

ログイン
取引アカウント 決済アカウント 貸仮想通貨アカウント
新規登録
取引アカウント 決済アカウント 貸仮想通貨アカウント
総資産
ヘッダーに表示
総資産
¥

残高

利用中

- 取引所 API 概要 認証 ライブラリ ページネーション
- Public API ティッカー 全取引履歴 板情報 レート取得 販売レート取得
- Private API 注文
  - 新規注文 未決済の注文一覧 注文のキャンセル 注文のキャンセルステータス 取引履歴 取引履歴(ページネーション)
- アカウント
  - 残高 ビットコインの送金 ビットコインの送金履歴 ビットコインの受け取り履歴 アカウント情報
- 日本円出金
  - 銀行口座一覧 銀行口座の登録 銀行口座の削除 出金履歴 出金申請の作成 出金申請のキャンセル
- WebSocket API 概要 パブリックチャンネル
  - 取引履歴 板情報

取引所 API 概要

取引所APIでは、大きく分けて2つのAPIが利用できます。認証の必要ない Public API と必要のある Private API です。

Public API では取引所の注文状況や公開されている取引の履歴、板情報を参照することができます。

Private API では取引所での新規注文やそのキャンセル、自分の残高などを確認することができます。

リクエスト先

```
https://coincheck.com
```

認証

Private API では認証が必要となります。ここではその認証方法について説明します。

API Key の作成

まずは、API Keyを作成しなければなりません。APIキーから作成することができます。

ここで生成される、アクセスキー、シークレットアクセスキーは外部に公開してはなりません。

この作成されたキーを使ってリクエストを作成します。

パーミッションの設定

なお、キーを生成する際、機能ごとにパーミッションを設定することができます。

また、任意のIPアドレスに使用を許可するよう設定することも可能です。



APIパーミッション設定のイメージ

リクエストの作成

認証が必要なリクエストでは、以下の情報を **HTTP Header** に含めてリクエストする必要があります。

- ACCESS-KEY APIキー で作成したアクセスキー
- ACCESS-NONCE 毎リクエストごとに増加する必要のある正の整数。通常はUNIXタイムスタンプを用います。最大値は 9223372036854775807 です。APIキーごとに管理されます。
- ACCESS-SIGNATURE 後述するSIGNATURE

SIGNATUREの作成

SIGNATUREは、**ACCESS-NONCE**, **リクエスト先URL**, **リクエストのボディ** を全て文字列にし連結したものを、HMAC-SHA256 hash形式でシークレットキーを使って署名した結果です。

Ruby

```
require "openssl"

nonce = Time.now.to_i.to_s
url = "https://coincheck.com/api/accounts/balance"
body = "hoge=foo"
message = nonce + url + body
secret = "API_SECRET"
OpenSSL::HMAC.hexdigest(OpenSSL::Digest.new("sha256"), secret, message)
# => "3919705fea4b0cd073b9c6e01e139f3b056782c57c5cffd5aea6d8c4ac98bef7"
```

PHP

```
$strUrl = "https://coincheck.com/api/accounts/balance";
$intNonce = time();
$arrQuery = array("hoge" => "foo");
$strAccessSecret = "API_SECRET";
$strMessage = $intNonce . $strUrl . http_build_query($arrQuery);
$strSignature = hash_hmac("sha256", $strMessage, $strAccessSecret);
# => "3bc1f33d802056c61ba8c8108f6ffb7527bcd184461a3ea0fed3cee0a22ae15d"
```

リクエストのサンプル

Ruby

```
require 'net/http'
require 'uri'
require 'openssl'

key = "API_KEY"
secret = "API_SECRET"
uri = URI.parse "https://coincheck.com/api/accounts/balance"
nonce = Time.now.to_i.to_s
message = nonce + uri.to_s
signature = OpenSSL::HMAC.hexdigest(OpenSSL::Digest.new("sha256"), secret, message)
headers = {
  "ACCESS-KEY" => key,
  "ACCESS-NONCE" => nonce,
  "ACCESS-SIGNATURE" => signature
}

https = Net::HTTP.new(uri.host, uri.port)
https.use_ssl = true
response = https.start {
  https.get(uri.request_uri, headers)
}

puts response.body
```

