Trading Platform API

v.1.0

Introduction

The purpose of this document is to provide detailed information about API that can be used by third parties for automation. Trading platform provides **REST** and **WebSocket** based APIs.
There are two different type of API calls: **Public/Private**.
For private calls there is a need to send "**Authorization**" header with token provided by login call.
In case of having API Key enabled some of calls are also available with API Key as well. Also need to have another custom header named "**Consumer**". Value for this one varies from environment to environment.

Production Environment

REST API URL: *https://trade.swixxo.com/api*
Socket URL: *wss://trade.swixxo.com/ws*
Consumer Token:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjViZTEzM2Y0ZTNlNDgyNzFmZmZmNGQyNSIsImlhdCI6MTU0MTQ4NTU1NiwiZXhwIjoyOTg1OTg0MTU0MTQ4NTU1Nn0.cMJ5sOe2u6k493OZUs5NEuK6wUZyTnQt_w4SIMPYJx0

User management

REST API calls that allows to get user specific information. Also it allows to authorize user and get corresponding session token that can should be used for other private calls.

Login Check

URL: */auth/login-check*
Method: GET
Headers: Authorization: sx <auth-token>
Consumer: sx <consumer-id>
Params: no params

Check if given auth-token is valid and authorized or not.

Login

URL: */auth/login*
Method: POST
Headers: Authorization: sx <username:password>
Consumer: sx <consumer-id>
Params: no params

Login user to platform and get **auth-token** in response. **Username** and **password** for login should be provided in **Authorization** header.

Logout

URL: /auth/logout
Method: GET
Headers: Authorization: sx <auth-token>

Consumer: sx <consumer-id>
Params: no params

Logout user from platform. After this call corresponding auth-token is not going to be valid and authorized.

Get User Transactions (deposit, withdrawal, adjustment, referral fees)

URL: */user/transactions?startDate=<Start-Date-Timestamp>&endDate=<End-Date-Timestamp>&page=<current-page>&itemsPerPage=<items-per-page>&currency=<currency-code>&type=<transaction-type>*
Method: GET
Headers: Authorization: sx <auth-token> **OR** Authorization: api-sx <api-key>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
startDate	Mandatory	non negative integer	timestamp
endDate	Mandatory	non negative integer	timestamp
page	Mandatory	Positive integer	page number
itemsPerPage	Mandatory	Positive integer	number of items per page
currency	Optional	Currency code	should be valid currency code available in platform
type	Optional	Enum: ['any', 'deposit', 'withdraw', 'adjustment']	Allows to filter only by specified transaction type

Returns list of transactions related to users balance change. It includes deposit/withdraw/adjustment from admin and also funds that used got through referral programs.

Get User Trades

URL: */user/trades?startDate=<Start-Date-Timestamp>&endDate=<End-Date-Timestamp>&page=<current-page>&itemsPerPage=<items-per-page>&symbol=<symbol-code>*
Method: GET
Headers: Authorization: sx <auth-token> **OR** Authorization: api-sx <api-key>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
startDate	Mandatory	non negative integer	timestamp
endDate	Mandatory	non negative integer	timestamp
page	Mandatory	Positive integer	page number

Name	Mandatory/ Optional	Accepted Values	Description
itemsPerPage	Mandatory	Positive integer	number of items per page
symbol	Optional	Tradable Pair	Should be valid currency pair separated with '-' symbol, or 'any' if trades are required for all pairs. (ETH-BTC)

Returns trades for logged in user. It's possible to specify symbol for which to get trades.

Get User Group

URL: /user/group/:id
Method: GET
Headers: Authorization: sx <auth-token>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
id	Mandatory	User id	User ID from database

Returns the group name for authorized user. That group defines all the fees and imitations for user.

Get User Info

URL: */user/:id*
Method: GET
Headers: Authorization: sx <auth-token>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
id	Mandatory	User id	User ID from database

Short user info.

Get User Profile Info

URL: */user/profile/:id*
Method: GET
Headers: Authorization: sx <auth-token>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
id	Mandatory	User id	User ID from database

User profile info including KYC related data.

Get User Balances

URL: */user/:userId/balances/:currency*
Method: GET
Headers: Authorization: sx <auth-token> **OR** Authorization: api-sx <api-key>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
userId	Mandatory	User id	User ID from database
currency	Optional	Currency Code	Will return balance for specific currency if provided

If currency is specified it returns balance for specified currency. Otherwise it returns balance for all active currencies including summary info.

Public API

REST API calls that does not need any authorization. They are public and can be used by anyone.

