

Package ‘okxAPI’

July 22, 2025

Title An Unofficial Wrapper for 'okx exchange v5' API

Version 0.1.1

Description An unofficial wrapper for 'okx exchange v5' API <<https://www.okx.com/docs-v5/en/>>, including 'REST' API and 'WebSocket' API.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports R6, data.table, httr, base64enc, jsonlite, websocket, digest

NeedsCompilation no

Author Yongchao Fang [aut, cre, cph]

Maintainer Yongchao Fang <yongchao.fang@outlook.com>

Repository CRAN

Date/Publication 2023-04-22 07:10:02 UTC

Contents

get_functions	2
get_history_candles	2
get_positions_history	3
restAPI	4
restAPIaccount	7
restAPImarket	8
restAPItrade	9
websocketAPIprivate	10
websocketAPIpublic	13
Index	17

get_functions

Wrapper for some frequently used APIs to get data easily

Description

The main purpose is to handle APIs that have limitations on the number of results returned per single request.

See Also

[get_positions_history](#) [get_history_candles](#)

get_history_candles

Retrieve the candlestick charts

Description

Wrapper for API [Get candlesticks](#) and [Get candlesticks history](#).

Usage

```
get_history_candles(
    api_key,
    secret_key,
    passphrase,
    bar = c("1m", "3m", "5m", "15m", "30m", "1H", "4H", "6H", "12H", "1D", "2D", "3D"),
    count,
    instId,
    ...
)
```

Arguments

api_key	Okx API key.
secret_key	Okx API secret key.
passphrase	Okx API passphrase.
bar	Bar size, the default is 1m, e.g. 1m/3m/5m/15m/30m/1H/2H/4H, Hong Kong time opening price k-line: 6H/12H/1D/2D/3D.
count	Number of Bars.
instId	Instrument ID, e.g. BTC-USDT-SWAP.
...	Other request parameters to be passed, See Get candlesticks history for more information.

Value

Candlestick charts data

Examples

```
## Not run:
candles <- get_history_candles(
  api_key, secret_key, passphrase, bar = "1m",
  count = 24*60, instId = "CFX-USDT-SWAP"
)

## End(Not run)
```

get_positions_history *Retrieve the position data*

Description

Wrapper for API [Get positions history](#).

Usage

```
get_positions_history(
  api_key,
  secret_key,
  passphrase,
  count = 90,
  period = 10,
  ...
)
```

Arguments

api_key	Okx API key.
secret_key	Okx API secret key.
passphrase	Okx API passphrase.
count	Retrieve position data for a specified number of past days, with a maximum of 90(days)
period	Due to the 'Number of results per request' limitation of the API, the period parameter must be specified to ensure that the number of position data entries within each period does not exceed 100.
...	Other request parameters to be passed, See Get positions history for more information.

Value

Position data

Examples

```
## Not run:
positions <- get_positions_history(
  api_key, secret_key, passphrase, count = 90, period = 10,
  instType = "SWAP", mgnMode = "isolated"
)

## End(Not run)
```

restAPI	<i>restAPI Class</i>
---------	----------------------

Description

Base class for **Okx exchange v5 API**.

Details

You can implement all REST API requests in Okx exchange by inheriting the restAPI class. Please refer to the example provided at the end of the document.

For commonly used interfaces, they have been implemented in this package. The naming convention is as follows: for "/api/v5/AAA/BBB", a new class named restAPIAAA, which inherits from restAPI, has been defined in this package. The BBB method in this new class is used to call the API.

Public fields

url Okx REST API url, which is <https://www.okx.com>.

api_key Okx API key.

secret_key Okx API secret key.

passphrase Okx API passphrase.

simulate Whether to use demo trading service.

Methods**Public methods:**

- `restAPI$new()`
- `restAPI$get_timestamp()`
- `restAPI$get_request_path()`
- `restAPI$get_body()`

- `restAPI$get_message()`
- `restAPI$get_signature()`
- `restAPI$get_header()`
- `restAPI$get_result()`
- `restAPI$clone()`

Method `new()`: Create a new REST API object.

Usage:

```
restAPI$new(api_key, secret_key, passphrase, simulate = FALSE)
```

Arguments:

`api_key` Okx API key.

`secret_key` Okx API secret key.

`passphrase` Okx API passphrase.

`simulate` Whether to use demo trading service.

Method `get_timestamp()`: Get UTC timestamp.

