

Package ‘binxr’

August 2, 2026

Type Package

Title 'Binance' REST API Client

Version 0.1.2

Author Oliver Zhou [aut, cre],
Lily Li [aut]

Maintainer Oliver Zhou <oliver.yxzhou@gmail.com>

Description Client for the 'Binance' <<https://www.binance.com/>> Spot, USD-M Futures, and Options REST APIs. Provides helper functions for signed and unsigned requests, market data retrieval, account access, and order management with 'data.table' output by default. COIN-M Futures, Portfolio Margin, WebSocket, SBE, and FIX APIs are not included.

License MIT + file LICENSE

URL <https://github.com/OliverLDS/binxr>,
<https://oliverlds.github.io/binxr/>

BugReports <https://github.com/OliverLDS/binxr/issues>

Encoding UTF-8

Depends R (>= 4.1.0)

Imports htr2, jsonlite, digest, data.table, rlang

Suggests testthat (>= 3.0.0), waldo

Config/testthat/edition 3

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-08-02 18:40:02 UTC

Contents

binxr_config_futures	5
binxr_config_options	5

binxr_config_spot	6
coerce_account_frames	7
config_futures	7
config_options	8
config_spot	9
futures_cancel_algo_order	10
futures_cancel_all_algo_orders	10
futures_cancel_all_orders	11
futures_cancel_order	11
futures_countdown_cancel_all	12
futures_get_24hr_ticker	13
futures_get_account	13
futures_get_account_summary	14
futures_get_account_trades	14
futures_get_aggregate_trades	15
futures_get_algo_order	16
futures_get_algo_orders	17
futures_get_balance	17
futures_get_book_ticker	18
futures_get_commission_rate	18
futures_get_continuous_klines	19
futures_get_exchange_info	20
futures_get_force_orders	20
futures_get_funding_info	21
futures_get_funding_rate_history	22
futures_get_historical_trades	22
futures_get_index_constituents	23
futures_get_index_price_klines	24
futures_get_insurance_balance	24
futures_get_klines	25
futures_get_mark_price	26
futures_get_mark_price_klines	27
futures_get_multi_assets_mode	27
futures_get_open_algo_orders	28
futures_get_open_interest	29
futures_get_open_orders	29
futures_get_order	30
futures_get_orders	31
futures_get_order_amendments	31
futures_get_order_book	32
futures_get_order_rate_limit	33
futures_get_positions	33
futures_get_position_mode	34
futures_get_position_risk	34
futures_get_premium_index_klines	35
futures_get_recent_trades	35
futures_get_rpi_depth	36
futures_get_server_time	37

futures_get_symbol_adl_risk	37
futures_get_ticker_price	38
futures_get_trading_schedule	38
futures_modify_order	39
futures_ping	40
futures_place_algo_order	40
futures_place_order	41
futures_set_leverage	43
futures_set_margin_type	44
futures_set_multi_assets_mode	44
futures_set_position_mode	45
futures_test_order	46
options_cancel_all_orders	47
options_cancel_all_orders_by_underlying	47
options_cancel_order	48
options_get_24hr_ticker	48
options_get_account_trades	49
options_get_commission	50
options_get_exchange_info	50
options_get_exercise_history	51
options_get_exercise_records	51
options_get_funding_flow	52
options_get_index_price	53
options_get_klines	54
options_get_margin_account	54
options_get_mark_price	55
options_get_open_interest	55
options_get_open_orders	56
options_get_order	57
options_get_order_book	57
options_get_order_history	58
options_get_positions	59
options_get_recent_block_trades	59
options_get_recent_trades	60
options_get_server_time	61
options_ping	61
options_place_order	62
round_price_qty	63
spot_amend_order_keep_priority	64
spot_cancel_all_orders	65
spot_cancel_order	65
spot_cancel_order_list	66
spot_cancel_replace_order	67
spot_get_24hr_ticker	69
spot_get_account	69
spot_get_account_trades	70
spot_get_aggregate_trades	71
spot_get_allocations	71

spot_get_all_order_lists	72
spot_get_average_price	73
spot_get_book_ticker	74
spot_get_commission_rates	74
spot_get_exchange_info	75
spot_get_execution_rules	76
spot_get_historical_block_trades	76
spot_get_historical_trades	77
spot_get_klines	78
spot_get_open_orders	79
spot_get_open_order_lists	79
spot_get_order	80
spot_get_orders	80
spot_get_order_amendments	81
spot_get_order_book	82
spot_get_order_list	82
spot_get_prevented_matches	83
spot_get_recent_trades	84
spot_get_reference_price	84
spot_get_reference_price_calculation	85
spot_get_relevant_filters	85
spot_get_rolling_window_ticker	86
spot_get_server_time	87
spot_get_ticker_price	87
spot_get_trading_day_ticker	88
spot_get_ui_klines	89
spot_get_unfilled_order_count	90
spot_ping	90
spot_place_oco_order	91
spot_place_order	92
spot_place_order_list_oco	94
spot_place_order_list_opo	96
spot_place_order_list_opoco	98
spot_place_order_list_oto	101
spot_place_order_list_otoco	104
spot_place_sor_order	107
spot_test_order	108
spot_test_sor_order	110

binxr_config_futures *Backward-compatible futures config constructor*

Description

Backward-compatible futures config constructor

Usage

```
binxr_config_futures(  
    api_key = Sys.getenv("BINX_API_KEY", unset = ""),  
    secret_key = Sys.getenv("BINX_SECRET_KEY", unset = ""),  
    base = "https://fapi.binance.com",  
    recvWindow = 10000L  
)
```

Arguments

api_key	character API key. Leave NULL for unsigned/public requests.
secret_key	character Secret key. Leave NULL for unsigned/public requests.
base	Legacy alias for base_url.
recvWindow	Legacy alias for recv_window.

Value

A binxr_config list.

binxr_config_options *Backward-compatible options config constructor*

Description

Backward-compatible options config constructor

Usage

```
binxr_config_options(  
    api_key = Sys.getenv("BINX_API_KEY", unset = ""),  
    secret_key = Sys.getenv("BINX_SECRET_KEY", unset = ""),  
    base = "https://eapi.binance.com",  
    recvWindow = 10000L  
)
```

Arguments

api_key	character API key. Leave NULL for unsigned/public requests.
secret_key	character Secret key. Leave NULL for unsigned/public requests.
base	Legacy alias for base_url.
recvWindow	Legacy alias for recv_window.

Value

A binxr_config list.

binxr_config_spot	<i>Backward-compatible spot config constructor</i>
-------------------	--

Description

Backward-compatible spot config constructor

Usage

```
binxr_config_spot(  
  api_key = Sys.getenv("BINX_API_KEY", unset = ""),  
  secret_key = Sys.getenv("BINX_SECRET_KEY", unset = ""),  
  base = "https://api.binance.com",  
  recvWindow = 10000L  
)
```

Arguments

api_key	character API key. Leave NULL for unsigned/public requests.
secret_key	character Secret key. Leave NULL for unsigned/public requests.
base	Legacy alias for base_url.
recvWindow	Legacy alias for recv_window.

Value

A binxr_config list.

coerce_account_frames *(Optional) Coerce account to data.frames*

Description

(Optional) Coerce account to data.frames

Usage

```
coerce_account_frames(account)
```

Arguments

account get_account() result

Value

list(assets=..., positions=...)

Examples

```
account <- list(
  assets = list(list(asset = "USDT", walletBalance = "10")),
  positions = list(list(symbol = "ETHUSD", positionAmt = "0"))
)
coerce_account_frames(account)
```

config_futures *Create a Binance futures configuration*

Description

Create a Binance futures configuration

Usage

```
config_futures(
  api_key = Sys.getenv("BINX_API_KEY", unset = ""),
  secret_key = Sys.getenv("BINX_SECRET_KEY", unset = ""),
  base_url = "https://fapi.binance.com",
  recv_window = 10000L,
  timeout = 30,
  verbose = FALSE
)
```

Arguments

api_key	character API key. Leave NULL for unsigned/public requests.
secret_key	character Secret key. Leave NULL for unsigned/public requests.
base_url	character Base URL for the Binance Futures API.
recv_window	integer recvWindow in milliseconds used for signed requests.
timeout	numeric request timeout in seconds.
verbose	logical whether to enable verbose request debugging in future helpers.

Value

A binxr_config list.

Examples

```
cfg <- config_futures(api_key = NULL, secret_key = NULL)
cfg$product
```

config_options	<i>Create a Binance options configuration</i>
----------------	---

Description

Create a Binance options configuration

Usage

```
config_options(
  api_key = Sys.getenv("BINX_API_KEY", unset = ""),
  secret_key = Sys.getenv("BINX_SECRET_KEY", unset = ""),
  base_url = "https://eapi.binance.com",
  recv_window = 10000L,
  timeout = 30,
  verbose = FALSE
)
```

Arguments

api_key	character API key. Leave NULL for unsigned/public requests.
secret_key	character Secret key. Leave NULL for unsigned/public requests.
base_url	character Base URL for the Binance Options API.
recv_window	integer recvWindow in milliseconds used for signed requests.
timeout	numeric request timeout in seconds.
verbose	logical whether to enable verbose request debugging in future helpers.

Value

A binxr_config list.

Examples

```
cfg <- config_options(api_key = NULL, secret_key = NULL)
cfg$product
```

 config_spot

Create a Binance spot configuration

Description

Create a Binance spot configuration

Usage

```
config_spot(
  api_key = Sys.getenv("BINX_API_KEY", unset = ""),
  secret_key = Sys.getenv("BINX_SECRET_KEY", unset = ""),
  base_url = "https://api.binance.com",
  recv_window = 10000L,
  timeout = 30,
  verbose = FALSE
)
```

Arguments

api_key	character API key. Leave NULL for unsigned/public requests.
secret_key	character Secret key. Leave NULL for unsigned/public requests.
base_url	character Base URL for the Binance Spot API.
recv_window	integer recvWindow in milliseconds used for signed requests.
timeout	numeric request timeout in seconds.
verbose	logical whether to enable verbose request debugging in future helpers.

Value

A binxr_config list.

Examples

```
cfg <- config_spot(api_key = NULL, secret_key = NULL)
cfg$product
```

futures_cancel_algo_order

Cancel a Binance Futures algo order

Description

Cancel a Binance Futures algo order

Usage

```
futures_cancel_algo_order(  
    symbol,  
    algo_id = NULL,  
    client_algo_id = NULL,  
    config = config_futures()  
)
```

Arguments

symbol	Trading pair symbol.
algo_id	Optional exchange algo-order ID.
client_algo_id	Optional client algo-order ID.
config	A futures configuration created by config_futures() .