Get tradable symbols

URL: */symbols/:symbol*
Method: GET
Headers: Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
symbol	Optional	Tradable pair	Should be valid currency pair separated with '-' symbol, or ‘any' (e.g. ETH-BTC)

Provides the list of all pairs that are active currently. If **symbol** is specified it will provide only that symbol specific information.

Get all active currencies

URL: */currencies*
Method: GET
Headers: Consumer: sx <consumer-id>
Params: no params

Get prices

URL: */prices/:symbol*
Method: GET

Headers: Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
symbol	Optional	Tradable pair	Should be valid currency pair separated with '-' symbol (e.g. ETH-BTC)

Returns list of prices for all symbols if nothing is specified or current symbol specific price info.

Get current order book

URL: */book/:symbol*
Method: GET
Headers: Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
symbol	Optional	Tradable pair	Should be valid currency pair separated with '-' symbol (e.g. ETH-BTC)

Returns order book for specified symbol all for all active symbols if nothing is specified.

Get trade list

URL: */trades-all?symbol=<tradable-pair>&startDate=<Start-Date-Timestamp>&endDate=<End-Date-Timestamp>&page=<current-page>&itemsPerPage=<items-per-page>*
Method: GET
Headers: Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
startDate	Mandatory	non negative integer	timestamp
endDate	Mandatory	non negative integer	timestamp
page	Mandatory	Positive integer	page number
itemsPerPage	Mandatory	Positive integer	number of items per page
symbol	Optional	Tradable Pair	Should be valid currency pair separated with '-' symbol, or 'any' if trades are required for all pairs. (ETH-BTC)

Returns trade history for specified symbol if provided. Otherwise for all active symbols.

Order Management API (Private)

REST API that deals with orders and requires authorization.

Get order list

URL: */orders/history?symbol=<tradable-pair>&page=<current-page>&itemsPerPage=<items-per-page>&startDate=<Start-Date-Timestamp>&endDate=<End-Date-Timestamp>&hideCanceled=<true/false>*
Method: GET
Headers: Authorization: sx <auth-token> **OR** Authorization: api-sx <api-key>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
startDate	Mandatory	non negative integer	timestamp
endDate	Mandatory	non negative integer	timestamp
page	Mandatory	Positive integer	page number
itemsPerPage	Mandatory	Positive integer	number of items per page
symbol	Optional	Tradable Pair	Should be valid currency pair separated with '-' symbol, or 'any' if trades are required for all pairs. (ETH-BTC)
hideCanceled	Mandatory	Boolean (true/false)	Return canceled orders or not

Returns order list for user.

Get open orders

URL: */orders/open-orders*
Method: GET
Headers: Authorization: sx <auth-token> **OR** Authorization: api-sx <api-key>
Consumer: sx <consumer-id>
Params: no params

List of open orders.

Order update check

URL: */orders/update-check*
Method: GET
Headers: Authorization: sx <auth-token> **OR** Authorization: api-sx <api-key>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
symbol	Optional	Tradable pair	Should be valid currency pair separated with '-' symbol, or ‘any' (e.g. ETH-BTC)
lastUpdateTime	Mandatory	non negative integer	timestamp

Check if there are updates in orders after specified lastUpdateTime.

Create new order

URL: */orders*
Method: POST
Headers: Authorization: sx <auth-token> **OR** Authorization: api-sx <api-key>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
symbol	Optional	Tradable pair	Should be valid currency pair separated with '-' symbol, or ‘any' (e.g. ETH-BTC)
side	Mandatory	enum: ['buy', 'sell']	Define order side
type	Mandatory	enum: ['market', 'limit']	Define order type
subType	Optional	enum: ['ioc', 'gtc', 'fok']	Defines advanced type for order. Default value is 'gtc'
price	Mandatory	positive number	Price for order
quantity	Mandatory	positive number	Amount for order

Cancel Active Order

URL: /orders/:orderID
Method: DELETE
Headers: Authorization: sx <auth-token> **OR** Authorization: api-sx <api-key>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
OrderId	Mandatory	String	Autogenerated string identifying order

Cancel order with specified order ID.

Cancel All Orders for given pair

URL: */orders?symbol=<tradable-pair>*
Method: DELETE
Headers: Authorization: sx <auth-token> **OR** Authorization: api-sx <api-key>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
symbol	Optional	Tradable pair	Should be valid currency pair separated with '-' symbol, or ‘any' (e.g. ETH-BTC)

Cancel all orders for specified symbol.