Java

```
import com.google.api.client.http.*;
import com.google.api.client.http.apache.ApacheHttpTransport;
import com.google.api.client.json.jackson2.JacksonFactory;
import org.apache.commons.codec.binary.Hex;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.IOException;
```

```
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.HashMap;
import java.util.Map;

public class CoincheckApi {
    private String apiKey;
    private String apiSecret;

    public static void main(String[] args) {
        String key = "API_KEY";
        String secret = "API_SECRET";
        CoincheckApi api = new CoincheckApi(key, secret);
        System.out.println(api.getTicker());
        System.out.println(api.getBalance());
    }

    public CoincheckApi(String apiKey, String apiSecret) {
        this.apiKey = apiKey;
        this.apiSecret = apiSecret;
    }

    public String getTicker() {
        String url = "https://coincheck.com/api/accounts/ticker";
        String jsonString = requestByUrlWithHeader(url, createHeader(url));
        return jsonString;
    }

    public String getBalance() {
        String url = "https://coincheck.com/api/accounts/balance";
        String jsonString = requestByUrlWithHeader(url, createHeader(url));
        return jsonString;
    }

    private Map createHeader(String url) {
        Map map = new HashMap();
        String nonce = createNonce();
        map.put("ACCESS-KEY", apiKey);
        map.put("ACCESS-NONCE", nonce);
        map.put("ACCESS-SIGNATURE", createSignature(apiSecret, url, nonce));
        return map;
    }

    private String createSignature(String apiSecret, String url, String nonce) {
        String message = nonce + url;
        return HMAC_SHA256Encode(apiSecret, message);
    }

    private String createNonce() {
        long currentUnixTime = System.currentTimeMillis() / 1000L;
        String nonce = String.valueOf(currentUnixTime);
        return nonce;
    }

    private String requestByUrlWithHeader(String url, final Map headers){
        ApacheHttpTransport transport = new ApacheHttpTransport();
        HttpRequestFactory factory = transport.createRequestFactory(new HttpRequestInitializer() {
            public void initialize(final HttpRequest request) throws IOException {
                request.setConnectTimeout(0);
                request.setReadTimeout(0);
                request.setParser(new JacksonFactory().createJsonObjectParser());
                final HttpHeaders httpHeaders = new HttpHeaders();
                for (Map.Entry e : headers.entrySet()) {
                    httpHeaders.set(e.getKey(), e.getValue());
                }
                request.setHeaders(httpHeaders);
            }
        });
        String jsonString;
```

```
        try {
            HttpRequest request = factory.buildGetRequest(new GenericUrl(url));
            HttpResponse response = request.execute();
            jsonString = response.parseAsString();
        } catch (IOException e) {
            e.printStackTrace();
            jsonString = null;
        }
        return jsonString;
    }


    public static String HMAC_SHA256Encode(String secretKey, String message) {

        SecretKeySpec keySpec = new SecretKeySpec(
                secretKey.getBytes(),
                "hmacSHA256");

        Mac mac = null;
        try {
            mac = Mac.getInstance("hmacSHA256");
            mac.init(keySpec);
        } catch (NoSuchAlgorithmException e) {
            // can't recover
            throw new RuntimeException(e);
        } catch (InvalidKeyException e) {
            // can't recover
            throw new RuntimeException(e);
        }
        byte[] rawHmac = mac.doFinal(message.getBytes());
        return Hex.encodeHexString(rawHmac);
    }
}
```

ライブラリ

coincheckのAPIを利用する際に役立つ各プログラミング言語向けのライブラリを開発、配布しています。

Ruby

コマンドラインより、以下のコマンドを実行するか、

```
$ gem install ruby_coincheck_client
```

Bundler環境下でGemfileに以下の通りに記述して利用できます。

```
gem 'ruby_coincheck_client'
```

また、gemファイルはGitHubからダウンロードできます。

coincheckjp/ruby_coincheck_client

PHP

composer.json にパッケージを定義します。

```
  {
    "require": {
```

```
        "coincheck/coincheck": "1.0.0"
    }
}
```

installします。

```
$ composer install
```

また、パッケージはGitHubからダウンロードできます。

coincheckjp/coincheck-php

Scala

```
val key = "YOUR-KEY-HERE"
val secret = "YOUR-SECRET-HERE"

import net.pocketengineer.coincheckscala.CoinCheckApiService
import net.pocketengineer.coincheckscala.model._

val cc = new CoinCheckApiService(key, secret)
// Ticker
val ticker: Ticker = cc.getTicker

// OrderBook
val orderBook: OrderBook = cc.getOrderBook

// Account Info
val balance: Balance = cc.getAccountInfo

// Trade
val orderId: Long = cc.tradeBtc(28400, 0.01F, BUY)

// OpenOrder
val openOrders: List[OpenOrder] = cc.getOpenOrder

// Cancel Order
val isSuccess: Boolean = cc.cancelOrder(ORDER_ID)
```