Value

A parsed list.

futures_cancel_all_algo_orders

Cancel all open Binance Futures algo orders

Description

Cancel all open Binance Futures algo orders

Usage

```
futures_cancel_all_algo_orders(symbol, config = config_futures())
```

Arguments

symbol	Trading pair symbol.
config	A futures configuration created by config_futures() .

Value

A parsed list.

futures_cancel_all_orders

Cancel all open Binance Futures orders for a symbol

Description

Cancel all open Binance Futures orders for a symbol

Usage

```
futures_cancel_all_orders(symbol, config = config_futures())
```

```
cancel_fapi_trade_orders_all(symbol, config = binxr_config_futures())
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
config	A futures configuration created by config_futures() .

Value

A parsed list.

futures_cancel_order *Cancel a Binance Futures order*

Description

Cancel a Binance Futures order

Usage

```
futures_cancel_order(
  symbol,
  order_id = NULL,
  orig_client_order_id = NULL,
  config = config_futures()
)

cancel_fapi_trade_order(
  symbol,
  orderId = NULL,
  origClientOrderId = NULL,
  config = binxr_config_futures()
)
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
order_id	Optional exchange order ID.
orig_client_order_id	Optional client order ID.
config	A futures configuration created by <code>config_futures()</code> .
orderId	Legacy alias for order_id.
origClientId	Legacy alias for orig_client_order_id.

Value

A parsed list.

futures_countdown_cancel_all

Set Binance Futures countdown cancel-all

Description

Set Binance Futures countdown cancel-all

Usage

```
futures_countdown_cancel_all(symbol, countdown_time, config = config_futures())
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
countdown_time	Countdown time in milliseconds.
config	A futures configuration created by <code>config_futures()</code> .

Value

A parsed list.

`futures_get_24hr_ticker`*Get Binance Futures 24hr ticker statistics*

Description

Get Binance Futures 24hr ticker statistics

Usage

```
futures_get_24hr_ticker(symbol = NULL, config = config_futures())
```

Arguments

<code>symbol</code>	Optional trading pair symbol.
<code>config</code>	A futures configuration created by config_futures() .

Value

A parsed list.

`futures_get_account`*Get Binance Futures account info*

Description

Get Binance Futures account info

Usage

```
futures_get_account(config = config_futures())  
get_fapi_account(config = binxr_config_futures())
```

Arguments

<code>config</code>	A futures configuration created by config_futures() .
---------------------	---

Value

A parsed list.

futures_get_account_summary
Get Binance Futures account summary

Description

Get Binance Futures account summary

Usage

```
futures_get_account_summary(
  acc = NULL,
  json_list = FALSE,
  config = config_futures()
)

get_fapi_account_summary(acc = NULL, config = binxr_config_futures())
```

Arguments

acc	Optional account payload from futures_get_account() . If NULL, account data is fetched.
json_list	If TRUE, return the parsed list instead of a <code>data.table</code> .
config	A futures configuration created by config_futures() .

Value

A one-row `data.table` by default, or a parsed list when `json_list = TRUE`.

futures_get_account_trades
Get Binance Futures account trades

Description

Get Binance Futures account trades

Usage

```
futures_get_account_trades(
  symbol,
  order_id = NULL,
  startTime = NULL,
  endTime = NULL,
  fromId = NULL,
```

```
    limit = 500,  
    json_list = FALSE,  
    config = config_futures()  
)
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
order_id	Optional order ID.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
fromId	Optional trade ID to start from.
limit	Maximum number of rows to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by <code>config_futures()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

`futures_get_aggregate_trades`*Get Binance Futures aggregate trades*

Description

Get Binance Futures aggregate trades

Usage

```
futures_get_aggregate_trades(  
  symbol,  
  fromId = NULL,  
  startTime = NULL,  
  endTime = NULL,  
  limit = 500,  
  json_list = FALSE,  
  config = config_futures()  
)
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
fromId	Optional aggregate trade ID.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
limit	Maximum number of trades to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by config_futures() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

futures_get_algo_order

Get a Binance Futures algo order

Description

Get a Binance Futures algo order

Usage

```
futures_get_algo_order(  
  symbol,  
  algo_id = NULL,  
  client_algo_id = NULL,  
  config = config_futures()  
)
```

Arguments

symbol	Trading pair symbol.
algo_id	Optional exchange algo-order ID.
client_algo_id	Optional client algo-order ID.
config	A futures configuration created by config_futures() .

Value

A parsed list.

`futures_get_algo_orders`*Get Binance Futures algo order history*

Description

Get Binance Futures algo order history

Usage

```
futures_get_algo_orders(  
    symbol,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 500,  
    json_list = FALSE,  
    config = config_futures()  
)
```

Arguments

<code>symbol</code>	Trading pair symbol.
<code>startTime</code>	Optional start time in milliseconds since Unix epoch.
<code>endTime</code>	Optional end time in milliseconds since Unix epoch.
<code>limit</code>	Maximum number of rows to return. Must not exceed 1000.
<code>json_list</code>	If TRUE, return the parsed list instead of a data.table.
<code>config</code>	A futures configuration created by config_futures() .

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

`futures_get_balance`*Get Binance Futures balances*

Description

Get Binance Futures balances

Usage

```
futures_get_balance(json_list = FALSE, config = config_futures())
```

Arguments

json_list If TRUE, return the parsed list instead of a data.table.
 config A futures configuration created by `config_futures()`.

Value

A data.table by default, or a parsed list when json_list = TRUE.

futures_get_book_ticker

Get Binance Futures order book ticker

Description

Get Binance Futures order book ticker

Usage

```
futures_get_book_ticker(symbol = NULL, config = config_futures())
```

Arguments

symbol Optional trading pair symbol.
 config A futures configuration created by `config_futures()`.

Value

A parsed list.

futures_get_commission_rate

Get Binance Futures commission rate

Description

Get Binance Futures commission rate

Usage

```
futures_get_commission_rate(symbol, config = config_futures())
```

Arguments

symbol Trading pair symbol, for example "ETHUSD".
 config A futures configuration created by `config_futures()`.

Value

A parsed list.

`futures_get_continuous_klines`*Get Binance Futures continuous contract klines*

Description

Get Binance Futures continuous contract klines

Usage

```
futures_get_continuous_klines(  
    pair,  
    contract_type = c("PERPETUAL", "CURRENT_MONTH", "NEXT_MONTH", "CURRENT_QUARTER",  
                      "NEXT_QUARTER"),  
    interval,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 500,  
    json_list = FALSE,  
    config = config_futures()  
)
```

Arguments

<code>pair</code>	Futures pair, for example "BTCUSDT".
<code>contract_type</code>	One of "PERPETUAL", "CURRENT_MONTH", "NEXT_MONTH", "CURRENT_QUARTER", or "NEXT_QUARTER".
<code>interval</code>	Kline interval string.
<code>startTime</code>	Optional start time in milliseconds since Unix epoch.
<code>endTime</code>	Optional end time in milliseconds since Unix epoch.
<code>limit</code>	Maximum number of rows to return. Must not exceed 1500.
<code>json_list</code>	If TRUE, return the parsed list instead of a <code>data.table</code> .
<code>config</code>	A futures configuration created by <code>config_futures()</code> .

Value

A `data.table` by default, or a parsed list when `json_list = TRUE`.

`futures_get_exchange_info`*Get Binance Futures exchange info*

Description

Get Binance Futures exchange info

Usage

```
futures_get_exchange_info(config = config_futures())
```

```
get_fapi_exchange_info(config = binxr_config_futures())
```

Arguments

`config` A futures configuration created by `config_futures()`.

Value

A parsed list. Symbol and asset lists are named when present.

`futures_get_force_orders`*Get Binance Futures force orders*

Description

Get Binance Futures force orders

Usage

```
futures_get_force_orders(  
  symbol = NULL,  
  auto_close_type = NULL,  
  startTime = NULL,  
  endTime = NULL,  
  limit = 50,  
  json_list = FALSE,  
  config = config_futures()  
)
```

Arguments

symbol	Optional trading pair symbol.
auto_close_type	Optional auto-close type.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
limit	Maximum number of rows to return. Must not exceed 100.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by config_futures() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

futures_get_funding_info
Get Binance Futures funding info

Description

Get Binance Futures funding info

Usage

```
futures_get_funding_info(  
  symbol = NULL,  
  json_list = FALSE,  
  config = config_futures()  
)
```

Arguments

symbol	Optional trading pair symbol.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by config_futures() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

`futures_get_funding_rate_history`*Get Binance Futures funding rate history*

Description

Get Binance Futures funding rate history

Usage

```
futures_get_funding_rate_history(  
    symbol = NULL,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 100,  
    json_list = FALSE,  
    config = config_futures()  
)
```

Arguments

<code>symbol</code>	Optional trading pair symbol.
<code>startTime</code>	Optional start time in milliseconds since Unix epoch.
<code>endTime</code>	Optional end time in milliseconds since Unix epoch.
<code>limit</code>	Maximum number of rows to return. Must not exceed 1000.
<code>json_list</code>	If TRUE, return the parsed list instead of a data.table.
<code>config</code>	A futures configuration created by <code>config_futures()</code> .

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

`futures_get_historical_trades`*Get Binance Futures historical trades*

Description

This endpoint returns trades from only the previous month and has request weight 200. Use it sparingly.

Usage

```
futures_get_historical_trades(
  symbol,
  limit = 100,
  fromId = NULL,
  json_list = FALSE,
  config = config_futures()
)
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
limit	Maximum number of trades to return. Must not exceed 500.
fromId	Optional trade identifier to fetch from.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by <code>config_futures()</code> .