Usage:

```
restAPI$get_timestamp()
```

Method `get_request_path()`: Get request path.

Usage:

```
restAPI$get_request_path(api, method = c("GET", "POST"), ...)
```

Arguments:

`api` Request api e.g. `/api/v5/account/positions-history`.

`method` Request method, GET or POST.

`...` Other request parameters.

Method `get_body()`: Get request body.

Usage:

```
restAPI$get_body(method = c("GET", "POST"), ...)
```

Arguments:

`method` Request method, GET or POST.

`...` Other request parameters.

Method `get_message()`: Get the signing messages.

Usage:

```
restAPI$get_message(timestamp, request_path, body, method = c("GET", "POST"))
```

Arguments:

`timestamp` Retrieve through method `get_timestamp`.

`request_path` Retrieve through method `get_request_path`.

`body` Retrieve through method `get_body`.

`method` Request method, GET or POST.

Method `get_signature()`: Get the signature.

Usage:

```
restAPI$get_signature(msg, secret_key = self$secret_key)
```

Arguments:

`msg` Retrieve through method `get_message`.
`secret_key` Okx API secret key.

Method `get_header()`: Get request headers.

Usage:

```
restAPI$get_header(timestamp, msg)
```

Arguments:

`timestamp` Retrieve through method `get_timestamp`.
`msg` Retrieve through method `get_message`.

Method `get_result()`: Retrieve data from api.

Usage:

```
restAPI$get_result(api, method = c("GET", "POST"), process, ...)
```

Arguments:

`api` Request api e.g. `/api/v5/account/positions-history`.
`method` Request method, GET or POST.
`process` A function to process the data received from the API.
`...` Other request parameters.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
restAPI$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

[restAPItrade](#) [restAPIaccount](#) [restAPImarket](#)

Examples

```
## Not run:
# for [Get currencies](https://www.okx.com/docs-v5/en/#rest-api-funding-get-currencies)
# you can define the class like this
myRestAPI <- R6::R6Class(
  inherit = restAPI,
  public = list(
    get_currencies = function(ccy, process = "identity") {
      self$get_result(
        api = "/api/v5/asset/currencies", method = "GET", process = process,
        ccy = ccy
      )
    }
  )
)
```

```

    )
  }
)
)
# And call it like this
tmp <- myRestAPI$new(api_key, secret_key, passphrase)
tmp$get_currencies("BTC")

## End(Not run)

```

restAPIaccount	<i>restAPIaccount Class</i>
----------------	-----------------------------

Description

Wrapper for **REST API ACCOUNT**.

Super class

`okxAPI::restAPI` -> restAPIaccount

Methods

Public methods:

- `restAPIaccount$balance()`
- `restAPIaccount$positions()`
- `restAPIaccount$positions_history()`
- `restAPIaccount$clone()`

Method `balance()`: See **Get balance** for more information.

Usage:

```
restAPIaccount$balance(ccy, process = "identity")
```

Arguments:

`ccy` Single currency or a vector composed of multiple currencies. (no more than 20).
`process` A function to process the data received from the API. Default to identity.

Method `positions()`: See **Get positions** for more information.

Usage:

```
restAPIaccount$positions(process = "identity", ...)
```

Arguments:

`process` A function to process the data received from the API. Default to identity.
`...` Other request parameters.

Method `positions_history()`: See **Get positions history** for more information.

Usage:

```
restAPIaccount$positions_history(process = "identity", ...)
```

Arguments:

process A function to process the data received from the API. Default to identity.
 ... Other request parameters.

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
restAPIaccount$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

restAPImarket

restAPImarket Class

Description

Wrapper for **REST API MARKET**.

Super class

```
okxAPI::restAPI -> restAPImarket
```

Methods**Public methods:**

- `restAPImarket$candles()`
- `restAPImarket$history_candles()`
- `restAPImarket$clone()`

Method candles(): See **Get candlesticks** for more information.

Usage:

```
restAPImarket$candles(instId, process = "identity", ...)
```

Arguments:

instId Instrument ID, e.g. BTC-USD-190927-5000-C.
 process A function to process the data received from the API. Default to identity.
 ... Other request parameters.

Method history_candles(): See **Get candlesticks history** for more information.

Usage:

```
restAPImarket$history_candles(instId, process = "identity", ...)
```

Arguments:

instId Instrument ID, e.g. BTC-USD-190927-5000-C.

process A function to process the data received from the API. Default to identity.
 ... Other request parameters.