Build

```
sbt assembly
```

dsaki/CoinCheckScala

ページネーション

coincheckの一部APIではページネーションにてデータを分割して取得することが可能です。

PARAMETERS

- limit 1ページあたりの取得件数を指定できます。
- order "desc", "asc" を指定できます。
- starting_after IDを指定すると絞り込みの開始位置を設定できます。
- ending_before IDを指定すると絞り込みの終了位置を設定できます。

```
{
  "success": true,
  "pagination": {
    "limit": 1,
    "order": "desc",
    "starting_after": null,
    "ending_before": null
  },
  "data": [
    {
      "id": 10,
      "pair": "btc_jpy",
      "status": "open",
      "created_at": "2015-12-02T05:27:53.000Z",
      "closed_at": null,
      "open_rate": "43553.0",
      "closed_rate": null,
      "amount": "1.51347797",
      "all_amount": "1.51045705",
      "side": "sell",
      "pl": "-8490.81029287",
      "new_order": {
        "id": 23104033,
        "side": "sell",
        "rate": null,
        "amount": null,
        "pending_amount": "0",
        "status": "complete",
        "created_at": "2015-12-02T05:27:52.000Z"
      },
      "close_orders": [
        {
          "id": 23755132,
          "side": "buy",
          "rate": "10000.0",
          "amount": "1.0",
          "pending_amount": "0.0",
          "status": "cancel",
          "created_at": "2015-12-05T05:03:56.000Z"
        }
      ]
    }
  ]
}
```


Public API

取引所の注文状況や公開されている取引の履歴、板情報を参照することができます。

ティッカー

各種最新情報を簡易に取得することができます。

HTTP REQUEST

```
GET /api/ticker
```


```
{
  "last": 27390,
  "bid": 26900,
  "ask": 27390,
```

```
  "high": 27659,
  "low": 26400,
  "volume": "50.29627103",
  "timestamp": 1423377841
}
```

RESPONSE ITEMS

- last 最後の取引の価格
- bid 現在の買い注文の最高価格
- ask 現在の売り注文の最安価格
- high 24時間での最高取引価格
- low 24時間での最安取引価格
- volume 24時間での取引量
- timestamp 現在の時刻

全取引履歴

最新の取引履歴を取得できます。

HTTP REQUEST

```
GET /api/trades
```

PARAMETERS

- *pair 取引ペア。現在は "btc_jpy" "etc_jpy" "fct_jpy" "mona_jpy" が利用可能です。

```
{
  "success": true,
  "pagination": {
    "limit": 1,
    "order": "desc",
    "starting_after": null,
    "ending_before": null
  },
  "data": [
    {
      "id": 82,
      "amount": "0.28391",
      "rate": 35400,
      "pair": "btc_jpy",
      "order_type": "sell",
      "created_at": "2015-01-10T05:55:38.000Z"
    },
    {
      "id": 81,
      "amount": "0.1",
      "rate": 36120,
      "pair": "btc_jpy",
      "order_type": "buy",
      "created_at": "2015-01-09T15:25:13.000Z"
    }
  ]
}
```

板情報

板情報を取得できます。

HTTP REQUEST

```
GET /api/order_books
```

```
{
  "asks": [
    [
      27330,
      "2.25"
    ],
    [
      27340,
      "0.45"
    ]
  ],
  "bids": [
    [
      27240,
      "1.1543"
    ],
    [
      26800,
      "1.2226"
    ]
  ]
}
```

RESPONSE ITEMS

- asks 売り注文の情報
- bids 買い注文の情報

レート取得

取引所の注文を元にレートを算出します。

HTTP REQUEST

```
GET /api/exchange/orders/rate
```

PARAMETERS

- *order_type 注文のタイプ（ "sell" or "buy" ）
- *pair 取引ペア。現在は "btc_jpy" "etc_jpy" "fct_jpy" "mona_jpy"が利用可能です。
- amount 注文での量。（例）0.1
- price 注文での金額。（例）28000
- ※ priceまたはamountのいずれかをパラメータとして指定する必要があります。

```
{
  "success": true,
  "rate": 60000,
  "price": 60000,
  "amount": 1
}
```