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

futures_get_index_constituents

Get Binance Futures index constituents

Description

Get Binance Futures index constituents

Usage

```
futures_get_index_constituents(symbol, config = config_futures())
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
config	A futures configuration created by <code>config_futures()</code> .

Value

A parsed list.

`futures_get_index_price_klines`*Get Binance Futures index price klines*

Description

Get Binance Futures index price klines

Usage

```
futures_get_index_price_klines(  
    symbol,  
    interval,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 500,  
    json_list = FALSE,  
    config = config_futures()  
)
```

Arguments

<code>symbol</code>	Trading pair symbol, for example "ETHUSD".
<code>interval</code>	Kline interval string.
<code>startTime</code>	Optional start time in milliseconds since Unix epoch.
<code>endTime</code>	Optional end time in milliseconds since Unix epoch.
<code>limit</code>	Maximum number of rows to return.
<code>json_list</code>	If TRUE, return the parsed list instead of a <code>data.table</code> .
<code>config</code>	A futures configuration created by <code>config_futures()</code> .

Value

A `data.table` by default, or a parsed list when `json_list = TRUE`.

`futures_get_insurance_balance`*Get Binance Futures insurance-fund balance snapshots*

Description

Get Binance Futures insurance-fund balance snapshots

Usage

```
futures_get_insurance_balance(  
    symbol = NULL,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 30,  
    json_list = FALSE,  
    config = config_futures()  
)
```

Arguments

symbol	Optional trading pair symbol.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
limit	Maximum number of rows to return. Must not exceed 100.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by <code>config_futures()</code> .

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

futures_get_klines	<i>Get Binance Futures klines</i>
--------------------	-----------------------------------

Description

Get Binance Futures klines

Usage

```
futures_get_klines(  
    symbol,  
    interval,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 500,  
    json_list = FALSE,  
    config = config_futures()  
)  
  
get_fapi_klines(  
    symbol,  
    interval,
```

```

    startTime = NULL,
    endTime = NULL,
    limit = 500,
    config = binxr_config_futures()
)

```

Arguments

symbol	Trading pair symbol, for example "ETHUSDT".
interval	Kline interval string.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
limit	Maximum number of rows to return.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by <code>config_futures()</code> .

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

futures_get_mark_price

Get Binance Futures mark price

Description

Get Binance Futures mark price

Usage

```
futures_get_mark_price(symbol = NULL, config = config_futures())
```

```
get_fapi_mark_price(symbol = NULL, config = binxr_config_futures())
```

Arguments

symbol	Optional trading pair symbol such as "ETHUSDT". If NULL, returns all symbols.
config	A futures configuration created by <code>config_futures()</code> .

Value

A parsed list.

`futures_get_mark_price_klines`*Get Binance Futures mark price klines*

Description

Get Binance Futures mark price klines

Usage

```
futures_get_mark_price_klines(  
    symbol,  
    interval,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 500,  
    json_list = FALSE,  
    config = config_futures()  
)
```

Arguments

<code>symbol</code>	Trading pair symbol, for example "ETHUSD".
<code>interval</code>	Kline interval string.
<code>startTime</code>	Optional start time in milliseconds since Unix epoch.
<code>endTime</code>	Optional end time in milliseconds since Unix epoch.
<code>limit</code>	Maximum number of rows to return.
<code>json_list</code>	If TRUE, return the parsed list instead of a <code>data.table</code> .
<code>config</code>	A futures configuration created by <code>config_futures()</code> .

Value

A `data.table` by default, or a parsed list when `json_list = TRUE`.

`futures_get_multi_assets_mode`*Get Binance Futures multi-assets mode*

Description

Get Binance Futures multi-assets mode

Usage

```
futures_get_multi_assets_mode(config = config_futures())
```

Arguments

config A futures configuration created by `config_futures()`.

Value

A parsed list.

futures_get_open_algo_orders

Get open Binance Futures algo orders

Description

Get open Binance Futures algo orders

Usage

```
futures_get_open_algo_orders(  
  symbol = NULL,  
  json_list = FALSE,  
  config = config_futures()  
)
```

Arguments

symbol Optional trading pair symbol.

json_list If TRUE, return the parsed list instead of a data.table.

config A futures configuration created by `config_futures()`.

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

futures_get_open_interest
Get Binance Futures open interest

Description

Get Binance Futures open interest

Usage

```
futures_get_open_interest(symbol, config = config_futures())
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
config	A futures configuration created by config_futures() .

Value

A parsed list.

futures_get_open_orders
Get Binance Futures open orders

Description

Get Binance Futures open orders

Usage

```
futures_get_open_orders(  
  symbol = NULL,  
  json_list = FALSE,  
  config = config_futures()  
)
```

```
get_fapi_trade_open_orders(symbol = NULL, config = binxr_config_futures())
```

Arguments

symbol	Optional trading pair symbol. If NULL, returns all open orders.
json_list	If TRUE, return the parsed list instead of a data table.
config	A futures configuration created by config_futures() .

Value

A `data.table` by default, or a parsed list when `json_list = TRUE`.

futures_get_order	<i>Get a Binance Futures order</i>
-------------------	------------------------------------

Description

Get a Binance Futures order

Usage

```
futures_get_order(  
  symbol,  
  order_id = NULL,  
  orig_client_order_id = NULL,  
  config = config_futures()  
)  
  
get_fapi_trade_order(  
  symbol,  
  orderId = NULL,  
  origClientId = NULL,  
  config = binxr_config_futures()  
)
```

Arguments

<code>symbol</code>	Trading pair symbol, for example "ETHUSDT".
<code>order_id</code>	Optional exchange order ID.
<code>orig_client_order_id</code>	Optional client order ID.
<code>config</code>	A futures configuration created by <code>config_futures()</code> .
<code>orderId</code>	Legacy alias for <code>order_id</code> .
<code>origClientId</code>	Legacy alias for <code>orig_client_order_id</code> .

Value

A parsed list.

futures_get_orders	<i>Get Binance Futures order history</i>
--------------------	--

Description

Get Binance Futures order history

Usage

```
futures_get_orders(  
    symbol,  
    limit = 500,  
    json_list = FALSE,  
    config = config_futures()  
)  
  
get_fapi_trade_orders(symbol, limit = 500, config = binxr_config_futures())
```

Arguments

symbol	Trading pair symbol, for example "ETHUSDT".
limit	Maximum number of orders to return.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by config_futures() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

futures_get_order_amendments	<i>Get Binance Futures order modification history</i>
------------------------------	---

Description

Get Binance Futures order modification history

Usage

```
futures_get_order_amendments(
    symbol,
    order_id = NULL,
    orig_client_order_id = NULL,
    startTime = NULL,
    endTime = NULL,
    limit = 500,
    json_list = FALSE,
    config = config_futures()
)
```

Arguments

symbol	Trading pair symbol.
order_id	Optional exchange order ID.
orig_client_order_id	Optional client order ID.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional start time in milliseconds since Unix epoch.
limit	Maximum number of rows to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by <code>config_futures()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

futures_get_order_book

Get Binance Futures order book

Description

Get Binance Futures order book

Usage

```
futures_get_order_book(symbol, limit = NULL, config = config_futures())
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
limit	Optional depth limit. One of 5, 10, 20, 50, 100, 500, or 1000.
config	A futures configuration created by <code>config_futures()</code> .

Value

A parsed list.

futures_get_order_rate_limit	<i>Get Binance Futures order rate limits</i>
------------------------------	--

Description

Get Binance Futures order rate limits

Usage

```
futures_get_order_rate_limit(json_list = FALSE, config = config_futures())
```

Arguments

json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by config_futures() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

futures_get_positions	<i>Get standardized Binance Futures positions</i>
-----------------------	---

Description

Get standardized Binance Futures positions

Usage

```
futures_get_positions(acc = NULL, json_list = FALSE, config = config_futures())

get_fapi_account_positions(acc = NULL, config = binxr_config_futures())
```

Arguments

acc	Optional account payload from futures_get_account() . If NULL, account data is fetched.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by config_futures() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

`futures_get_position_mode`*Get Binance Futures position mode*

Description

Get Binance Futures position mode

Usage

```
futures_get_position_mode(config = config_futures())
```

Arguments

`config` A futures configuration created by `config_futures()`.

Value

A parsed list.

`futures_get_position_risk`*Get Binance Futures position risk*

Description

Get Binance Futures position risk

Usage

```
futures_get_position_risk(json_list = FALSE, config = config_futures())
```

```
get_fapi_account_position_risk(config = binxr_config_futures())
```

Arguments

`json_list` If TRUE, return the parsed list instead of a `data.table`.

`config` A futures configuration created by `config_futures()`.

Value

A `data.table` by default, or a parsed list when `json_list = TRUE`.

```
futures_get_premium_index_klines
```

Get Binance Futures premium index klines

Description

Get Binance Futures premium index klines

Usage

```
futures_get_premium_index_klines(  
    symbol,  
    interval,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 500,  
    json_list = FALSE,  
    config = config_futures()  
)
```

Arguments

symbol	Trading pair symbol, for example "ETHUSDT".
interval	Kline interval string.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
limit	Maximum number of rows to return.
json_list	If TRUE, return the parsed list instead of a <code>data.table</code> .
config	A futures configuration created by <code>config_futures()</code> .

Value

A `data.table` by default, or a parsed list when `json_list = TRUE`.

```
futures_get_recent_trades
```

Get Binance Futures recent trades

Description

Get Binance Futures recent trades

Usage

```
futures_get_recent_trades(
    symbol,
    limit = 500,
    json_list = FALSE,
    config = config_futures()
)
```

Arguments

symbol	Trading pair symbol, for example "ETHUSDT".
limit	Maximum number of trades to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A futures configuration created by config_futures() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

futures_get_rpi_depth *Get Binance Futures RPI order book*

Description

Get Binance Futures RPI order book

Usage

```
futures_get_rpi_depth(symbol, limit = NULL, config = config_futures())
```

Arguments

symbol	Trading pair symbol, for example "ETHUSDT".
limit	Optional depth limit. One of 5, 10, 20, 50, 100, 500, or 1000.
config	A futures configuration created by config_futures() .

Value

A parsed list.

`futures_get_server_time`*Get Binance Futures server time*

Description

Get Binance Futures server time

Usage

```
futures_get_server_time(config = config_futures())
```

```
get_fapi_system_time(config = binxr_config_futures())
```

Arguments

`config` A futures configuration created by [config_futures\(\)](#).

Value

A POSIXct timestamp.

`futures_get_symbol_adl_risk`*Get Binance Futures symbol-level ADL risk*

Description

Get Binance Futures symbol-level ADL risk

Usage

```
futures_get_symbol_adl_risk(symbol, config = config_futures())
```

Arguments

`symbol` Trading pair symbol, for example "ETHUSD".

`config` A futures configuration created by [config_futures\(\)](#).

Value

A parsed list.

futures_get_ticker_price
Get Binance Futures ticker price

Description

Get Binance Futures ticker price

Usage

```
futures_get_ticker_price(  
  symbol = NULL,  
  version = c("v2", "deprecated"),  
  config = config_futures()  
)
```

Arguments

symbol	Optional trading pair symbol.
version	One of "v2" or "deprecated".
config	A futures configuration created by config_futures() .

Value

A parsed list.

futures_get_trading_schedule
Get Binance Futures trading schedules

Description

Get Binance Futures trading schedules

Usage

