機能しない場合は十分なエラー情報が提供されるはずです。

用途に適した証明書を提供するかどうかは、ユーザー次第です。たとえば、認証局によって署名されていない場合は、独自の認証局で証明書に署名し、それをアプリケーションまたはブラウザーに追加するなど、通常の方法でこれを処理する必要があります。

Google Chrome では、このガイドを使用できます。

Firefox では、このガイドを使用できます。

Safari では、このガイドを使用できます。

他のブラウザについては、オンラインで見つけることができるガイドがあります。

単一の証明書構成のみがサポートされています。パスワードが提供される場合、セキュリティのために難読化されます。

5.4.cbrx_certificate (セット)

証明書と秘密鍵を API に提供します。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_certificate",
"パラメータ": [
"設定"、{
"private-key": key-filename, "certificate": certificate-filename,
"password": password
    }
  ]
}
```

パラメータ	説明
キーファイル名	The filename including the path of the private key
証明書ファイル名	The filename including the path of the certificate
パスワード	秘密鍵で必要な場合はオプションのパスワード

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

注

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

例

JSON-RPC リクエストの例:

```
{
  "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"
    }
  ]
}
```

成功した応答の例:

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

5.5.cbrx_certificate (削除)

API から証明書と秘密鍵を削除します。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_certificate",
"params": ["削除"]
}
```

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.6.cbrx_certificate (get)

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

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_certificate",
"params": "get"
}
```

戻り値 :

```
"結果": {
"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	説明
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

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

成功した応答の例:

```
{
  "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

この関数を使用すると、永続的な構成オプションを設定できます。Multiple configuration keys can be set at once as the Example below shows.

構文：参照API Call Structure

```
{
"メソッド": "cbrx_config_set",
"パラメータ": {
"構成キー:構成値
  }
}
```

パラメータ	説明
構成キー	以下の表に従って、変更する構成キー。
構成値	構成を変更する値。

構成キー	説明
adb_path	Android™ Developer Tools からのADB実行可能ファイルへのフルパス名。
バッテリー更新対応	バッテリーの更新はまったく実行されますか。
バッテリー更新同時実行	同時に実行されるバッテリー更新の数 同時に。
バッテリー更新頻度秒	バッテリーの更新間隔の秒数。

戻り値：

```
{
「結果」:真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.8.cbrx_connection_cli

接続されたハブでコマンド ライン インターフェイス操作を実行し、完全な結果を返します。これにより、API サービスを停止することなく、ハブのコマンド ラインでコマンドを直接実行できます。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 設定 command.

構文: 参照API Call Structure

```
{
"メソッド": "cbrx_connection_cli",
"パラメータ": [
接続ハンドル
cli コマンド
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
cli コマンド	送信する CLI コマンド。すべての CLI コマンドについては、CLI ドキュメントを参照してください。www.cambrionix.com/cambrionix-cli

戻り値:

```
{
「結果」: [ cli 応答] }
```

*cli-response*は、コマンドから返されたすべての出力行を含む文字列の配列です。詳細については、www.cambrionix.com/cliを参照してください。

例

JSON-RPC リクエストの例

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

成功した応答の例:

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

5.9.cbrx_connection_close

接続ハンドルで指定されているように、以前に開いたハブへの接続を閉じます。

構文：参照API Call Structure

```
{
"メソッド": "cbrx_connection_close",
"params": [接続ハンドル]
}
```

*接続ハンドル*はConnection Handles整数として

戻り値：

```
{
「結果」：真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例：

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

成功した応答の例：

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

5.10.cbrx_connection_closeandlock

ハブへのすべての接続を閉じ、それ以上使用できないようにロックします。

cbrx_connection_unlock.これらの接続を使用していた他のプロセスがこのハブにアクセスしようとすると、エラーが返されます。

構文：参照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.

戻り値：

```
{
「結果」：真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例：

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

成功した応答の例：

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

5.11.cbrx_connection_get

接続ハンドルで指定されたハブから、キー値を取得します。

構文: 参照API Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル、「辞書キー」]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
辞書キー	への呼び出しによって返されるcbrx_connection_getdictionary見る 辞書を取得詳細については

戻り値:

```
{
「結果」: [辞書の値] }
```

*dictionary-value*は、ディクショナリキーの値です。を参照してください。辞書を取得詳細については。

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

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

成功した応答の例:

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

5.12.cbrx_connection_getdictionary

指定されたハブに関する情報を返すことができるすべてのキーを取得します。

構文：参照API Call Structure

```
{
"method": "cbrx_connection_getdictionary", "params": [接続ハンドル]
}
```

*接続ハンドル*はConnection Handles整数として

戻り値：

```
{
"結果": [辞書] }
```

*dictionary*は、デバイスのキーと値の名前を含む文字列の配列です。参照してください辞書を取得セクション。

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例：

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

成功した応答の例：

```
{
"jsonrpc": "2.0",
"id": 0,
"結果": [辞書] }
```

*dictionary*は、デバイスのキーと値の名前を含む文字列の配列です。参照してください辞書を取得セクション。

5.13.cbrx_connection_open

指定されたハブへの接続を開きます。正常に開くと、以降の呼び出しに使用できる接続ハンドルが生成されます。

構文：参照API Call Structure

```
{
"メソッド": "cbrx_connection_open",
"パラメータ": [
 hub-serial ,  location
  ]
}
```

パラメータ	説明
hub-serial	から返されたハブのシリアル番号です。cbrx_discover
位置	以下の表を参照してください。含まれていない場合は、デフォルトでローカルになります

ロケーション パラメータ	説明
ローカル	ローカルハブに接続する
ドック	でハブに接続するドックから見つけることができるIDを指定する必要があります cbrx_discover

戻り値：

```
{
「結果」: [ 接続ハンドル] }
```

*接続ハンドル*はConnection Handles整数として

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.14.cbrx_connection_set

接続ハンドルで指定されたハブで、キー値を設定します。ドックを呼び出すと、適切な充電器のみに送信されるポート固有のキーを除いて、関連するキーが両方の充電器に設定されます。

構文：参照API Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「辞書キー」、値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
辞書キー	への呼び出しによって返されるcbrx_connection_setdictionary見る 辞書の設定詳細については
価値	キーにも設定したい値

戻り値：

```
{
「結果」:真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例：

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

成功した応答の例:

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

5.15.cbrx_connection_setdictionary

によって指定されたハブのすべての書き込み可能な値キーとコマンド キーを一覧表示します。
Connection Handles.

構文: 参照API Call Structure

```
{
"method": "cbrx_connection_setdictionary", "params": [接続ハンドル]
}
```

*接続ハンドル*はConnection Handles整数として

戻り値:

```
{
"結果": [辞書]
}
```

*dictionary*は、デバイスの書き込み可能なキーとコマンド キーの名前を含む文字列の配列です。参照してください辞書の設定セクション。

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

成功した応答の例:

```
{
"jsonrpc": "2.0",
"ID": 0,
"結果": [辞書]
}
```

*辞書*についてはこちらをご覧ください辞書の設定セクション

5.16.cbrx_connection_unlock

以前にロックされていたハブのロックを解除します。

構文：参照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.

戻り値：

```
{
「結果」：真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例

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

成功した応答の例:

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

5.17.cbrx_device_get

シリアル番号で指定されたハブから、キー値を取得します。に似ているcbrx_connection_getポートに関連するget値のみが受け入れられます。

同じデバイスで複数の操作を行う必要がある場合、これは他の方法よりも遅いことに注意してください。

構文：参照API Call Structure

```
{
"メソッド": "cbrx_device_get",
"パラメータ": [
usb-serial, dictionary-key
  ]
}
```

パラメータ	説明
usb-serial	USB serial number
辞書キー	への呼び出しによって返されるcbrx_connection_getdictionary見る 辞書を取得詳細については

戻り値：

```
{
「結果」: [辞書の値] }
```

*dictionary-value*は、指定されたキー値です。辞書を取得詳細については。

エラー

APIメソッドにエラーがある場合、JSON-RPC Error Objectが返されます。

例

JSON-RPCリクエストの例：

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

成功した応答の例:

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

5.18.cbrx_discover

Cambrionix ユニットを検出し、ハブのシリアル番号を取得します。

構文: 参照API Call Structure

```
{
"メソッド": "cbrx_discover",
"params": [単位] }
```

単位パラメータ	説明
ローカル	接続されたハブのユニット ID
ドック	一緒に接続され、接続された複数のハブのユニット ID

戻り値:

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

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

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.19.cbrx_discover (「すべて」)

すべてのユニットを検出し、ハブとそれらに接続されたデバイスに関する詳細情報を返します。USB スキャンで表示されるデバイスのみが含まれることに注意してください。他の検出方法とは異なり、シリアル番号の配列ではなく、コンテンツを含むシリアル番号のオブジェクトになります。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_discover",
"params": ["all"] }
```

戻り値:

```
{
"結果": {
"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}
    }
  }
}
```

出力	説明
hub-serial	から返されたハブのシリアル番号です。cbrx_discover
状態	シリアルポートが開いているかどうか、を参照してください状態

出力	説明
メーカー名	メーカーの定義名、デフォルトは「Cambrionix」
ファームウェアバージョン	ファームウェアのバージョン番号。フォーマット「N.nn」
ブートローダーのバージョン	ブートローダーのバージョン番号。フォーマット「N.nn」
グループオーダー	下流の製品が最初に更新されて再起動されるように、接続された製品を更新するときに便利なハブを注文するために使用されます。
ファームウェアの種類	製品が受け入れるファームウェアを示すために使用されます
ハードウェアID	フロントパネル製品のハードウェアID番号
商品名	製品のハードウェア名
シリアルポート	のシリアルポート製品が接続されています。
USBバージョン	ハブへの接続のUSBバージョン番号。フォーマット「N.nn」
ロケーションID	のLocation IDs整数として
ポート数量	製品のポート数
外部電源	製品に外部電源ユニットがあるかどうか
ランタイム	製品の電源が入っていた時間 (ミリ秒)。制限なし
再起動フラグ	再起動フラグがtrueかfalseか
同期可能	製品が同期できるかどうか。正しいか間違っているか
5V-現在	製品に5Vが供給されているかどうか。正しいか間違っているか
12V-現在	製品に12Vが供給されているかどうか。正しいか間違っているか
一時的に可能	製品が温度を監視できるかどうか。正しいか間違っているか

出力	説明
ハードウェアフラグ	に詳述されているハードウェアフラグ辞書を取得
デバイス文字列	接続されたデバイスからの情報。デバイス文字列

例

JSON-RPC リクエストの例:

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

成功した応答の例:

```
{
  "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

検出されたハブのユニット ID を、OS で使用されるデバイス名にマップします。これは、ローカルに取り付けられた Cambrionix 製品にのみ使用できます。

構文: 参照 API Call Structure

```
{
"メソッド": "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.

戻り値:

```
{
"結果": ["デバイス名"]
}
```

device-name is what the OS uses for the connection that the hub which is identified by the *hub-serial*.詳細については、次を参照してください。シリアルポート

例:

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.21.cbrx_exit

API を再起動します。

構文: 参照API Call Structure

```
{
「方法」:「cbrx_exit」
}
```

戻り値 :

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.22.cbrx_find

ローカルの Cambrionix ユニットに接続されているデバイスを検索します。

構文: 参照 API Call Structure

```
{
"メソッド": "cbrx_find",
"params": [ ID ]
}
```

*ID*は次のいずれかの形式です。

ID パラメータ	説明
VID	ベンダー ID に一致するデバイスを検索します。整数として表示
VID & PID	ベンダー ID と製品 ID が完全に一致するデバイスを検索します。整数として表示
名前	指定された正規表現に一致するものを検索します。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)."<メーカー名>\x1D<製品名>\x1D<シリアル番号>".正規表現は一致ではなく検索として実行されるため、部分文字列を一致させるために「.*iPhone.*」などを実行する必要はありません。「iPhone」で十分です。好きなだけ厳しくすることができます。さらに、電話機の ID または内部シリアル番号が検出された場合、これらも照合されます。

戻り値:

```
{
"結果": {
"usb-serial": {
"HostDevice": "hub-serial", "HostPort": device-port,
"HostDescription": "product-name", "HostSerial": "serial-port",
"Device": ["device-string"] }
```

```
      }
    }
}
```

出力	説明
usb-serial	デバイスのシリアル番号
hub-serial	から返されたハブのシリアル番号です。cbrx_discover
デバイスポート	デバイスも接続されているハブのポート番号
商品名	製品のハードウェア名
シリアルポート	のシリアルポート製品が接続されています。
デバイス文字列	接続されたデバイスからの情報。デバイス文字列

返されるデータは、検索条件に一致するデバイスのシリアル番号に基づいています。各ノードの値には、場所の詳細と正確なデバイスの詳細が保持されます。

USB ツリー全体で指定されたアイテムが検索され、Cambrionix ハブの下のどこかに見つかった場合は、接続の詳細が返されます。これは、Cambrionix ハブに直接接続されているのではなく、中間のハブ デバイスに接続されているデバイス (拡張バッテリと追加の USB スロットを備えた電話など) に特に役立ちます。

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.