RESPONSE ITEMS

- rate 注文のレート
- price 注文の金額
- amount 注文の量

販売レート取得

販売所のレートを取得します。

HTTP REQUEST

```
GET /api/rate/[pair]
```

PARAMETERS

- *pair 通貨ペア (例 "btc_jpy")

```
{
  "rate": "60000"
}
```

RESPONSE ITEMS

- rate レート

Private API

取引所での新規注文やそのキャンセル、自分の残高などを確認することができます。

注文

取引所での注文に関するAPIを利用できます。

新規注文

取引所に新規注文を発行します。

たとえば、10 BTC を 30000 JPY/BTC で買いたい場合は、以下のように指定します。
rate: 30000, amount: 10, order_type: "buy", pair: "btc_jpy"

買い注文の場合、指定した価格よりも低い価格での売り注文が既に存在していれば、その売り注文を安い順に決済します。売り注文が足りなかった場合は未決済の注文として残ります。売り注文の場合も同様です。

注文方法について

order_type は全部で4つあります。

- "buy" 指値注文 現物取引 買い
- "sell" 指値注文 現物取引 売り
- "market_buy" 成行注文 現物取引 買い
- "market_sell" 成行注文 現物取引 売り

"buy"

指値注文 現物取引 買い

- *rate 注文のレート。（例）28000
- *amount 注文での量。（例）0.1

"sell"

指値注文 現物取引 売り

- *rate 注文のレート。（例）28000
- *amount 注文での量。（例）0.1

"market_buy"

成行注文 現物取引 買い

買いの成行注文ではBTCではなく利用するJPYの数量を送信する必要があります。

- *market_buy_amount 成行買で利用する日本円の金額。（例）10000

"market_sell"

成行注文 現物取引 売り

- *amount 注文での量。（例）0.1

HTTP REQUEST

```
POST /api/exchange/orders
```

PARAMETERS

- *pair 取引ペア。現在は "btc_jpy" "etc_jpy" "fct_jpy" "mona_jpy" が利用可能です。
- *order_type 注文方法
- rate 注文のレート。（例）28000
- amount 注文での量。（例）0.1
- market_buy_amount 成行買で利用する日本円の金額。（例）10000
- stop_loss_rate 逆指値レート（逆指値とは？）

```
{
  "success": true,
```

```
  "id": 12345,
  "rate": "30010.0",
  "amount": "1.3",
  "order_type": "sell",
  "stop_loss_rate": null,
  "pair": "btc_jpy",
  "created_at": "2015-01-10T05:55:38.000Z"
}
```

RESPONSE ITEMS

- id 新規注文のID
- rate 注文のレート
- amount 注文の量
- order_type 注文方法
- stop_loss_rate 逆指値レート
- pair 取引ペア
- created_at 注文の作成日時

未決済の注文一覧

アカウントの未決済の注文を一覧で表示します。

HTTP REQUEST

```
GET /api/exchange/orders/opens
```

```
{
  "success": true,
  "orders": [
    {
      "id": 202835,
      "order_type": "buy",
      "rate": 26890,
      "pair": "btc_jpy",
      "pending_amount": "0.5527",
      "pending_market_buy_amount": null,
      "stop_loss_rate": null,
      "created_at": "2015-01-10T05:55:38.000Z"
    },
    {
      "id": 202836,
      "order_type": "sell",
      "rate": 26990,
      "pair": "btc_jpy",
      "pending_amount": "0.77",
      "pending_market_buy_amount": null,
      "stop_loss_rate": null,
      "created_at": "2015-01-10T05:55:38.000Z"
    },
    {
      "id": 38632107,
      "order_type": "buy",
      "rate": null,
      "pair": "btc_jpy",
      "pending_amount": null,
      "pending_market_buy_amount": "10000.0",
      "stop_loss_rate": "50000.0",
      "created_at": "2016-02-23T12:14:50.000Z"
    }
```

```
  ]
}
```

RESPONSE ITEMS

- id 注文のID (新規注文でのIDと同一です)
- rate 注文のレート (null の場合は成り行き注文です)
- pending_amount 注文の未決済の量
- pending_market_buy_amount 注文の未決済の量 (現物成行買いの場合のみ)
- order_type 注文のタイプ ("sell" or "buy")
- stop_loss_rate 逆指値レート
- pair 取引ペア
- created_at 注文の作成日時