Get Order by ID

URL: */orders/:orderID*
Method: GET
Headers: Authorization: sx <auth-token> **OR** Authorization: api-sx <api-key>
Consumer: sx <consumer-id>
Params:

Name	Mandatory/ Optional	Accepted Values	Description
OrderId	Mandatory	String	Autogenerated string identifying order

Returns order details for order with given OrderId.

Web Socket API

Authorize Socket

This allows to authorize socket and allow user to make private calls.

Request:

```
{
    "method": "authorize",
    "params": {
        "token": “<auth-token or api-key>”,
        "consumerId": “<consumer-id>"
    },
    "id": 1
}
```

Response:

```
{
    "method": "authorize",
    "result": {
        "success": true
    },
    "jsonrpc": "2.0"
}
```

Get User Balances

This allows authorized users to get balance information for all wallets or for specified currency.

Request:

```
{
    "method": "balances",
    "params": {
        "includeSummary": <true/false>
        "currency": "BTC"
    },
    "id": 1
}
```

Response:

Currencies with corresponding balances. If currency is specified it will return balance for specified currency only.

Subscribe to order book

Subscribe to order book information for given pair.
After getting subscribed user will get response as “snapshotBook” that reflects current situation.
After that user will get updates on order book changes as “bookUpdate”.

Request:

```
{
    "method": "subscribeBook",
    "params": {
        "symbol": “BTC/USDT"
    },
    "id": 1
}
```

Response:

```
 {
    "method": "snapshotBook",
    "result": {
        "buy": [{
            "price": "6052.2000000000",
            "quantity": "0.7000"
        },
        ….
```

```
            ],
            "sell": [{
                "price": "6063.0000000000",
                "quantity": "0.4000"
            },
            ....
            ],
            "symbol": "BTC/USDT"
        },
        "jsonrpc": "2.0"
    }
```


Update:


```
    {
        "method": "bookUpdate",
        "result": {
            "buy": [{
                "price": "6052.3000000000",
                "quantity": "0.6400"
            }],
            "sell": [],
            "symbol": "BTC/USDT"
        },
        "jsonrpc": "2.0"
    }
```


Subscribe to trades

Subscribe to trades feed.
Once user gets subscribed it will get snapshot for last trades for specified market.
After that user will get updates on every new trade.

Request:


```
    {
        "method": "subscribeTrades",
        "params": {
            "symbol": “BTC/USDT”,
            "limit": 100
        },
        "id": 1
    }
```


Response:


```
    {
        "method": "snapshotTrades",
        "result": [{
            "tradeId": "2ab25a-865-1542213990040",
            "pair": "BTC/USDT",
            "side": "SELL",
```

```
            "quantity": "0.0002",
            "price": "6061.2200000000",
            "created": "2018-11-14T16:46:30.041Z"
        }
        ....
        ],
        "jsonrpc": "2.0"
    }
```


Update:


```
    {
        "method": "newTrade",
        "result": {
            "tradeId": "2ab25a-875-1542214006026",
            "pair": "BTC/USDT",
            "side": "SELL",
            "quantity": "0.0008",
            "price": "6068.4100000000",
            "created": "2018-11-14T16:46:46.026Z"
        },
        "jsonrpc": "2.0"
    }
```


Subscribe to candles

Subscribe to get candles. It will return snapshot as a response to subscription. After that user will get periodic updated for each new candle. Frequency will depend on “interval” parameter provided during subscription.

Request:


```
    {
        "method": "subscribeCandles",
        "params": {
            "symbol": "BTC/USDT",
            "limit": 1000,
            "interval": "M1"
        },
        "id": 1
    }
```


Response:


```
    {
        "method": "snapshotCandles",
        "result": [{
            "timestamp": "2018-11-14T15:01:00.000Z",
            "symbol": "BTC/USDT",
            "interval": "M1",
            "open": "6307.5100000000",
            "close": "6307.8900000000",
```

```
            "min": "6307.5100000000",
            "max": "6307.8900000000",
            "volume": "0.0039",
            "volumeQuote": "24.6007"
        }
        ....
        ],
        "jsonrpc": "2.0"
    }
```


Update:


```
    {
        "method": "newCandle",
        "result": {
            "timestamp": "2018-11-14T16:51:00.000Z",
            "symbol": "BTC/USDT",
            "interval": "M1",
            "open": "6024.5100000000",
            "close": "6024.5100000000",
            "min": "6024.5100000000",
            "max": "6033.1600000000",
            "volume": "0.0196",
            "volumeQuote": "118.0803"
        },
        "jsonrpc": "2.0"
    }
```