例

JSON-RPC リクエストの例:

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

成功応答の例

```
{
  "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_ファームウェア

ファームウェア メソッドは、Cambrionix ハブのファームウェア更新のすべての側面を制御できます。API のローカル ストレージにファームウェア ファイルを追加または削除したり、現在使用可能なファームウェア ファイルを一覧表示したり、提供されたファイルからファームウェアを更新したり、既存のファームウェア更新のステータスを確認したりできるサブコマンドがいくつかあります。

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.

ファームウェア	部品番号	商品名
ユニバーサル	PP15S	PowerPad15S
ユニバーサル	PP15C	PowerPad15C
ユニバーサル	PP8S	PowerPad8S
ユニバーサル	SS15	SuperSync15
ユニバーサル	TS3-16	ThunderSync3-16
頭いい	TS3-C10	ThunderSync3-C10
ユニバーサル	U16S スペード	U16S スペード
ユニバーサル	U8S	U8S
PDシンク	PDSync-C4	PDSync-C4
ユニバーサル	ModIT-Max	ModIT-Max
モーター制御	モーター制御盤	ModIT-Max

構文: 参照API Call Structure

```
{
"メソッド": "cbrx_firmware",
"params": [ファームウェア呼び出し]
}
```

ファームウェア呼び出し	説明
追加	利用可能な更新元としてファームウェアを API に提供します。見る cbrx_firmware (追加)
削除する	ファームウェアを API で使用できないようにします。見る cbrx_firmware (削除)
リスト	利用可能なすべてのファームウェアを一覧表示します。見るcbrx_firmware (リスト)
アップデート	ファームウェアの更新を開始します。見るcbrx_firmware (アップデート)
状態	ファームウェア更新のステータスを取得します。見るcbrx_firmware (ステータス)

5.24.cbrx_firmware (追加)

APIを実行しているホストのAPIローカルストレージにファームウェアファイルを追加します。ファームウェアファイルを追加するには、Base64でエンコードされたファイルのzipを提供します。Base64エンコーディングの詳細については、次のリンクを参照してください。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_firmware",
"パラメータ": [
"追加"、
「ファイル名」、エンコードされたバイト
  ]
}
```

パラメータ	説明
ファイル名	ファームウェアファイルの名前
エンコードされたバイト	Base64でエンコードされたファイルのzip

戻り値:

```
{
「結果」: 真
}
```

例

JSON-RPCリクエストの例: (この例では、base64でエンコードされたテキストは、マニュアル内で簡素化と明確化のために削減されていることに注意してください)

```
  {
```

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

成功した応答の例:

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

5.25.cbrx_firmware (リスト)

利用可能なすべてのファームウェア バージョンのリストを取得する

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_firmware",
"params": ["リスト"]
}
```

戻り値:

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

パラメータ	説明
ファイル名	ファイル名
ファームウェアバージョン	ファームウェアのバージョン番号

例

JSON-RPC リクエストの例:

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

成功した応答の例:

```
{
  "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 (削除)

API を実行しているホストの API ローカル ストレージからファームウェア ファイルを削除します。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_firmware",
"パラメータ": [
「削除」、「ファイル名」
  ]
}
```

*filename*は、ファームウェア ファイルの名前です。

戻り値:

```
{
「結果」: 真
}
```

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.27.cbrx_firmware (ステータス)

このメソッドを使用して、ファームウェア更新のステータスを取得できます。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_firmware",
"パラメータ": [
"状態"、
"接続ハンドル",
  ]
}
```

*接続ハンドル*はConnection Handles整数として

戻り値:

```
{
"結果": [
"Version": " firmware-version ", "Type": " firmware-type ",
"Progress":進捗率, "Stage": " stage-value " ]
}
```

パラメータ	説明
ファームウェアバージョン	ファームウェアのバージョン番号
ファームウェアの種類	ファームウェアのタイプを示すために使用されます
進捗率	パーセンテージで表した更新の進行状況
ステージ値	ファームウェアの更新が現在進行中の「ステージ」

ステージ値	説明
なし	ファームウェアが更新されていません

ステージ値	説明
connecting	Connecting to the hub to update the firmware
初期化	アップデートを初期化しています
消去	現在のファームウェアの消去
消された	現在のファームウェアが消去されました
updating	新しいファームウェアをインストールしています
updated	新しいファームウェアのインストールが完了しました
verifying	ファームウェアが正しくインストールされたことの確認
完了	チェックが完了しました
再起動中	すべてのチェックとインストールが完了した後のハブの再起動
再起動した	ハブが再起動され、使用できるようになりました
skipped	Update skipped as hub already updated

エラー

ステージにエラーがある場合、以下のいずれかのエラーがステージ値に表示されます。これらのステージエラーはいずれも、ハブのファームウェアが無効な状態にあり、やり直す必要があることを意味します。

ステージエラー	説明
crypt-init-failed	選択したデバイスに間違ったタイプのファームウェアが使用されました
初期化失敗	初期化ステージに失敗しました
erase-failed	現在のファームウェアを消去できませんでした
フラッシュ失敗	新しいファームウェアをハブにインストールできませんでした
チェック失敗	インストールのチェックに失敗しました
再起動失敗	ハブを再起動できませんでした

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.28.cbrx_firmware (アップデート)

この方法を使用して、ファームウェアの更新を開始します。一部の製品には、ディスプレイ、モーター制御ファームウェアなど、複数のファームウェアが含まれている場合があります。例に記載されているように、1つのコマンドで複数のファームウェアの更新を開始できます。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_firmware",
"パラメータ": [
"アップデート"、
"接続ハンドル",
"firmware-type""filename" ]
}
```

パラメータ	説明
ファームウェアの種類	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_ファームウェア section
ファイル名	ファイルの名前。使用可能なファイルを取得するために、使用可能なファイルから特定のファームウェア ファイルを使用できます。cbrx_firmware (リスト)
接続ハンドル	のConnection Handles整数として

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

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON エラーオブジェクトが返されます。

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.29.cbrx_get_usb (tree)

発見されたUSBツリー全体を返します。

構文: 参照Call Structure

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

戻り値

```
{
"結果": [
    {
"VID": vendor-id , "PID": product-id , "Description": "
description ", "LocationID": location-id , "USBVersion": USB-
version , "USBPower": {
"State": " power-state ", "Description": " power-description "
},
"ホストコントローラー": {
"Type": "ホスト コントローラー タイプ"、"EndpointTotal":アクティブ エンドポイント、"EndpointPeakTotal":ピーク エンドポイント、"EndpointMemoryUsed": エンドポイントメモリ、"EndpointPeakMemoryUsed":ピーク エンドポイント メモリ
        },
"子供": [
        {
"VID": vendor-id 、"PID": product-id 、"LocationID": location-id 、
"USBVersion": USB-version 、"USBPower":
            {
"State": " power-state ", "Description": " power-description "
},
```

```
"USBスピード":
            {
"Speed": " USB-speed ", "Description": " speed-name " },
「エンドポイント」:
            {
"Active": active-endpoints, "Maximum": maximum-endpoints,
"Memory": endpoint-memory,
"TotalInTree": {
"Endpoints": tree-endpoints
"Memory": tree-memory
            }
          }
        }
      ]
    }
  ]
}
```

出力	説明
ベンダーID	Device Vendor ID, or VID.整数として表示
製品番号	Product ID, PID.整数として表示
説明	ハードウェア名
ロケーションID	のLocation IDs整数として
USBバージョン	ハブへの接続のUSBバージョン番号。フォーマット「N.nn」
電源状態	USB Power Statesコード
電源説明	USB電源のオン/オフ
ホストコントローラータイプ	USBホスト コントローラのタイプ

出力	説明
ピークエンドポイント	USB ホスト コントローラーのエンドポイント使用率のピーク。見るエンドポイント
ピーク エンドポイント メモリ	エンドポイントのメモリ使用量のピーク。見るエンドポイント
USB速度	最大速度の USB 接続が可能
速度名	USB 接続の名前スーパースピード USB 5Gbps
アクティブエンドポイント	デバイスが使用しているエンドポイントの数
最大エンドポイント	デバイスが使用できるエンドポイントの数
エンドポイント メモリ	エンドポイントによって使用されているメモリの量
tree-endpoints	How many endpoints the tree is using
tree-memory	Amount of memory being used by endpoints in the tree

例

JSON-RPC リクエストの例:

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

成功した応答の例:

```
{
  "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).

構文: 参照 Call Structure

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

Variable	説明
locationID	のLocation IDs整数として
hub-serial	から返されたハブのシリアル番号です。cbrx_discover

戻り値

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.

例

JSON-RPC リクエストの例:

```
{
  "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": "SuperSpeedEndpointCcompanion",
              "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

ハブのシリアル番号で指定されたハブから、キー値を取得します。に似ているcbrx_connection_get .

同じハブで複数の操作を行う必要がある場合、この関数は遅くなることに注意してください。

構文: 参照API Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
 hub-serial,  dictionary-key
  ]
}
```

パラメータ	説明
hub-serial	から返されたハブのシリアル番号です。cbrx_discover
辞書キー	見る辞書を取得詳細については

戻り値:

```
{
「結果」: [辞書の値] }
```

*dictionary-value*は、指定されたキー値です。辞書を取得詳細については。

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.32.cbrx_hub_set

接続ハンドルで指定されたハブで、キー値を設定します。に似ているcbrx_connection_set

同じハブで複数の操作を行う必要がある場合、この関数は遅くなることに注意してください。

構文: 参照API Call Structure

```
{
"メソッド": "cbrx_hub_set",
"パラメータ": [
 hub-serial,  dictionary-key,  Value
  ]
}
```

パラメータ	説明
hub-serial	から返されたハブのシリアル番号です。cbrx_discover
辞書キー	見る辞書の設定詳細については
価値	キーに適用する値

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

成功した応答の例:

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

5.33.cbrx_notifications

API は、特定のイベントの通知の送信をサポートしています。通知パケットは、「id」フィールドがないことを除いて、他の JSON-RPC 応答オブジェクトと同じです。

注: 通知は、通知を要求したアクティブなソケット接続にのみ送信されます。ソケットを閉じて別のソケットを開くと、通知を再要求する必要があります。を参照してくださいクイックスタート例えば

構文: 参照API Call Structure

```
{
"メソッド": "cbrx_notifications",
"params": ["通知"], }
```

*通知*は文字列の配列から送信時に表示されますcbrx_apidetails可能な通知の完全なリストは、セクションにありますAPI 通知

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

```
{
  "jsonrpc": "2.0",
  "id": 0,
  "method": "cbrx_notifications",
  "params": [
    "usb-device-attached"
  ]
}
```

成功した応答の例:

```
{
  "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.

構文：参照API Call Structure

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

Variable	説明
UDID	The device USB serial number

戻り値

```
{
「結果」：真
}
```

例

JSON-RPC リクエストの例：

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

成功した応答の例：

```
{
```

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

6.API 通知

API 通知パケットは、「id」フィールドがないことを除いて、他の JSON-RPC パケットと同じです。これらの通知のほとんどは、「params」フィールドに何も提供しません。これらの通知は、cbrx_notifications方法。

以下の表に、API を使用して利用可能な通知のリストを示します。これらの通知のほとんどは、「params」フィールドに何も提供しません。

通知	説明
全て	すべての通知をリクエストする
ディスカバー - 変更	API は、使用可能なハブの変更を検出しました。再実行する必要がありますcbrx_discoverこの時点で。
デッドハブ変更	API は、ハブが応答しなくなったか、ハブに接続できないことを検出しました
ファームウェアの進行状況	ファームウェアの更新の進行状況に関する更新を要求する
過熱	ハブが過熱しています
過電圧	ハブが過電圧
RFID受信	RFID カードがセンサーに提示されたときに通知を受け取る
rfid 削除	RFID カードがセンサーから取り外されたときに通知を受け取る
USB デバイス接続	usb-changed よりも詳細で、接続されている特定のデバイスが表示されます。
USB デバイスの取り外し	usb-changed よりも詳細で、切り離された特定のデバイスが表示されます。
低電圧	ハブに電圧がかかっています
USB変更	API が USB ツリーの変更を検出しました。

ディスカバー - 変更

API で使用できるハブの変更が検出されました。

通知パケットの例:

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

デッドハブ変更

API は、ハブが応答しなくなったか、ハブに接続できないことを検出しました。たとえば、別のプログラムがシリアルポートを開いていることが原因です。

通知パケットの例:

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

ファームウェアの進行状況

ファームウェア更新の進行状況に関する情報。出力の詳細については、次を参照してください。cbrx_firmware (ステータス)

通知パケットの例:

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

過熱

ハブが動作温度を超えています。詳細については、製品のユーザーマニュアルを参照してください。

通知パケットの例:

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

過電圧

ハブが推奨電圧を超えています。詳細については、製品のユーザーマニュアルを参照してください。

通知パケットの例:

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

RFID受信

RFID センサーが RFID カードの存在を検出しました。

通知パケットの例:

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

rfid 削除

RFID センサーが、RFID カードが取り外されたことを検出しました。

通知パケットの例:

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

USB デバイス接続

デバイスが API で使用可能になり、デバイスに関する詳細な情報出力が表示されます。

通知パケットの例:

```
 {
  "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 デバイスの取り外し

デバイスは API で使用できなくなり、デバイスに関する詳細な情報出力が表示されます。

通知パケットの例:

```
{
  "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
    }
  }
}
```

低電圧

ハブは推奨電圧以下です。詳細については、製品のユーザーマニュアルを参照してください。

通知パケットの例:

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

USB変更

USB ツリーに変更がありました。

通知パケットの例:

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

7.非推奨のメソッド

これらのメソッドは下位互換性をサポートするためだけに存在し、使用しないでください。これらのメソッドは、将来のバージョンで削除される可能性があります。

API 呼び出し	説明
cbrx_apiversion (真)	API の詳細バージョンを取得する

7.1.cbrx_apiversion (真)

> *!cbrx_apidetailsこのメソッドは API バージョン 3.0 で廃止されました。'を使用してください!*

実行中の API の詳細バージョンを返します。

構文: 参照API Call Structure

```
{
"メソッド": "cbrx_apiversion",
"params": [真]
}
```

戻り値:

```
{
"結果": {
"version": [バージョン番号]、"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情報" }
}
```

出力	説明
バージョンナンバー	整数としての API のバージョン番号 (メジャー、マイナー、リビジョン、ビルド)

出力	説明
セミバリアント	API バージョンの完全な名前
commitid バリアント	コミット ID の数値
支店名	インストールされた API のブランチ
API機能	Cambrionix 内部使用のための API 情報
可能な通知	可能な通知を表示する文字列の配列。見るAPI 通知
インストール場所	インストールファイルの場所
ログの場所	ログが保存される場所
設定場所	API 設定の場所
ドキュメントの場所	API ドキュメントの Web アドレス
ブランド情報	CPUのブランド
CPU アーキテクチャ	CPUのアーキテクチャ
CPU 機能	CPUで利用できる機能
コア値	CPUのコア数
OS情報	ローカル マシンで実行されているオペレーティング システム

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

JSON-RPC リクエストの例:

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

例 成功応答


```
{
  "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)'!

発見されたUSBツリー全体を返します。

構文: 参照API Call Structure

```
{
"メソッド": "cbrx_get_usbtree",
}
```

戻り値

```
{
"結果": [
    {
"VID": vendor-id , "PID": product-id , "Description": "
description ", "LocationID": location-id , "USBVersion": USB-
version , "USBPower": {
"State": " power-state ", "Description": " power-description "
},
"ホストコントローラー": {
"Type": "ホスト コントローラー タイプ"、"EndpointTotal":アクティブ エンドポイント、"EndpointPeakTotal":ピーク エンドポイント、"EndpointMemoryUsed": エンドポイントメモリ、"EndpointPeakMemoryUsed":ピーク エンドポイント メモリ
        },
"子供": [
        {
"VID": vendor-id 、"PID": product-id 、"LocationID": location-id 、
"USBVersion": USB-version 、"USBPower":
            {
"State": " power-state ", "Description": " power-description "
```

```
},
"USBスピード":
            {
"Speed": " USB-speed ", "Description": " speed-name " },
「エンドポイント」:
            {
「アクティブ」: アクティブ エンドポイント、「最大」: 最大エンドポイント、「メモリ」: エンドポイントメモリ
          }
        }
      ]
    }
  ]
}
```

出力	説明
ベンダーID	Device Vendor ID, VID.整数として表示
製品番号	Product ID, PID.整数として表示
説明	ハードウェア名
ロケーション ID	のLocation IDs整数として
USB バージョン	ハブへの接続の USB バージョン番号。フォーマット「N.nn」
電源状態	USB Power Statesコード
電源説明	USB電源のオン/オフ
ホストコントローラータイプ	USB ホスト コントローラのタイプ
ピークエンドポイント	USB ホスト コントローラーのエンドポイント使用率のピーク。見るエンドポイント

出力	説明
ピークエンドポイントメモリ	エンドポイントのメモリ使用量のピーク。見るエンドポイント
USB速度	最大速度のUSB接続が可能
速度名	USB接続の名前スーパースピードUSB 5Gbps
アクティブエンドポイント	デバイスが使用しているエンドポイントの数
最大エンドポイント	デバイスが使用できるエンドポイントの数
エンドポイントメモリ	エンドポイントによって使用されているメモリの量

例

JSON-RPCリクエストの例:

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

成功した応答の例:

```
{
  "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.デバイス文字列

API がデバイスにクエリを実行すると、次の文字列が返されます。

```
"デバイス": {
"VID": vendor-ID,
"PID":製品 ID ,
"メーカー": "デバイスメーカー",
"説明": "説明",
"SerialNumber": "usb-serial",
"LocationID":ロケーション ID ,
"DevicePath": device-path
"USBVersion": USB-バージョン,
"USBPower": {
"状態": "電源状態",
「説明」:「電源の説明」
    },
"USB速度": {
"速度": " USB速度",
"説明": " USB 説明"
"容量": {
"速度": "可能な速度",
「説明」: 「有能な説明」
      }
    },
"エンドポイント": {
「アクティブ」: アクティブエンドポイント、
「最大」: 最大エンドポイント、
「メモリー」: エンドポイントメモリー
```

```
    },
"バッテリー": {
"DataSource": "バッテリーデータソース",
"TrustLevel": "信頼レベル",
"PairingSupported":サポート　ペアリング、
"CurrentLevel":バッテリー電流レベル,
"CurrentTime":現在のハブ時間、
"StartingLevel":充電開始レベル,
"StartingTime":充電開始時間、
"CapacityNew":新しいバッテリー容量、
「容量」:現在のバッテリー容量、
"ChargingStatus": "充電ステータス",
"HealthPercent": battery-health,
"Temperature": device-temperature
    },
"PhoneSerialNumber": "電話シリアル",
"PhoneIdentity": "電話名",
"PhoneModel": "phone-model",
"IMEI": " IMEI番号",
"MacAddress": " MacAddress ",
"PhoneSoftwareVersion": "phone-OS-version",
"PhoneECID": ECID,
"PhoneProductType": "phone-product",
"PhoneOSType": "phone-OS",
"PhoneColour": "phone-colour"
  }
}
```

出力	説明
vendor-ID	Device Vendor ID number, VID.整数として表示
製品番号	Product ID number, PID.整数として表示
デバイスメーカー	デバイスの製造元の名前
説明	ハードウェアの名前
usb-serial	USB serial number
ロケーション ID	のLocation IDs整数として
device-path	The platform specific path for the device
USB バージョン	The USB version number of the connection to the hub
電源状態	USB Power Statesコード
電源説明	USB電源のオン/オフ
USB速度	最大速度の USB 接続が可能
USB の説明	USB 接続の名前スーパースピード USB 5Gbps
対応速度	可能な最大データ速度デバイス
有能な説明	可能な最大データ速度デバイスの名前
アクティブエンドポイント	How many endpoints the device is using, displayed as a integer
最大エンドポイント	How many endpoints the device is capable of using, displayed as a integer
エンドポイント メモリ	Amount of memory being used by endpoints, displayed as a integer
バッテリーデータソース	デバイスのバッテリー情報のソース
信頼レベル	デバイスが信頼/ペアリングされているかどうか
サポート ペアリング	Whether the device supports being trusted/ paired

出力	説明
バッテリー電流レベル	Current battery percent level of device, displayed as a integer
現在のハブ時間	The hub time in ms, shown as an integer in ms
充電開始レベル	Battery percentage level when device connected, displayed as a integer
充電開始時間	The hub time in ms charging started, shown as an integer in ms
新しいバッテリー容量	The battery capacity of device from new, displayed as a integer
現在のバッテリー容量	The battery capacity of the device now, displayed as a integer
充電状態	バッテリーの充電状態満杯
バッテリーの状態	Battery health percentage, displayed as a integer
device-temperature	The temperature the device is reporting
電話シリアル	電話のシリアル番号
電話名	電話の名前
phone-model	The model of the phone i.e."iPhone 12"
IMEI番号	The IMEI number of the phone
マカアドレス	モバイルデバイスに割り当てられた一意のアドレス。12 の 16 進数文字で構成される 48 ビット値です。
電話 OS バージョン	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.

ウィンドウズ

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
```

マックOS

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.

ウィンドウズ

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
```

マックOS

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.追加情報

Cambrionix Connect Recorder Service

Recorder サービスはオプションのインストールコンポーネントで、デバイスの状態、充電履歴、接続イベントなどのイベントを記録できます。これらは、後でクライアント ソフトウェアで表示できます。

Stopping the API service

- ウィンドウズ

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

- ウィンドウズ

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
```

制限事項

API は、Cambrionix 製品のほとんどの機能を制御する手段を提供しますが、いくつかの制限があります。The API can only be used with the following products.

ファームウェア	部品番号	商品名
ユニバーサル	PP15S	PowerPad15S
ユニバーサル	PP15C	PowerPad15C
ユニバーサル	PP8S	PowerPad8S
ユニバーサル	SS15	SuperSync15
ユニバーサル	TS3-16	ThunderSync3-16
頭いい	TS3-C10	ThunderSync3-C10
ユニバーサル	U16S スペード	U16S スペード
ユニバーサル	U8S	U8S
PDシンク	PDSync-C4	PDSync-C4
ユニバーサル	ModIT-Max	ModIT-Max
モーター制御	モーター制御盤	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

接続ハンドルは、物理接続を表します。接続ハンドルは、使用するドライバーだけでなく、そのドライバーで使用するデータソースも定義します。コードのセグメント内で、接続ハンドルは接続情報を含む構造体を識別します。

を使用して接続ハンドルを取得できます。cbrx_connection_open

開いているハンドルで 30 秒以上アクティビティがない場合、ハンドルは無効になります。削除されたハンドルを使用しようとする後続の呼び出しは、CBRXAPI_ERRORCODE_INVALIDHANDLE で失敗します。API を使用するソフトウェアは、この状況に対処し、それに応じて対応できなければなりません。ソフトウェアは、新しいハンドルを取得するために単に cbrx_connection_open を再度呼び出すことができます。または、通知を登録すると、無期限に開いたままになります。

Location IDs

ロケーション ID は、USB ツリー内のデバイスの場所を表す 32 ビットの符号なし整数です。たとえば、ドライバーが a&b&c の形式でデバイスのロケーション文字列を作成する場合、ロケーション ID は 0x00000abc になります。

シリアルポート

デバイスが接続されているシリアル ポートの説明は、API が実行されている OS によって異なります。macOS および Linux の場合、これは場所として表示されます。Windows の場合、これは COM ポートとして表示されます。以下の例を参照してください。

ウィンドウズ

```
COM5
```

マックOS

```
dev/tty.usbmodem141502
```

Linux

```
/dev/ttyUSB0
```

エンドポイント

エンドポイントの制限が発生した場合、ホスト コンピューターで「USB リソースが不足している」ことを示すエラーが表示される可能性がありますが、エラー メッセージがまったく表示されず、(いずれかの) USB デバイスがランダムに失敗する可能性があります正しく動作するか、断続的になる。

USB エンドポイントの制限は、最近のマザーボードの xHCI (USB3) ホスト コントローラーにのみ適用されます。これらの USB ホスト コントローラーはメモリが限られており、通常は 64 ~ 128 個の USB エンドポイントを提供します。

xHCI ホスト コントローラーによって提供される限られた数のエンドポイントの問題は、ほとんどの USB デバイス、特に携帯電話やタブレットが複数のエンドポイントとして列挙されることです。一般的なモバイル デバイスは、5 つのエンドポイントとして列挙される場合があります。したがって、ホスト コントローラーに 64 個の使用可能なエンドポイントがある場合、そのホスト コントローラーに接続できるのは 64/5 デバイスのみであり、12 個のデバイスに相当します。USB ハブを含むすべての USB デバイスにはエンドポイントが必要であることに注意してください。そのため、ワークフローを設定するときは、この制限を考慮に入れることが重要です。

Power States

デバイスの電源状態には、D0、D1、D2、および D3 という名前が付けられます。D0 は完全にオンの状態で、D1、D2、および D3 は低電力状態です。状態番号は消費電力に反比例します。状態番号が大きいほど消費電力は少なくなります。

状態

API は、ハブ接続のステータスに関する次のオプションを返すことができます。

ステータス応答	説明
アイドル	ハブは接続されていますが、通信していません
アクティブ	ハブは接続され、使用中です
ない	ハブは USBtree にありません
反応しない	ハブが応答しなくなりました
ロックされた	ハブ接続がロックされています

11.ロギング

API は、すべての USB イベントのログ情報を生成し、何が起こったかに関する情報と特定のハードウェア情報を保存できます。これは、API で何が起こっているかを確認するのに役立ち、障害や問題をキャプチャできます。

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
```