注文のキャンセル

新規注文または未決済の注文一覧のIDを指定してキャンセルすることができます。

HTTP REQUEST

```
DELETE /api/exchange/orders/[id]
```

PARAMETERS

- *id 新規注文または未決済の注文一覧のID

```
{
  "success": true,
  "id": 12345
}
```

RESPONSE ITEMS

- id キャンセルした注文のID

注文のキャンセルステータス

オーダーのキャンセル処理状況を参照出来ます。

HTTP REQUEST

```
GET /api/exchange/orders/cancel_status?id=[id]
```

PARAMETERS

- *id 新規注文または未決済の注文一覧のID

```
{
  "success": true,
  "id": 12345,
  "cancel": true,
```

```
  "created_at": "2020-07-29T17:09:33.000Z"
}
```

RESPONSE ITEMS

- id 注文のID
- cancel キャンセル済みか (true or false)
- created_at 注文の作成日時

取引履歴

自分の最近の取引履歴を参照できます。

HTTP REQUEST

```
GET /api/exchange/orders/transactions
```

```
{
  "success": true,
  "transactions": [
    {
      "id": 38,
      "order_id": 49,
      "created_at": "2015-11-18T07:02:21.000Z",
      "funds": {
        "btc": "0.1",
        "jpy": "-4096.135"
      },
      "pair": "btc_jpy",
      "rate": "40900.0",
      "fee_currency": "JPY",
      "fee": "6.135",
      "liquidity": "T",
      "side": "buy"
    },
    {
      "id": 37,
      "order_id": 48,
      "created_at": "2015-11-18T07:02:21.000Z",
      "funds": {
        "btc": "-0.1",
        "jpy": "4094.09"
      },
      "pair": "btc_jpy",
      "rate": "40900.0",
      "fee_currency": "JPY",
      "fee": "-4.09",
      "liquidity": "M",
      "side": "sell"
    }
  ]
}
```

RESPONSE ITEMS

- id ID
- order_id 注文のID
- created_at 取引が行われた時間
- funds 各残高の増減分

- pair 取引ペア
- rate 約定価格
- fee_currency 手数料の通貨
- fee 発生した手数料
- liquidity "T" (Taker) or "M" (Maker)
- side "sell" or "buy"

取引履歴 (ページネーション)

自分の最近の取引履歴を参照できます。

HTTP REQUEST

```
GET /api/exchange/orders/transactions_pagination
```


```
{
  "success": true,
  "pagination": {
    "limit": 1,
    "order": "desc",
    "starting_after": null,
    "ending_before": null
  },
  "data": [
    {
      "id": 38,
      "order_id": 49,
      "created_at": "2015-11-18T07:02:21.000Z",
      "funds": {
        "btc": "0.1",
        "jpy": "-4096.135"
      },
      "pair": "btc_jpy",
      "rate": "40900.0",
      "fee_currency": "JPY",
      "fee": "6.135",
      "liquidity": "T",
      "side": "buy"
    },
    {
      "id": 37,
      "order_id": 48,
      "created_at": "2015-11-18T07:02:21.000Z",
      "funds": {
        "btc": "-0.1",
        "jpy": "4094.09"
      },
      "pair": "btc_jpy",
      "rate": "40900.0",
      "fee_currency": "JPY",
      "fee": "-4.09",
      "liquidity": "M",
      "side": "sell"
    }
  ]
}
```


RESPONSE ITEMS

- id ID

- order_id 注文のID
- created_at 取引が行われた時間
- funds 各残高の増減分
- pair 取引ペア
- rate 約定価格
- fee_currency 手数料の通貨
- fee 発生した手数料
- liquidity "T" (Taker) or "M" (Maker)
- side "sell" or "buy"

アカウント

自分のアカウントの残高や、各種情報を取得することができます。

残高

アカウントの残高を確認できます。

jpy, btc には未決済の注文に利用している jpy_reserved, btc_reserved は含まれていません。

HTTP REQUEST

```
GET /api/accounts/balance
```

```
{
  "success": true,
  "jpy": "0.8401",
  "btc": "7.75052654",
  "jpy_reserved": "3000.0",
  "btc_reserved": "3.5002",
  "jpy_lend_in_use": "0",
  "btc_lend_in_use": "0.3",
  "jpy_lent": "0",
  "btc_lent": "1.2",
  "jpy_debt": "0",
  "btc_debt": "0"
}
```

