HTTP Protocol

Jump to bottom

haipo yang edited this page on Nov 7, 2017 · 44 revisions

API protocol

The API is based on HTTP protocol JSON RPC, whose request method must be POST. Its URL is：/ and the Content-Type is：application/json

Request

- method: method，String
- params: parameters，Array
- id: Request id, Integer

Response

- result: Json object，null for failure
- error: Json object，null for success，non-null for failure

1. code: error code
2. message: error message

- id: Request id, Integer

General error code:

- 1: invalid argument
- 2: internal error
- 3: service unavailable
- 4: method not found
- 5: service timeout

Asset API

Asset inquiry

- method: `balance.query`
- params: unfixed parameters, first parameter is user ID, followed by list of asset names. Return to user's overall asset if the list is blank.

1. user_id: user ID，Integer

- result: {"asset": {"available": "amount", "freeze": "amount"}}
- example:

```
"params": [1, "BTC"]
"result": {"BTC": {"available": "1.10000000","freeze": "9.90000000"}}
```

Asset change

- method: `balance.update`
- params:

1. user_id: user ID，Integer
2. asset: asset name，String
3. business: business type，String
4. business_id: business ID，Integer, but it will only succeed once with multiple operations of the same user_id, asset, business or business_id
5. change: balance change，String, negative numbers for deduction
6. detail: Json object，attached information

- result: "success"
- error code:

10. repeat update
11. balance not enough

- example:

```
"params": [1, "BTC", "deposit", 100, "1.2345"]
"result": "success"
```

Asset history

- method: `balance.history`
- params:

1. user_id: user ID, Integer
2. asset: asset name，which can be null
3. business: business，which can be null，use ',' to separate types
4. start_time: start time，0 for unlimited，Integer
5. end_time: end time，0 for unlimited, Integer
6. offset: offset position，Integer
7. limit: count limit，Integer

- result:

```
{
    "offset":
    "limit":
    "records": [
        {
            "time": timestamp,
            "asset": asset,
            "business": business,
            "change": change,
            "balance"：balance,
            "detail": detail
        }
        ...
    ]
```

Asset list

- method: `asset.list`
- params: none

Asset summary

- method： `asset.summary`
- params: asset list，return to overall asset if null

Trade API

Place limit order

- method: `order.put_limit`
- params:

1. user_id: user ID，Integer
2. market: market name，String
3. side: 1: sell, 2: buy，Integer
4. amount: count，String
5. pride: price，String
6. taker_fee_rate: String, taker fee
7. maker_fee_rate: String, maker fee
8. source: String, source，up to 30 bytes

- result: order detail
- error:

10. balance not enough

- example:

```
params: [1, "BTCCNY", 1, "10", "8000", "0.002", "0.001"]
```

Place market order

- method: `order.put_market`
- params:

1. user_id: user ID，Integer
2. market: market name，String
3. side: 1: sell, 2: buy，Integer
4. amount: count or amount，String
5. taker_fee_rate: taker fee
6. source: String, source，up to 30 bytes

- result: order detail
- error:

10. balance not enough

- example:

```
params: '[1, "BTCCNY", 1, "10","0.002"]'
```

Cancel order

- method: `order.cancel`
- params:

1. user_id: user ID
2. market：market
3. order_id： order ID

- result: order detail
- error:

10. order not found
11. user not match

Order details

- method: `order.deals`
- params:

1. order_id: order ID, Integer
2. offset
3. limit

- result:
- example:

```
"result": {
    "offset":
    "limit":
    "records": [
        {
            "id": Executed ID
            "time": timestamp
            "user": user ID
            "role": role, 1：Maker, 2: Taker
            "amount": count
            "price": price
            "deal": order amount
            "fee": fee
            "deal_order_id": Counterpart transaction ID
        }
    ...
    ]
```

Order list

- method: `order.book`
- params:

1. market:
2. side: side，1：sell，2：buy
3. offset:
4. limit:

- result:

Order depth

- method: `order.depth`
- params:

1. market：market name
2. limit: count limit，Integer
3. interval: interval，String, e.g. "1" for 1 unit interval, "0" for no interval

- result:

```
"result": {
    "asks": [
        [
            "8000.00",
            "9.6250"
        ]
    ],
    "bids": [
        [
            "7000.00",
            "0.1000"
        ]
    ]
}
```

Inquire unexecuted orders

- method: `order.pending`
- params:

1. user_id: user ID，Integer
2. market: market name，String
3. offset: offset，Integer
4. limit: limit，Integer

- result:
- example:

```
"params": [1, "BTCCNY", 0, 100]"
"result": {
    "offset": 0,
    "limit": 100,
    "total": 1,
    "records": [
        {
            "id": 2,
            "ctime": 1492616173.355293,
            "mtime": 1492697636.238869,
            "market": "BTCCNY",
            "user": 2,
            "type": 1, // 1: limit order，2：market order
            "side": 2, // 1：sell，2：buy
            "amount": "1.0000".
            "price": "7000.00",
            "taker_fee": "0.0020",
            "maker_fee": "0.0010",
            "source": "web",
            "deal_money": "6300.0000000000",
            "deal_stock": "0.9000000000",
            "deal_fee": "0.0009000000"
        }
    ]
}
```

Unexecuted order details

- method: `order.pending_detail`
- params:

1. market:
2. order_id: order ID，Integer

- result:

Inquire executed orders

- method: `order.finished`
- params:

1. user_id: user ID，Integer
2. market: market name，String
3. start_time: start time，0 for unlimited，Integer
4. end_time: end time，0 for unlimited, Integer
5. offset: offset，Integer
6. limit: limit，Integer
7. side: side，0 for no limit，1 for sell，2 for buy

- result:

Executed order details

- method: `order.finished_detail`
- params:

1. order_id: order ID，Integer

- result:

Market API

Market price

- method: `market.last`
- params:

1. market

- result: "price"

Executed history

- method: `market.deals`
- params:

1. market:
2. limit: count，no more than 10000
3. last_id: id limit

- result:

```
"result": [
    {
        "id": 5,
        "time": 1492697636.238869,
        "type": "sell",
        "amount": "0.1000",
        "price": "7000.00"
    },
    {
        "id": 4,
        "time": 1492697467.1411841,
        "type": "sell",
        "amount": "0.1000"
        "price": "7000.00",
    }
}
```

User Executed history

- method: `market.user_deals`
- params:

1. user_id: user ID，Integer
2. market: market name，String
3. offset: offset，Integer
4. limit: limit，Integer

- result:

```
"result": [
    "offset":
    "limit":
    "records": [
        {
            "id": Executed ID
            "time": timestamp
            "user": user ID
            "side": side，1：sell，2：buy
            "role": role，1：Maker, 2: Taker
            "amount": count
            "price": price
            "deal": amount
            "fee": fee
            "deal_order_id": Counterpart Transaction ID
        }
    ...
    ]
}
```

KLine

- method: `market.kline`
- params:

1. market: market
2. start: start ，Integer
3. end: end, Integer
4. interval: interval, Integer

- result:

```
"result": [
    [
        1492358400, time
        "7000.00",  open
        "8000.0",   close
        "8100.00",  highest
        "6800.00",  lowest
        "1000.00"   volume
        "123456.78" amount
        "BTCCNY"    market name
    ]
]
```

Market status

- method: `market.status`
- params:

1. market: market name
2. period: cycle period ，Integer, e.g. 86400 for last 24 hours

- result:

```
"result": {
    "period": 86400,
    "last": "7000.00",
    "open": "0",
    "close": "0",
    "high": "0",
    "low": "0",
    "volume": "0"
}
```

Market status today

- method: `market.status_today`
- params:

1. market: market name

- result:

```
{
    "open": Open today
    "last": Latest price
    "high": Highest price
    "low":  Lowest price
    "deal": 24H amount
    "volume": 24H volume
}
```

Market list

- method: `market.list`
- params: none

Market summary

- method： `market.summary`
- params: market list, return to market if null