次に、問題を再現した後、フォルダーからログを圧縮できます

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

作業が終了したら、以下のファイルを削除できます。

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

デフォルトの場所

CambrionixApiService によって生成されたログメッセージは、syslog に送られます。

Windows を使用すると、ログはデフォルトで以下の場所に保存されます

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

macOS を使用すると、ログはデフォルトで以下の場所に保存されます

```
ライブラリ>ログ>Cambrionix
```

Linux を使用すると、ログはデフォルトで以下の場所に保存されます

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

動作を調査するためのロギング

バグまたは問題が発生している場合は、動作のログを取得して、何が起こっているかをより詳細に確認できます。

1. API でのロギングを有効にする
2. 発生している問題を引き起こす方法でハブを使用します。
3. 問題が発生するまで待ちます
4. 問題が発生した時刻をメモし、ログが保存されているフォルダーを圧縮します。

この情報を取得したら、自分でログを確認するか、サポートに関して Cambrionix に連絡している場合は、サポート チケット システム経由でログを送信できます。

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.ドック

複数の製品が接続されている場合、2番目の充電器が1番目の充電器の拡張ポートに接続されるなど、これはDockと呼ばれます。一部の操作では、これら2つの充電器を1つのユニットとして扱い、両方の充電器のポートを組み合わせると便利な場合があります。

単一のユニットとしてドックにアクセスするには、最初にcbrx_discoverをパラメーターdocksで呼び出して、使用可能なドックのリストを取得します。次に、問題のIDと"docks"を指定してcbrx_connection_openを呼び出します。

ドックユニットは、nrOfPortsやTotalCurrent_mAなどのキーの合計を返します。ポートの範囲は、接続された製品の合計ポート数をカバーするように拡張されています。コンピュータに直接接続された製品のポートが最初に参照され、次に最初の充電器の拡張ポートに接続された充電器のポートが参照されます。

ハードウェアやファームウェアなどの一部のキーは結合されないため、これらのキーは親充電器の値を返します。これらのキーの値を下流の充電器から取得したい場合は、通常の方法でその充電器を開いて取得することができます。設定が変更された場合を除いて、充電器を開いてドックへのアクセスを妨げることはありません。

13.Dynamic Hubs

他のさまざまなハブを組み合わせた動的ハブを開くことができます。これは、ドック行う。動的ハブを開くには、この例に示すように、開きたいすべてのハブのシリアル番号を特別な「Dynamic:」プレフィックス付きの名前に単純に結合します。

```
# 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
```

この動的ハブは、ポートに1からNまでの番号が付けられた単一のエンティティとして扱われます。ここで、Nは含まれるすべてのハブのポートの総数です。

14.辞書

ハブごとに、API は 2 つの辞書を返すことができます。

- 読み取り可能なキーを含む Get ディクショナリ。
- 設定可能なキーを含む Set ディクショナリ。

返されるキーと値のペアは、ユニットがサポートする機能セットによって異なります。

14.1.機能セット

次の機能セットを利用できます。

機能セット	説明
ベース	すべての Cambrionix ユニットでサポートされる基本レベルの機能
同期	同期機能
5V	ユニットには5Vの固定電源があります
12V	本体は12V電源です
温度	本体には温度センサーが付いています
PD	ユニットは USB Power Delivery Specification を実装しています
ゲート	ユニットには、ロックゲートを制御してポートに接続されたデバイスを保護するモーター制御製品があります。

すべての製品が基本機能セットをサポートしています。

追加の機能セットも使用できる場合は、基本機能セットのキーの可能な値の範囲を拡張できます。

Hardware キーは、ハブのタイプの値を返します。

これらは、CambrionixApiService がさまざまなタイプのハブでサポートする追加の機能セットです。

「Hardware」によって返されるハブタイプ	同期	5V	12V	温度	PD	ゲート
PP8C		はい	はい	はい		
PP8S	はい	はい	はい	はい		
PP15C		はい	はい	はい		

「Hardware」によって返されるハブ タイプ	同期	5V	12V	温度	PD	ゲート
PP15S	はい	はい	はい	はい		
SS15	はい	はい	はい	はい		
シリーズ8		はい				
U8C-EXT		はい	はい	はい		
U8C		はい				
U8RA	はい	はい				
U8S-EXT	はい	はい	はい	はい		
U8S	はい	はい				
U10C		はい				
U10S	はい	はい				
U12S	はい	はい				
U16S Spade-NL	はい	はい				
PD 同期 4	はい*1			はい	はい	
ThunderSync2-16	はい	はい		はい		
ThunderSync3-16	はい	はい		はい		
ModIT Max*2	はい	はい		はい		はい

*1 PDSync-C4 は「同期」機能セット自体を実装していませんが、同期機能を備えており、これらは常に利用可能であることに注意してください。つまり、充電モードと同期モードを切り替える必要はありません。

*2 ModIT Max は ThunderSync3-16 として認識されますが、ゲート制御用のハードウェアが追加されています。

14.2.辞書を取得

鍵	機能セット
添付	5V
編集済み	ベース
有効なプロファイル	5V
ファームウェア	ベース
Firmware Requirements	ベース
FiveVoltRail_flags	5V
FiveVoltRail_Limit_Max_V	5V
FiveVoltRail_Limit_Min_V	5V
FiveVoltRail_V	5V
FiveVoltRailMax_V	5V
FiveVoltRailMin_V	5V
ゲート	ゲート
グループ	ベース
ハードウェア	ベース
ハードウェアフラグ	ベース
HardwareInformation	ベース
健康	ベース
HostPresent	PD
InputRail_flags	PD
InputRail_Limit_Max_V	PD
InputRail_Limit_Min_V	PD
InputRail_V	PD
InputRailMax_V	PD

鍵	機能セット
InputRailMin_V	PD
キー.N	5V
モードチェンジオート	同期
nrOfPorts	ベース
パネルID	ベース
Port.N.Battery	同期
Port.N.Current_mA	ベース
Port.N.Description	PD
Port.N.Energy_Wh	ベース
Port.N.Flags	ベース
Port.N.FlashDrive	同期
Port.N.LocationID	同期
Port.N.Manufacturer	PD
Port.N.Mode	ベース
ポート.N.PID	PD
Port.N.ProfileID	5V
Port.N.Profiles	5V
Port.N.SerialNumber	PD
Port.N.TimeCharged_sec	ベース
Port.N.TimeCharging_sec	ベース
Port.N.USBStrings	PD
Port.N.VID	PD
Port.N.Voltage_10mV	PD
PortInfo.N	ベース
ポート情報	ベース

鍵	機能セット
Profile.N.enabled	5V
pwm_percent	温度
再起動しました	ベース
セキュリティ武装	5V
設定	ベース
システムタイトル	ベース
温度_C	温度
温度_フラグ	温度
Temperature_Limit_Max_C	温度
温度最大_C	温度
合計電流_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	ベース

添付

デバイスが接続されているポートごとに1つのビットが設定されたビット フィールド。ポート 1 はビット 0、ポート 2 はビット 1 など。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"添付"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 添付ビット
}
```

*attach-bit*は整数値です。

ポート 1、2、3 に接続された 3 つのデバイスの例

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

編集済み

ファームウェア バージョンのタイムスタンプ。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"編集済み"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{ "結果": "コンパイル日"
}
```

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

タイムスタンプ	説明
うーん	英語の月の最初の3文字
DD	整数としての月の日付
YYYY	整数としての年
HH	ビルド時間、0 ~ 23
んん	ビルドの分、0 ~ 59
SS	ビルドの秒、0-59

例

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

有効なプロファイル

現在有効になっているグローバルプロファイルのリスト

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「有効なプロファイル」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "プロファイル"
}
```

*プロファイル*は、ハブに適用されるすべての充電プロファイルのリストです。プロファイルは、各プロファイル間にスペースを入れた単一の数字として表示されます。

例

充電プロファイル1、2、3、および4が有効な場合の戻り値。

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

ファームウェア

ファームウェア バージョン文字列。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"ファームウェア"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」:「ファームウェアバージョン」
}
```

*firmware-version ファームウェア*のバージョン番号です。フォーマット「N.nn」

例

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

Firmware Requirements

このハブに適用可能なファームウェアのタイプを取得します。ハブが受け入れるすべてのタイプのファームウェアの情報を含む配列として返されます。各エントリのフォームファクタフィールドは、そのパーツのファームウェアタイプを示します。これは、Universal の場合は「un」、PDSync の場合は「pd」、TS3-C10 の場合は「st」、モーター制御ボードの場合は「mc」のいずれかになります。

ファームウェア	部品番号	商品名
ユニバーサル	PP15S	PowerPad15S
ユニバーサル	PP15C	PowerPad15C
ユニバーサル	PP8S	PowerPad8S
ユニバーサル	SS15	SuperSync15
ユニバーサル	TS3-16	ThunderSync3-16
頭いい	TS3-C10	ThunderSync3-C10
ユニバーサル	U16S スペード	U16S スペード
ユニバーサル	U8S	U8S
PDシンク	PDSync-C4	PDSync-C4
ユニバーサル	ModIT-Max	ModIT-Max
モーター制御	モーター制御盤	ModIT-Max

構文：参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「ファームウェア要件」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値：

現在インストールされている各タイプのファームウェアに関する情報の配列と、それらを更新するために必要なファームウェアタイプの詳細を返します。

```
{
"結果": [
    {
"メーカー": "メーカー名",
"ハードウェア": "製品名",
"ファームウェア": "ファームウェアのバージョン",
"ブートローダー": "ブートローダーのバージョン",
"Group": "group-order",
"FormFactor": "ファームウェアの種類",
"HardwareID":ハードウェア ID ,
"グループ": "グループ順"
"SerialNumber": "hub-serial",
    }
  ]
}
```

出力	説明
hub-serial	から返されたハブのシリアル番号です。cbrx_discover
メーカー名	メーカーの定義名、デフォルトは「Cambrionix」
ファームウェアバージョン	ファームウェアのバージョン番号。フォーマット「N.nn」
ブートローダーのバージョン	ブートローダーのバージョン番号。フォーマット「N.nn」
グループオーダー	下流の製品が最初に更新されて再起動されるように、接続された製品を更新するときに便利なハブを注文するために使用されます。

出力	説明
ファームウェアの種類	製品が受け入れるファームウェアを示すために使用されます
ハードウェア ID	フロントパネル製品のハードウェア ID 番号
商品名	製品のハードウェア名

例

```
{
    "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

検出された5V電源レールエラーフラグのリストを返します(存在する場合)。

構文:参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"FiveVoltRail_flags"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "フラグ"
}
```

国旗	説明
紫外線	不足電圧発生
OV	過電圧発生
紫外線 紫外線	不足電圧と過電圧の両方が発生しました。

例

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

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

FiveVoltRail_Limit_Max_V

エラー フラグをトリガーする 5V レールの上 限 。

構 文 : 参 照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"FiveVoltRail_Limit_Max_V"
  ]
}
```

*接続ハンドル*はConnection Handles整 数 として。

戻 り値 :

```
{
「結果」: FiveVoltRail-Limit-Max
}
```

*FiveVoltRail-Limit-Max*は、n.nn 形 式 のボルト 単 位 の 10 進 数 です。

例

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

FiveVoltRail_Limit_Min_V

エラーフラグをトリガーする5Vレールの下限

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"FiveVoltRail_Limit_Min_V"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値：

```
{
「結果」: FiveVoltRail-Limit-Min
}
```

*FiveVoltRail-Limit-Min*は、n.nn形式のボルト単位の10進数です。

例

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

FiveVoltRail_V

現在の5V電源電圧 (ボルト (V))

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「FiveVoltRail_V」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: FiveVoltRail_V
}
```

*FiveVoltRail_V*は、n.nn形式のボルト単位の10進数です。

例

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

FiveVoltRailMax_V

ボルト (V) で測定された最高 5V 供給電圧

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"FiveVoltRailMax_V"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: FiveVoltRailMax_V
}
```

*FiveVoltRailMax_V*は、n.nn 形式のボルト単位の10進数です。

例

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

FiveVoltRailMin_V

ボルト (V) で測定された最低 5V 供給電圧

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"FiveVoltRailMin_V"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: FiveVoltRailMin_V
}
```