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
restAPImarket$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

restAPItrade

restAPItrade Class

Description

Wrapper for **REST API TRADE**.

Super class

`okxAPI::restAPI -> restAPItrade`

Methods

Public methods:

- `restAPItrade$order()`
- `restAPItrade$cancel_order()`
- `restAPItrade$clone()`

Method order(): See **Place order** for more information.

Usage:

```
restAPItrade$order(
  instId,
  tdMode = c("isolated", "cross", "cash"),
  side = c("buy", "sell"),
  sz,
  ordType = c("market", "limit", "post_only", "fok", "ioc", "optimal_limit_ioc"),
  process = "identity",
  ...
)
```

Arguments:

instId Instrument ID, e.g. BTC-USD-190927-5000-C.

tdMode Trade mode. Margin mode: cross or isolated. Non-Margin mode: cash.

side Order side, buy or sell.

sz Quantity to buy or sell.

ordType Order type. market: Market order, limit: Limit order, post_only: Post-only order, fok: Fill-or-kill order, ioc: Immediate-or-cancel order, optimal_limit_ioc: Market order with immediate-or-cancel order (applicable only to Futures and Perpetual swap).
 process A function to process the data received from the API. Default to identity.
 ... Other request parameters.

Method cancel_order(): See [Cancel order](#) for more information.

Usage:

```
restAPItrade$cancel_order(instId, ordId, clOrdId, process = "identity", ...)
```

Arguments:

instId Instrument ID, e.g. BTC-USD-190927.
 ordId Order ID, Either ordId or clOrdId is required. If both are passed, ordId will be used.
 clOrdId Client Order ID as assigned by the client.
 process A function to process the data received from the API. Default to identity.
 ... Other request parameters.

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
restAPItrade$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

websocketAPIprivate	<i>websocketAPIprivate Class</i>
---------------------	----------------------------------

Description

Private channel of WebSocket API for [Okx exchange v5 API](#). See [Private Channel](#) for more information.

Public fields

channel Private WebSocket url.
 api_key Okx API key.
 secret_key Okx API secret key.
 passphrase Okx API passphrase.
 simulate Whether to use demo trading service.
 ws A websocket::WebSocket object to establish a connection to the server.

Methods

Public methods:

- `websocketAPIprivate$new()`
- `websocketAPIprivate$get_timestamp()`
- `websocketAPIprivate$get_message()`
- `websocketAPIprivate$get_signature()`
- `websocketAPIprivate$connect()`
- `websocketAPIprivate$login()`
- `websocketAPIprivate$on_open()`
- `websocketAPIprivate$on_close()`
- `websocketAPIprivate$on_message()`
- `websocketAPIprivate$on_error()`
- `websocketAPIprivate$send()`
- `websocketAPIprivate$close()`
- `websocketAPIprivate$clone()`

Method `new()`: Craeate a new websocketAPIprivate object.

Usage:

```
websocketAPIprivate$new(api_key, secret_key, passphrase, simulate = FALSE)
```

Arguments:

`api_key` Okx API key.

`secret_key` Okx API secret key.

`passphrase` Okx API passphrase.

`simulate` Whether to use demo trading service.

Method `get_timestamp()`: Get UTC timestamp.

Usage:

```
websocketAPIprivate$get_timestamp()
```

Method `get_message()`: Get the signing messages.

Usage:

```
websocketAPIprivate$get_message(timestamp)
```

Arguments:

`timestamp` Retrieve through method `get_timestamp`.

Method `get_signature()`: Get the signature.

Usage:

```
websocketAPIprivate$get_signature(secret_key, msg)
```

Arguments:

`secret_key` Okx API secret key.

`msg` Retrieve through method `get_message`.

Method `connect()`: Initiate the connection to the server.

Usage:

```
websocketAPIprivate$connect()
```

Method login(): Log in.

Usage:

```
websocketAPIprivate$login()
```

Method on_open(): Called when the connection is established.

Usage:

```
websocketAPIprivate$on_open(func)
```

Arguments:

func A Callback function.

Method on_close(): Called when a previously-opened connection is closed. The event will have 'code' (integer) and 'reason' (one-element character) elements that describe the remote's reason for closing.

Usage:

```
websocketAPIprivate$on_close(func)
```

Arguments:

func A Callback function.

Method on_message(): Called each time a message is received from the server. The event will have a 'data' element, which is the message content.

Usage:

```
websocketAPIprivate$on_message(func)
```

Arguments:

func A Callback function.

