WolfBot REST and WebSocket JSON APIs

For an overview of the latest API parameters and response you may also check out: https://wolfbot.org/wp-json/tradebot/v1/ (although I will keep this document updated of course)

General
All wolfbot.org API calls require your personal affiliate API key.
All some-bot-instance.wolfbot.org API calls require the current bot API key which you must query from the server first.

All responses have the following JSON format:

```
{
        error: boolean;
        errorCode: int;
        errorMsg: string;
        data: any[]; // array with data depending on your API call (see below)
}
```

Current error codes are (0 means “no error”):
API_ERR_INVALID_EMAIL = 1;
API_ERR_NO_INSTANCE = 2;
API_ERR_INTERNAL = 3;
API_USER_DOESNT_EXIST = 4;
NO_BOT_SUBSCRIPTIONS_FOUND = 5;
TRADEBOOK_NOT_AVAILABLE = 100;
TRADEBOOK_ERROR_QUERY = 101;
TRADEBOOK_ERROR_PARAMS = 102;

API flow
You should only call the /new call once and store the username and token in your database.
You can call /getApiKey every time you start a new session (for example when a user logs into your website).
All bot instance API calls are also available in the open source version of WolfBot.

Create a new bot instance
This must be called once for every new customer.
POST: https://wolfbot.org/wp-json/tradebot/v1/new
Parameters:

- apiKey (string): Your wolfbot.org API key
- email (string): The email of the customer. His password will be sent to that address.
- username (string): The username of the customer
- days (int): The number of days the bot instance shall run. Use 0 for a demo instance. Valid values: 0|30|180|360
- Affiliate (string, optional): The username of the affiliate to give a commission for this referral.

data-response:
[{
bot: {
id (int): unique ID of this cloud bot instance
url (string): The URL to access this bot. The user must open this url in his browser and login with his account. Also needed for bot API calls such as live trades, importing trade book,.....
token (string): A unique token for this bot instance. Needed to request the current bot API key.
}
}]

Get the current API key of an active bot instance
This must be called every time before you make a REST API call to a bot instance or open a websocket connection. The API key can change if the user logs in again.
POST: https://wolfbot.org/wp-json/tradebot/v1/getApiKey
Parameters:

- apiKey (string): Your wolfbot.org API key
- token (string): The bot instance token from the /new API call
- username (string): The username this bot belongs to (must be the same value as in POST /new API call)

data-response:
[{
data: {
subscriptions: [{
bot_url: {
id (int): The unique ID of this bot instance
bot_api_key (int): The current API key you need to make requests to this bot
}
}]
}
// there are more keys in that response which you can ignore
}]

Get the tradebook for this user

This can be called at any time to get historical trades of this bot instance.
It will return the most recent trades (at most 9000 results).
POST: bot-url/tradebook/ - for example: https://bot1234.wolfbot.org:3143/tradebook/
bot-url received in the /new API call
Parameters:

- apiKey (string): The API key for this bot you received from the /getApiKey call.
- mode (string, optional, default trading): The trading mode you want to get trades from. Values: trading|arbitrage|lending
- currencyStr (string): The currency pair for which you want to get the history. Examples: USD_BTC, BTC_ETH,...
  For lending mode only 1 currency: USD, BTC, ETH,...
- configName (string, optional): The name of a user-defined configuration file with his set of trading strategies. If present, only trades from that set of strategies will be returned.
- endDate (int, optional): The end date as a unix timestamp (in seconds) for the trades history. The current date will be used if omitted.

