api-doc Documentation

リリース *v2.0.0*

fcce

2019 年 09 月 12 日

目次

第 1 章

現物公開 API

1.1 共通情報

現物公開 API の共通情報です。

1.1.1 リクエスト方法

- エンドポイント：https://api.zaif.jp/api/1
- メソッド：GET

1.1.2 戻り値

- 全て json 形式となっています。フォーマットは API によって変わります。

1.1.3 エラーメッセージ

メッセージ	詳細
unsupported method	サポート外の method です。

1.1.4 補足

- 呼び出しは 1 秒間に 10 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。

1.2 個別情報

現物公開 API の個別情報です。

1.2.1 通貨情報の取得

通貨情報を取得します。

リクエスト方法

- /currencies/{currency}

例）https://api.zaif.jp/api/1/currencies/btc

パラメータ

- なし

戻り値

キー	詳細	型
name	通貨の名前	str
is_token	token 種別	boolean

currency に all を指定した場合、有効な全ての通貨情報を取得します。

```
[
    {
        "name": "btc",
        "is_token": false
    },
    {
        "name": "XCP",
        "is_token": true
    },
    ...
]
```

currency に btc 等、有効な通貨情報を指定した場合、その情報のみを取得します。

```
[
    {
        "name": "btc",
        "is_token": false
    }
]
```

補足

- token 種別

  token の場合、true。

1.2.2 通貨ペア情報の取得

通貨ペア情報を取得します。

リクエスト方法

- /currency_pairs/{currency_pair}

例）https://api.zaif.jp/api/1/currency_pairs/btc_jpy

パラメータ

- なし

戻り値

キー	詳細	型
name	通貨ペアの名前	str
title	通貨ペアのタイトル	str
currency_pair	通貨ペアのシステム文字列	str
description	通貨ペアの詳細	str
is_token	token 種別	boolean
event_number	イベントトークンの場合、0 以外	int
seq	通貨シークエンス	int
item_unit_min	基軸通貨最小値	float
item_unit_step	基軸通貨最小単位	float
item_japanese	基軸通貨日本語表記	str
aux_unit_min	決済通貨最小値	float
aux_unit_step	決済通貨最小単位	float
aux_unit_point	決済通貨小数点桁数	int
aux_japanese	決済通貨日本語表記	str

currency_pair に all を指定した場合、有効な全ての通貨ペア情報を取得します。


```
[
    {
        "name": "BTC/JPY",
        "title": "BTC/JPY",
        "currency_pair": "btc_jpy",
        "description":
↪"\u30d3\u30c3\u30c8\u30b3\u30a4\u30f3\u30fb\u65e5\u672c\u5186\u306e\u53d6\u5f15\u3092\u884c\u3046\u
↪",
        "is_token": false,
        "event_number": 0,
        "item_unit_min": 0.0001,
        "item_unit_step": 0.0001,
```


(次のページに続く)

(前のページからの続き)


```
        "aux_unit_min": 5.0,
        "aux_unit_step": 5.0,
        "seq": 0,
        "aux_japanese": "\u65e5\u672c\u5186",
        "item_japanese": "\u30d3\u30c3\u30c8\u30b3\u30a4\u30f3",
        "aux_unit_point": 0,
    },
    {
        "name": "KINOKOUSAKA/JPY",
        "title": "KINOKOUSAKA/JPY \u53d6\u5f15\u6240 - ZAIF Exchange",
        "currency_pair": "kinokousaka_jpy",
        "description": "KINOKOUSAKA/JPY␣
↪\u53d6\u5f15\u6240\u3002KINOKOUSAKA\u3068\u65e5\u672c\u5186\u306e\u53d6\u5f15\u304c\u884c\u3048\u30
↪",
        "is_token": true,
        "event_number": 1,
        "item_unit_min": 0.01,
        "item_unit_step": 0.01,
        "aux_unit_min": 0.01,
        "aux_unit_step": 0.01,
        "seq": 134,
        "aux_japanese": "\u65e5\u672c\u5186",
        "item_japanese": "KINOKOUSAKA",
        "aux_unit_point": 2,
    }
    ...
]
```


currency_pair に btc_jpy 等、有効な通貨ペア情報を指定した場合、その情報のみを取得します。


```
[
    {
        "name": "BTC/JPY",
        "title": "BTC/JPY",
        "currency_pair": "btc_jpy",
        "description":
↪"\u30d3\u30c3\u30c8\u30b3\u30a4\u30f3\u30fb\u65e5\u672c\u5186\u306e\u53d6\u5f15\u3092\u884c\u3046\u
↪",
        "is_token": false,
        "event_number": 0,
        "item_unit_min": 0.0001,
        "item_unit_step": 0.0001,
        "aux_unit_min": 5.0,
        "aux_unit_step": 5.0,
        "seq": 0,
        "aux_japanese": "\u65e5\u672c\u5186",
        "item_japanese": "\u30d3\u30c3\u30c8\u30b3\u30a4\u30f3",
```


(次のページに続く)

(前のページからの続き)

```
        "aux_unit_point": 0,
    }
]
```

補足

- token 種別

  token の場合、true。

1.2.3 現在の終値を取得

現在の終値を取得します。

リクエスト方法

- /last_price/{currency_pair}

例）https://api.zaif.jp/api/1/last_price/btc_jpy

注釈: currency_pair に指定できる値は **通貨ペア情報の取得** を参照してください。

パラメータ

- なし

戻り値

キー	詳細	型
last_price	現在の終値	float

```
{
    "last_price": 134820.0
}
```

エラーメッセージ

メッセージ	詳細
unsupported currency_pair	サポートされていない通貨ペアです。
currency_pair missing	リクエストされた通貨ペアが不適切です。

1.2.4 ティッカーの取得

ティッカーを取得します。

リクエスト方法

- /ticker/{currency_pair}

例）https://api.zaif.jp/api/1/ticker/btc_jpy

注釈: currency_pair に指定できる値は **通貨ペア情報の取得** を参照してください。

パラメータ

- なし

戻り値

キー	詳細	型
last	終値	float
high	過去 24 時間の高値	float
low	過去 24 時間の安値	float
vwap	過去 24 時間の加重平均	float
volume	過去 24 時間の出来高	float
bid	買気配値	float
ask	売気配値	float

```
{
    "last": 135875.0,
    "high": 136000.0,
```

(次のページに続く)

(前のページからの続き)

```
    "low": 131570.0,
    "vwap": 133301.7489,
    "volume": 6889.215,
    "bid": 135875.0,
    "ask": 135920.0
}
```

エラーメッセージ

メッセージ	詳細
unsupported currency_pair	サポートされていない通貨ペアです。
currency_pair missing	リクエストされた通貨ペアが不適切です。

補足

- vwap 算出方法

  個々の取引価格*個々の取引量　→　A

  A の過去 24 時間分を合算　→　B

  過去 24 時間分の個々の取引量を合算　→　C

  B/C →　vwap

1.2.5 全ユーザー取引履歴の取得

全ユーザの取引履歴を取得します。取得できる取引履歴は最新のものから最大 150 件となります。

リクエスト方法

- /trades/{currency_pair}

例）https://api.zaif.jp/api/1/trades/btc_jpy

注釈: currency_pair に指定できる値は **通貨ペア情報の取得** を参照してください。

パラメータ

- なし

戻り値

キー	詳細	型
date	取引日時	UNIX_TIMESTAMP
price	取引価格	float
amount	取引量	float
tid	取引 ID	int
currency_pair	通貨ペア	str
trade_type	取引種別	str

```
[
    {
        "date": 1491756592,
        "price": 135340.0,
        "amount": 0.02,
        "tid": 43054307,
        "currency_pair": "btc_jpy",
        "trade_type": "ask"
    },
    {
        "date": 1491756591,
        "price": 135345.0,
        "amount": 0.01,
        "tid": 43054306,
        "currency_pair": "btc_jpy",
        "trade_type": "bid"
    },
    ...
]
```

エラーメッセージ

メッセージ	詳細
unsupported currency_pair	サポートされていない通貨ペアです。
currency_pair missing	リクエストされた通貨ペアが不適切です。

補足

- 取引種別

  bid：買い　ask：売り

1.2.6 板情報の取得

板情報を取得します。売り情報は価格の昇順、買い情報は価格の降順でソートされた状態で返却されます。情報数は最大 150 件となります。

リクエスト方法

- /depth/{currency_pair}

例）https://api.zaif.jp/api/1/depth/btc_jpy

注釈: currency_pair に指定できる値は **通貨ペア情報の取得** を参照してください。

パラメータ

- なし

戻り値

キー	詳細	型
asks	売り板情報	list
bids	買い板情報	list

```
{
    "asks": [
        [
            134875.0,
            0.0063
        ],
        [
            134885.0,
            0.1639
        ],
```

(次のページに続く)

(前のページからの続き)

```
        ...
    ],
    "bids": [
        [
            134870.0,
            0.01
        ],
        [
            134865.0,
            0.3066
        ],
        ...
    ]
}
```

エラーメッセージ

メッセージ	詳細
unsupported currency_pair	サポートされていない通貨ペアです。
currency_pair missing	リクエストされた通貨ペアが不適切です。

補足

- 売り (買い) 板情報

  配列の最初が価格、最後が量。

第 2 章

現物取引 API

2.1 共通情報

現物取引 API の共通情報です。

2.1.1 事前準備

- 取引 API を利用するには、アカウント情報 のページから API Key の発行をおこなってください。

2.1.2 リクエスト方法

- エンドポイント：https://api.zaif.jp/tapi
- メソッド：POST

2.1.3 認証

- 取得した API Keys を利用して、下記のように HTTP ヘッダを設定し、認証情報を送信します。

パラメータ	詳細	例
key	API キー	490f983a-5fab-49b2-b789-9d1f130874d3
sign	署名	詳細は下記

注釈: sign は POST する全てのパラメータ（nonce と method およびメソッド毎のパラメータ）を URL エンコードしたクエリ形式 (param1=val1&param2=val2) のメッセージとして、Secret Key を用いて HMAC-SHA512 で署名します。

2.1.4 パラメータ

パラメータ	詳細	例
nonce	1 以上の数	23123
method	API メソッド名	get_info

注釈: メソッド毎の固有のパラメータも全て POST パラメータにて送信してください。nonce パラメータの値は実効毎に増分されていないとエラーが発生します。また、増分量は少数点以下の値にも対応しております。

2.1.5 戻り値

キー	詳細	型
success	成功フラグ	int
return	実行結果	dict or string

```
{
    "success": 1,
    "return": {
        ...
    }
}
```