Method on_error(): Called when the connection fails to be established. The event will have an 'message' element, a character vector of length 1 describing the reason for the error.

Usage:

```
websocketAPIprivate$on_error(func)
```

Arguments:

func A Callback function.

Method send(): Send a message to the server.

Usage:

```
websocketAPIprivate$send(msg)
```

Arguments:

msg Messages.

Method close(): Close the connection.

Usage:

```
websocketAPIprivate$close()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
websocketAPIprivate$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## Not run:
tmp <- websocketAPIprivate$new(api_key, secret_key, passphrase)
tmp$connect()
Sys.sleep(1)
tmp$login()

# subscribe account information
msg <- list(
  op = "subscribe",
  args = list(
    list(channel = "account", ccy = "USDT")
  )
)
msg <- jsonlite::toJSON(msg, auto_unbox = TRUE, pretty = TRUE)
tmp$send(msg)

# pass your own callback function
tmp$on_message(function(event) {
  if (event$data == "pong") {
    cat("Bingo!!\n")
  }
})
tmp$send("ping")

tmp$close()

## End(Not run)
```

websocketAPIpublic	<i>websocketAPIpublic Class</i>
--------------------	---------------------------------

Description

Public channel of WebSocket API for **Okx exchange v5 API**. See **Public Channel** for more information.

Public fields

channel Public WebSocket url.

simulate Whether to use demo trading service.

ws A websocket::WebSocket object to establish a connection to the server.

Methods

Public methods:

- `websocketAPIpublic$new()`
- `websocketAPIpublic$connect()`
- `websocketAPIpublic$on_open()`
- `websocketAPIpublic$on_close()`
- `websocketAPIpublic$on_message()`
- `websocketAPIpublic$on_error()`
- `websocketAPIpublic$send()`
- `websocketAPIpublic$close()`
- `websocketAPIpublic$clone()`

Method `new()`: Create a new `websocketAPIpublic` object.

Usage:

```
websocketAPIpublic$new(simulate = FALSE)
```

Arguments:

`simulate` Whether to use demo trading service.

Method `connect()`: Initiate the connection to the server.

Usage:

```
websocketAPIpublic$connect()
```

Method `on_open()`: Called when the connection is established.

Usage:

```
websocketAPIpublic$on_open(func)
```

Arguments:

`func` A Callback function.

Method `on_close()`: Called when a previously-opened connection is closed. The event will have 'code' (integer) and 'reason' (one-element character) elements that describe the remote's reason for closing.

Usage:

```
websocketAPIpublic$on_close(func)
```

Arguments:

`func` A Callback function.

Method `on_message()`: Called each time a message is received from the server. The event will have a 'data' element, which is the message content.

Usage:

```
websocketAPIpublic$on_message(func)
```

Arguments:

`func` A Callback function.

Method `on_error()`: Called when the connection fails to be established. The event will have an 'message' element, a character vector of length 1 describing the reason for the error.

Usage:

```
websocketAPIpublic$on_error(func)
```

Arguments:

func A Callback function.

Method `send()`: Send a message to the server.

Usage:

```
websocketAPIpublic$send(msg)
```

Arguments:

msg Messages.

Method `close()`: Close the connection.

Usage:

```
websocketAPIpublic$close()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
websocketAPIpublic$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## Not run:
tmp <- websocketAPIpublic$new()
tmp$connect()

# subscribe BTC-USDT-SWAP 5m candlesticks data
msg <- list(
  op = "unsubscribe",
  args = list(
    list(channel = "candle5m", instId = "BTC-USDT-SWAP")
  )
)
msg <- jsonlite::toJSON(msg, auto_unbox = TRUE, pretty = TRUE)
tmp$send(msg)

# pass your own callback function
tmp$on_message(function(event) {
  if (event$data == "pong") {
    cat("Bingo!!\n")
  }
})
tmp$send("ping")

tmp$close()
```

```
## End(Not run)
```

Index

`get_functions`, [2](#)
`get_history_candles`, [2](#), [2](#)
`get_positions_history`, [2](#), [3](#)

`okxAPI::restAPI`, [7–9](#)

`restAPI`, [4](#)
`restAPIaccount`, [6](#), [7](#)
`restAPImarket`, [6](#), [8](#)
`restAPItrade`, [6](#), [9](#)

`websocketAPIprivate`, [10](#)
`websocketAPIpublic`, [13](#)