*FiveVoltRailMin_V*は、n.nn 形式のボルト単位の 10 進数です。

例

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

ゲート

存在する場合、拡張製品のすべてのゲートの状態を記述するオブジェクトを返します。現在、これはModIT範囲でのみ使用できます。

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「ゲイツ」
   ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値：

```
{
"結果":
   {
" N ":"状態"
   }
}
```

*N*はポート番号です

state	説明
開いた	ゲートは開いています
閉まっている	門は閉まっています
オープニング	門が開いています

state	説明
閉鎖	門が閉まっています
失速した	門が動かなくなった
タイムアウト	ゲートの応答に時間がかかりすぎました
わからない	上端スイッチも下端スイッチも作動していない
オープン失速	ゲートは開いた位置で失速しました
閉ざされた	ゲートが閉位置で失速した
オープニング失速	開門中にゲートが止まった
閉店失速	ゲートは閉鎖中に失速しました
無効	ゲートの開閉を禁止する無効フラグが設定されています

例

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

グループ

PCB ジャンパから読み取ったグループ文字。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"グループ"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "グループ順"
}
```

*group-order*ハブの順序付けに使用されます。これは、下流の製品が更新されて最初に再起動されるように、接続された製品を更新するときに役立ちます。または、グループ ジャンパが取り付けられていない場合は「-」。

例

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

ハードウェア

ハブの部品番号。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「ハードウェア」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "製品名"
}
```

*product-name*は製品のハードウェア名です

例

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

ハードウェアフラグ

機能が存在するかどうかを示すフラグ

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「ハードウェアフラグ」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "フラグ"
}
```

国旗	説明
S	同期機能セット
L	5V機能セット
え	12V 機能セット
T	温度機能セット
P	PD 機能セット

例

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

HardwareInformation

Information on the hub

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"HardwareInformation"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": {
"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	説明
商品名	Product name of the hub
product-webpage	The webpage for the hub
min-temperature	The minimum recommended temperature for the hubs enviroment (°C)

Variable	説明
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)

例


```
{
  "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.
      }
    }
  }
```

健康

ポート固有ではなく、動的に変化する使用可能なすべてのキー(ディクショナリとして)。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"健康"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": {
"稼働時間_秒":稼働時間_秒,
"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":温度_C ,
"TemperatureMax_C": TemperatureMax_C ,
"Temperature_flags": " Temperature_flags ",
「再起動」:再起動しました
  }
}
```

出力	説明
Uptime_sec	見るUptime_sec
FiveVoltRail_V	見るFiveVoltRail_V
FiveVoltRailMin_V	見るFiveVoltRailMin_V
FiveVoltRailMax_V	見るFiveVoltRailMax_V
FiveVoltRail_flags	見るFiveVoltRail_flags
TwelveVoltRail_V	見るTwelveVoltRail_V
TwelveVoltRailMin_V	見るTwelveVoltRailMin_V
TwelveVoltRailMax_V	見るTwelveVoltRailMax_V
InputRail_V	見るInputRail_V
InputRailMin_V	見るInputRail_Limit_Min_V
InputRailMax_V	見るInputRailMax_V
TwelveVoltRail_flags	見るTwelveVoltRail_flags
InputRail_flags	見るInputRail_flags
温度_C	見る温度_C
温度最大_C	見る温度最大_C
温度_フラグ	見る温度_フラグ
再起動しました	見る再起動しました

例


```
{
    "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

ハブは、接続されたホスト コンピュータのホスト USB ソケットを監視します。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「ホストプレゼント」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: ホスト
}
```

ホスト	説明
真実	ホストが検出されました
間違い	ホストが検出されません

例

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

InputRail_flags

入力レールエラーフラグのリスト(設定されている場合)。

構文:参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"InputRail_flags"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」:フラグ
}
```

国旗	説明
紫外線	不足電圧発生
OV	過電圧発生
フラグなし	電圧は許容可能です

例

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

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

InputRail_Limit_Max_V

エラーフラグをトリガーする入力レールの上限

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"InputRail_Limit_Max_V"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値：

```
{
「結果」: InputRail-Limit-Max
}
```

*InputRail-Limit-Max*は、n.nn形式のボルト単位の10進数です。

例

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

InputRail_Limit_Min_V

エラー フラグをトリガーする入力レールの下限。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"InputRail_Limit_Min_V"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: InputRail-Limit-Min
}
```

*InputRail-Limit-Min*は、n.nn 形式のボルト単位の 10 進数です。

例

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

InputRail_V

ボルト (V) 単位の電流入力レール供給。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"InputRail_V"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: InputRail_V
}
```

*InputRail_V*は、n.nn 形式のボルト単位の 10 進数です。

例

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

InputRailMax_V

ボルト (V) で測定された最大入力電圧。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"InputRailMax_V"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: InputRailMax_V
}
```

*InputRailMax_V*は、n.nn 形式のボルト単位の10進数です。

例

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

InputRailMin_V

ボルト (V) で測定された最低入力電圧。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"InputRailMin_V"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: InputRailMin_V
}
```

n.nn 形式のボルト単位の10進数です。

例

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

キー.N

ボタンが押された場合に情報を取得します。ダブルクリックは検出されません。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Key.N」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: ボタン
}
```

ボタン	説明
0	このエントリが最後に読み取られて以来、ボタン n は押されていません
1	このエントリが最後に読み取られてから、ボタン n が押されました

例

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

モードチェンジオート

Charge から Sync へのモード変更は自動です。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「モードチェンジオート」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 自動変更
}
```

自動変更	説明
真実	Charge から Sync へのモード変更は自動です。
間違い	Charge から Sync へのモード変更は手動です。

例

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

nrOfPorts

ハブの USB ポートの数。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"nrOfPorts"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: ポート番号
}
```

*ポート番号*は、使用可能なポート数の整数です

例

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

ノート

- PDSync-C4 には、追加のポート 0 があります。これは、ホスト ポートに関する情報です。

パネルID

取り付けられている場合は、フロントパネルボードのPanelID番号。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「パネルID」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "パネル ID "
}
```

*Panel-ID*はフロントパネル製品のID番号です。適合しない場合はAbsent/Noneを返します

例

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

Port.N.Battery

可能であれば、接続されているデバイスの現在のバッテリーレベルを取得します。バッテリ情報の収集に関する注意事項を参照してください。デバイスの種類 (Android™、iOS など) とホスト OS によっては、異なるデータが返される場合があります。

Apple の場合、「Apple Mobile Device Support」をインストールする必要があります (iTunes に含まれています)。

Android の場合、adb をインストールして実行する必要があります。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル
「Port.N.Battery」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": {
"CurrentLevel":バッテリー電流レベル,
"CurrentTime":現在のハブ時間、
"StartingLevel":充電開始レベル,
"StartingTime":充電開始時間、
```

```
    }
}
```

出力	説明
バッテリー電流レベル	デバイスの現在のバッテリーレベルをパーセンテージで表示
現在のハブ時間	ハブ時間 (ミリ秒単位の整数で表示)
充電開始レベル	デバイスが接続されているときのバッテリーのパーセンテージレベル
充電開始時間	ハブの充電開始時間 (ミリ秒単位の整数で表示)

例

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

Port.N.Current_mA

このUSBポートに接続されたUSBデバイスに供給される電流 (ミリアンペア (mA))。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.Current_mA」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: Current_mA
}
```

*Current_mA*は、デバイスに供給される電流 (mA (ミリアンペア)) です。

例

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

Port.N.Description

このUSBポートに接続されているUSBデバイスが検出された場合に報告される説明。検出できなかった場合は、空の文字列が返されます。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.Description」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "説明"
}
```

*説明*は、ハードウェアの名前です。

例

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

Port.N.Energy_Wh

このUSBポートのUSBデバイスが消費したエネルギー。

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.Energy_Wh」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値：

```
{
「結果」: エネルギー-Wh
}
```

*energy-Wh*はワット時単位の電力 (1秒ごとに計算)、nnの形式

例

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

Port.N.Flags

特定のポートのすべてのフラグのリストを取得します

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.Flags」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "フラグ"
}
```

スペースで区切られた、大文字と小文字を区別するフラグ文字のリスト。O、S、B、I、P、C、F は相互に エクスクルーシブ。A、D は相互に排他的です。	
国旗	**説明**
O	ポートはオフモードです
S	ポートは同期モードです
B	ポートはバイアスモードです
私	ポートは充電モードで、IDLE です

P	ポートは充電モードで、PROFILING です
ハ	ポートは充電モードで、充電中です
ふ	ポートは充電モードで、充電が完了しています
あ	デバイスはこのポートに接続されています
D	このポートにはデバイスが接続されていません。ポートは切り離されています
T	デバイスがポートから盗まれました: THEFT
え	エラーが存在します。ヘルスコマンドを参照
R	システムが再起動しました。crf コマンドを参照
r	モード変更中に Vbus がリセットされている

例

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

Port.N.FlashDrive

検出された場合、USB フラッシュドライブのマウント ポイントを返します。Windows の場合、これはドライブ文字になります。それ以外の場合は、ボリューム マウント ポイントになります。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.FlashDrive」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"パス:": "場所",
「容量」: 合計メモリ,
「利用可能」: 利用可能なメモリ
}
```

出力	説明
位置	ホスト システム内のディスクの場所
合計メモリ	ディスク上の合計メモリ
使用可能なメモリ	ディスク上のメモリが使用されていない

例:

ウィンドウズ:

```
{
    "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
}
```

フラッシュドライブがない場合、戻り値は単にfalseになります。

これと同じ情報は、PortsInfo、PortInfo.N、またはcbrx_discover('all')の「FlashDrive」フィールドでも提供されます(該当する場合)。該当しないか事前に設定されていない場合、このフィールドは表示されません。

Port.N.LocationID

特定のポートのロケーション ID を返します。これはデバイスを接続する必要がないため、USB スロットを一意に識別するために使用できます。ロケーション ID は、最初のバイトで USB ホスト コントローラーがオンになっているバス番号を示し、次に子デバイスのツリーの下にあるポート番号を示します。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.LocationID」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: ロケーション ID
}
```

*location-id*はLocation IDs整数として

例

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

ノート

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

Port.N.Manufacturer

このUSBポートに接続されたUSBデバイスによって報告された製造元(検出された場合)。検出できなかった場合は、空の文字列が返されます。

構文:参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.Manufacturer」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "デバイス メーカー"
}
```

*device-manufacturer*は、デバイスメーカーの名前です。

例

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

Port.N.Mode

現在のポート モード。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル
「Port.N.Mode」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "モード"
}
```

標準 USB ハブの場合、モードは次のいずれかになります。

モード キャラクター	説明
s	同期モード
c	充電モード
b	バイアスモード
o	オフ

Type-C ハブの場合、モードは次のいずれかになります。

モードキャラクター	説明
c	の上
o	オフ

例

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

ポート.N.PID

このUSBポートに接続されているUSBデバイスの製品ID(検出された場合)。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「ポート.N.PID」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 製品 ID
}
```

*product-id*は製品ID番号またはPIDです。整数として表示されます。検出できなかった場合は0(ゼロ)が返されます。

例

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

Port.N.ProfileID

プロファイル ID 番号。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「ポート.N.PID」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: プロファイル ID
}
```

*profile-ID*はプロファイル番号、または課金していない場合は 0 です。

例

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

Port.N.Profiles

このポートの有効なプロファイルのリスト。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.Profiles」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": "プロファイル"
}
```

*プロファイル*は、ポートに適用されるすべての課金プロファイルのリストです。プロファイルは、各プロファイル間にスペースを入れた単一の数字として表示されます。

例

充電プロファイル1、2、3、および4が有効になっています。

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

Port.N.SerialNumber

このUSBポートに接続されているUSBデバイスによって報告されたシリアル番号(検出された場合)。検出できなかった場合は、空の文字列が返されます。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.SerialNumber」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

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

usb-serial is the USB serial number

例

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

Port.N.TimeCharged_sec

このUSBポートがデバイスの充電完了を検出してからの時間 (秒)。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"Port.N.TimeCharged_sec"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"result":充電完了
}
```

*charge-complete*は、デバイスが充電を完了してから経過した秒数です。このポートが充電の完了を検出しなかった場合、-1が返されます。

例

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

Port.N.TimeCharging_sec

このUSBポートが接続されたデバイスの充電を開始してからの時間 (秒)。USBポートが接続されたデバイスの充電を開始していない場合は、0が返されます。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"Port.N.TimeCharging_sec"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"result":充電時間
}
```

*充電時間*は、ポートがデバイスを充電してから経過した時間 (秒単位) です。

例

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

Port.N.USBStrings

このUSBポートの「Manufacturer」、「Description」、および「SerialNumber」の値を含むディクショナリ。

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.USBStrings」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値：

```
{
"SerialNumber": "usb-serial",
"説明": "説明",
"メーカー": "デバイス メーカー"
}
```