RESPONSE ITEMS

- jpy 日本円の残高
- btc ビットコインの残高
- jpy_reserved 未決済の買い注文に利用している日本円の合計
- btc_reserved 未決済の売り注文に利用しているビットコインの合計
- jpy_lend_in_use 貸出申請をしている日本円の合計 (現在は日本円貸出の機能を提供していません)
- btc_lend_in_use 貸出申請をしているビットコインの合計 (現在はビットコイン貸出の機能を提供していません)
- jpy_lent 貸出をしている日本円の合計 (現在は日本円貸出の機能を提供していません)
- btc_lent 貸出をしているビットコインの合計 (現在はビットコイン貸出の機能を提供していません)

- jpy_debt 借りている日本円の合計
- btc_debt 借りているビットコインの合計

ビットコインの送金

指定のアドレスにビットコインを送ります。

HTTP REQUEST

```
POST /api/send_money
```

PARAMETERS

- *address 送り先のビットコインアドレス
- *amount 送りたいビットコインの量

```
{
  "success": true,
  "id": "276",
  "address": "1v6zFvyNPgdRvhUufkRoTtgyiw1xigncc",
  "amount": "1.5",
  "fee": "0.002"
}
```

RESPONSE ITEMS

- id 送金のIDです
- address 送った先のbitcoinアドレス
- amount 送ったbitcoinの量
- fee 手数料

送金履歴

ビットコインの送金履歴です。

HTTP REQUEST

```
GET /api/send_money
```

PARAMETERS

- *currency 通貨 (現在は BTC のみ対応)

```
{
  "success": true,
  "sends": [
    {
      "id": 2,
      "amount": "0.05",
      "currency": "BTC",
      "fee": "0.0",
      "address": "13PhzoK8me3u5nHzzFD85qT9RqEWR9M4Ty",
      "created_at": "2015-06-13T08:25:20.000Z"
```

```
    },
    {
      "id": 1,
      "amount": "0.05",
      "currency": "BTC",
      "fee": "0.0001",
      "address": "118ekvRwx6uc4241jkuQt5eYvLiuWdMW2",
      "created_at": "2015-06-13T08:21:18.000Z"
    }
  ]
}
```


RESPONSE ITEMS

- id 送金のIDです
- amount 送ったbitcoinの量
- fee 手数料
- currency 通貨
- address 送った先のbitcoinアドレス
- created_at 送金処理の作成日時

受け取り履歴

ビットコインの受け取り履歴です。

HTTP REQUEST

```
GET /api/deposit_money
```

PARAMETERS

- *currency 通貨（現在は BTC のみ対応）


```
{
  "success": true,
  "deposits": [
    {
      "id": 2,
      "amount": "0.05",
      "currency": "BTC",
      "address": "13PhzoK8me3u5nHzzFD85qT9RqEWR9M4Ty",
      "status": "confirmed",
      "confirmed_at": "2015-06-13T08:29:18.000Z",
      "created_at": "2015-06-13T08:22:18.000Z"
    },
    {
      "id": 1,
      "amount": "0.01",
      "currency": "BTC",
      "address": "13PhzoK8me3u5nHzzFD85qT9RqEWR9M4Ty",
      "status": "received",
      "confirmed_at": "2015-06-13T08:21:18.000Z",
      "created_at": "2015-06-13T08:21:18.000Z"
    }
  ]
}
```


RESPONSE ITEMS

- id 受け取りのID
- amount 受け取ったビットコインの量
- currency 通貨
- address 受け取り元のビットコインアドレス
- status ステータス
- confirmed_at 受け取りの承認日時
- created_at 受け取り処理の作成日時

アカウント情報

アカウントの情報を表示します。

HTTP REQUEST

```
GET /api/accounts
```

```
{
  "success": true,
  "id": 10000,
  "email": "test@gmail.com",
  "identity_status": "identity_pending",
  "bitcoin_address": "1v6zFvyNPgdRvhUufkRoTtgyiw1xigncc",
  "taker_fee": "0.15",
  "maker_fee": "0.0",
  "exchange_fees": {
    "btc_jpy": {
      "taker_fee": "0.15",
      "maker_fee": "0.0"
    },
    "etc_jpy": {
      "taker_fee": "0.0",
      "maker_fee": "0.0"
    },
    "fct_jpy": {
      "taker_fee": "0.1",
      "maker_fee": "0.1"
    },
    "mona_jpy": {
      "taker_fee": "0.0",
      "maker_fee": "0.0"
    }
  }
}
```