```
futures_get_trading_schedule(config = config_futures())
```

Arguments

config	A futures configuration created by config_futures() .
--------	---

Value

A parsed list.

futures_modify_order	<i>Modify a Binance Futures limit order</i>
----------------------	---

Description

Modify a Binance Futures limit order

Usage

```
futures_modify_order(  
    symbol,  
    side = c("BUY", "SELL"),  
    quantity,  
    price,  
    order_id = NULL,  
    orig_client_order_id = NULL,  
    modify_id = NULL,  
    config = config_futures()  
)
```

Arguments

symbol	Trading pair symbol.
side	One of "BUY" or "SELL".
quantity	Replacement order quantity.
price	Replacement limit price.
order_id	Optional exchange order ID.
orig_client_order_id	Optional client order ID.
modify_id	Optional user-defined modification identifier.
config	A futures configuration created by config_futures() .

Value

A parsed list.

futures_ping	<i>Test Binance Futures connectivity</i>
--------------	--

Description

Test Binance Futures connectivity

Usage

```
futures_ping(config = config_futures())
```

Arguments

config A futures configuration created by `config_futures()`.

Value

A parsed list.

futures_place_algo_order	<i>Place a Binance Futures algo order</i>
--------------------------	---

Description

Places a USD-M Futures conditional order through the Algo Service. Use this function for stop-loss, take-profit, and trailing-stop order types.

Usage

```
futures_place_algo_order(
  symbol,
  side = c("BUY", "SELL"),
  type = c("STOP", "STOP_MARKET", "TAKE_PROFIT", "TAKE_PROFIT_MARKET",
    "TRAILING_STOP_MARKET"),
  quantity = NULL,
  price = NULL,
  trigger_price = NULL,
  time_in_force = NULL,
  position_side = c("BOTH", "LONG", "SHORT"),
  reduce_only = FALSE,
  close_position = FALSE,
  working_type = c("CONTRACT_PRICE", "MARK_PRICE"),
  price_protect = FALSE,
  activation_price = NULL,
```

```

    callback_rate = NULL,
    new_client_algo_id = NULL,
    config = config_futures()
)

```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
side	One of "BUY" or "SELL".
type	Conditional order type.
quantity	Optional order quantity. Omit only when close_position is TRUE.
price	Optional limit price for "STOP" and "TAKE_PROFIT".
trigger_price	Trigger price for non-trailing conditional orders.
time_in_force	Optional time-in-force for limit conditional orders.
position_side	One of "BOTH", "LONG", or "SHORT".
reduce_only	Whether the order is reduce-only.
close_position	Whether to close the entire position when triggered.
working_type	Trigger price source.
price_protect	Whether to enable trigger price protection.
activation_price	Optional activation price for a trailing stop.
callback_rate	Required callback rate for a trailing stop.
new_client_algo_id	Optional client algo-order identifier.
config	A futures configuration created by <code>config_futures()</code> .

Value

A parsed list.

futures_place_order	<i>Place a Binance Futures order</i>
---------------------	--------------------------------------

Description

Place a Binance Futures order

Usage

```
futures_place_order(
    symbol,
    side = c("BUY", "SELL"),
    type = c("LIMIT", "MARKET", "STOP", "STOP_MARKET", "TAKE_PROFIT", "TAKE_PROFIT_MARKET",
            "TRAILING_STOP_MARKET"),
    quantity,
    price = NULL,
    time_in_force = NULL,
    position_side = c("BOTH", "LONG", "SHORT"),
    reduce_only = FALSE,
    stop_price = NULL,
    working_type = NULL,
    new_client_order_id = NULL,
    config = config_futures()
)

place_fapi_trade_order(
    symbol,
    side = c("BUY", "SELL"),
    type = c("LIMIT", "MARKET", "STOP", "STOP_MARKET", "TAKE_PROFIT", "TAKE_PROFIT_MARKET",
            "TRAILING_STOP_MARKET"),
    quantity,
    price = NULL,
    timeInForce = c("GTC", "IOC", "FOK", "GTX"),
    positionSide = c("BOTH", "LONG", "SHORT"),
    reduceOnly = FALSE,
    stopPrice = NULL,
    workingType = c("CONTRACT_PRICE", "MARK_PRICE"),
    newClientOrderId = NULL,
    config = binxr_config_futures()
)
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
side	One of "BUY" or "SELL".
type	Order type.
quantity	Order quantity.
price	Optional limit price.
time_in_force	Optional time-in-force value. Required for LIMIT.
position_side	One of "BOTH", "LONG", or "SHORT".
reduce_only	Whether the order is reduce-only.
stop_price	Optional trigger price for conditional orders.
working_type	Optional trigger source. Must only be supplied with stop_price.

new_client_order_id	Optional client order identifier.
config	A futures configuration created by <code>config_futures()</code> .
timeInForce	Legacy alias for <code>time_in_force</code> .
positionSide	Legacy alias for <code>position_side</code> .
reduceOnly	Legacy alias for <code>reduce_only</code> .
stopPrice	Legacy alias for <code>stop_price</code> .
workingType	Legacy alias for <code>working_type</code> .
newClientOrderId	Legacy alias for <code>new_client_order_id</code> .

Value

A parsed list.

futures_set_leverage	<i>Set Binance Futures leverage</i>
----------------------	-------------------------------------

Description

Set Binance Futures leverage

Usage

```
futures_set_leverage(symbol, leverage, config = config_futures())
```

```
set_fapi_account_leverage(symbol, leverage, config = binxr_config_futures())
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
leverage	Desired leverage.
config	A futures configuration created by <code>config_futures()</code> .

Value

A parsed list.

```
futures_set_margin_type
```

Set Binance Futures margin type

Description

Set Binance Futures margin type

Usage

```
futures_set_margin_type(
    symbol,
    margin_type = c("CROSSED", "ISOLATED"),
    config = config_futures()
)

set_fapi_account_margin_type(
    symbol,
    marginType = c("CROSSED", "ISOLATED"),
    config = binxr_config_futures()
)
```

Arguments

symbol	Trading pair symbol, for example "ETHUSD".
margin_type	One of "CROSSED" or "ISOLATED".
config	A futures configuration created by config_futures() .
marginType	Legacy alias for margin_type.

Value

A parsed list.

```
futures_set_multi_assets_mode
```

Set Binance Futures multi-assets mode

Description

Set Binance Futures multi-assets mode

Usage

```
futures_set_multi_assets_mode(  
    multi_assets_margin = TRUE,  
    config = config_futures()  
)
```

Arguments

`multi_assets_margin` TRUE to enable multi-assets mode, FALSE to disable it.

`config` A futures configuration created by [config_futures\(\)](#).

Value

A parsed list.

`futures_set_position_mode`

Set Binance Futures position mode

Description

Set Binance Futures position mode

Usage

```
futures_set_position_mode(dual_side_position = TRUE, config = config_futures())  
  
set_fapi_account_position_side(  
    dualSidePosition = TRUE,  
    config = binxr_config_futures()  
)
```

Arguments

`dual_side_position` TRUE for hedge mode, FALSE for one-way mode.

`config` A futures configuration created by [config_futures\(\)](#).

`dualSidePosition` Legacy alias for `dual_side_position`.

Value

A parsed list.

futures_test_order	<i>Test a Binance Futures order</i>
--------------------	-------------------------------------

Description

Test a Binance Futures order

Usage

```
futures_test_order(
    symbol,
    side = c("BUY", "SELL"),
    type = c("LIMIT", "MARKET", "STOP", "STOP_MARKET", "TAKE_PROFIT", "TAKE_PROFIT_MARKET",
            "TRAILING_STOP_MARKET"),
    quantity,
    price = NULL,
    time_in_force = NULL,
    position_side = c("BOTH", "LONG", "SHORT"),
    reduce_only = FALSE,
    stop_price = NULL,
    working_type = NULL,
    new_client_order_id = NULL,
    config = config_futures()
)
```

Arguments

symbol	Trading pair symbol, for example "ETHUSDT".
side	One of "BUY" or "SELL".
type	Order type.
quantity	Order quantity.
price	Optional limit price.
time_in_force	Optional time-in-force value. Required for LIMIT.
position_side	One of "BOTH", "LONG", or "SHORT".
reduce_only	Whether the order is reduce-only.
stop_price	Optional trigger price for conditional orders.
working_type	Optional trigger source. Must only be supplied with stop_price.
new_client_order_id	Optional client order identifier.
config	A futures configuration created by config_futures() .

Value

A parsed list.

`options_cancel_all_orders`*Cancel all Binance Options orders for a symbol*

Description

Cancel all Binance Options orders for a symbol

Usage

```
options_cancel_all_orders(symbol, config = config_options())
```

Arguments

<code>symbol</code>	Option symbol, for example "BTC-200730-9000-C".
<code>config</code>	An options configuration created by <code>config_options()</code> .

Value

A parsed list.

`options_cancel_all_orders_by_underlying`*Cancel all Binance Options orders by underlying*

Description

Cancel all Binance Options orders by underlying

Usage

```
options_cancel_all_orders_by_underlying(underlying, config = config_options())
```

Arguments

<code>underlying</code>	Underlying symbol, for example "BTCUSDT".
<code>config</code>	An options configuration created by <code>config_options()</code> .

Value

A parsed list.

options_cancel_order *Cancel a Binance Options order*

Description

Cancel a Binance Options order

Usage

```
options_cancel_order(  
    symbol,  
    order_id = NULL,  
    client_order_id = NULL,  
    config = config_options()  
)
```

Arguments

symbol	Option symbol, for example "BTC-200730-9000-C".
order_id	Optional exchange order ID.
client_order_id	Optional client order ID.
config	An options configuration created by config_options() .

Value

A parsed list.

options_get_24hr_ticker
 Get Binance Options 24hr ticker statistics

Description

Get Binance Options 24hr ticker statistics

Usage

```
options_get_24hr_ticker(  
    symbol = NULL,  
    json_list = FALSE,  
    config = config_options()  
)
```

Arguments

symbol	Optional option symbol.
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by <code>config_options()</code> .

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

options_get_account_trades

Get Binance Options account trades

Description

Get Binance Options account trades

Usage