出力	説明
usb-serial	USB serial number
説明	ハードウェアの名前
デバイスメーカー	デバイスの製造元の名前

例

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

Port.N.VID

このUSBポートに接続されているUSBデバイスのベンダーID。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.VID」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: ベンダー ID
}
```

Vendor-ID is the vendor ID, VID.整数として表示されます。検出できなかった場合は0 (ゼロ) が返されます。

例

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

Port.N.Voltage_10mV

ポートに供給される電圧は10mV です。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Port.N.Voltage_10mV」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 電圧-10mV
}
```

*voltage-10mV*は、10mV 単位の整数としてポートに供給される電圧です。

例

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

PortInfo.N

指定されたポートのすべてのポート情報を取得します。このポートで使用可能なすべてのキーと値をディクショナリとして。

構文：参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「PortInfo.N」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値：

```
{
"結果":{
「ポート」:ポート番号、
"電流_mA":電流_mA ,
"LocationID":ロケーション ID ,
"フラグ":フラグ,
"USBVersion": USB-バージョン,
「VID」:ベンダーID 、
"PID":製品 ID 、
"メーカー": "デバイスメーカー",
"説明": "説明",
"SerialNumber": "usb-serial",
```

```
"USBツリー": {
"LocationID":ロケーション ID ,
"USBVersion": USB-バージョン,
"USBPower": {
"状態": "電源状態",
「説明」:「電源の説明」
      },
"USB速度": {
"速度": " USB速度",
"説明": " USB 説明"
"容量": {
"速度": "可能な速度",
「説明」: 「有能な説明」
        }
      },
"エンドポイント": {
「アクティブ」: アクティブエンドポイント、
「メモリー」: エンドポイントメモリー
      }
    }
  }
}
```

出力	説明
ポート番号	ハブのポート番号
電流_mA	デバイスに供給される電流 (mA (ミリアンペア))
ロケーション ID	のLocation IDs整数として

出力	説明
フラグ	港の旗については、を参照してください。Port.N.Flags
USBバージョン	ハブへの接続の USB バージョン番号。フォーマット「N.nn」
ベンダーID	デバイス ベンダー ID 番号または VID。整数として表示
製品番号	製品 ID 番号または PID。整数として表示
デバイスメーカー	デバイスの製造元の名前
説明	ハードウェアの名前
usb-serial	USB serial number
電源状態	USB Power Statesコード
電源説明	USB power turned on/off
USB速度	Maximum speed USB connection capable of
USB の説明	USB 接続の名前スーパースピード USB 5Gbps
対応速度	可能な最大データ速度デバイス
有能な説明	可能な最大データ速度デバイスの名前
アクティブエンドポイント	デバイスが使用しているエンドポイントの数
エンドポイント メモリ	エンドポイントによって使用されているメモリの量

例

```
{
    "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
                }
            }
        }
    }
```

ポート情報

すべてのポートのすべてのポート情報を取得します。ディクショナリのディクショナリとして、すべてのポートに関するすべての利用可能な情報。これらの値のほとんどは、個別にクエリできます

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「ポート情報」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値

```
{
"Port.1": {
「ポート」:ポート番号、
"電流_mA":電流_mA ,
"フラグ":フラグ,
"ProfileID":プロファイル ID ,
"TimeCharging_sec":充電時間、
"TimeCharged_sec":充電完了、
"Energy_Wh":エネルギー-Wh ,
「VID」:ベンダーID 、
"PID":製品 ID 、
"メーカー": "デバイスメーカー",
```

```
"説明": "説明",
"SerialNumber": "usb-serial",
"PhoneSerialNumber": "電話シリアル",
"PhoneIdentity": "電話名",
"IMEI": " IMEI番号",
"MacAddress": " MacAddress ",
"PhoneSoftwareVersion": " phone-OS-version "
"USBツリー": {
"USB2": {
"LocationID":ロケーション ID ,
「VID」:ベンダーID 、
"PID":製品 ID 、
"メーカー": "デバイスメーカー",
"説明": "説明",
"SerialNumber": "usb-serial",
"USBVersion": USB-バージョン,
"バッテリー": {
"DataSource": "バッテリーデータソース",
"TrustLevel": "信頼レベル",
"PairingSupported":サポート ペアリング、
"HealthPercent":バッテリーの状態、
"CurrentLevel":バッテリー電流レベル,
"CurrentTime":現在のハブ時間、
"StartingLevel":充電開始レベル,
"StartingTime":充電開始時間、
"CapacityNew":新しいバッテリー容量、
「容量」:現在のバッテリー容量、
"ChargingStatus": "充電ステータス",
```

```
        },
"PhoneSerialNumber": "電話シリアル",
"PhoneIdentity": "電話名",
"IMEI": " IMEI番号",
"MacAddress": " MacAddress ",
"PhoneSoftwareVersion": " phone-OS-version "
      }
    },
"バッテリー": {
"DataSource": "バッテリーデータソース",
"TrustLevel": "信頼レベル",
"PairingSupported":サポート　ペアリング、
"CurrentLevel":バッテリー電流レベル,
"CurrentTime":現在のハブ時間、
"StartingLevel":充電開始レベル,
"StartingTime":充電開始時間、
"CapacityNew":新しいバッテリー容量、
「容量」:現在のバッテリー容量、
"ChargingStatus": "充電ステータス",
"HealthPercent":バッテリーの状態
    }
  }
}
```

出力	説明
ポート番号	ハブのポート番号
電流_mA	モバイル デバイスに供給される電流 (mA (ミリアンペア))
フラグ	港の旗については、を参照してください。Port.N.Flags

出力	説明
プロファイル ID	プロファイル番号、または充電していない場合は 0。
充電時間	ポートがデバイスを充電してから経過した時間 (秒単位)。
充電完了	デバイスが充電を完了してからの時間 (秒)
エネルギー-Wh	電力 (ワット時) (毎秒計算)、nn の形式
ベンダーID	デバイス ベンダー ID 番号または VID。整数として表示
製品番号	製品 ID 番号または PID。整数として表示
デバイスメーカー	デバイスの製造元の名前
説明	ハードウェアの名前
usb-serial	USB serial number
電話シリアル	電話のシリアル番号
電話名	電話の名前
IMEI番号	電話の IMEI 番号
マカアドレス	モバイル デバイスに割り当てられた一意のアドレス。12 の 16 進数文字で構成される 48 ビット値です。
電話 OS バージョン	電話機の OS のバージョン番号
ロケーション ID	のLocation IDs整数として
USB バージョン	ハブへの接続の USB バージョン番号。フォーマット「N.nn」
バッテリーデータソース	デバイスのバッテリー情報のソース
信頼レベル	デバイスが信頼/ペアリングされているかどうか
サポート ペアリング	デバイスが信頼/ペアリングをサポートしているもの
バッテリーの状態	パーセンテージで表示されるバッテリーの状態

出力	説明
バッテリー電流レベル	デバイスの現在のバッテリーレベルをパーセンテージで表示
現在のハブ時間	ハブ時間 (ミリ秒単位の整数で表示)
充電開始レベル	デバイスが接続されているときのバッテリーのパーセンテージレベル
充電開始時間	ハブの充電開始時間 (ミリ秒単位の整数で表示)
新しいバッテリー容量	新しいデバイスのバッテリー容量
現在のバッテリー容量	現在のデバイスのバッテリー容量
充電状態	バッテリーの充電状態満杯
サポートペアリング	デバイスが信頼/ペアリングをサポートしているもの
バッテリーの状態	パーセンテージで表示されるバッテリーの状態

例

返される情報の抜粋された例。


```
{
    "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

特定のプロファイルが有効になっている場合に情報を取得します。ハブで利用可能なプロファイルについては、特定の製品のユーザーマニュアルを参照してください。

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「Profile.N.enabled」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値：

```
{
「結果」: プロファイル対応
}
```

プロファイル対応	説明
真実	特定のプロファイルが有効になっています
間違い	特定のプロファイルが有効になっていません

例

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

pwm_percent

ファン回転速度。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"pwm_パーセント"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: ファンパーセント
}
```

*fan-percent*はファンの速度をパーセンテージで表したもので、0 ~ 100 の数値で表示されます

例

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

再起動しました

電源投入後にシステムが再起動されたかどうかを示すフラグ。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「再起動しました」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 再起動
}
```

再起動しました	説明
真実	システムが再起動されました
間違い	再起動は発生していません。

例

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

セキュリティ武装

セキュリティは武装していますか?

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「セキュリティアームド」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: セキュリティ武装
}
```

セキュリティ武装	説明
真実	セキュリティが武装している
間違い	セキュリティは武装していません

例

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

設定

Obtain current hub Internal hub settings.

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"設定"、
真実
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"結果": {
"company_name": "メーカー名",
"product_name": "製品名",
"local_name": "ローカル名",
"attach_threshold": " attach-threshold ",
"default_profile": [デフォルト プロファイル],
"remap_ports": [ポート順],
"ports_on": [ポートオン],
"sync_chrg": [同期チャージ],
"alt_sync_chrg": [ alt-sync-charge ],
"misc_flags":Internal hub-flags,
```

```
"display_mode": "表示モード",
"charged_threshold": " charged-threshold ",
"temperature_max": "シャットダウン温度",
"よろめく": "よろめく"
  }
}
```

変数	説明
メーカー名	メーカーの定義名、デフォルトは「Cambrionix」
商品名	製品のハードウェア名
ローカル名	Local name set by the user, "-" if not set
アタッチしきい値	デバイスが接続されていることをハブが検出する電流 (mA)。「d」は工場出荷時のデフォルトが設定されていることを意味します。
デフォルトプロファイル	各ポートのデフォルト プロファイル、コンマ区切りリスト
ポートオーダー	ポートはポート番号順、カンマ区切りのリストで並べ替えます
ポートオン	各ポートがデフォルトでオンかどうか、0 はデフォルトでオフ、1 はデフォルトでオン、カンマ区切りのリスト
同期充電	各ポートの CDP* がオンかどうか、0 はオフ、1 はオン、カンマ区切りリスト
代替同期チャージ	各ポートの代替 CDP* がオンかどうか、0 はオフ、1 はオン、コンマ区切りリスト
Internal hub-flags	If any Internal hub Misc flags are active
ディスプレイモード	ログの表示モードを変更します。「d」は工場出荷時のデフォルトが設定されていることを意味します
課金しきい値	デバイスが完全に充電されていることをハブが検出する電流 (mA)。「d」は、工場出荷時のデフォルト設定を意味します。

変数	説明
シャットダウン温度	Temperature that will shutdown the hub if reached in Celsius, "d" means factory default is set
よろめく	ポートがオンになる間の遅延 (ミリ秒)。「d」は、工場出荷時のデフォルトが設定されていることを意味します。

*Charging Downstream Port (CDP) が有効になっているということは、ポートがデータの転送とデバイスの充電を同時に、単なるデータ同期よりも高い電流で実行できることを意味します。With CDP enabled the hub can supply up to 1.5 A

CDP を無効にすると、「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.

例


```
{
    "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"
    }
}
```

システムタイトル

システム識別テキスト。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「システムタイトル」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: システムタイトル
}
```

*system-title*は、ハブの完全な記述名です

例

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

温度_C

現在のPCB温度 (摂氏)。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「温度_°C」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 温度
}
```

*温度*は、10進数として測定された温度です。≤0 °Cは0を返します。測定温度が100 °C以上の場合は100が返されます。

例

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

温度_フラグ

温度エラーフラグ:

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「温度_フラグ」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 温度フラグ
}
```

温度フラグ	説明
OT	過熱イベントが発生しました。
空の	温度は許容範囲です

例

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

Temperature_Limit_Max_C

エラー フラグをトリガーする許容温度範囲の上限。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"Temperature_Limit_Max_C"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 温度制限最大
}
```

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

例

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

温度最大_C

摂氏で記録されたPCBの最高温度。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「温度Max_C」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 最高温度
}
```

*温度*は、10進数で表示される測定温度です。≤0 °Cは0を返します。測定温度 ≥100 °Cは100を返します。

例

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

合計電流_mA

すべてのUSBポートの総電流(mA)。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「総電流_mA」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 総電流
}
```

*total-current*は、10進数で表示されるmA単位の全ポートの合計電流です。

例

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

TotalPower_W

すべてのUSBポートで消費される合計電力 (ワット (W))。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"TotalPower_W"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値

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

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

例

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

TwelveVoltRail_flags

12V 電源レール エラー フラグのリスト。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「TwelveVoltRail_flags」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 12 ボルト フラグ
}
```