2.1.6 エラーメッセージ

メッセージ	詳細
method not found	指定されたメソッドが存在しません。
no data found for the key	API キーが無効です。
time wait restriction, please later	同じメソッドが短時間に多く呼び出しされたときに発生します。
signature mismatch	署名が不適切です。
invalid access token	無効なトークンが指定されています。
expired access token	トークンの有効期限が切れています。トークン再発行 API を参考にし、トークンの再発行をしてください。
nonce not incremented	前回 API 実行時より nonce 値が加算されていません。
nonce out of range	値が最大値を超えています。新しい API キーを発行してください。
api key don't have {} permission	APIKey に権限がありません。
invalid {} parameter	指定されているパラメータが無効です。

2.1.7 補足

- 戻り値

  処理に成功した場合、success には 1 が、return には実行結果が設定されます。

  処理に失敗した場合、success には 0 が、return にはエラーメッセージが設定されます。

2.2 個別情報

現物取引 API の個別情報です。

2.2.1 残高情報の取得

現在の残高（余力および残高・トークン）、API キーの権限、過去のトレード数、アクティブな注文数、サーバーのタイムスタンプを取得します。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	get_info	str	

戻り値

キー	詳細	型
funds	残高	dict
deposit	残高に注文情報を加味した情報	dict
rights	キーが保持している権限	dict
trade_count	実行したトレード数	int
open_orders	アクティブな注文数	int
server_time	UNIX 時間で換算された日本時間	int

```
{
    "success":1,
    "return":{
        "funds":{
```

(次のページに続く)

(前のページからの続き)


```
            "jpy":15320,
            "btc":1.389,
            "xem":100.2,
            "mona":2600,
            "pepecash":0.1
        },
        "deposit":{
            "jpy":20440,
            "btc":1.479,
            "xem":100.2,
            "mona":3200,
            "pepecash":0.1
        },
        "rights":{
            "info":1,
            "trade":1,
            "withdraw":0,
            "personal_info":0,
            "id_info":0,
        },
        "trade_count":18,
        "open_orders":3,
        "server_time":1401950833
    }
}
```


補足

- 呼び出しは 10 秒間に 10 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- deposit 算出方法

  deposit は現在の資産の残高に注文情報を加味したものになります。

  買い注文が存在する場合、その注文の値段と量をかけ合わせたもので、売り注文が存在する場合は、その注文の量のみが加味されます。
- 取得できる情報は、API を実行した時点のものになります。

2.2.2 残高情報 (軽量) の取得

get_info の軽量版で、過去のトレード数を除く項目を返します。

パラメータ

パラメータ	必須	詳細	型
method	Yes	get_info2	str

戻り値

キー	詳細	型
funds	残高	dict
deposit	残高に注文情報を加味した情報	dict
rights	キーが保持している権限	dict
open_orders	アクティブな注文数	int
server_time	UNIX 時間で換算された日本時間	int

```
{
    "success": 1,
    "return": {
        "funds": {
            "jpy": 15320,
            "btc": 1.389,
            "xem": 100.2,
            "mona": 2600,
            "pepecash": 0.1
        },
        "deposit": {
            "jpy": 20440,
            "btc": 1.479,
            "xem": 100.2,
            "mona": 3200,
            "pepecash": 0.1
        },
        "rights": {
            "info": 1,
            "trade": 1,
            "withdraw": 0,
            "personal_info": 0
        },
        "open_orders": 3,
        "server_time": 1401950833
    }
}
```

補足

- 呼び出しは 10 秒間に 20 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- deposit 算出方法

  deposit は現在の資産の残高に注文情報を加味したものになります。

  買い注文が存在する場合、その注文の値段と量をかけ合わせたもので、売り注文が存在する場合は、その注文の量のみが加味されます。
- 取得できる情報は、API を実行した時点のものになります。

2.2.3 チャット情報の取得

チャットに使用されるニックネームと画像のパスを返します。

パラメータ

パラメータ	必須	詳細	型
method	Yes	get_personal_info	str

戻り値

キー	詳細	型
ranking_nickname	ニックネーム	str
icon_path	画像のパス	str

```
{
    "success": 1,
    "return": {
        "ranking_nickname": "ニックネーム",
        "icon_path": "https://abs.twimg.com/sticky/default_profile_images/default_
↪profile_0_normal.png"
    }
}
```

2.2.4 個人情報の取得

ユーザー ID やメールアドレスといった個人情報を取得します。

パラメータ

パラメータ	必須	詳細	型
method	Yes	get_id_info	str

戻り値

キー	詳細	型
id	ユーザー ID	str
email	メールアドレス	str
name	ユーザ名	int
kana	ユーザ名カナ	str
certified	認証済みかどうか	bool

```
{
    "success": 1,
    "return": {
    }
}
```

2.2.5 ユーザー自身の取引履歴を取得

ユーザー自身の取引履歴を取得します。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	trade_history	str	
from	No	この順番のレコードから取得	int	0
count	No	取得するレコード数	int	1000
from_id	No	このトランザクション ID のレコードから取得	int	0
end_id	No	このトランザクション ID のレコードまで取得	int	infinity
order	No	ソート順	str (ASC or DESC)	DESC
since	No	開始タイムスタンプ	UNIX_TIMESTAMP	0
end	No	終了タイムスタンプ	UNIX_TIMESTAMP	infinity
currency_pair	No	通貨ペア	str	指定なし
is_token	No	カウンターパーティトークンかどうか	bool	false

戻り値

キー	詳細	型
例）182	注文 ID	int
currency_pair	通貨ペア	str
action	bid(買い) or ask(売り)	str
amount	数量	float
price	価格	float
fee	手数料	int
your_action	bid(買い) or ask(売り)、自己取引の場合は both	str
bonus	マイナス手数料分	float
timestamp	取引日時	UNIX_TIMESTAMP
comment	注文のコメント	str


```
{
    "success": 1,
    "return": {
        "182": {
            "currency_pair": "btc_jpy",
            "action": "bid",
            "amount": 0.03,
            "price": 56000,
            "fee": 0,
            "your_action": "ask",
            "bonus": 1.6,
            "timestamp": 1402018713,
```


(次のページに続く)

(前のページからの続き)

```
            "comment" : "demo"
        }
    }
}
```

補足

- 呼び出しは 60 秒間に 12 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- “ since ” もしくは ” end ” をセットした場合、” order ” は強制的に ” ASC ” となります。
- “ from_id ” もしくは ” end_id ” をセットした場合、” order ” は強制的に ” ASC ” となります。
- “ currency_pair ” と ” is_token ” の両方を指定した場合は ” currency_pair ” が優先されます。両方指定しない場合はカウンターパーティトークン以外の情報を取得します。

2.2.6 未約定注文一覧の取得

現在有効な注文一覧を取得します（未約定注文一覧）。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	active_orders	str	
currency_pair	No	通貨ペア。指定なしで全通貨ペア	str	全通貨ペア
is_token	No	カウンターパーティトークンかどうか	bool	false
is_token_both	No	true：全てのアクティブなオーダー情報を取得	bool	false
		false：currency_pair や is_token に従ったオーダー情報を取得		

戻り値

キー	詳細	型
例）184	注文 ID	int
currency_pair	通貨ペア	str
action	bid(買い) or ask(売り)	str
amount	数量	int
price	価格	int
timestamp	取引日時	UNIX_TIMESTAMP
comment	注文のコメント	str

```
{
    "success": 1,
    "return": {
        "184": {
            "currency_pair": "btc_jpy",
            "action": "ask",
            "amount": 0.03,
            "price": 56000,
            "timestamp": 1402021125,
            "comment" : "demo"
        }
    }
}
```

is_token_both が true の時は下記

```
{
   "success": 1,
   "return": {
       "active_orders": {
           "184": {
               "currency_pair": "btc_jpy",
               "action": "ask",
               "amount": 0.03,
               "price": 56000,
               "timestamp": 1402021125,
               "comment" : "demo"
           },
           "token_active_orders": {
               "235": {
                   "currency_pair": "kaori_jpy",
                   "action": "ask",
                   "amount": 0.3,
                   "price": 10,
```

(次のページに続く)

(前のページからの続き)

```
                    "timestamp": 1402064525,
                    "comment" : "demo"
                }
            }
        }
    }
}
```

補足

- 呼び出しは 5 秒間に 10 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。

2.2.7 注文

取引注文を行います。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	trade	str	
currency_pair	Yes	通貨ペア	str(例)btc_jpy	
action	Yes	bid(買い) or ask(売り)	str	
price	Yes	指値注文価格	numerical	
amount	Yes	数量	numerical	
limit	No	リミット注文価格	numerical	
comment	No	コメントの追加	str	

戻り値

キー	詳細	型
received	今回の注文で約定した取引量	float
remains	今回の注文で約定せず、板に残った取引量	float
order_id	今回の注文がすべて成立した場合は 0、	int
	一部、もしくはすべて約定しなかった場合は板に残った注文の ID	
funds	残高	dict

```
{
    "success": 1,
    "return": {
        "received": 0.1,
        "remains": 0,
        "order_id": 0,
        "funds": {
            "jpy": 325,
            "btc": 1.392,
            "mona": 2600
        }
    }
}
```

エラーメッセージ

メッセージ	詳細
trade temporarily unavailable	取引が一時的に停止されています。
your account is restricted now, KYC required.	本人確認が完了していないため、取引ができません。本人確認を完了させて下さい。
insuffcient funds	取引に必要な残高が存在しません。

補足

- 呼び出しは 10 秒間に 9 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- パラメータ limit について

  リミット値（利確のための反対売買の指値）を指定することができます。リミット値を指定した場合、注文が成立した分だけの数量について、自動的にリミット注文が発行されます。

- パラメータ comment について

  コメントは 255 字以内で半角英数字記号のみに対応しています。また、スラッシュは使えませんのでご注意ください。コメントをつけた取引注文が約定した場合、該当する取引履歴にそのコメントが付与されます。取引注文の管理にご利用ください。

- 価格および数量の数値について

  適切な価格（price および limit）、もしくは数量 (amount) の単位以外で注文しようとした場合、invalid price parameter または invalid amount parameter というエラーが返されます。適切な価格や数量は現物公開 API の通貨ペア情報で取得できます。通貨ペアごとに適切な価格や数量の最低量や単位は変わりますので、ご注意ください

2.2.8 注文の取消し

注文の取消しを行います。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	cancel_order	str	
order_id	Yes	注文 ID（**注文** または 未約定注文一覧 で取得できます）	int	
currency_pair	No	通貨ペア	str(例)btc_jpy	
is_token	No	true：カウンターパーティトークンのオーダーを取り消したい時	bool	false
		false：カウンターパーティトークン以外のオーダーを取り消したい時		