```
options_get_account_trades(
  symbol = NULL,
  fromId = NULL,
  startTime = NULL,
  endTime = NULL,
  limit = 100,
  json_list = FALSE,
  config = config_options()
)
```

Arguments

symbol	Optional option symbol.
fromId	Optional trade ID to start from.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
limit	Maximum number of rows to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by <code>config_options()</code> .

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

`options_get_commission`*Get Binance Options user commission*

Description

Get Binance Options user commission

Usage

```
options_get_commission(config = config_options())
```

Arguments

`config` An options configuration created by `config_options()`.

Value

A parsed list.

`options_get_exchange_info`*Get Binance Options exchange info*

Description

Get Binance Options exchange info

Usage

```
options_get_exchange_info(config = config_options())
```

Arguments

`config` An options configuration created by `config_options()`.

Value

A parsed list. Option symbol and asset lists are named when present.

`options_get_exercise_history`*Get Binance Options historical exercise records*

Description

Get Binance Options historical exercise records

Usage

```
options_get_exercise_history(  
    underlying = NULL,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 100,  
    json_list = FALSE,  
    config = config_options()  
)
```

Arguments

<code>underlying</code>	Optional underlying symbol, for example "BTCUSDT".
<code>startTime</code>	Optional start time in milliseconds since Unix epoch.
<code>endTime</code>	Optional end time in milliseconds since Unix epoch.
<code>limit</code>	Maximum number of rows to return. Must not exceed 100.
<code>json_list</code>	If TRUE, return the parsed list instead of a <code>data.table</code> .
<code>config</code>	An options configuration created by <code>config_options()</code> .

Value

A `data.table` by default, or a parsed list when `json_list = TRUE`.

`options_get_exercise_records`*Get Binance Options exercise records*

Description

Get Binance Options exercise records

Usage

```
options_get_exercise_records(  
    symbol = NULL,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 1000,  
    json_list = FALSE,  
    config = config_options()  
)
```

Arguments

symbol	Optional option symbol.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
limit	Maximum number of rows to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by <code>config_options()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

options_get_funding_flow

Get Binance Options funding flow

Description

Get Binance Options funding flow

Usage

```
options_get_funding_flow(  
    currency = "USDT",  
    recordId = NULL,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 100,  
    json_list = FALSE,  
    config = config_options()  
)
```

Arguments

currency	Asset type. Currently Binance supports "USDT".
recordId	Optional record ID to start from.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
limit	Maximum number of rows to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by config_options() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

options_get_index_price

Get Binance Options underlying index price

Description

Get Binance Options underlying index price

Usage

```
options_get_index_price(underlying, config = config_options())
```

Arguments

underlying	Underlying spot pair, for example "BTCUSDT".
config	An options configuration created by config_options() .

Value

A parsed list.

options_get_klines	<i>Get Binance Options klines</i>
--------------------	-----------------------------------

Description

Get Binance Options klines

Usage

```
options_get_klines(
    symbol,
    interval,
    startTime = NULL,
    endTime = NULL,
    limit = 500,
    json_list = FALSE,
    config = config_options()
)
```

Arguments

symbol	Option symbol, for example "BTC-200730-9000-C".
interval	Kline interval string.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
limit	Maximum number of rows to return. Must not exceed 1500.
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by config_options() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

options_get_margin_account	<i>Get Binance Options margin account information</i>
----------------------------	---

Description

Get Binance Options margin account information

Usage

```
options_get_margin_account(config = config_options())
```

Arguments

config An options configuration created by `config_options()`.

Value

A parsed list.

`options_get_mark_price`*Get Binance Options mark prices*

Description

Get Binance Options mark prices

Usage

```
options_get_mark_price(  
  symbol = NULL,  
  json_list = FALSE,  
  config = config_options()  
)
```

Arguments

symbol Optional option symbol.

json_list If TRUE, return the parsed list instead of a data. table.

config An options configuration created by `config_options()`.

Value

A data. table by default, or a parsed list when `json_list = TRUE`.

`options_get_open_interest`*Get Binance Options open interest*

Description

Get Binance Options open interest

Usage

```
options_get_open_interest(
    underlying_asset,
    expiration,
    json_list = FALSE,
    config = config_options()
)
```

Arguments

underlying_asset	Underlying asset, for example "BTCUSDT".
expiration	Expiration date, for example "221225".
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by <code>config_options()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

options_get_open_orders

Get Binance Options open orders

Description

Get Binance Options open orders

Usage

```
options_get_open_orders(
    symbol = NULL,
    order_id = NULL,
    startTime = NULL,
    endTime = NULL,
    json_list = FALSE,
    config = config_options()
)
```

Arguments

symbol	Optional option symbol.
order_id	Optional order ID to start from.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by <code>config_options()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

options_get_order	<i>Get a Binance Options order</i>
-------------------	------------------------------------

Description

Get a Binance Options order

Usage

```
options_get_order(  
  symbol,  
  order_id = NULL,  
  client_order_id = NULL,  
  config = config_options()  
)
```

Arguments

symbol	Option symbol, for example "BTC-200730-9000-C".
order_id	Optional exchange order ID.
client_order_id	Optional client order ID.
config	An options configuration created by config_options() .

Value

A parsed list.

options_get_order_book	<i>Get Binance Options order book</i>
------------------------	---------------------------------------

Description

Get Binance Options order book

Usage

```
options_get_order_book(symbol, limit = NULL, config = config_options())
```

Arguments

symbol	Option symbol, for example "BTC-200730-9000-C".
limit	Optional depth limit. One of 10, 20, 50, 100, 500, or 1000.
config	An options configuration created by <code>config_options()</code> .

Value

A parsed list.

options_get_order_history
<i>Get Binance Options order history</i>

Description

Get Binance Options order history

Usage

```
options_get_order_history(  
  symbol,  
  order_id = NULL,  
  startTime = NULL,  
  endTime = NULL,  
  limit = 100,  
  json_list = FALSE,  
  config = config_options()  
)
```

Arguments

symbol	Option symbol, for example "BTC-200730-9000-C".
order_id	Optional order ID to start from.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
limit	Maximum number of rows to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by <code>config_options()</code> .

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

options_get_positions *Get Binance Options positions*

Description

Get Binance Options positions

Usage

```
options_get_positions(  
    symbol = NULL,  
    json_list = FALSE,  
    config = config_options()  
)
```

Arguments

symbol	Optional option symbol.
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by config_options() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

options_get_recent_block_trades
 Get Binance Options recent block trades

Description

Get Binance Options recent block trades

Usage

```
options_get_recent_block_trades(  
    symbol = NULL,  
    limit = 100,  
    json_list = FALSE,  
    config = config_options()  
)
```

Arguments

symbol	Optional option symbol.
limit	Maximum number of trades to return. Must not exceed 500.
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by <code>config_options()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

options_get_recent_trades

Get Binance Options recent trades

Description

Get Binance Options recent trades

Usage

```
options_get_recent_trades(  
  symbol,  
  limit = 100,  
  json_list = FALSE,  
  config = config_options()  
)
```

Arguments

symbol	Option symbol, for example "BTC-200730-9000-C".
limit	Maximum number of trades to return. Must not exceed 500.
json_list	If TRUE, return the parsed list instead of a data.table.
config	An options configuration created by <code>config_options()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

`options_get_server_time`*Get Binance Options server time*

Description

Get Binance Options server time

Usage

```
options_get_server_time(config = config_options())
```

Arguments

`config` An options configuration created by `config_options()`.

Value

A POSIXct timestamp.

`options_ping`*Test Binance Options connectivity*

Description

Test Binance Options connectivity

Usage

```
options_ping(config = config_options())
```

Arguments

`config` An options configuration created by `config_options()`.

Value

A parsed list.

options_place_order	<i>Place a Binance Options order</i>
---------------------	--------------------------------------

Description

Place a Binance Options order

Usage

```
options_place_order(
    symbol,
    side = c("BUY", "SELL"),
    type = c("LIMIT"),
    quantity,
    price,
    time_in_force = c("GTC", "IOC", "FOK"),
    reduce_only = FALSE,
    post_only = FALSE,
    new_order_resp_type = c("ACK", "RESULT"),
    client_order_id = NULL,
    is_mmp = FALSE,
    self_trade_prevention_mode = NULL,
    config = config_options()
)
```

Arguments

symbol	Option symbol, for example "BTC-200730-9000-C".
side	One of "BUY" or "SELL".
type	Order type. Only "LIMIT" is currently supported.
quantity	Order quantity.
price	Order price.
time_in_force	Time in force. Default is "GTC".
reduce_only	Whether the order is reduce-only.
post_only	Whether the order is post-only.
new_order_resp_type	Response type. One of "ACK" or "RESULT".
client_order_id	Optional client order ID.
is_mmp	Whether the order is an MMP order.
self_trade_prevention_mode	Optional STP mode.
config	An options configuration created by <code>config_options()</code> .

Value

A parsed list.

round_price_qty	<i>Round price and quantity to exchange increments</i>
-----------------	--

Description

Round price and quantity to exchange increments

Usage

```
round_price_qty(  
    exchangeInfo,  
    symbol,  
    price = NULL,  
    quantity = NULL,  
    round_dir = c("down", "up")  
)  
  
util_round_price_qty(  
    exchangeInfo,  
    symbol,  
    price = NULL,  
    quantity = NULL,  
    round_dir = c("down", "up")  
)
```

Arguments

exchangeInfo	Exchange info from futures_get_exchange_info() .
symbol	Trading pair symbol, for example "ETHUSD".
price	Optional price to round using the symbol tick size.
quantity	Optional quantity to round using the symbol step size.
round_dir	Direction: "down" or "up".

Value

A list with elements price and quantity.

Examples

```
exchange_info <- list(
  symbols = list(
    ETHUSDT = list(
      filters = list(
        list(filterType = "PRICE_FILTER", tickSize = "0.01"),
        list(filterType = "LOT_SIZE", stepSize = "0.001")
      )
    )
  )
)
round_price_qty(exchange_info, "ETHUSDT", price = 1800.123, quantity = 0.12345)
```

spot_amend_order_keep_priority

Amend a Binance Spot order while keeping priority

Description

Amend a Binance Spot order while keeping priority

Usage

```
spot_amend_order_keep_priority(
  symbol,
  order_id = NULL,
  orig_client_order_id = NULL,
  new_client_order_id = NULL,
  new_qty,
  config = config_spot()
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
order_id	Optional exchange order ID.
orig_client_order_id	Optional client order ID.
new_client_order_id	Optional new client order ID after amendment.
new_qty	New order quantity. Must be positive.
config	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list.