十二ボルト旗	説明
「紫外線」	不足電圧発生
「OV」	過電圧発生
「OV UV」	過電圧と不足電圧の両方が発生しました
""	電圧は許容可能です

例

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

TwelveVoltRail_Limit_Max_V

エラー フラグをトリガーする 12V レールの上 限 。

構 文 : 参 照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"TwelveVoltRail_Limit_Max_V"
  ]
}
```

*接続ハンドル*はConnection Handles整 数 として。

戻 り値 :

```
{
「結果」: 12 ボルト制限最大
}
```

*12 ボルト制限最大値*は、10 進 数 として表 示 されるボルト でエラーがフラグされる前 の最 大 ボルト 量 です。

例

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

TwelveVoltRail_Limit_Min_V

エラー フラグをトリガーする 12V レールの下 限 。

構 文 : 参 照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"TwelveVoltRail_Limit_Min_V"
  ]
}
```

*接続ハンドル*はConnection Handles整 数 として。

戻 り値 :

```
{
「結果」: 12 ボルト制限分
}
```

*12 ボルト制限分*は、10 進 数 として表 示 されるボルト でエラーがフラグされる前 のボルト の最 小 量 です。

例

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

TwelveVoltRail_V

現在の12V供給電圧 (ボルト (V))。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
「TwelveVoltRail_V」
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 12ボルト
}
```

*12ボルト*は、10進数で表示されるボルト単位で供給されるボルトの量です。

例

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

TwelveVoltRailMax_V

測定された最高の12V供給電圧。

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"TwelveVoltRailMax_V"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値：

```
{
「結果」: 12 ボルト最大
}
```

*12ボルト最大*は、10進数として表示されるボルトで表示されるボルトの最高量です。

例

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

TwelveVoltRailMin_V

ボルト (V) で測定された最低 12V 供給電圧。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"TwelveVoltRailMin_V"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 12 ボルト分
}
```

*12 ボルト分*は、10 進数として表示されるボルトで表示されるボルトの最小量です。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 12.31
}
```

Uptime_sec

最後のリセット以降、ハブが実行されている時間 (秒単位)。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル,
"稼働時間_秒"
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
「結果」: 稼働時間
}
```

*uptime*は、整数として表示される連続実行時間 (秒単位) です。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "result": 151304
}
```

14.3.辞書の設定

鍵	機能セット
ビープ	5V
ClearErrorFlags	ベース
クリアLCD	5V
ClearRebootFlag	ベース
FiveVoltRail.過電圧	5V
FiveVoltRail.UnderVoltage	5V
InputRail.過電圧	PD
InputRail.UnderVoltage	PD
LCDText.ROW.COL	5V
モード	ベース
Port.N.gate	ゲート
Port.N.led1	ベース
Port.N.led2	ベース
Port.N.led3	ベース
Port.N.leds	ベース
ポートNモード	ベース
Port.N.プロファイル	同期
ポート.N.RGB	ゲート
ProfileEnable.n	5V
リブート	ベース
リモコン	ベース
RGBコントロール	ゲート
セキュリティ武装	5V

鍵	機能セット
設定	ベース
温度.過熱	温度
TwelveVoltRail.過電圧	12V
TwelveVoltRail.UnderVoltage	12V

ビープ

渡されたミリ秒数だけビープ音を鳴らします。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"ビープ"、
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
期間	ビープ音に必要な時間 (ミリ秒) の整数

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "beep",
        250
    ]
}
```

ClearErrorFlags

すべてのエラー フラグをクリアする

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"ClearErrorFlags",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	エラー フラグをクリアする

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "ClearErrorFlags",
        true
    ]
}
```

クリアLCD

LCD をクリアします。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル, "ClearLCD",
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
```

```
            7654,
            "ClearLCD",
        ]
    }
```

ClearRebootFlag

再起動フラグをクリアします。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「ClearRebootFlag」、値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	再起動フラグをクリアする

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "ClearRebootFlag",
        true
    ]
}
```

FiveVoltRail.過電圧

5V 過電圧状態の動作を強制します。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"FiveVoltRail.OverVoltage"、値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	5V過電圧フラグを設定

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        ""FiveVoltRail.OverVoltage"",
        true
    ]
}
```

FiveVoltRail.UnderVoltage

電圧条件下で5Vの動作を強制します。

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"FiveVoltRail.UnderVoltage"、値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	5V低電圧フラグを設定

戻り値：

```
{
「結果」:真
}
```

エラー

APIメソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "FiveVoltRail.UnderVoltage",
        true
    ]
}
```

InputRail.過電圧

入力レール過電圧条件の動作を強制します。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「InputRail.OverVoltage」、
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	電圧入力過電圧フラグの設定

戻り値:

```
{
「結果」:真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "InputRail.OverVoltage",
        true
    ]
}
```

InputRail.UnderVoltage

電圧条件下で入力レールの動作を強制します。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル, "InputRail.UnderVoltage",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	電圧入力不足電圧フラグをセット

戻り値:

```
{
「結果」:真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "LCDText.4.5",
        "hello"
    ]
}
```

LCDText.ROW.COL

文字列を LCD の(行、列)に書き込みます。行と列はゼロベースです。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「LCDText.ROW.COL" ,
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
行	書き込みを開始したい LCD 行
コル	書き始めたい液晶コラム
価値	キーに設定したい値

価値	説明
弦	LCDに表示したい文字列

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "jsonrpc": "2.0",
    "id": 0,
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "InputRail.UnderVoltage",
        true
    ]
}
```

モード

すべての USB ポートに同じモードを設定します。各ハブでサポートされているモードの詳細については、製品のユーザー マニュアルを参照してください。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「モード」、
「値」
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
c	充電モード
s	同期および充電モード
b	バイアスモード
o	オフ

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Mode",
        "s"
    ]
}
```

Port.N.gate

指定されたゲートを開閉します。cbrx_connection_get(handle, “Gates”) を介して必要なゲートの状態を監視し、それが完了したことを確認する必要があります。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「ポート。N .gate",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
N	ポート番号
価値	キーに設定したい値

価値	説明
開いた	ゲートを開く
近い	ゲートを閉じる
止まる	現在のゲート動作を停止

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.gate",
        "open"
    ]
}
```

Port.N.led1

最初のLEDのステータスを設定します

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"ポート。N .led1",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
N	ポート番号
価値	キーに設定したい値

価値	説明
点滅パターン	0 ～ 255で、値で表されるビット パターンに従ってLEDが点滅します。

戻り値：

```
{
「結果」:真
}
```

エラー

APIメソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.led1",
        170
    ]
}
```

Port.N.led2

2 番目の LED のステータスを設定します

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「ポート。N .led2",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
N	ポート番号
価値	キーに設定したい値

価値	説明
点滅パターン	0 ~ 255 で、値で表されるビット パターンに従って LED が点滅します。

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.led2",
        170
    ]
}
```

Port.N.led3

3 番目の LED のステータスを設定します

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「ポート。N .led3",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
N	ポート番号
価値	キーに設定したい値

価値	説明
点滅パターン	0 ~ 255 で、値で表されるビット パターンに従って LED が点滅します。

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.led3",
        170
    ]
}
```

Port.N.leds

3 つの LED すべてのステータスを設定します

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「ポート。N .leds",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
N	ポート番号
価値	キーに設定したい値

価値

個々の LED 設定をシフトして OR 演算した 8 ビット値として構成される 24 ビット数値。すなわちled1 | (led2 << 8) | (led3 << 16) であるため、led1 と led2 がゼロで、led3 が 0b10101010 (10 進数の 170) の場合、結果は 10 進数で 11,141,120 になります。

ThunderSync3 では、255 が緑、65,280 が赤、16,711,680 が黄です。

ModIT では、黄色の代わりに青が使用されますが、もちろん、任意の RGB ミックスに色を混ぜることができます。

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.leds",
        11193404
    ]
}
```

ポートNモード

単一のUSBポートのモードを設定します。同期モードは、同期機能セットを実装するデバイスでのみ設定できます。バイアスモードは、5V機能セットを実装するデバイスでのみ設定できます。

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"ポート。N .mode",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
N	ポート番号
価値	キーに設定したい値

価値	説明
c	充電モード
s	同期および充電モード
b	バイアスモード
o	オフ

戻り値：

```
{
```

```
「結果」: 真

}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.Mode",
        "c"
    ]
}
```

Port.N.プロファイル

有効なプロファイルのリストを設定します。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"ポート。N .プロファイル",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
N	ポート番号
価値	キーに設定したい値

価値	説明
プロフィール	有効にするプロファイルのコンマ区切りリスト。ハブに適用可能なプロファイルの詳細については、製品のユーザーマニュアルを参照してください。

戻り値:

```
{
「結果」:真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.profiles",
        "1,2,3"
    ]
}
```

ポート.N.RGB

ModIT LED の RGB カラーを設定します。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"ポート。N .RGB",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
N	ポート番号
価値	キーに設定したい値

価値

色の値は、整数 (完全な RGBA を指定する必要があります) または文字列のいずれかです。文字列の場合、RGB、RGBA、RRGGBB、または RRGGBBAA として指定できます。HTML の色でできることとよく似ています。たとえば、赤には「FF0000」または「F00」、白には「FFFFFF」などを使用します。必要に応じて、アルファ(強度) の数字を指定します。したがって、「FFFFFF80」は半明るい白を表します。

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Port.1.RGB",
        "ff08"
    ]
}
```

ProfileEnable.n

グローバル プロファイルを有 効 または無 効 にします n

構 文 : 参 照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「 ProfileEnable.n 」
  ]
}
```

パラメータ	説 明
接続ハンドル	のConnection Handles整 数 として
n	プロファイル番 号

戻 り値 :

```
{
「結 果 」: 真
}
```

エラー

API メソッドにエラーがある場 合 、JSON-RPC Error Object返 されます。

例

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

```
        "method": "cbrx_connection_set",
        "params": [
            7654,
            "ProfileEnable.n"
        ]
    }
```

リブート

今すぐハブを再起動します。API は自動的に接続を再確立しようとしますが、更新された結果を数秒間受信することは期待できません。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「再起動」、
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	ハブを再起動します。

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Reboot",
        "true"
    ]
}
```

リモコン

ユニットコントロールの制御を有効/無効にします。これにより、LED または LCD を更新したり、パネルボタンの押下を検出したりできます。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"リモコン"、
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	手動リモート コントロール モードを有効にする
間違い	手動リモコンモードを無効にする
「オート」	API 経由で自動制御モードを有効にする

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "RemoteControl",
        "true"
    ]
}
```

RGB コントロール

ポートの ModIT RGB LED コントロールを有効/無効にします。これには、RemoteControl を有効にする必要はありません。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「RGBControl」、
    {
「ポート」: N 、
「有効にする」: 値
    }
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
N	ポート番号
価値	キーに設定したい値

価値	説明
真実	RGB LED の制御を有効にします
間違い	RGB LED の制御を無効にする

戻り値:

```
{
```

「結果」: 真

}

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "RGBControl",
        {
            "port": 8,
            "enable": true
        }
    ]
}
```

複数のポート

ポートの範囲で制御を設定したい場合は、パラメーターが変更されます。開始するポートの「開始」値と終了するポートの「終了」値の 2 つの値を入力する必要があります。

構文: 参照 Call Structure

{

"メソッド": "cbrx_connection_set",

"パラメータ": [

接続ハンドル、「RGBControl」、

{ "開始": *N* 、"終了": *N* 、"有効": *値*

}

]

}

パラメータ	説明
接続ハンドル	のConnection Handles整数として
N	ポート番号
価値	キーに設定したい値

価値	説明
真実	RGB LED の制御を有効にします
間違い	RGB LED の制御を無効にする

戻り値：

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "RGBControl",
        {
            "start": 1,
            "end": 8,
            "enable": true
        }
    ]
}
```

セキュリティ武装

セキュリティ機能を有効/無効にします。セキュリティが有効になっている場合、デバイスをポートから取り外すと、アラーム(取り付けられている場合)が鳴り、ライトが点滅します(取り付けられている場合)。

構文:参照Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"SecurityArmed",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	セキュリティを有効にする
間違い	セキュリティーを無効にする

戻り値:

```
{
「結果」:真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "SecurityArmed",
        "true"
    ]
}
```

設定

接続されたハブでコマンド ライン インターフェイス操作を実行し、完全な結果を返します。これにより、API サービスを停止することなく、ハブのコマンド ラインでコマンドを直接実行できます。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.