注釈: “ currency_pair ” と ” is_token ” の両方を指定した場合は ” currency_pair ” が優先されます。両方指定しない場合はカウンターパーティトークン以外の情報を操作します。

戻り値

キー	詳細	型
order_id	注文 ID	int
funds	残高	dict

```
{
    "success": 1,
    "return": {
        "order_id": 184,
        "funds": {
            "jpy": 15320,
            "btc": 1.392,
            "mona": 2600,
```

(次のページに続く)

(前のページからの続き)

```
            "kaori": 0.1
        }
    }
}
```

エラーメッセージ

メッセージ	詳細
order not found	注文が見つかりません。
order is too new	注文から一定時間の経過が必要です。

2.2.9 出金

資金の引き出しリクエストを送信します。 2015 年 12 月 15 日より、Zaif 内の振替を除くリクエストには一旦トランザクション ID は空で返されるようになりました。 通常 1 ～ 2 分でトランザクションが発生しますので、後ほど withdraw_history メソッドを利用して確認してください xem の出金時には、手数料は自動計算され、opt_fee に値をセットして送信しますとエラーが返されますのでご注意ください。 不正送金の対策として、アカウントに対する最初の日本円入金から 7 日間は、API による仮想通貨の出金を制限しております。Zaif からの XEM の出金は multisig トランザクションになります。multisig トランザクションに対応していない他の取引所やサービスへ送金されましても、Zaif では対応致し兼ねますのでご注意ください。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	withdraw	str	
currency	Yes	引き出す通貨。現物公開 API の currencies で取得できるものが指定できます。ただし jpy は指定できません。	str(例)btc 等	
address	Yes	送信先のアドレス	str	
message	No	送信メッセージ (XEM のみ)	ASCII str	
amount	Yes	出金額	numerical	
opt_fee	No	採掘者への手数料。ただし currency が btc、mona 以外の時に指定するとエラーとなります。	numerical	

戻り値

キー	詳細	型
id	出金 ID	int
txid	振替 ID	str
fee	今回の引き出しにかかった手数料	float
funds	残高	dict

```
{
  "success": 1,
  "return": {
      "id": 23634,
      "fee": 0.001,
      "txid":,
      "funds": {
          "jpy": 15320,
          "btc": 1.392,
          "xem": 100.2,
          "mona": 2600
      }
}
```

エラーメッセージ

メッセージ	詳細
kyc is not finished	郵送による本人確認が完了していません。
insufficient funds	取引に必要な残高が存在しません。

2.2.10 入金履歴の取得

入金履歴を取得します。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	deposit_history	str	
currency	Yes	通貨	str(例)jpy 等	指定なし
from	No	この順番のレコードから取得	int	0
count	No	取得するレコード数	int	1000
from_id	No	この入金 ID のレコードから取得	int	0
end_id	No	この入金 ID のレコードまで取得	int	infinity
order	No	ソート順	str (ASC or DESC)	DESC
since	No	開始タイムスタンプ	UNIX_TIMESTAMP	0
end	No	終了タイムスタンプ	UNIX_TIMESTAMP	infinity

戻り値

キー	詳細	型
timestamp	出金日時	UNIX_TIMESTAMP
address	出金先アドレス	str
amount	取引量	float
txid	トランザクション ID	str


```
{
    "success":1,
    "return":{
        "3816":{
          "timestamp":1435745065,
          "address":"12qwQ3sPJJAosodSUhSpMds4WfUPBeFEM2",
          "amount":0.001,
          "txid":"64dcf59523379ba282ae8cd61d2e9382c7849afe3a3802c0abb08a60067a159f",
        },
        "3814":{
          "timestamp":1435548083,
          "address":"12qwQ3sPJJAosodSUhSpMds4WfUPBeFEM2",
          "amount":0.001,
          "txid":"7d012cfff6e67a8938f93215367eef4177604459631ea62c85550980dca71819"
        },
    }
}
```

補足

- 呼び出しは 60 秒間に 10 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- “ since ” もしくは ” end ” をセットした場合、” order ” は強制的に ” ASC ” となります。
- “ from_id ” もしくは ” end_id ” をセットした場合、” order ” は強制的に ” ASC ” となります。

2.2.11 出金履歴の取得

出金履歴を取得します。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	withdraw_history	str	
currency	Yes	通貨	str(例)jpy 等	指定なし
from	No	この順番のレコードから取得	int	0
count	No	取得するレコード数	int	1000
from_id	No	この出金 ID のレコードから取得	int	0
end_id	No	この出金 ID のレコードまで取得	int	infinity
order	No	ソート順	str (ASC or DESC)	DESC
since	No	開始タイムスタンプ	UNIX_TIMESTAMP	0
end	No	終了タイムスタンプ	UNIX_TIMESTAMP	infinity

戻り値

キー	詳細	型
timestamp	出金日時	UNIX_TIMESTAMP
address	出金先アドレス	str
amount	取引量	float
txid	トランザクション ID	str

```
{
    "success":1,
    "return":{
        "3816":{
```

(次のページに続く)

(前のページからの続き)


```
            "timestamp":1435745065,
            "address":"12qwQ3sPJJAosodSUhSpMds4WfUPBeFEM2",
            "amount":0.001,
            "txid":"64dcf59523379ba282ae8cd61d2e9382c7849afe3a3802c0abb08a60067a159f",
          },
          "3814":{
            "timestamp":1435548083,
            "address":"12qwQ3sPJJAosodSUhSpMds4WfUPBeFEM2",
            "amount":0.001,
            "txid":"7d012cfff6e67a8938f93215367eef4177604459631ea62c85550980dca71819"
          },
      }
}
```


補足

- 呼び出しは 60 秒間に 10 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- “ since ” もしくは ” end ” をセットした場合、” order ” は強制的に ” ASC ” となります。
- “ from_id ” もしくは ” end_id ” をセットした場合、” order ” は強制的に ” ASC ” となります。

第 3 章

信用取引 API

3.1 共通情報

信用取引 API の共通情報です。

3.1.1 事前準備

- 信用取引 API を利用するには、アカウント情報 のページから API Key の発行をおこなってください。

3.1.2 リクエスト方法

- エンドポイント：https://api.zaif.jp/tlapi
- メソッド：POST

3.1.3 認証

- 取得した API Keys を利用して、下記のように HTTP ヘッダを設定し、認証情報を送信します。

キー	詳細	例
key	API キー	490f983a-5fab-49b2-b789-9d1f130874d3
sign	署名	詳細は下記

注釈: sign は POST する全てのパラメータ（nonce と method およびメソッド毎のパラメータ）を URL エンコードしたクエリ形式 (param1=val1&param2=val2) のメッセージとして、Secret Key を用いて HMAC-SHA512 で署名します。

3.1.4 パラメータ

キー	詳細	例
nonce	1 以上の数	23123
method	API メソッド名	get_info
type	取引タイプ	margin

注釈: メソッド毎の固有のパラメータも全て POST パラメータにて送信してください。 nonce パラメータの値は実効毎に増分されていないとエラーが発生します。また、増分量は少数点以下の値にも対応しております。

3.1.5 戻り値

キー	詳細	型
success	成功フラグ	int
return	実行結果	dict or string

```
{
    "success": 1,
    "return": {
        ...
    }
}
```

3.1.6 エラーメッセージ

メッセージ	詳細
method not found	指定されたメソッドが存在しません。
no data found for the key	API キーが無効です。
time wait restriction, please later	同じメソッドが短時間に多く呼び出しされたときに発生します。しばらく待ってから、再度お試しください。
signature mismatch	署名が不適切です。
invalid access token	無効なトークンが指定されています。
expired access token	トークンの有効期限が切れています。トークン再発行 API を参考にし、トークンの再発行をしてください。
nonce not incremented	前回 API 実行時より nonce 値が加算されていません。
nonce out of range	値が最大値を超えています。新しい API キーを発行してください。
api key don ’t have {} permission	API Key に権限がありません。
invalid {} parameter	指定されているパラメータが無効です。
invalid type	取引タイプが不正です。

3.1.7 補足

- 戻り値

  処理に成功した場合、success には 1 が、return には実行結果が設定されます。

  処理に失敗した場合、success には 0 が、return にはエラーメッセージが設定されます。

3.2 個別情報

信用取引 API の個別情報です。

3.2.1 ユーザー自身の取引履歴を取得

信用取引のユーザー自身の取引履歴を取得します。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	get_positions	str	
group_id	No	グループ ID	int	
from	No	この順番のレコードから取得	int	0
count	No	取得するレコード数	int	1000
from_id	No	このトランザクション ID のレコードから取得	int	0
end_id	No	このトランザクション ID のレコードまで取得	int	infinity
order	No	ソート順	str (ASC or DESC)	DESC
since	No	開始タイムスタンプ	UNIX_TIMESTAMP	0
end	No	終了タイムスタンプ	UNIX_TIMESTAMP	infinity
currency_pair	No	通貨ペア。指定なしで全通貨ペア	str (例) btc_jpy	全通貨ペア

戻り値

キー	詳細	型
例）182	注文 ID	int
group_id	グループ ID	int
currency_pair	通貨ペア	str
action	bid(買い) or ask(売り)	str
amount	数量	float
price	価格	float
limit	リミット注文価格	float
stop	ストップ注文価格	float
timestamp	発注日時	UNIX_TIMESTAMP
term_end	注文の有効期限	UNIX_TIMESTAMP
leverage	レバレッジ	float
fee_spent	支払い手数料	float
timestamp_closed	クローズ日時	UNIX_TIMESTAMP
price_avg	建玉平均価格	float
amount_done	建玉数	float
close_avg	決済平均価格	float
close_done	決済数	float
deposit_xxx	実際にデポジットした額 (xxx は通貨コード)	float
deposit_price_xxx	デポジット時計算レート (xxx は通貨コード)	float
refunded_xxx	実際に返却した額 (xxx は通貨コード)	float
refunded_price_xxx	実際に返却した額 (xxx は通貨コード)	float
guard_fee	追証ガード手数料	float

```
{
    "success": 1,
    "return": {
        "182": {
            "group_id": 1,
            "currency_pair": "btc_jpy",
            "action": "bid",
            "leverage": 2.5,
            "price": 110005,
            "limit": 130000,
            "stop": 90000,
            "amount": 0.03,
            "fee_spent": 0,
            "timestamp": 1402018713,
            "term_end": 1404610713,
            "timestamp_closed": 1402019000,
```

(次のページに続く)

(前のページからの続き)

```
            "deposit": 35.76 ,
            "deposit_jpy": 35.76,
            "refunded": 35.76 ,
            "refunded_jpy": 35.76,
            "swap": 0,
        }
    }
}
```