`spot_cancel_all_orders`*Cancel all open Binance Spot orders for a symbol*

Description

Cancel all open Binance Spot orders for a symbol

Usage

```
spot_cancel_all_orders(symbol, config = config_spot())
```

Arguments

<code>symbol</code>	Trading pair symbol, for example "BTCUSDT".
<code>config</code>	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list by default.

`spot_cancel_order`*Cancel a Binance Spot order*

Description

Cancel a Binance Spot order

Usage

```
spot_cancel_order(  
  symbol,  
  order_id = NULL,  
  orig_client_order_id = NULL,  
  new_client_order_id = NULL,  
  cancel_restrictions = NULL,  
  config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
order_id	Optional exchange order ID.
orig_client_order_id	Optional client order ID.
new_client_order_id	Optional client order ID for the cancel request.
cancel_restrictions	Optional cancel restriction. One of "ONLY_NEW" or "ONLY_PARTIALLY_FILLED".
config	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list.

spot_cancel_order_list

Cancel a Binance Spot order list

Description

Cancel a Binance Spot order list

Usage

```
spot_cancel_order_list(
  symbol,
  order_list_id = NULL,
  list_client_order_id = NULL,
  new_client_order_id = NULL,
  config = config_spot()
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
order_list_id	Optional exchange order-list ID.
list_client_order_id	Optional order-list client ID.
new_client_order_id	Optional client order ID for the cancel request.
config	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list.

spot_cancel_replace_order*Cancel and replace a Binance Spot order*

Description

Cancel and replace a Binance Spot order

Usage

```
spot_cancel_replace_order(  
    symbol,  
    side = c("BUY", "SELL"),  
    type = c("LIMIT", "MARKET", "STOP_LOSS", "STOP_LOSS_LIMIT", "TAKE_PROFIT",  
            "TAKE_PROFIT_LIMIT", "LIMIT_MAKER"),  
    cancel_replace_mode = c("STOP_ON_FAILURE", "ALLOW_FAILURE"),  
    time_in_force = NULL,  
    quantity = NULL,  
    quote_order_qty = NULL,  
    price = NULL,  
    cancel_new_client_order_id = NULL,  
    cancel_orig_client_order_id = NULL,  
    cancel_order_id = NULL,  
    new_client_order_id = NULL,  
    strategy_id = NULL,  
    strategy_type = NULL,  
    stop_price = NULL,  
    trailing_delta = NULL,  
    iceberg_qty = NULL,  
    new_order_resp_type = NULL,  
    self_trade_prevention_mode = NULL,  
    cancel_restrictions = NULL,  
    order_rate_limit_exceeded_mode = NULL,  
    peg_price_type = NULL,  
    peg_offset_value = NULL,  
    peg_offset_type = NULL,  
    config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
side	One of "BUY" or "SELL".
type	Order type.
cancel_replace_mode	One of "STOP_ON_FAILURE" or "ALLOW_FAILURE".

time_in_force	Optional time-in-force value.
quantity	Optional order quantity.
quote_order_qty	Optional quote-order quantity for MARKET orders.
price	Optional limit price.
cancel_new_client_order_id	Optional client order ID for the cancel request.
cancel_orig_client_order_id	Optional original client order ID to cancel.
cancel_order_id	Optional exchange order ID to cancel.
new_client_order_id	Optional client order identifier.
strategy_id	Optional numeric strategy identifier.
strategy_type	Optional numeric strategy type. Must be at least 1000000.
stop_price	Optional trigger price.
trailing_delta	Optional trailing delta value.
iceberg_qty	Optional iceberg quantity.
new_order_resp_type	Optional response type. One of "ACK", "RESULT", or "FULL".
self_trade_prevention_mode	Optional self-trade prevention mode.
cancel_restrictions	Optional cancel restriction. One of "ONLY_NEW" or "ONLY_PARTIALLY_FILLED".
order_rate_limit_exceeded_mode	Optional order-rate-limit behavior. One of "DO_NOTHING" or "CANCEL_ONLY".
peg_price_type	Optional pegged price type. One of "PRIMARY_PEG" or "MARKET_PEG".
peg_offset_value	Optional peg offset value.
peg_offset_type	Optional peg offset type. Only "PRICE_LEVEL" is supported.
config	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list.

spot_get_24hr_ticker	<i>Get Binance Spot 24-hour ticker statistics</i>
----------------------	---

Description

Get Binance Spot 24-hour ticker statistics

Usage

```
spot_get_24hr_ticker(  
  symbol = NULL,  
  symbols = NULL,  
  type = c("FULL", "MINI"),  
  symbol_status = NULL,  
  json_list = FALSE,  
  config = config_spot()  
)
```

Arguments

symbol	Optional trading pair symbol.
symbols	Optional character vector of trading pair symbols.
type	One of "FULL" or "MINI".
symbol_status	Optional trading status filter.
json_list	If TRUE, return the parsed list instead of converting array responses to <code>data.table</code> .
config	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list for single-symbol requests, or a `data.table` for multi-symbol/all-symbol requests unless `json_list = TRUE`.

spot_get_account	<i>Get Binance Spot account information</i>
------------------	---

Description

Get Binance Spot account information

Usage

```
spot_get_account(omit_zero_balances = FALSE, config = config_spot())
```

Arguments

omit_zero_balances	Whether to omit zero balances from the response.
config	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list.

spot_get_account_trades

Get Binance Spot account trades

Description

Get Binance Spot account trades

Usage

```
spot_get_account_trades(
  symbol,
  order_id = NULL,
  start_time = NULL,
  end_time = NULL,
  from_id = NULL,
  limit = 500,
  json_list = FALSE,
  config = config_spot()
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
order_id	Optional exchange order ID.
start_time	Optional start time in milliseconds.
end_time	Optional end time in milliseconds.
from_id	Optional trade ID to start from.
limit	Maximum number of trades to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by <code>config_spot()</code> .

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

spot_get_aggregate_trades *Get Binance Spot aggregate trades*

Description

Get Binance Spot aggregate trades

Usage

```
spot_get_aggregate_trades(  
    symbol,  
    fromId = NULL,  
    startTime = NULL,  
    endTime = NULL,  
    limit = 500,  
    json_list = FALSE,  
    config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
fromId	Optional aggregate trade identifier to fetch from, inclusive.
startTime	Optional start time in milliseconds since Unix epoch, inclusive.
endTime	Optional end time in milliseconds since Unix epoch, inclusive.
limit	Maximum number of rows to return.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by config_spot() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

spot_get_allocations *Get Binance Spot allocations*

Description

Get Binance Spot allocations

Usage

```
spot_get_allocations(  
    symbol,  
    start_time = NULL,  
    end_time = NULL,  
    from_allocation_id = NULL,  
    limit = 500,  
    order_id = NULL,  
    json_list = FALSE,  
    config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
start_time	Optional start time in milliseconds.
end_time	Optional end time in milliseconds.
from_allocation_id	Optional allocation ID to start from.
limit	Maximum number of records to return. Must not exceed 1000.
order_id	Optional exchange order ID.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by <code>config_spot()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

spot_get_all_order_lists

Get all Binance Spot order lists

Description

Get all Binance Spot order lists

Usage

```
spot_get_all_order_lists(  
    from_id = NULL,  
    start_time = NULL,  
    end_time = NULL,  
    limit = 500,  
    json_list = FALSE,  
    config = config_spot()  
)
```

Arguments

from_id	Optional order-list ID to start from.
start_time	Optional start time in milliseconds.
end_time	Optional end time in milliseconds.
limit	Maximum number of order lists to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by <code>config_spot()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

`spot_get_average_price`*Get Binance Spot average price*

Description

Get Binance Spot average price

Usage

```
spot_get_average_price(symbol, config = config_spot())
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
config	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list.

spot_get_book_ticker *Get Binance Spot book ticker*

Description

Get Binance Spot book ticker

Usage

```
spot_get_book_ticker(  
  symbol = NULL,  
  symbols = NULL,  
  symbol_status = NULL,  
  json_list = FALSE,  
  config = config_spot()  
)
```

Arguments

symbol	Optional trading pair symbol.
symbols	Optional character vector of trading pair symbols.
symbol_status	Optional trading status filter.
json_list	If TRUE, return the parsed list instead of converting array responses to <code>data.table</code> .
config	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list for single-symbol requests, or a `data.table` for multi-symbol/all-symbol requests unless `json_list = TRUE`.

spot_get_commission_rates
 Get Binance Spot commission rates

Description

Get Binance Spot commission rates

Usage

```
spot_get_commission_rates(symbol, config = config_spot())
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
config	A spot configuration created by config_spot() .

Value

A parsed list.

spot_get_exchange_info

Get Binance Spot exchange information

Description

Get Binance Spot exchange information

Usage

```
spot_get_exchange_info(  
    symbol = NULL,  
    symbols = NULL,  
    permissions = NULL,  
    show_permission_sets = TRUE,  
    symbol_status = NULL,  
    config = config_spot()  
)
```

Arguments

symbol	Optional trading pair symbol.
symbols	Optional character vector of trading pair symbols.
permissions	Optional character vector of permission strings.
show_permission_sets	Logical flag forwarded as showPermissionSets.
symbol_status	Optional trading status filter.
config	A spot configuration created by config_spot() .

Value

A parsed list.

`spot_get_execution_rules`*Get Binance Spot execution rules*

Description

Get Binance Spot execution rules

Usage

```
spot_get_execution_rules(  
    symbol = NULL,  
    symbols = NULL,  
    symbol_status = NULL,  
    config = config_spot()  
)
```

Arguments

<code>symbol</code>	Optional trading pair symbol.
<code>symbols</code>	Optional character vector of trading pair symbols.
<code>symbol_status</code>	Optional trading status filter.
<code>config</code>	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list.