RESPONSE ITEMS

- id アカウントのID。日本円入金の際に指定するIDと一致します。
- email 登録されたメールアドレス
- identity_status 本人確認書類の提出状況を表示します。
- bitcoin_address あなたのデポジット用ビットコインのアドレス
- taker_fee Takerとして注文を行った場合の手数料を表示します。(BTC_JPY)
- maker_fee Makerとして注文を行った場合の手数料を表示します。(BTC_JPY)
- exchange_fees 板ごとの手数料を表示します。

日本円出金

日本円を銀行振込で出金できます。

銀行口座一覧

お客様の出金用に登録された銀行口座の一覧を返します。

HTTP REQUEST

```
GET /api/bank_accounts
```

```
{
  "success": true,
  "data": [
    {
      "id": 243,
      "bank_name": "みずほ",
      "branch_name": "東京営業部",
      "bank_account_type": "futsu",
      "number": "0123456",
      "name": "タナカ　タロウ"
    }
  ]
}
```

RESPONSE ITEMS

- id ID
- bank_name 銀行名
- branch_name 支店名
- bank_account_type 銀行口座の種類（futsu : 普通口座, toza : 当座預金口座）
- number 口座番号
- name 口座名義

銀行口座の登録

出金先の銀行口座を登録します。

HTTP REQUEST

```
POST /api/bank_accounts
```

PARAMETERS

- *bank_name 銀行名
- *branch_name 支店名
- *bank_account_type 銀行口座の種類（futsu : 普通口座, toza : 当座預金口座）
- *number 口座番号（例）"0123456"
- *name 口座名義

```
{
  "success": true,
  "data": {
    "id": 641,
```

```
    "bank_name": "熊本",
    "branch_name": "田中",
    "bank_account_type": "futsu",
    "number": "0123456",
    "name": "hoge"
  }
}
```

RESPONSE ITEMS

- id ID
- bank_name 銀行名
- branch_name 支店名
- bank_account_type 銀行口座の種類（futsu : 普通口座, toza : 当座預金口座）
- number 口座番号（例）"0123456"
- name 口座名義

銀行口座の削除

出金先の銀行口座を削除します。

HTTP REQUEST

```
DELETE /api/bank_accounts/[id]
```

PARAMETERS

- *id 銀行口座一覧のID

```
{
  "success": true
}
```

出金履歴

日本円出金の申請の履歴を表示します。

HTTP REQUEST

```
GET /api/withdraws
```

```
{
  "success": true,
  "pagination": {
    "limit": 25,
    "order": "desc",
    "starting_after": null,
    "ending_before": null
  },
  "data": [
    {
      "id": 398,
      "status": "finished",
      "amount": "242742.0",
      "currency": "JPY",
```

```
      "created_at": "2014-12-04T15:00:00.000Z",
      "bank_account_id": 243,
      "fee": "400.0",
      "is_fast": true
    }
  ]
}
```

RESPONSE ITEMS

- id ID
- status 出金の状態 (pending 未処理, processing 手続き中, finished 完了, canceled キャンセル済み)
- amount 金額
- currency 通貨
- created_at 作成日時
- bank_account_id 銀行口座のID
- fee 振込手数料
- is_fast 高速出金のオプション 現在、停止中

出金申請の作成

日本円の出金申請をします。

HTTP REQUEST

```
POST /api/withdraws
```

PARAMETERS

- *bank_account_id 銀行口座のID
- *amount 金額
- *currency 通貨 (現在は "JPY" のみ対応)

```
{
  "success": true,
  "data": {
    "id": 1043,
    "status": "pending",
    "amount": "10000.0",
    "currency": "JPY",
    "created_at": "2015-08-31T11:32:45.213Z",
    "bank_account_id": 243,
    "fee": "300.0"
  }
}
```

RESPONSE ITEMS

- id ID
- status 出金の状態 (pending 未処理, processing 手続き中, finished 完了, canceled キャンセル済み)
- amount 金額
- currency 通貨 (現在は "JPY" のみ対応)

- created_at 作成日時
- bank_account_id 銀行口座のID
- fee 振込手数料

出金申請のキャンセル

出金申請をキャンセルします。

status が pending の出金申請のみキャンセルできます。

HTTP REQUEST

```
DELETE /api/withdraws/[id]
```

PARAMETERS

- *id 出金申請のID

```
{
  "success": true
}
```

WebSocket API β版

WebSocket APIはHTTP通信では実装が難しかった、リアルタイムでのやり取りが容易になり、取引所で公開されている取引の履歴、板情報を取得することができます。