補足

- 呼び出しは 60 秒間に 10 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- " since " もしくは " end " をセットした場合、" order " は強制的に " ASC " となります。
- " from_id " もしくは " end_id " をセットした場合、" order " は強制的に " ASC " となります。

3.2.2 ユーザー自身の取引履歴明細を取得

信用取引のユーザー自身の取引履歴の明細を取得します。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	position_history	str	
group_id	No	グループ ID	int	
leverage_id	Yes	注文 ID	int	

戻り値

キー	詳細	型
例）182	注文 ID	int
group_id	グループ ID	int
currency_pair	通貨ペア	str
action	bid(買い) or ask(売り)	str
amount	数量	float
price	価格	float
timestamp	発注日時	UNIX_TIMESTAMP
your_action	bid(買い) or ask(売り)、自己取引の場合は both	str
bid_leverage_id	買い注文 ID(自分の注文の場合のみ)	int
ask_leverage_id	売り注文 ID(自分の注文の場合のみ)	int


```
{
    "success": 1,
    "return": {
        "182": {
            "group_id": 1,
            "currency_pair": "btc_jpy",
            "action": "bid",
            "amount": 0.0001,
            "price": 499000
            "timestamp": 1504251232
            "your_action": "bid",
            "bid_leverage_id": 182,
        },
        "183": {
            "group_id": 1,
            "currency_pair": "btc_jpy",
            "action": "ask",
            "amount": 0.0001,
            "price": 450000
            "timestamp": 1504251267
            "your_action": "ask",
            "ask_leverage_id": 182,
        },

    }
}
```

エラーメッセージ

メッセージ	詳細
order not found	注文が見つかりません。

補足

- 呼び出しは 60 秒間に 10 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- leverage_id は**ユーザー自身の取引履歴**または**未約定注文一覧**で取得できます。

3.2.3 未約定注文一覧の取得

信用取引の現在有効な注文一覧を取得します（未約定注文一覧）。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	active_positions	str	
group_id	No	グループ ID	int	
currency_pair	No	通貨ペア。指定なしで全通貨ペア	str(例) btc_jpy	全通貨ペア

戻り値

キー	詳細	型
例）182	注文 ID	int
group_id	グループ ID	int
currency_pair	通貨ペア	str
action	bid(買い) or ask(売り)	str
amount	数量	float
price	価格	float
limit	リミット注文価格	float
stop	ストップ注文価格	float
timestamp	発注日時	UNIX_TIMESTAMP
term_end	注文の有効期限	UNIX_TIMESTAMP
leverage	レバレッジ	float
fee_spent	支払い手数料	float
price_avg	建玉平均価格	float
amount_done	建玉数	float
close_avg	決済平均価格	float
close_done	決済数	float
deposit_xxx	実際にデポジットした額 (xxx は通貨コード)	float
deposit_price_xxx	デポジット時計算レート (xxx は通貨コード)	float

```
{
    "success": 1,
    "return": {
        "184": {
            "group_id": "1",
            "currency_pair": "btc_jpy",
            "action": "ask",
            "amount": 0.0001,
            "price": 450000,
            "timestamp": 1402021125,
            "term_end": 1404613125,
            "leverage": 1,
            "fee_spent": 0.0015,
            "price_avg": 450000,
            "amount_done": 0.0001,
            "deposit_jpy": 48.72
        }
    }
}
```

補足

- 呼び出しは 10 秒間に 20 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。

3.2.4 注文

信用取引の注文を行います。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	create_position	str	
group_id	No	グループ ID	int	
currency_pair	Yes	通貨ペア	str(例) btc_jpy	
action	Yes	bid(買い) or ask(売り)	str	
amount	Yes	数量	numerical	
price	Yes	価格	numerical	
leverage	Yes	レバレッジ	numerical	
limit	No	リミット注文価格	numerical	
stop	No	ストップ注文価格	numerical	

戻り値

キー	詳細	型
leverage_id	注文 ID	int
timestamp	注文日時	UNIX_TIMESTAMP
term_end	注文の有効期限	UNIX_TIMESTAMP
price_avg	建玉平均価格	float
amount_done	建玉数	float
deposit_xxx	実際にデポジットした額 (xxx は通貨コード)	float
deposit_price_xxx	デポジット時計算レート (xxx は通貨コード)	float
funds	残高	dict

```
{
    "success": 1,
    "return": {
```

(次のページに続く)

(前のページからの続き)


```
        "leverage_id": 22258,
        "timestamp": 1504253833,
        "term_end": 1506845833,
        "price_avg": 118000,
        "amount_done": 0.0001,
        "deposit_jpy": 11.92,
        "funds": {
            "jpy": 325,
            "btc": 1.392,
            "mona": 2600
        }
    }
}
```


エラーメッセージ

メッセージ	詳細
trade temporarily unavailable	取引が一時的に停止されています。
your account is restricted now, KYC required.	本人確認が完了していないため、取引ができません。本人確認を完了させて下さい。
insufficient funds	取引に必要な残高が存在しません。

補足

- 呼び出しは 10 秒間に 3 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- パラメータ limit について

  リミット値（利確のための反対売買の指値）を指定することができます。リミット値を指定した場合、注文が成立した分だけの数量について、自動的にリミット注文が発行されます。

- パラメータ stop について

  ストップ値を指定した場合、設定価格まで下落（あるいは上昇）した場合に成行にて決済を試みます。成行注文のため必ずしも設定価格で決済されることを保証する物ではございません。

- 価格および数量の数値について

  下記の単位以外で注文しようとした場合、invalid price parameter または invalid amount parameter というエラーが返されます。

  - 価格（price および limit）

btc_jpy : 5 円単位

– 数量（amount）

btc_jpy : 0.0001BTC 単位

3.2.5 注文の変更

信用取引の注文の変更を行います。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	change_position	str	
group_id	No	グループ ID	int	
leverage_id	Yes	注文 ID	int	
price	Yes	価格	numerical	
limit	No	リミット注文価格	numerical	
stop	No	ストップ注文価格	numerical	

戻り値

キー	詳細	型
leverage_id	注文 ID	int
timestamp_closed	クローズ日時	UNIX_TIMESTAMP
price_avg	建玉平均価格	float
amount_done	建玉数	float
close_avg	決済平均価格	float
close_done	決済数	float
refunded_xxx	実際に返却した額 (xxx は通貨コード)	float
refunded_price_xxx	実際に返却した額 (xxx は通貨コード)	float
guard_fee	追証ガード手数料	float

```
{
    "success": 1,
    "return": {
        "leverage_id": 22258,
```

(次のページに続く)

(前のページからの続き)

```
        "price_avg": 118000,
        "amount_done": 0.0001,
    }
}
```

エラーメッセージ

メッセージ	詳細
order not found	注文が見つかりません。
order already closed	注文が既にクローズしています。
trade temporarily unavailable	取引が一時的に停止されています。

補足

- 呼び出しは 10 秒間に 3 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- パラメータ limit について

  limit を指定しなかった場合、設定済みのリミット注文は取り消されます。継続してリミット注文を出し続けたい場合は毎回セットしてください。リミット値を指定した場合、注文が成立した分だけの数量について、自動的にリミット注文が発行されます。
- パラメータ stop について

  stop を指定しなかった場合、設定済みのストップ注文は取り消されます。継続してストップ注文を出し続けたい場合は毎回必ずセットしてください。ストップ値を指定した場合、設定価格まで下落（あるいは上昇）した場合に成行にて決済を試みます。成行注文のため必ずしも設定価格で決済されることを保証する物ではございません。
- 価格および数量の数値について

  下記の単位以外で注文しようとした場合、invalid price parameter または invalid amount parameter というエラーが返されます。

  - 価格（price および limit）

    btc_jpy : 5 円単位

  - 数量（amount）

    btc_jpy : 0.0001BTC 単位
- leverage_id は**ユーザー自身の取引履歴**または**未約定注文一覧**で取得できます。

3.2.6 注文の取消し

信用取引の注文の取消しを行います。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	cancel_position	str	
group_id	No	グループ ID	int	
leverage_id	Yes	注文 ID	int	

戻り値

キー	詳細	型
leverage_id	注文 ID	int
fee_spent	支払い手数料	float
timestamp_closed	クローズ日時	UNIX_TIMESTAMP
price_avg	建玉平均価格	float
amount_done	建玉数	float
close_avg	決済平均価格	float
close_done	決済数	float
refunded_xxx	実際に返却した額 (xxx は通貨コード)	float
refunded_price_xxx	実際に返却した額 (xxx は通貨コード)	float
guard_fee	追証ガード手数料	float
funds	残高	dict

```
{
    'success': 1,
    'return': {
        'leverage_id': 2072,
        'refunded_jpy': 645.96,
        'funds': {
            'btc': 0.496,
            'jpy': 1564.96,
            'xem': 0.0,
            'mona': 10.0
        },
```

(次のページに続く)

(前のページからの続き)

```
        'fee_spent': 0.0,
        'timestamp_closed': '1508384951',
        'swap': 0.0
    }
}
```

エラーメッセージ

メッセージ	詳細
order not found	注文が見つかりません。
order already closed	注文が既にクローズしています。
order is too new	注文から一定時間の経過が必要です。

補足

- 呼び出しは 10 秒間に 3 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- leverage_id は**ユーザー自身の取引履歴**または**未約定注文一覧**で取得できます。

第 4 章

AirFX API

4.1 共通情報

AirFX API の共通情報です。

4.1.1 事前準備

- AirFX API を利用するには、アカウント情報のページから API Key の発行をおこなってください。

4.1.2 リクエスト方法

- エンドポイント：https://api.zaif.jp/tlapi
- メソッド：POST

4.1.3 認証

- 取得した API Keys を利用して、下記のように HTTP ヘッダを設定し、認証情報を送信します。

キー	詳細	例
key	API キー	490f983a-5fab-49b2-b789-9d1f130874d3
sign	署名	詳細は下記

注釈: sign は POST する全てのパラメータ（nonce と method およびメソッド毎のパラメータ）を URL エンコードしたクエリ形式 (param1=val1&param2=val2) のメッセージとして、Secret Key を用いて HMAC-SHA512 で署名します。

4.1.4 パラメータ

キー	詳細	例
nonce	1 以上の数	23123
method	API メソッド名	get_info
type	取引タイプ	futures
group_id	グループ ID	1

注釈: メソッド毎の固有のパラメータも全て POST パラメータにて送信してください。nonce パラメータの値は実効毎に増分されていないとエラーが発生します。また、増分量は少数点以下の値にも対応しております。