`spot_get_historical_block_trades`*Get Binance Spot historical block trades*

Description

Get Binance Spot historical block trades

Usage

```
spot_get_historical_block_trades(  
    symbol,  
    limit = 500,  
    fromId = NULL,  
    json_list = FALSE,  
    config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
limit	Maximum number of rows to return.
fromId	Optional block trade identifier to fetch from.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by <code>config_spot()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

`spot_get_historical_trades`*Get Binance Spot historical trades*

Description

Get Binance Spot historical trades

Usage

```
spot_get_historical_trades(  
  symbol,  
  limit = 500,  
  fromId = NULL,  
  json_list = FALSE,  
  config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
limit	Maximum number of rows to return.
fromId	Optional trade identifier to fetch from.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by <code>config_spot()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

spot_get_klines	<i>Get Binance Spot klines</i>
-----------------	--------------------------------

Description

Get Binance Spot klines

Usage

```
spot_get_klines(  
  symbol,  
  interval,  
  startTime = NULL,  
  endTime = NULL,  
  timeZone = NULL,  
  limit = 500,  
  json_list = FALSE,  
  config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
interval	Kline interval string.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
timeZone	Optional timezone string accepted by Binance.
limit	Maximum number of rows to return.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by <code>config_spot()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

spot_get_open_orders *Get Binance Spot open orders*

Description

Get Binance Spot open orders

Usage

```
spot_get_open_orders(symbol = NULL, json_list = FALSE, config = config_spot())
```

Arguments

symbol	Optional trading pair symbol. If NULL, returns all open orders.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by config_spot() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

spot_get_open_order_lists
 Get Binance Spot open order lists

Description

Get Binance Spot open order lists

Usage

```
spot_get_open_order_lists(json_list = FALSE, config = config_spot())
```

Arguments

json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by config_spot() .

Value

A data.table by default, or a parsed list when json_list = TRUE.

spot_get_order	<i>Get a Binance Spot order</i>
----------------	---------------------------------

Description

Get a Binance Spot order

Usage

```
spot_get_order(  
    symbol,  
    order_id = NULL,  
    orig_client_order_id = NULL,  
    config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
order_id	Optional exchange order ID.
orig_client_order_id	Optional client order ID.
config	A spot configuration created by config_spot() .

Value

A parsed list.

spot_get_orders	<i>Get Binance Spot order history</i>
-----------------	---------------------------------------

Description

Get Binance Spot order history

Usage

```
spot_get_orders(  
    symbol,  
    order_id = NULL,  
    start_time = NULL,  
    end_time = NULL,  
    limit = 500,  
    json_list = FALSE,  
    config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
order_id	Optional exchange order ID to start from.
start_time	Optional start time in milliseconds.
end_time	Optional end time in milliseconds.
limit	Maximum number of orders to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by <code>config_spot()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

spot_get_order_amendments

Get Binance Spot order amendments

Description

Get Binance Spot order amendments

Usage

```
spot_get_order_amendments(
  symbol,
  order_id,
  from_execution_id = NULL,
  limit = 500,
  json_list = FALSE,
  config = config_spot()
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
order_id	Exchange order ID.
from_execution_id	Optional execution ID to start from.
limit	Maximum number of records to return. Must not exceed 1000.
json_list	If TRUE, return the parsed list instead of a data.table.
config	A spot configuration created by <code>config_spot()</code> .

Value

A data.table by default, or a parsed list when json_list = TRUE.

spot_get_order_book	<i>Get Binance Spot order book</i>
---------------------	------------------------------------

Description

Get Binance Spot order book

Usage

```
spot_get_order_book(  
    symbol,  
    limit = NULL,  
    symbol_status = NULL,  
    config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
limit	Optional order book depth limit.
symbol_status	Optional trading status filter.
config	A spot configuration created by config_spot() .

Value

A parsed list.

spot_get_order_list	<i>Get a Binance Spot order list</i>
---------------------	--------------------------------------

Description

Get a Binance Spot order list

Usage

```
spot_get_order_list(  
    order_list_id = NULL,  
    orig_client_order_id = NULL,  
    config = config_spot()  
)
```

Arguments

order_list_id Optional exchange order-list ID.
 orig_client_order_id Optional order-list client ID.
 config A spot configuration created by `config_spot()`.

Value

A parsed list.

spot_get_prevented_matches

Get Binance Spot prevented matches

Description

Get Binance Spot prevented matches

Usage

```
spot_get_prevented_matches(
  symbol,
  prevented_match_id = NULL,
  order_id = NULL,
  from_prevented_match_id = NULL,
  limit = 500,
  json_list = FALSE,
  config = config_spot()
)
```

Arguments

symbol Trading pair symbol, for example "BTCUSDT".
 prevented_match_id Optional prevented-match ID.
 order_id Optional exchange order ID.
 from_prevented_match_id Optional prevented-match ID to start from.
 limit Maximum number of records to return. Must not exceed 1000.
 json_list If TRUE, return the parsed list instead of a data.table.
 config A spot configuration created by `config_spot()`.

Value

A data.table by default, or a parsed list when `json_list = TRUE`.

`spot_get_recent_trades`*Get Binance Spot recent trades*

Description

Get Binance Spot recent trades

Usage

```
spot_get_recent_trades(  
    symbol,  
    limit = 500,  
    json_list = FALSE,  
    config = config_spot()  
)
```

Arguments

<code>symbol</code>	Trading pair symbol, for example "BTCUSDT".
<code>limit</code>	Maximum number of rows to return.
<code>json_list</code>	If TRUE, return the parsed list instead of a <code>data.table</code> .
<code>config</code>	A spot configuration created by <code>config_spot()</code> .

Value

A `data.table` by default, or a parsed list when `json_list = TRUE`.

`spot_get_reference_price`*Get Binance Spot reference price*

Description

Get Binance Spot reference price

Usage

```
spot_get_reference_price(symbol, config = config_spot())
```

Arguments

<code>symbol</code>	Trading pair symbol, for example "BTCUSDT".
<code>config</code>	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list.

spot_get_reference_price_calculation

Get Binance Spot reference price calculation metadata

Description

Get Binance Spot reference price calculation metadata

Usage

```
spot_get_reference_price_calculation(  
    symbol,  
    symbol_status = NULL,  
    config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
symbol_status	Optional trading status filter.
config	A spot configuration created by config_spot() .

Value

A parsed list.

spot_get_relevant_filters

Get Binance Spot relevant account filters

Description

Get Binance Spot relevant account filters

Usage

```
spot_get_relevant_filters(symbol, config = config_spot())
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
config	A spot configuration created by config_spot() .

Value

A parsed list.

spot_get_rolling_window_ticker
<i>Get Binance Spot rolling window ticker statistics</i>

Description

Get Binance Spot rolling window ticker statistics

Usage

```
spot_get_rolling_window_ticker(  
  symbol = NULL,  
  symbols = NULL,  
  windowSize = NULL,  
  type = c("FULL", "MINI"),  
  symbol_status = NULL,  
  json_list = FALSE,  
  config = config_spot()  
)
```

Arguments

symbol	Optional trading pair symbol.
symbols	Optional character vector of trading pair symbols.
windowSize	Optional rolling window size such as "1m", "4h", or "7d".
type	One of "FULL" or "MINI".
symbol_status	Optional trading status filter.
json_list	If TRUE, return the parsed list instead of converting array responses to data.table.
config	A spot configuration created by config_spot() .

Value

A parsed list for single-symbol requests, or a data.table for multi-symbol requests unless json_list = TRUE.

spot_get_server_time *Get Binance Spot server time*

Description

Get Binance Spot server time

Usage

```
spot_get_server_time(config = config_spot())
```

Arguments

config A spot configuration created by `config_spot()`.

Value

A POSIXct timestamp.

spot_get_ticker_price *Get Binance Spot ticker price*

Description

Get Binance Spot ticker price

Usage

```
spot_get_ticker_price(
  symbol = NULL,
  symbols = NULL,
  symbol_status = NULL,
  json_list = FALSE,
  config = config_spot()
)

get_spot_mark_price(symbol = NULL, config = binxr_config_spot())
```

Arguments

symbol Optional trading pair symbol.

symbols Optional character vector of trading pair symbols.

symbol_status Optional trading status filter.

json_list If TRUE, return the parsed list instead of converting array responses to `data.table`.

config A spot configuration created by `config_spot()`.

Value

A parsed list for single-symbol requests, or a `data.table` for multi-symbol/all-symbol requests unless `json_list = TRUE`.

`spot_get_trading_day_ticker`

Get Binance Spot trading day ticker statistics

Description

Get Binance Spot trading day ticker statistics

Usage

```
spot_get_trading_day_ticker(
  symbol = NULL,
  symbols = NULL,
  timeZone = NULL,
  type = c("FULL", "MINI"),
  symbol_status = NULL,
  json_list = FALSE,
  config = config_spot()
)
```

Arguments

<code>symbol</code>	Optional trading pair symbol.
<code>symbols</code>	Optional character vector of trading pair symbols.
<code>timeZone</code>	Optional timezone string accepted by Binance.
<code>type</code>	One of "FULL" or "MINI".
<code>symbol_status</code>	Optional trading status filter.
<code>json_list</code>	If TRUE, return the parsed list instead of converting array responses to <code>data.table</code> .
<code>config</code>	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list for single-symbol requests, or a `data.table` for multi-symbol requests unless `json_list = TRUE`.

spot_get_ui_klines	<i>Get Binance Spot UI klines</i>
--------------------	-----------------------------------

Description

Get Binance Spot UI klines

Usage

```
spot_get_ui_klines(  
  symbol,  
  interval,  
  startTime = NULL,  
  endTime = NULL,  
  timeZone = NULL,  
  limit = 500,  
  json_list = FALSE,  
  config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
interval	Kline interval string.
startTime	Optional start time in milliseconds since Unix epoch.
endTime	Optional end time in milliseconds since Unix epoch.
timeZone	Optional timezone string accepted by Binance.
limit	Maximum number of rows to return.
json_list	If TRUE, return the parsed list instead of a <code>data.table</code> .
config	A spot configuration created by <code>config_spot()</code> .

Value

A `data.table` by default, or a parsed list when `json_list = TRUE`.

spot_get_unfilled_order_count	<i>Get Binance Spot unfilled order counts</i>
-------------------------------	---

Description

Get Binance Spot unfilled order counts

Usage

```
spot_get_unfilled_order_count(json_list = FALSE, config = config_spot())
```

Arguments

json_list	If TRUE, return the parsed list instead of a <code>data.table</code> .
config	A spot configuration created by <code>config_spot()</code> .