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"設定"、
"settings_unlock\nCommand"
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
コマンド	The Command you wish to send to the hub For all CLI commands see the CLI Documentation https://www.cambrionix.com/cambrionix-cli

戻り値:

```
{
"jsonrpc": "2.0",
"ID": 0,
"result": "Unlocked \nSetting updated"
```

```
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

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."
}
```

例

```
{
  "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"
  ]
}
```

温度.過熱

過熱状態の動作を強制します。

構文: 参照Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル,
"Temperature.OverTemperature",
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	過熱フラグを設定する

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "Temperature.OverTemperature",
        "true"
    ]
}
```

TwelveVoltRail.過電圧

12V 過電圧状態の動作を強制します。TwelveVoltRail は InputRail と同じです。

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「TwelveVoltRail.OverVoltage」、
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	12V過電圧フラグをセット

戻り値:

```
{
「結果」: 真
}
```

エラー

API メソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "TwelveVoltRail.OverVoltage",
        "true"
    ]
}
```

TwelveVoltRail.UnderVoltage

電圧条件下で12Vの動作を強制します。

TwelveVoltRailはInputRailと同じです。

構文：参照Call Structure

```
{
"メソッド": "cbrx_connection_set",
"パラメータ": [
接続ハンドル、「TwelveVoltRail.UnderVoltage」、
価値
  ]
}
```

パラメータ	説明
接続ハンドル	のConnection Handles整数として
価値	キーに設定したい値

価値	説明
真実	12V不足電圧フラグを設定する

戻り値：

```
{
「結果」:真
}
```

エラー

APIメソッドにエラーがある場合、JSON-RPC Error Object返されます。

例

```
{
    "id": 0,
    "jsonrpc": "2.0",
    "method": "cbrx_connection_set",
    "params": [
        7654,
        "TwelveVoltRail.UnderVoltage",
        "true"
    ]
}
```

14.4.非推奨の辞書

これらの辞書は、下位互換性をサポートするためだけに存在するものであり、使用しないでください。これらのキー値は、将来のバージョンで削除される可能性があります。

API 呼び出し	説明
設定	Obtain current hub Internal hub settings.

設定

Obtain current hub Internal hub settings, returns as text.

構文: 参照 Call Structure

```
{
"メソッド": "cbrx_connection_get",
"パラメータ": [
接続ハンドル、「設定」、
  ]
}
```

*接続ハンドル*はConnection Handles整数として。

戻り値:

```
{
"jsonrpc": "2.0",
"ID": 5,
"結果": [
"現在のメモリ設定:",
    "",
"settings_set company_nameメーカー名",
"settings_set product_name製品名",
"settings_setローカル名　ローカル名",
"settings_set attach_threshold attach-threshold ",
"settings_set default_profileデフォルトプロファイル",
"settings_set remap_portsポート順",
"settings_set ports_onポートオン",
"settings_set sync_chrg同期チャージ",
```

```
"settings_set alt_sync_chrg alt-sync-charge ",
"settings_set misc_flags Internal hub-flags",
"settings_set display_modeディスプレイモード",
"settings_set charge_threshold課金しきい値",
"settings_set temperature_maxシャットダウン温度",
"settings_set stagger stagger "
  ]
}
```

変数	説明
メーカー名	メーカーの定義名、デフォルトは「Cambrionix」
商品名	製品のハードウェア名
ローカル名	Local name set by the user, "-" if not set
アタッチしきい値	デバイスが接続されていることをハブが検出する電流 (mA)。「d」は工場出荷時のデフォルトが設定されていることを意味します。
デフォルトプロファイル	各ポートのデフォルト プロファイル
ポートオーダー	ポートはポート番号順に並べます
ポートオン	各ポートがデフォルトでオンかどうか、0 はデフォルトでオフ、1 はデフォルトでオン
同期充電	各ポートの CDP* がオンかどうか、0 はオフ、1 はオン、カンマ区切りリスト
代替同期チャージ	各ポートの代替 CDP* がオンかどうか、0 がオフ、1 がオン
Internal hub-flags	If any Internal hub Misc flags are active
ディスプレイモード	ログの表示モードを変更します。「d」は工場出荷時のデフォルトが設定されていることを意味します

変数	説明
課金しきい値	デバイスが完全に充電されていることをハブが検出する電流 (mA)。「d」は、工場出荷時のデフォルト設定を意味します。
シャットダウン温度	Temperature that will shutdown the hub if reached in Celsius, "d" means factory default is set
よろめく	ポートがオンになる間の遅延 (ミリ秒)。「d」は、工場出荷時のデフォルトが設定されていることを意味します。

*Charging Downstream Port (CDP) が有効になっているということは、ポートがデータの転送とデバイスの充電を同時に、単なるデータ同期よりも高い電流で実行できることを意味します。With CDP enabled the hub can supply up to 1.5 A

CDP を無効にすると、「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.

例


```
{
    "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.ソケット接続

cbrxapi モジュールを提供する Python ラッパーを使用すると、API が呼び出されるたびにソケットが作成されます。このソケットは、閉じられる前にコマンドを送信し、応答を受信するために使用されます。

独自のプログラムを作成している場合は、API との通信の開始時に単一のソケットを作成し、API の使用を停止するまでこのソケットを開いたままにしておくことを検討してください。API との通信の存続期間中ソケットを開いたままにしておくと、システムの負荷が軽減され、API との通信サイクルが短くなります。

API への独自のソケット接続を管理することを選択した場合は、最後のコマンドからの応答を受け取る前にソケットを閉じないことが重要です。応答の受信を待たずにソケットを閉じると、要求された操作が完了しない可能性があります。これは、set および close 操作で特に重要です。

16.LED の制御

API は、製品の LED を制御できます。デフォルトでは、これらの LED は製品によって自動的に制御され、ポートの状態を示します。

LED を API で制御するには、この自動制御を無効にする必要があります。これを行うには、「RemoteControl」キーを「True」に設定します。LED の制御を自動制御に戻したい場合は、「RemoteControl」を「False」に設定します。見るcbrx_connection_setこのメソッドの使用方法の詳細については。

LED の制御は、連続的に繰り返されるパターンとしてバイナリで解釈される 8 ビット値を提供することによって実現されます。したがって、値を 11110000b に設定すると、LED はゆっくりと点滅します。LED は「1」の場所で点灯し、「0」の場所で消灯します。または、値 10101010b を設定すると、LED が速く点滅します。パターンは対称である必要はないので、10010000b は、サイクルが繰り返される前に、2 つの短い閃光を生成し、短い休止時間を長くします。

RemoteControl が False のときに LED に設定された値は上書きされるため、効果がありません。

True の代わりに「auto」という特別な引数を使用すると、ハブは、そのポートに接続されているデバイスが取り外されたときに、ユーザーが設定した LED パターンをオーバーライドできます。

17.バッテリー情報

接続されたデバイスのバッテリー情報を取得できます。Android Debug Bridge (ADB) を使用する Android™ デバイスの場合、および iOS デバイスの場合は libimobile の組み込みビルド。

ADB を使用して、いくつかの条件が満たされていれば、Android™ デバイスのバッテリーレベルを照会できます。

- Android プラットフォームツールがインストールされます。これらはここからダウンロードできます。
- ADB バイナリがパスに含まれているか、そのパスが cbrx_config_set を介して API に提供されます。
- デバイスで USB デバッグが有効になっています。
- 電話で必要な場合は、電話からコンピューターを信頼しています。

Android™ デバイスでデバッグモードを有効にする方法の詳細については、このページを参照してください。必要な唯一のオプションは、開発者モードと USB デバッグを有効にすることです。

```
# 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.
```

パスなしでadbを見つける

パスを設定する代わりに、これらのプログラムの場所を API に伝えることができます。

```
{
  "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	説明
"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

コード	価値	説明
CBRXAPI_ERRORCODE_IDNOTFOUND	-10001	IDが見つかりません。渡されたユニット ID がハブを表していないか、ディスカバリーが最後に実行されてから切断されています。1分後に未使用のハンドルを閉じる内部タイムアウトがあることに注意してください。
CBRXAPI_ERRORCODE_NOHANDLINGTHREAD	-10002	スレッドの処理を開始できません。このエラーは、2.1 以降には該当しません。
CBRXAPI_ERRORCODE_KEYNOTFOUND	-10003	キーが見つかりません。渡されたキーが見つかりません。つづりが間違っているか、このユニットの辞書に存在しない可能性があります。
CBRXAPI_ERRORCODE_ERRORSETTINGVALUE	-10004	値を設定できませんでした。(キーと値) のペアは受け入れられませんでした。これは、キーが存在しないか、スペルが間違っているか、値が間違った型であるか、渡された値が無効であるか範囲外であることを意味する可能性があります。
CBRXAPI_ERRORCODE_INVALIDHANDLE	-10005	ハンドルが無効です。関数に渡されたハンドルが無効であるか、無効になっています。これは、誤った値を渡すか、ハンドルが既に閉じられている (つまり、cbrx_closeandlock が呼び出されることによって)、またはユニットがコンピュータから切断されました。
CBRXAPI_ERRORCODE_TIMEOUT	-10006	通信タイムアウト。ハブに対する操作の完了に時間がかかりすぎました。接続が切断されたか、単に応答が遅い可能性があります。操作を再試行する価値があります。
CBRXAPI_ERRORCODE_DROPPED	-10007	リモートへのソケット接続が切断されました。
CBRXAPI_ERRORCODE_METHOD_REMOVED	-10008	メソッドは削除されました。
CBRXAPI_ERRORCODE_AGAIN	-10009	システムの準備ができていません。再試行。これは、API 関数への非常に迅速な呼び出しが原因である可能性があり、システムはサービスを開始するのに十分なほど進行していません。
CBRXAPI_ERRORCODE_FIRMWARE_UPDATE	-10010	ファームウェアの更新中にエラーが発生しました。

コード	価値	説明
CBRXAPI_ERRORCODE_FIRMWARE_FILE	-10011	ファームウェア ファイル エラー。これは通常、ファイル形式のエラーが原因です。
CBRXAPI_ERRORCODE_DEVICE_NOT_FOUND	-10012	デバイスが見つかりません。
CBRXAPI_ERRORCODE_CONNECTION_ERROR	-10014	ハブへのシリアル ポート接続を開くことができませんでした。
CBRXAPI_ERRORCODE_HUB_NOT_FOUND	-10013	ハブが見つかりません。

商標や登録商標などの保護された名称と記号の使用

このマニュアルでは、Cambrionix とは一切関係のない第三者企業の商標、登録商標、その他の保護された名前やシンボルを参照している場合があります。これらの参照は説明目的のみであり、Cambrionix による製品またはサービスの推奨、またはこのマニュアルが適用される製品の当該サードパーティ企業による推奨を表すものではありません。

Cambrionixは、このマニュアルおよび関連ドキュメントに含まれるすべての商標、登録商標、サービスマークなどの保護された名称や記号が、各々の所有者に帰属することをここに認めます

「Mac®およびmacOS®は、米国およびその他の国と地域で登録されたApple Inc.の商標です。」

「Intel®およびIntelのロゴは、Intel Corporationまたはその子会社の商標です。」

「Thunderbolt™ および Thunderbolt ロゴは、Intel Corporation またはその子会社の商標です。」

「Android™はGoogle LLCの商標です」

「Chromebook™はGoogle LLCの商標です。」

「iOS™は、米国およびその他の国におけるApple Inc.の商標または登録商標であり、

ライセンスに基づいて使用されています。」

「Linux®は、米国およびその他の国におけるLinus Torvaldsの登録商標です。」

「Microsoft™およびMicrosoft Windows™は、Microsoftグループ企業の商標です。」

「Cambrionix® およびロゴは Cambrionix Limited の商標です。」

記載されているすべての商標および登録商標は、それぞれの所有者の財産として認められ、尊重されます。

保護情報に関する重要なお知らせ

Cambrionix テクノロジーの特定のコンポーネントは、Cambrionix の保護された知的財産 (IP) とみなされることにご注意ください。具体的には:

- ソースコード: 当社のソフトウェアのソースコードは独自のものであり、提供することはできません。
- 独自の方法: 当社の独自の方法の詳細な説明と実装も保護されています。

したがって、ソースコードやその他の保護された情報へのアクセス要求は丁重にお断りさせていただきます。ご理解とご協力をよろしくお願いいたします。

カンブリオニクスの特許

タイトル	リンク	出願番号	助成金番号
同期および充電ポート	GB2489429	1105081.2	2489429
カンブリオニクス	UK00002646615	2646615	00002646615
CAMBRIONIX 非常にインテリジェント...	UK00002646617	2646617	00002646617

ライセンス

の用法Cambrionix Hub API Cambrionix Connect SaaS 条件の対象となります。このドキュメントは、次のリンクを使用してダウンロードおよび表示できます。

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

の用法Cambrionix Hub API Cambrionix ライセンス契約の対象となります。このドキュメントは、次のリンクを使用してダウンロードおよび表示できます。

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

Cambrionix
CHARGE • CONNECT • MANAGE

カンブリオニクス株式会社
The Maurice Wilkes Building
Cowley Road
Cambridge CB4 0DS
United Kingdom

+44 (0) 1223 755520
https://www.cambrionix.com

Cambrionix Ltd は、イングランドとウェールズで登録された会社です。
会社番号 06210854で