4.1.5 戻り値

キー	詳細	型
success	成功フラグ	int
return	実行結果	dict or string

```
{
    "success": 1,
    "return": {
        ...
    }
}
```

4.1.6 エラーメッセージ

メッセージ	詳細
method not found	指定されたメソッドが存在しません。
no data found for the key	API キーが無効です。
time wait restriction, please later	同じメソッドが短時間に多く呼び出しされたときに発生します。しばらく待ってから、再度お試しください。
signature mismatch	署名が不適切です。
invalid access token	無効なトークンが指定されています。
expired access token	トークンの有効期限が切れています。トークン再発行 API を参考にし、トークンの再発行をしてください。
nonce not incremented	前回 API 実行時より nonce 値が加算されていません。
nonce out of range	値が最大値を超えています。新しい API キーを発行してください。
api key don ’t have {} permission	API Key に権限がありません。
invalid {} parameter	指定されているパラメータが無効です。
invalid type	取引タイプが不正です。

4.1.7 補足

- 戻り値

  処理に成功した場合、success には 1 が、return には実行結果が設定されます。

  処理に失敗した場合、success には 0 が、return にはエラーメッセージが設定されます。

4.2 個別情報

AirFX API の個別情報です。

4.2.1 ユーザー自身の取引履歴を取得

AirFX のユーザー自身の取引履歴を取得します。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	get_positions	str	
group_id	Yes	グループ ID	int	
from	No	この順番のレコードから取得	int	0
count	No	取得するレコード数	int	1000
from_id	No	このトランザクション ID のレコードから取得	int	0
end_id	No	このトランザクション ID のレコードまで取得	int	infinity
order	No	ソート順	str (ASC or DESC)	DESC
since	No	開始タイムスタンプ	UNIX_TIMESTAMP	0
end	No	終了タイムスタンプ	UNIX_TIMESTAMP	infinity
currency_pair	No	通貨ペア。指定なしで全通貨ペア	str (例) btc_jpy	全通貨ペア

戻り値

キー	詳細	型
例）182	注文 ID	int
group_id	グループ ID	int
currency_pair	通貨ペア	str
action	bid(買い) or ask(売り)	str
amount	数量	float
price	価格	float
limit	リミット注文価格	float
stop	ストップ注文価格	float
timestamp	発注日時	UNIX_TIMESTAMP
term_end	注文の有効期限	UNIX_TIMESTAMP
leverage	レバレッジ	float
fee_spent	支払い手数料	float
timestamp_closed	クローズ日時	UNIX_TIMESTAMP
price_avg	建玉平均価格	float
amount_done	建玉数	float
close_avg	決済平均価格	float
close_done	決済数	float
deposit_xxx	実際にデポジットした額 (xxx は通貨コード)	float
deposit_price_xxx	デポジット時計算レート (xxx は通貨コード)	float
refunded_xxx	実際に返却した額 (xxx は通貨コード)	float
refunded_price_xxx	実際に返却した額 (xxx は通貨コード)	float
swap	受け取ったスワップの額	float


```
{
    "success": 1,
    "return": {
        "182": {
            "group_id": 1,
            "currency_pair": "btc_jpy",
            "action": "bid",
            "leverage": 2.5,
            "price": 110005,
            "limit": 130000,
            "stop": 90000,
            "amount": 0.03,
            "fee_spent": 0,
            "timestamp": 1402018713,
            "term_end": 1404610713,
            "timestamp_closed": 1402019000,
```


(次のページに続く)

(前のページからの続き)

```
            "deposit": 35.76 ,
            "deposit_jpy": 35.76,
            "refunded": 35.76 ,
            "refunded_jpy": 35.76,
            "swap": 0,
        }
    }
}
```

補足

- 呼び出しは 60 秒間に 10 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- " since " もしくは " end " をセットした場合、" order " は強制的に " ASC " となります。
- " from_id " もしくは " end_id " をセットした場合、" order " は強制的に " ASC " となります。

4.2.2 ユーザー自身の取引履歴明細を取得

AirFX のユーザー自身の取引履歴の明細を取得します。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	create_position	str	
group_id	Yes	グループ ID	int	
leverage_id	Yes	注文 ID	int	

戻り値

キー	詳細	型
例）182	注文 ID	int
group_id	グループ ID	int
currency_pair	通貨ペア	str
action	bid(買い) or ask(売り)	str
amount	数量	float
price	価格	float
timestamp	発注日時	UNIX_TIMESTAMP
your_action	bid(買い) or ask(売り)、自己取引の場合は both	str
bid_leverage_id	買い注文 ID(自分の注文の場合のみ)	int
ask_leverage_id	売り注文 ID(自分の注文の場合のみ)	int


```
{
    "success": 1,
    "return": {
        "182": {
            "group_id": 1,
            "currency_pair": "btc_jpy",
            "action": "bid",
            "amount": 0.0001,
            "price": 499000
            "timestamp": 1504251232
            "your_action": "bid",
            "bid_leverage_id": 182,
        },
        "183": {
            "group_id": 1,
            "currency_pair": "btc_jpy",
            "action": "ask",
            "amount": 0.0001,
            "price": 450000
            "timestamp": 1504251267
            "your_action": "ask",
            "ask_leverage_id": 182,
        },

    }
}
```

エラーメッセージ

メッセージ	詳細
order not found	注文が見つかりません。

補足

- 呼び出しは 60 秒間に 10 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- leverage_id は**ユーザー自身の取引履歴**または**未約定注文一覧**で取得できます。

4.2.3 未約定注文一覧の取得

AirFX の現在有効な注文一覧を取得します（未約定注文一覧）。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	active_positions	str	
group_id	Yes	グループ ID	int	
currency_pair	No	通貨ペア。指定なしで全通貨ペア	str(例) btc_jpy	全通貨ペア

戻り値

キー	詳細	型
例）182	注文 ID	int
group_id	グループ ID	int
currency_pair	通貨ペア	str
action	bid(買い) or ask(売り)	str
amount	数量	float
price	価格	float
limit	リミット注文価格	float
stop	ストップ注文価格	float
timestamp	発注日時	UNIX_TIMESTAMP
term_end	注文の有効期限	UNIX_TIMESTAMP
leverage	レバレッジ	float
fee_spent	支払い手数料	float
price_avg	建玉平均価格	float
amount_done	建玉数	float
close_avg	決済平均価格	float
close_done	決済数	float
deposit_xxx	実際にデポジットした額 (xxx は通貨コード)	float
deposit_price_xxx	デポジット時計算レート (xxx は通貨コード)	float
swap	受け取ったスワップの額	float

```
{
    "success": 1,
    "return": {
        "184": {
            "group_id": "1",
            "currency_pair": "btc_jpy",
            "action": "ask",
            "amount": 0.0001,
            "price": 450000,
            "timestamp": 1402021125,
            "term_end": 1404613125,
            "leverage": 1,
            "fee_spent": 0.0015,
            "price_avg": 450000,
            "amount_done": 0.0001,
            "deposit_jpy": 48.72
        }
    }
}
```

補足

- 呼び出しは 10 秒間に 20 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。

4.2.4 注文

AirFX の注文を行います。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	create_position	str	
group_id	Yes	グループ ID	int	
currency_pair	Yes	通貨ペア	str(例) btc_jpy	
action	Yes	bid(買い) or ask(売り)	str	
amount	Yes	数量	numerical	
price	Yes	価格	numerical	
leverage	Yes	レバレッジ	numerical	
limit	No	リミット注文価格	numerical	
stop	No	ストップ注文価格	numerical	

戻り値

キー	詳細	型
leverage_id	注文 ID	int
timestamp	注文日時	UNIX_TIMESTAMP
term_end	注文の有効期限	UNIX_TIMESTAMP
price_avg	建玉平均価格	float
amount_done	建玉数	float
deposit_xxx	実際にデポジットした額 (xxx は通貨コード)	float
deposit_price_xxx	デポジット時計算レート (xxx は通貨コード)	float
funds	残高	dict

```
{
    "success": 1,
    "return": {
```

(次のページに続く)

(前のページからの続き)


```
        "leverage_id": 22258,
        "timestamp": 1504253833,
        "term_end": 1506845833,
        "price_avg": 118000,
        "amount_done": 0.0001,
        "deposit_jpy": 11.92,
        "funds": {
            "jpy": 325,
            "btc": 1.392,
            "mona": 2600
        }
    }
}
```


エラーメッセージ

メッセージ	詳細
trade temporarily unavailable	取引が一時的に停止されています。
your account is restricted now, KYC required.	本人確認が完了していないため、取引ができません。本人確認を完了させて下さい。
insufficient funds	取引に必要な残高が存在しません。

補足

- 呼び出しは 10 秒間に 3 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- パラメータ limit について

  リミット値（利確のための反対売買の指値）を指定することができます。リミット値を指定した場合、注文が成立した分だけの数量について、自動的にリミット注文が発行されます。

- パラメータ stop について

  ストップ値を指定した場合、設定価格まで下落（あるいは上昇）した場合に成行にて決済を試みます。成行注文のため必ずしも設定価格で決済されることを保証する物ではございません。

- 価格および数量の数値について

  下記の単位以外で注文しようとした場合、invalid price parameter または invalid amount parameter というエラーが返されます。

  - 価格（price および limit）

btc_jpy : 5 円単位

– 数量（amount）

btc_jpy : 0.0001BTC 単位

4.2.5 注文の変更

AirFX 注文の変更を行います。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	change_position	str	
group_id	Yes	グループ ID	int	
leverage_id	Yes	注文 ID	int	
price	Yes	価格	numerical	
limit	No	リミット注文価格	numerical	
stop	No	ストップ注文価格	numerical	

戻り値

キー	詳細	型
leverage_id	注文 ID	int
timestamp_closed	クローズ日時	UNIX_TIMESTAMP
price_avg	建玉平均価格	float
amount_done	建玉数	float
close_avg	決済平均価格	float
close_done	決済数	float
refunded_xxx	実際に返却した額 (xxx は通貨コード)	float
refunded_price_xxx	実際に返却した額 (xxx は通貨コード)	float
swap	受け取ったスワップの額	float

```
{
    "success": 1,
    "return": {
        "leverage_id": 22258,
```

(次のページに続く)

(前のページからの続き)

```
        "price_avg": 118000,
        "amount_done": 0.0001,
    }
}
```

エラーメッセージ

メッセージ	詳細
order not found	注文が見つかりません。
order already closed	注文が既にクローズしています。
trade temporarily unavailable	取引が一時的に停止されています。