WebSocket APIはβ版であるため、仕様に変更があったり、動作が不安定になる可能性があります。また、利用可能な通貨ペアは"btc_jpy" "etc_jpy" "fct_jpy" "mona_jpy"になります。
(2020/09時点)

概要

WebSocket 接続先

wss://ws-api.coincheck.com/

Subscribe

メッセージの受信を開始するには、まずSubscribeを示すリクエストをサーバーに送信する必要があります。このリクエストはJSONエンコードされている必要があります。

テスト方法

//JavaScript
//Chrome環境にて実行
socket = new WebSocket("wss://ws-api.coincheck.com/")

```
socket.send(JSON.stringify({type: "subscribe", channel: "btc_jpy-orderbook"}))
socket.send(JSON.stringify({type: "subscribe", channel: "btc_jpy-trades"}))
```

パブリックチャンネル

パブリックチャンネルは認証を必要としないチャンネルです。

取引履歴

最新の取引履歴を取得できます。

例) [pair] = btc_jpy

REQUEST

```
{
  "type": "subscribe",
  "channel": "[pair]-trades"
}
```

RESPONSE

```
[
  2357062,
  "[pair]",
  "148638.0",
  "5.0",
  "sell"
]

[
  2357068,
  "[pair]",
  "148642.0",
  "0.7828",
  "buy"
]
```

RESPONSE ITEMS

```
[
  "ID",
  "取引ペア",
  "注文のレート",
  "注文での量",
  "注文方法"
]
```

板情報

板情報の差分を一定間隔で送信します。

例) [pair] = btc_jpy

REQUEST

```
{
  "type": "subscribe",
  "channel": "[pair]-orderbook"
}
```

RESPONSE

```
[
  "[pair]",
  {
    "bids": [
      [
        "148634.0",
        "0"
      ],
      [
        "148633.0",
        "0.0574"
      ]
    ],
    "asks": [
      [
        "148834.0",
        "0"
      ],
      [
        "148833.0",
        "1.0581"
      ]
    ]
  }
]
```

RESPONSE ITEMS

- asks 売り注文の情報
- bids 買い注文の情報

```
[
  "注文のレート",
  "注文での量"
]
```

- Coincheck
- ドキュメント一覧
- 取引所 API 概要

Language 日本語



Coincheck

コインチェック株式会社

暗号資産交換業者
登録番号：関東財務局長00014号

一般社団法人
日本暗号資産取引業協会会員

-
-
-

PRODUCTS

- ビットコイン販売所
- アルトコイン販売所
- ビットコイン取引所
- コインチェックアプリ
- ビットコイン決済
- ビットコイン決済アプリ
- 貸仮想通貨サービス
- Coincheckでんき
- Coincheckガス
- Coincheckアンケート
- Coincheckつみたて
- OTC取引サービス

SERVICE GUIDES

- 初めての方へ
- 個人口座開設の流れ
- 法人口座開設の流れ
- サービスの安全性・セキュリティ
- 安心してお取引いただくための取組
- ビットコイン購入方法
- ビットコイン送金
- ビットコイン決済導入方法
- ビットコイン取引所 API
- ビットコイン決済 API
- チャットガイドライン

SUPPORT

- ヘルプセンター/お問い合わせ
- 取扱い通貨の終値一覧
- 過去のレート検索
- 手数料
- 注文数量について
- 苦情処理・紛争解決に係る業務運営体制

INFO

- 仮想通貨とは？
- ビットコインとは？
- アルトコインとは？
- 仮想通貨のお役立ち情報
- 仮想通貨チャート一覧
- 取引説明書
- Coincheckつみたて取引説明書
- 取扱い暗号資産の概要
- 取扱い暗号資産に関する参考情報
- ロゴについて
- 公式ブログ
- Coincheck Tech Blog
- マネックス仮想通貨研究所

ABOUT

- 会社概要
- 採用情報
- 利用規約
- Coincheckつみたて利用規約
- 特商法について
- プライバシーポリシー
- 情報セキュリティポリシー
- 反社会的勢力対応基本方針
- マネー・ローンダリング等防止基本方針
- 計画されたHF及び新コインへの当社対応指針
- 債務の履行に関する方針
- 最良執行方針
- 利益相反管理方針
- サイトマップ
- サイトポリシー