data-response:

```
[{
        tradesMeta: // aggregated statistics about all trades in this response
        {
                avgBuyPrice: number;
                avgSellPrice: number;
                totalBuys: number;
                totalSells: number;
                totalBuyAmountBase: number;
                totalSellAmountBase: number;
                remainingCoins: number;
                breakEvenPrice: number;
                profitLoss: number;
                baseCurrency: string;
                quoteCurrency: string;
        },
        trades: // array with single trades
        [
                {
                        time: string; // ISO date string
                        rate: number;
                        amount: number;
                        configName: string;
                        strategies:
                        [
```

```
                                        string; // array of strategy names in this
                                        config file name
                                ],
                                reason: string;
                                market: number; // see marketStr
                                type: number; // see typeStr
                                exchange: number; // see exchangeStr;
                                currencyPair: // currency pair as stored in DB, see
                                currencyPairStr for a string representation
                                {
                                        from: number;
                                        to: number;
                                },
                                fees: number;
                                arbitrage: boolean;
                                currencyPairStr: string;
                                _id: string; // unique ID
                                exchangeStr: string; // the name of the exchange
                                marketStr: string; //
                                EXCHANGE|MARGIN|LENDING
                                typeStr: string; //
                                BUY|SELL|CLOSE_LONG|CLOSE_SHORT|LEND|INTE
                                REST_PAY|INTEREST_GET
                        }, ...
                ]
}]
```

Open a Websocket connection to receive live trades & strategy data

WS: wss://bot-url/?apiKey=BOT_API_KEY&format=json

For bot-url use the hostname:port you received from the /new API call.

Parameters:

- apiKey (string): The API key for this bot you received from the /getApiKey call.

After the connection has been established you can subscribe to channels via WebSocket messages of the form [opcode, payload]:

- subscribe to strategy live data: [12, {action: "sub"}]
- subscribe to live traders: [24, {action: "sub"}]

You may use {action: "unsub"} as payload to unsubscribe from updates of a channel.

response:

The initial full response when subscribing to strategy data (opcode 12):

```
[12, {
   "full": [{
      "nr": 1,
```

```
      "exchanges": [
         string; // array of strings, usually only 1 exchange name
      ],
      "marginTrading": boolean;
      "tradeTotalBtc": number;
      "warmUpMin": number;
      "strategies": {
         "StrategyName": {
            // key value pairs with live strategy data (depending on the strategy)
         }
      },
      "currencyPairStr": string; // USD_BTC
      "baseCurrency": string; // USD
      "quoteCurrency": string; // BTC
      "position": string;
      "positionAmount": number;
      "pl": number;
      "plPercent": number;
      "activeNr": number;
      "mainStrategyName": string;
      "userToken": string;
      "meta": {
         "importLabel": string;
      }
   }]
}]
```

This will give you updates for the 1st currency nur (see “nr” key in response above):

```
[12, {
   "config": string; // unique string as nr-currencyPair (or currency for lending). The currencyPair alone is not unique because you can run multiple configs on the same pair, example: "1-USD_BTC"
   "strategies": {
      "StrategyName": {
         // key value pairs with live strategy data (depending on the strategy)
      }
   },
   "meta": {
      "importLabel": string;
   }
}]
```

To change the config number (and thus currencyPair) you want to receive updates for you have to send a WebSocket message: [12, {tabNr: config-nr}]

The responses when subscribing to trades data (opcode 24):

```
[24, [
        “buy”, // the first array element is the name of the trade
        // payload objects starting from array index 1
]]
```

Possible trade events (for trading and arbitrage):
[“buy”, order: Order.Order, trades: Trade.Trade[], info?: TradeInfo]
[“sell”, order: Order.Order, trades: Trade.Trade[], info?: TradeInfo]
[“close”, order: Order.Order, trades: Trade.Trade[], info?: TradeInfo]
[“syncPortfolio”, currencyPair: Currency.CurrencyPair, coins: number, position: MarginPosition, exchangeLabel: Currency.Exchange]

Possible trade events (in lending mode):
[“takenAll”, currency: Currency.Currency]
[“placed”, offer: Funding.FundingOrder, result: OfferResult, info: LendingTradeInfo]
[“cancelled”, offer: Funding.FundingOrder, result: CancelOrderResult, info: LendingTradeInfo]
[“synced”, balances: CoinMap, balancesTaken: CoinMap, balancesTakenPercent: CoinMap, lendingRates: LendingRateMap, globalInterestStats: GlobalLendingStatsMap, minLendingRates: GlobalInterestMap, coinStatus: CoinStatusMap, totalEquity: TotalEquityMap]

Sample NodeJS client:
https://github.com/Ekliptor/WolfBot/blob/master/tests/samples/WebSocketAPI.js