補足

- 呼び出しは 10 秒間に 3 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- パラメータ limit について

  limit を指定しなかった場合、設定済みのリミット注文は取り消されます。継続してリミット注文を出し続けたい場合は毎回セットしてください。リミット値を指定した場合、注文が成立した分だけの数量について、自動的にリミット注文が発行されます。
- パラメータ stop について

  stop を指定しなかった場合、設定済みのストップ注文は取り消されます。継続してストップ注文を出し続けたい場合は毎回必ずセットしてください。ストップ値を指定した場合、設定価格まで下落（あるいは上昇）した場合に成行にて決済を試みます。成行注文のため必ずしも設定価格で決済されることを保証する物ではございません。
- 価格および数量の数値について

  下記の単位以外で注文しようとした場合、invalid price parameter または invalid amount parameter というエラーが返されます。

  - 価格（price および limit）

    btc_jpy : 5 円単位

  - 数量（amount）

    btc_jpy : 0.0001BTC 単位
- leverage_id は**ユーザー自身の取引履歴**または**未約定注文一覧**で取得できます。

4.2.6 注文の取消し

AirFX 注文の取消しを行います。

パラメータ

パラメータ	必須	詳細	型	デフォルト
method	Yes	cancel_order	str	
group_id	Yes	グループ ID	int	
leverage_id	Yes	注文 ID	int	

戻り値

キー	詳細	型
leverage_id	注文 ID	int
fee_spent	支払い手数料	float
timestamp_closed	クローズ日時	UNIX_TIMESTAMP
price_avg	建玉平均価格	float
amount_done	建玉数	float
close_avg	決済平均価格	float
close_done	決済数	float
refunded_xxx	実際に返却した額 (xxx は通貨コード)	float
refunded_price_xxx	実際に返却した額 (xxx は通貨コード)	float
swap	受け取ったスワップの額	float
funds	残高	dict

```
{
    'success': 1,
    'return': {
        'leverage_id': 2072,
        'refunded_jpy': 645.96,
        'funds': {
            'btc': 0.496,
            'jpy': 1564.96,
            'xem': 0.0,
            'mona': 10.0
        },
```

(次のページに続く)

(前のページからの続き)

```
        'fee_spent': 0.0,
        'timestamp_closed': '1508384951',
        'swap': 0.0
    }
}
```

エラーメッセージ

メッセージ	詳細
order not found	注文が見つかりません。
order already closed	注文が既にクローズしています。
order is too new	注文から一定時間の経過が必要です。

補足

- 呼び出しは 10 秒間に 3 回以下におさまるようにしてください。呼び出しが多すぎるとアクセス拒否されることがあります。
- leverage_id は**ユーザー自身の取引履歴**または**未約定注文一覧**で取得できます。

第 5 章

WebSocket API

5.1 WebSocket API 利用手順

websocket を利用したリアルタイム板情報と終値の API を配信しています。 本 API が切断され再接続する場合は、１つの IP アドレスからの接続開始が 4 回/秒程度におさまるようにしてください。

5.2 リクエスト方法

wss://ws.zaif.jp/stream?currency_pair={currency_pair}

ws://ws.zaif.jp/stream?currency_pair={currency_pair}

- ストリーミングは上記 2 つの接続方法があります。
- currency_pair に指定できる値は currency_pairs で取得できる通貨ペア情報の内、is_token が false になっているものです。

5.3 戻り値

```
{
        "asks":[
            [
                30000.0,
                0.1
            ],
            [
                30010.0,
                0.2
            ],
            ...
```

(次のページに続く)

(前のページからの続き)

```
        ],
        "bids":[
            [
                29500.0,
                0.5
            ],
            [
                29300.0,
                0.1
            ],
            ...
        ],
        "trades":[
            {
                "currenty_pair":"btc_jpy",
                "trade_type":"ask",
                "price":30001,
                "tid":123,
                "amount":0.02,
                "date":1427879761
            },
        ...,
        "timestamp":"2015-04-01 18:16:01.739990",
        "last_price:{
            "action":"ask",
            "price":30001
        },
        "currency_pair":"btc_jpy"
}
```

第 6 章

OAuth API

6.1 OAuth 認証機能の利用手順

Zaif のアカウント情報を利用したサービス (以下クライアントサービス) を展開する業者 (以下クライアント) が OAuth 認証機能を利用できるようにするための手順を記述します。

6.1.1 1. サービス基本情報の登録

Zaif にログインし、『アカウント』→『連携アプリケーション基本設定』から「連携アプリケーションの情報を新しく作成する」ボタンを押下し、サービスの詳細な情報を設定してください。

項目名	必須	詳細	入力例
名称	Yes	クライアントが各設定を見分けるための名称です。	自動取引 BOT 設定
リダイレクト URI	No ※1	認可クリア後、必要なパラメータを受け取るためのリダイレクト URL 設定してください。**設定することを強く推奨しています。**	https://auto.bot.co.jp/zaif_oauth
サービス名称	Yes	サービスの名称を設定してください。	自動取引 BOT
サービスイメージ URL	No	サービスのイメージ画像の URL を設定してください。	https://auto.bot.co.jp/service_image.png
サービス URL	No	サービスの URL を設定してください。	https://auto.bot.co.jp/
サービス利用規約 URL	No	サービスの利用規約の URL を設定してください。	https://auto.bot.co.jp/terms
サービスプライバシーポリシー URL	No	サービスのプライバシープリシーの URL を設定してください。	https://auto.bot.co.jp/policy

注釈: ※ 1 未設定の場合、後述する認可画面リダイレクトパラメータに指定される redirect_uri が利用されます。

6.1.2 2.Zaif アカウント認可画面へのリダイレクト

クライアントサービスのログイン画面等に、Zaif アカウント認可画面 (https://zaif.jp/oauth) へ必要なパラメータを追加してリダイレクトを行えるようにしてください。

項目名	必須	詳細	入力例
client_id	Yes	連携アプリケーション基本情報登録時に発行される ID を指定してください。	a3823d2a30f24e39980ebe7943b55737
response_type	Yes	固定	code
scope	Yes	認可したい機能を半角空白 (エンコード時は %20) で区切って指定してください、詳細は下記 scope 詳細をご確認ください。	info%20trade
state	Ye	リダイレクトするたびに一意の文字列を指定してください。	2a99cc45cef04c358dbc26db880f9d03
redirect_uri	No ※ 1	発行された code(後述) を取得するためのリダイレクト先を指定してください。	http://your_domain/zaif_redirect
lang	No ※ 2	表示言語を明示的に指定したい場合に指定してください。	en

注釈: ※ 1 未指定の場合はサービス基本情報登録時に指定したリダイレクト URL が使用されます。

※ 2 未指定の場合ブラウザの言語設定を参照して自動で言語情報を切り替えます。

scope 詳細

名称	詳細	補足
info	データ参照	
trade	通貨のトレード	
withdraw	口座への引き出し	すべてのクライアントが指定できるわけではありません。詳しくはサポートまでご連絡ください。

リダイレクトに成功すると、下記のような認証画面が表示されます。

6.1.3 3. リダイレクトされた情報の解析

サービス利用ユーザ (以下ユーザ) が認証に成功すると、指定したリダイレクト URL にリクエストが発行されるので、パラメータを解析し、認可処理を行ってください。

キー	詳細	例
code	Zaif が発行した一意の文字列です。後述のトークン発行時に利用します。	cb533e906a984b0e8ba4efae3af0c7cb
state	リダイレクト時に付与した state パラメータと同値です。成りすまし対策として、必ず同値であるかチェックしてください。	2a99cc45cef04c358dbc26db880f9d03

6.1.4 4. トークン発行リクエスト

前述の code パラメータを使って、token が取得できる Web API を実行し、token を取得します。なお、戻り値は JSON になります。

- エンドポイント：https://oauth.zaif.jp/oauth/v1/token
- メソッド：POST

トークン発行 API パラメータ

パラメータ	必須	詳細	例
grant_type	Yes	固定	authorization_code
code	Yes	code	リダイレクトされた code 値を指定してください。
client_id	Yes	9r88i445cef04c358dbc26db880f9d03	アプリケーション基本情報登録時に発行されたクライアント ID を指定してください。
client_secret	Yes	2a99cc45cef04c358dbc26db880f9d03	アプリケーション基本情報登録時に発行されたクライアントシークレットを指定してください。
redirect_uri	No ※ 1	http://your_domain/zaif_redirect	リダイレクトしたい URL を指定してください。

注釈: ※ 1 認可画面リダイレクト時に指定している場合必ず同値を指定してください。

トークン発行 API 戻り値

キー	詳細	例
token_type	固定	bearer
state	リダイレクト時に付与した state パラメータと同値です。	2a99cc45cef04c358dbc26db880f9d03
access_token	API を利用する時に指定するトークンです。	bb12f3de5df2472290ff15331824a9cf
refresh_token	利用期限が切れた access token を再発行するために使用します。	ef972ad13e484e17abffbfd5dba51750
expires_in	access token の期限です。単位は秒です。	3600

6.1.5 5.API 実行

今まで HTTP ヘッダに key、sign パラメータを付与して実行していた取引 API ですが、取得した token を利用すればそれらは必要なくなります。 発行された access token を token パラメータとしてリクエスト発行時に HTTP ヘッダに付与し、API を実行して下さい。

6.1.6 6.access token の期限が切れた場合

期限が切れた access token は利用できなくなります。下記 token の再発行 Web API を利用して、token を再発行して下さい。

- エンドポイント:https://oauth.zaif.jp/oauth/v1/refresh_token
- メソッド：POST

トークン再発行 API パラメータ

パラメータ	必須	詳細	例
grant_type	Yes	固定	refresh_token
refresh_token	Yes	トークン発行 API 実行時に取得した refresh token を指定してください。	ef972ad13e484e17abffbfd5dba51750
client_id	Yes	アプリケーション基本情報登録時に発行されたクライアント ID を指定してください。	9r88i445cef04c358dbc26db880f9d03
client_secret	Yes	アプリケーション基本情報登録時に発行されたクライアントシークレットを指定してください。	2a99cc45cef04c358dbc26db880f9d03

トークン再発行 API 戻り値

キー	詳細	例
token_type	固定	bearer
access_token	API を利用する時に指定するトークンです。	bb12f3de5df2472290ff15331824a9cf
refresh_token	利用期限が切れた access token を再発行するために使用します。	ef972ad13e484e17abffbfd5dba51750
expires_in	access token の期限です。単位は秒です。	3600

6.1.7 補足

- ユーザが認証したアプリケーションの情報を削除したい場合は、『アカウント』→『連携アプリケーション一覧』を選択し、削除したいアプリケーション情報の削除ボタンを押下してください。

第 7 章

決済 API

7.1 共通情報

決済 API の共通情報です。決済 API は、決済サービス『Zaif Payment』を導入するために使用します。

7.1.1 事前準備

- 決済 API の利用には事業者用アカウントおよび API Key が必要になります。
- アカウント情報 のページから API Key の発行をおこなってください。

7.1.2 リクエスト方法

- エンドポイント：https://api.zaif.jp/ecapi
- メソッド：POST

7.1.3 認証

- 取得した API Keys を利用して、下記のように HTTP ヘッダを設定し、認証情報を送信します。

パラメータ	詳細	例
key	API キー	490f983a-5fab-49b2-b789-9d1f130874d3
md5secret	Secret Key を md5 ハッシュ化した文字列	
sha1secret	Secret Key を sha1 ハッシュ化した文字列	

注釈: "md5secret"と"Secret Key"はどちらかをセット

7.1.4 パラメータ

キー	詳細	例
nonce	1 以上の数	23123
method	API メソッド名	get_info

注釈: メソッド毎の固有のパラメータも全て POST パラメータにて送信してください。nonce パラメータの値は実効毎に増分されていないとエラーが発生します。また、増分量は少数点以下の値にも対応しております。

7.1.5 戻り値

キー	詳細	型
success	成功フラグ	int
return	実行結果	dict or string

```
{
    "success": 1,
    "return": {
        ...
    }
}
```

7.1.6 エラーメッセージ

メッセージ	詳細
nonce not incremented	前回 API 実行時より nonce 値が加算されていません。
account has not permission to use ec api	アカウントが決済 API を使用する権限がありません。
invalid parameter {}	パラメータが不正です。

7.1.7 補足

- 戻り値

  処理に成功した場合、success には 1 が、return には実行結果が設定されます。処理に失敗した場合、success には 0 が、return にはエラーメッセージが設定されます。

インボイスの表示

作成したインボイスから支払フォームを表示することにより、利用者から Bitcoin/Monacoin による支払いを促します。 支払いフォームの URL は以下の通りです。

https://zaif.jp/invoice/form/{invoiceId}

{invoiceId}はインボイスの作成時に発行された ID になります。 下記のようにして iframe による表示を行うことも可能です。

```
<iframe id="zaif_ec_iframe"
scrolling="no"
allowtransparency="true"
```

(次のページに続く)

(前のページからの続き)

```
frameborder="0"  src='https://zaif.jp/invoice/iframe/{invoiceId}'
style='width:500px; overflow: hidden; padding:10px;'></iframe>
```

また、インボイス作成時に取得したデータを利用し、事業者様の EC サイト上で独自にフォームを表示していただくことも可能です。

7.2 個別情報

決済 API の個別情報です。

7.2.1 インボイスの作成

決済金額・商品名・通貨などの情報を送信してインボイスを作成し、暗号通貨による決済を開始します。 インボイスを作成すると、決済用の Bitcoin または Monacoin アドレスが発行され、暗号通貨建ての請求が行なわれます。インボイスには有効期限（現在３０分としていますが、これは変更になる可能性があります）があり、有効期限内に顧客が決済用のアドレスへ支払を行うことにより、決済が完了します。

パラメータ

パラメータ	必須	詳細	型	例
method	Yes	createInvoice	str	
speed	No	決済完了とみなすスピード。	str	high medium low
notificationUri	No	決済完了したタイミングでの通知先 URI 事業者様の EC サイトシステムに通知を行うためのものになります。	str	
notificationMethod	No	決済完了したタイミングでの通知先 URI へ通知する際に使用される HTTP メソッド。デフォルトは POST になります。	str	GET または POST
redirectUri	No	決済フォームで着金後、EC サイトへ戻るためのリダイレクト先の URI。設定されなかった場合はリダイレクトせず着金後のステータスが表示されます。	str	
currency	Yes	決済に使用する暗号通貨	str	btc または mona
amount	Yes	決済金額（日本円）。実際の請求対象金額。1 円単位、カンマ無し。	int	
subTotal	No	小計（日本円）	int	
tax	No	消費税（日本円）	int	
regularPrice	No	定価（日本円）	int	
discount	No	割引額（日本円）	int	
merchantName	No	店舗名	str	
orderNumber	No	注文番号。店舗側での識別用に任意の番号やコードを利用することができます。	str	
referenceNumber	No	リファレンス番号。店舗側での識別用に任意の番号やコードを利用することができます。	str	
itemName	No	商品名	str	
buyerId	No	利用者 ID	str	
buyerName	No	利用者名	str	
buyerKana	No	利用者ふりがな	str	
buyerZip	No	利用者郵便番号	str	
buyerAddr1	No	利用者住所 1	str	
buyerAddr2	No	利用者住所 2	str	

注釈: speed（決済スピード）について

high とすると、暗号通貨ネットワーク上での送金トランザクションについて、確認前の状態でも、着金次第決済完了とみなします。 medium とすると、1 件以上の確認が入ったタイミングで決済完了とみなします。これは bitcoin で平均 10 分、monacoin で平均 2 分程度になります。 low とすると、6 件以上の確認が入ったタイミングで決済完了とみなします。これは bitcoin で平均 1 時間、monacoin で平均 12 分程度になります。 high とすると顧客側の送金が完了したタイミングとほぼ同時に着金し、決済完了とみなされますが、万一このトランザクションがネットワーク上で認証されなかった場合、決済が後から取り消しされる場合があります。

注釈: 利用者の氏名・住所などについて

利用者の氏名・住所・電話番号などのフィールドについては、送信していただくと決済フォームに表示されます。ただし、利用者 ID については表示されません。 決済フォームはインボイス ID がもれない限りアクセスすることができませんが、インターネット上ではアクセス制限なしに公開される状態になりますので、個人情報保護の観点から、必要でない場合（注文番号などから顧客が関連付けできる場合）は顧客の情報を送信されないことをお勧めします。

戻り値

キー	詳細	型
invoiceId	作成したインボイスを識別するための ID	str
invoiceUri	作成したインボイスに対する支払フォームの URI	str
invoiceIframeUri	作成したインボイスに対する iframe 版支払フォームの URI	str
created	インボイス作成日時	int
expired	インボイスの有効期限	int
amount	決済対象金額（送信された金額）	int
currency	決済対象の暗号通貨	str
rate	決済時の換算レート	int
btc	Bitcoin による請求額（bitcoin による決済時のみ）	int
mona	Monacoin による請求額（monacoin による決済時のみ）	int
address	Bitcoin また Monacoin の決済用支払先アドレス	str
speed	決済スピード（送信されたものまたはデフォルトで適用されたもの)	str
orderNumber	送信された注文番号（送信された場合のみ）	str
referenceNumber	送信されたリファレンス番号（送信された場合のみ）	str
buyerId	送信された利用者 ID（送信された場合のみ）	str

```
{
    "success": 1,
    "return": {
        "invoiceId": "d7dd735c-1650-11e5-b412-4437e6999eec",
        "invoiceUri": "https://zaif.jp/invoice/form/d7dd735c-1650-11e5-b412-
↪4437e6999eec",
        "invoiceIframeUri": "https://zaif.jp/invoice/iframe/d7dd735c-1650-11e5-b412-
↪4437e6999eec",
        "created": 1434696690,
        "expired": 1434698490,
        "amount": 10800,
        "currency": "btc",
        "rate": "30012",
        "btc": "0.359856",
        "address": "19yhwoY8ysDNy1J1JBZf6nRBsUfLTe2Lvb",
        "BIP21": "bitcoin:19yhwoY8ysDNy1J1JBZf6nRBsUfLTe2Lvb?amount=0.359856",
        "speed": "high",
        "orderNumber": "<the order number if you sent>"
    }
}
```


注釈: 決済完了通知 (notificationUri) について

notificationUri を設定した場合、speed で設定した状態となったタイミングで、決済完了の通知が HTTP(S) で送信されます。

キー	詳細	型
invoiceId	作成したインボイスを識別するための ID	str
settled	決済完了日時	int
amount	決済対象金額（送信された金額）	int
btc	Bitcoin による請求額（bitcoin による決済時のみ）	int
mona	Monacoin による請求額（monacoin による決済時のみ）	int
orderNumber	送信された注文番号（送信された場合のみ）	str
referenceNumbe	送信されたリファレンス番号（送信された場合のみ）	str
buyerId	送信された利用者 ID（送信された場合のみ）	str

警告: notificationMethod に GET を設定した場合は、パラメータは送信されません。 notificationMethod に GET を設定する場合、notificationUri に注文を識別できるような工夫をして設定してください。

通知のエラー時の対応について エラー時の再送については準備中です。

注釈: 決済完了時のリダイレクト (redirectUri) について

顧客が zaif 上の決済フォームを表示したまま送金（支払い）したとき、暗号通貨ネットワーク上で着金を確認したタイミングで自動的にリダイレクトされます。 redirectUri を設定してない場合はリダイレクトされず、こちらのフォームが表示されたままになります。その際、入金ステータスは自動的に更新されます。

注釈: Bitcoin 建てまたは Monacoin 建ての決済

- 円建てではなく、Bitcoin 建てまたは Monacoin 建てでの決済を行うことができます。 createInvoice の billingCurrency パラメータ（一覧にはないパラメータです）に"btc"または"mona"を指定して下さい。このとき currency パラメータも同じ暗号通貨を指定する必要があります。
- 戻り値から rate は削除されることに注意してください。
- BTC または MONA がそのまま決済事業者様のアカウントに精算されますので、決済手数料は完全にゼロ％になりますが、円換算を行う際の相場の変動リスクはそのまま決済事業者様が担うことになりますことにご注意ください。

7.2.2 インボイス情報の取得

作成したインボイスの情報を得ることができます。

パラメータ

パラメータ	必須	詳細	型	例
method	Yes	getInvoice	str	
invoiceId	Yes	発行された invoiceId	str	

戻り値

キー	詳細	型
invoiceId	作成したインボイスを識別するための ID	str
created	インボイス作成日時	int
expired	インボイスの有効期限	int
status	インボイスの状態	str
settled	決済完了日時	int
amount	決済対象金額（送信された金額）	int
currency	決済対象の暗号通貨	str
rate	決済時の換算レート	int
btc	Bitcoin による請求額（bitcoin による決済時のみ）	int
mona	Monacoin による請求額（monacoin による決済時のみ）	int
address	Bitcoin また Monacoin の決済用支払先アドレス	str
BIP21	bitcoin または monacoin の支払い URI	str
speed	決済スピード（送信されたものまたはデフォルトで適用されたもの)	str
orderNumber	送信された注文番号（送信された場合のみ）	str

- インボイスの状態 (status)
  - インボイスには下記のような「状態」があります。
    1. new : 作成され、請求が開始された状態。
    2. paid : 支払先アドレスに対して入金が行なわれ、着金した状態（speed=high の場合の決済完了タイミング)。
    3. confirmed : 支払先アドレスに対する入金が暗号通貨ネットワーク上で 1 確認以上された状態（speed=medium の場合の決済完了タイミング)。
    4. complete : 支払先アドレスに対する入金が暗号通貨ネットワーク上で 6 確認以上された状態（speed=low の場合の決済完了タイミング)。
    5. expired : 支払先アドレスに対して入金が行なわれず、有効期限が切れた状態。
    6. invalid : 支払先アドレスに対して入金が開始されたが、なんらかの理由で確認されなかった状態や、入金金額が不足した状態で有効期限が切れた状態。
    7. canceled : 作成者によりキャンセルされた状態。

       インボイスの状態と他に、決済完了とみなすかどうかについては、インボイス作成時に設定された speed に従って下記の項目が対応します。

    8. settled: 決済完了日時（unixtime)、speed に応じて決済を完了とみなしたタイミングでセットされます。

7.2.3 インボイスの検索

注文番号でインボイスを検索し、インボイス ID を返します。 インボイスの詳細については インボイス情報の取得メソッド を使用してください。

パラメータ

パラメータ	必須	詳細	型	例
method	Yes	getInvoiceIdsByOrderNumber	str	
orderNumber	Yes	注文番号	str	

戻り値

キー	詳細	型
invoiceIds	検索結果	dict
counts	検索結果の件数	int

7.2.4 インボイスのキャンセル

作成したインボイスを取消します。 支払が完了していたり既に有効期限が切れている場合はエラーとなります。部分的に入金されていた場合は、過払いとして処理されます。

パラメータ

パラメータ	必須	詳細	型	例
method	Yes	cancelInvoice	str	
invoiceId	Yes	キャンセルしたい invoiceId	str	

戻り値

キー	詳細	型
invoiceId	作成したインボイスを識別するための ID	str
created	インボイス作成日時	int
expired	インボイスの有効期限	int
status	インボイスの状態	str
settled	決済完了日時	int
amount	決済対象金額（送信された金額）	int
currency	決済対象の暗号通貨	str
rate	決済時の換算レート	int
btc	Bitcoin による請求額（bitcoin による決済時のみ）	int
mona	Monacoin による請求額（monacoin による決済時のみ）	int
address	Bitcoin また Monacoin の決済用支払先アドレス	str
BIP21	bitcoin または monacoin の支払い URI	str
speed	決済スピード（送信されたものまたはデフォルトで適用されたもの)	str
orderNumber	送信された注文番号（送信された場合のみ）	str

注釈: キャンセル失敗時はエラーが帰るため、成功時の status は常に’canceled’ となります。

第 8 章

Q&A

8.1 よくあるお問い合わせ

よくあるお問い合わせ一覧です。

Q. 取引履歴の確認で order_id を使って取引結果を照合できない order_id は取引注文の id で、trade_history API の id は取引結果の id ですので、別物になります。したがって、trade_history API では、order_id を使ってデータを参照することはできません。

Q.withdraw で出金ができない Zaif では不正送金の対策として、アカウントに対する最初の日本円入金から 7 日間は、API による仮想通貨の出金を制限しております。

Q.active_orders の注文 ID が部分約定したのに変わらない active_orders の注文 ID は現在板にある注文の ID になります。その注文が、全約定しない限り同じ ID で残り続けます。また、約定した数量は amount から引かれるため、その注文の残量などを確認したい場合はもう一度 active_orders を実行し、ご確認ください。

Q.count の値を大きくできない一部 API で count パラメータを指定するメソッドがありますが、count にセットできる値の最大値は 1000 になります。もし 1000 より大きい値がセットされた場合、1000 として処理されるのでご注意ください。

第 9 章

その他

9.1 ライブラリへのリンク

Zaif の公開 API/取引 API を各種プログラミング言語から簡単に使用するためのライブラリをご紹介させていただきます。

注釈: ここで紹介させていただいているライブラリは、Zaif と直接関係のない一般の方々が作成して公開しておられるもの、いわゆる「非公式ライブラリ」となります。これ以外にもご存知の場合や作成して頂いた場合で、ここにリンクを希望されます方は、お手数ですが zaifdotjp の Twitter アカウントや関係者などにご一報いただければ幸いです。また、もしここへの掲載を希望されない場合も、お手数おかけしますがご連絡頂ければ対処いたしますので、よろしくおねがいします。

9.1.1 Java

- zaif.jp API for Java. - Java から Zaif.jp の API を使うためのライブラリです。https://github.com/techbureau/JZaif

9.1.2 JavaScript

- node.js zaif.jp module

  https://www.npmjs.com/package/zaif.jp

  https://github.com/you21979/node-zaif

- Qiita : Node.js - zaif（取引所）で API を使って取引する http://qiita.com/you21979@github/items/32da1d0e1782e9e31938

9.1.3 Ruby

- RubyGems zaif

  https://github.com/techbureau/zaif-ruby

9.1.4 PHP

- Zaif4PHP

  https://github.com/tk1024/Zaif4PHP/

9.1.5 Golang

- Zaif 取引所 (Bitcoin/Monacoin) の API を Golang から使う

  http://qiita.com/yanakend/items/e516f250800c4b876e65

9.1.6 Python

- zaif の API を簡単にコール出来るようにしました

  https://github.com/techbureau/zaifapi

- [Zaif]Python で簡単に仮想通貨の取引が出来るようにしてみた

  http://qiita.com/Akira-Taniguchi/items/e52930c881adc6ecfe07

9.2 サンプルコード

サンプルコードをご紹介させていただきます。

9.2.1 Python

requests は外部ライブラリとなります。利用するためには任意の環境に pip などを使ってインストールする必要があります。

現物公開 API

```
import requests
import json


response = requests.get('https://api.zaif.jp/api/1/last_price/btc_jpy')
if response.status_code != 200:
    raise Exception('return status code is {}'.format(response.status_code))
json.loads(response.text)
>>>{"last_price": 130065.0}
```

現物取引 API

```
import json
import hmac
import hashlib
import requests
from future.moves.urllib.parse import urlencode


secret = 'your secret key'
key = 'your key'
params = {
    'method': 'actiuve_orders',
    'nonce': 123,
    'currency_pairs': 'btc_jpy'
}
encoded_params = urlencode(params)
signature = hmac.new(bytearray(secret.encode('utf-8')), digestmod=hashlib.sha512)
signature.update(encoded_params.encode('utf-8'))
headers = {
    'key': key,
    'sign': signature.hexdigest()
}
response = requests.post('https://api.zaif.jp/tapi', data=encoded_params,␣
↪headers=headers)
if response.status_code != 200:
    raise Exception('return status code is {}'.format(response.status_code))
print(json.loads(response.text))
>>>{
>>>    "success": 1,
>>>    "return": {
>>>        "184": {
>>>            "currency_pair": "btc_jpy",
>>>            "action": "ask",
>>>            "amount": 0.03,
>>>            "price": 56000,
>>>            "timestamp": 1402021125
>>>        }
```

(次のページに続く)

(前のページからの続き)

```
>>>     }
>>>}
```

9.2.2 Go

現物公開 API

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
)

func main() {
    uri := "https://api.zaif.jp/api/1/last_price/btc_jpy"
    req, _ := http.NewRequest("GET", uri, nil)

    client := new(http.Client)
    resp, _ := client.Do(req)
    defer resp.Body.Close()

    byteArray, _ := ioutil.ReadAll(resp.Body)
    fmt.Println(string(byteArray))
}
>>>{"last_price": 130065.0}
```

現物取引 API

```
package main

import (
    "fmt"
    "time"
    "strconv"
    "crypto/hmac"
    "crypto/sha512"
    "io/ioutil"
    "net/http"
    "encoding/hex"
    "net/url"
    "strings"
)
```

(次のページに続く)

(前のページからの続き)

```
var key = "<your_key>"
var secret = "<your_secret>"

func main() {

    uri := "https://api.zaif.jp/tapi"
    values := url.Values{}
    values.Add("method", "get_info")
    values.Add("nonce", strconv.FormatInt(time.Now().Unix(), 10))

    encodedParams := values.Encode()
    req, _ := http.NewRequest("POST", uri, strings.NewReader(encodedParams))

    hash := hmac.New(sha512.New, []byte(secret))
    hash.Write([]byte(encodedParams))
    signature := hex.EncodeToString(hash.Sum(nil))

    req.Header.Add("Key", key)
    req.Header.Add("Sign", signature)
    client := new(http.Client)
    resp, _ := client.Do(req)
    defer resp.Body.Close()

    byteArray, _ := ioutil.ReadAll(resp.Body)
    fmt.Println(string(byteArray))
}
```

9.2.3 Swift

現物公開 API

```
let apiUrl = URL(string: "https://api.zaif.jp/api/1/last_price/btc_jpy")!
var request = URLRequest(url: apiUrl)
request.addValue("application/json", forHTTPHeaderField: "Content-type")
request.addValue("application/json", forHTTPHeaderField: "Accept")
request.httpMethod = "GET"
URLSession.shared.dataTask(with: request) {data, response, err in
    if let data = data {
        do {
            let json = try JSONSerialization.jsonObject(with: data, options:␣
↪JSONSerialization.ReadingOptions.allowFragments)
            print(json)
        } catch {
            print("Serialize Error")
        }
    } else {
```

(次のページに続く)

(前のページからの続き)

```
        print("Error")
    }
}.resume()
>>>{"last_price": 130065.0}
```

現物取引 API

```
import Foundation
import CryptoSwift

let key = "<your_key>"
let secret = "<your_secret>"

let url = URL(string: "https://api.zaif.jp/tapi")
let postStr = "method=get_info&nonce=" + String(Date().timeIntervalSince1970)
var request = URLRequest(url: url!)
request.httpMethod = "POST"
request.httpBody = postStr.data(using: .utf8)!

let signature = try HMAC(key: Array(secret.utf8), variant: .sha512).
↪authenticate(Array(postStr.utf8))
request.addValue(key, forHTTPHeaderField: "Key")
request.addValue(signature.toHexString(), forHTTPHeaderField: "Sign")

URLSession.shared.dataTask(with: request) { data, response, err in
    if let data = data {
        do {
           let json = try JSONSerialization.jsonObject(with: data, options:␣
↪JSONSerialization.ReadingOptions.allowFragments)
            print(json)
        } catch {
            print("Serialize Error")
        }
    } else {
        print("Error")
    }
}.resume()
```