Value

A `data.table` by default, or a parsed list when `json_list = TRUE`.

spot_ping	<i>Test Binance Spot API connectivity</i>
-----------	---

Description

Test Binance Spot API connectivity

Usage

```
spot_ping(config = config_spot())
```

Arguments

config	A spot configuration created by <code>config_spot()</code> .
--------	--

Value

An empty parsed list on success.

spot_place_oco_order	<i>Place a deprecated Binance Spot OCO order</i>
----------------------	--

Description

Place a deprecated Binance Spot OCO order

Usage

```
spot_place_oco_order(  
    symbol,  
    side = c("BUY", "SELL"),  
    quantity,  
    price,  
    stop_price,  
    list_client_order_id = NULL,  
    limit_client_order_id = NULL,  
    limit_strategy_id = NULL,  
    limit_strategy_type = NULL,  
    limit_iceberg_qty = NULL,  
    trailing_delta = NULL,  
    stop_client_order_id = NULL,  
    stop_strategy_id = NULL,  
    stop_strategy_type = NULL,  
    stop_limit_price = NULL,  
    stop_iceberg_qty = NULL,  
    stop_limit_time_in_force = NULL,  
    new_order_resp_type = NULL,  
    self_trade_prevention_mode = NULL,  
    config = config_spot()  
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
side	One of "BUY" or "SELL".
quantity	Order quantity shared by both legs.
price	Limit price for the maker leg.
stop_price	Stop trigger price.
list_client_order_id	Optional order-list client ID.
limit_client_order_id	Optional limit-leg client ID.
limit_strategy_id	Optional limit-leg strategy ID.

limit_strategy_type
Optional limit-leg strategy type.

limit_iceberg_qty
Optional limit-leg iceberg quantity.

trailing_delta
Optional shared trailing delta.

stop_client_order_id
Optional stop-leg client ID.

stop_strategy_id
Optional stop-leg strategy ID.

stop_strategy_type
Optional stop-leg strategy type.

stop_limit_price
Optional stop-limit price.

stop_iceberg_qty
Optional stop-leg iceberg quantity.

stop_limit_time_in_force
Optional stop-limit time-in-force.

new_order_resp_type
Optional response type. One of "ACK", "RESULT", or "FULL".

self_trade_prevention_mode
Optional self-trade prevention mode.

config
A spot configuration created by `config_spot()`.

Value

A parsed list.

spot_place_order	<i>Place a Binance Spot order</i>
------------------	-----------------------------------

Description

Place a Binance Spot order

Usage

```
spot_place_order(  
    symbol,  
    side = c("BUY", "SELL"),  
    type = c("LIMIT", "MARKET", "STOP_LOSS", "STOP_LOSS_LIMIT", "TAKE_PROFIT",  
            "TAKE_PROFIT_LIMIT", "LIMIT_MAKER"),  
    time_in_force = NULL,  
    quantity = NULL,  
    quote_order_qty = NULL,  
    price = NULL,
```

```

    new_client_order_id = NULL,
    strategy_id = NULL,
    strategy_type = NULL,
    stop_price = NULL,
    trailing_delta = NULL,
    iceberg_qty = NULL,
    new_order_resp_type = NULL,
    self_trade_prevention_mode = NULL,
    peg_price_type = NULL,
    peg_offset_value = NULL,
    peg_offset_type = NULL,
    config = config_spot()
)

```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
side	One of "BUY" or "SELL".
type	Order type.
time_in_force	Optional time-in-force value.
quantity	Optional order quantity.
quote_order_qty	Optional quote-order quantity for MARKET orders.
price	Optional limit price.
new_client_order_id	Optional client order identifier.
strategy_id	Optional numeric strategy identifier.
strategy_type	Optional numeric strategy type. Must be at least 1000000.
stop_price	Optional trigger price.
trailing_delta	Optional trailing delta value.
iceberg_qty	Optional iceberg quantity.
new_order_resp_type	Optional response type. One of "ACK", "RESULT", or "FULL".
self_trade_prevention_mode	Optional self-trade prevention mode.
peg_price_type	Optional pegged price type. One of "PRIMARY_PEG" or "MARKET_PEG".
peg_offset_value	Optional peg offset value.
peg_offset_type	Optional peg offset type. Only "PRICE_LEVEL" is supported.
config	A spot configuration created by config_spot() .

Value

A parsed list.

spot_place_order_list_oco

Place a Binance Spot OCO order list

Description

Place a Binance Spot OCO order list

Usage

```
spot_place_order_list_oco(
    symbol,
    side = c("BUY", "SELL"),
    quantity,
    above_type = c("STOP_LOSS_LIMIT", "STOP_LOSS", "LIMIT_MAKER", "TAKE_PROFIT",
        "TAKE_PROFIT_LIMIT"),
    above_client_order_id = NULL,
    above_iceberg_qty = NULL,
    above_price = NULL,
    above_stop_price = NULL,
    above_trailing_delta = NULL,
    above_time_in_force = NULL,
    above_strategy_id = NULL,
    above_strategy_type = NULL,
    above_peg_price_type = NULL,
    above_peg_offset_type = NULL,
    above_peg_offset_value = NULL,
    below_type = c("STOP_LOSS", "STOP_LOSS_LIMIT", "TAKE_PROFIT", "TAKE_PROFIT_LIMIT"),
    below_client_order_id = NULL,
    below_iceberg_qty = NULL,
    below_price = NULL,
    below_stop_price = NULL,
    below_trailing_delta = NULL,
    below_time_in_force = NULL,
    below_strategy_id = NULL,
    below_strategy_type = NULL,
    below_peg_price_type = NULL,
    below_peg_offset_type = NULL,
    below_peg_offset_value = NULL,
    list_client_order_id = NULL,
    new_order_resp_type = NULL,
    self_trade_prevention_mode = NULL,
    config = config_spot()
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
side	One of "BUY" or "SELL".
quantity	Order quantity shared by both legs.
above_type	Type for the above leg.
above_client_order_id	Optional above-leg client ID.
above_iceberg_qty	Optional above-leg iceberg quantity.
above_price	Optional above-leg limit price.
above_stop_price	Optional above-leg stop price.
above_trailing_delta	Optional above-leg trailing delta.
above_time_in_force	Optional above-leg time-in-force.
above_strategy_id	Optional above-leg strategy ID.
above_strategy_type	Optional above-leg strategy type.
above_peg_price_type	Optional above-leg peg price type.
above_peg_offset_type	Optional above-leg peg offset type.
above_peg_offset_value	Optional above-leg peg offset value.
below_type	Type for the below leg.
below_client_order_id	Optional below-leg client ID.
below_iceberg_qty	Optional below-leg iceberg quantity.
below_price	Optional below-leg limit price.
below_stop_price	Optional below-leg stop price.
below_trailing_delta	Optional below-leg trailing delta.
below_time_in_force	Optional below-leg time-in-force.
below_strategy_id	Optional below-leg strategy ID.
below_strategy_type	Optional below-leg strategy type.

below_peg_price_type
 Optional below-leg peg price type.
 below_peg_offset_type
 Optional below-leg peg offset type.
 below_peg_offset_value
 Optional below-leg peg offset value.
 list_client_order_id
 Optional order-list client ID.
 new_order_resp_type
 Optional response type. One of "ACK", "RESULT", or "FULL".
 self_trade_prevention_mode
 Optional self-trade prevention mode.
 config
 A spot configuration created by `config_spot()`.

Value

A parsed list.

spot_place_order_list_opo

Place a Binance Spot OPO order list

Description

Place a Binance Spot OPO order list

Usage

```

spot_place_order_list_opo(
    symbol,
    list_client_order_id = NULL,
    new_order_resp_type = NULL,
    self_trade_prevention_mode = NULL,
    working_type = c("LIMIT", "LIMIT_MAKER"),
    working_side = c("BUY", "SELL"),
    working_client_order_id = NULL,
    working_price,
    working_quantity,
    working_iceberg_qty = NULL,
    working_time_in_force = NULL,
    working_strategy_id = NULL,
    working_strategy_type = NULL,
    working_peg_price_type = NULL,
    working_peg_offset_type = NULL,
    working_peg_offset_value = NULL,
    pending_type = c("LIMIT", "MARKET", "STOP_LOSS", "STOP_LOSS_LIMIT", "TAKE_PROFIT",

```

```

    "TAKE_PROFIT_LIMIT", "LIMIT_MAKER"),
    pending_side = c("BUY", "SELL"),
    pending_client_order_id = NULL,
    pending_price = NULL,
    pending_stop_price = NULL,
    pending_trailing_delta = NULL,
    pending_iceberg_qty = NULL,
    pending_time_in_force = NULL,
    pending_strategy_id = NULL,
    pending_strategy_type = NULL,
    pending_peg_price_type = NULL,
    pending_peg_offset_type = NULL,
    pending_peg_offset_value = NULL,
    config = config_spot()
)

```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
list_client_order_id	Optional order-list client ID.
new_order_resp_type	Optional response type. One of "ACK", "RESULT", or "FULL".
self_trade_prevention_mode	Optional self-trade prevention mode.
working_type	Working leg type. One of "LIMIT" or "LIMIT_MAKER".
working_side	Working leg side. One of "BUY" or "SELL".
working_client_order_id	Optional working-leg client ID.
working_price	Working leg price.
working_quantity	Working leg quantity.
working_iceberg_qty	Optional working-leg iceberg quantity.
working_time_in_force	Optional working-leg time-in-force.
working_strategy_id	Optional working-leg strategy ID.
working_strategy_type	Optional working-leg strategy type.
working_peg_price_type	Optional working-leg peg price type.
working_peg_offset_type	Optional working-leg peg offset type.
working_peg_offset_value	Optional working-leg peg offset value.

pending_type Pending leg order type.
 pending_side Pending leg side. One of "BUY" or "SELL".
 pending_client_order_id
 Optional pending-leg client ID.
 pending_price Optional pending-leg price.
 pending_stop_price
 Optional pending-leg stop price.
 pending_trailing_delta
 Optional pending-leg trailing delta.
 pending_iceberg_qty
 Optional pending-leg iceberg quantity.
 pending_time_in_force
 Optional pending-leg time-in-force.
 pending_strategy_id
 Optional pending-leg strategy ID.
 pending_strategy_type
 Optional pending-leg strategy type.
 pending_peg_price_type
 Optional pending-leg peg price type.
 pending_peg_offset_type
 Optional pending-leg peg offset type.
 pending_peg_offset_value
 Optional pending-leg peg offset value.
 config A spot configuration created by `config_spot()`.

Value

A parsed list.

spot_place_order_list_opoco

Place a Binance Spot OPOCO order list

Description

Place a Binance Spot OPOCO order list

Usage

```
spot_place_order_list_opoco(
    symbol,
    list_client_order_id = NULL,
    new_order_resp_type = NULL,
    self_trade_prevention_mode = NULL,
```

```

working_type = c("LIMIT", "LIMIT_MAKER"),
working_side = c("BUY", "SELL"),
working_client_order_id = NULL,
working_price,
working_quantity,
working_iceberg_qty = NULL,
working_time_in_force = NULL,
working_strategy_id = NULL,
working_strategy_type = NULL,
working_peg_price_type = NULL,
working_peg_offset_type = NULL,
working_peg_offset_value = NULL,
pending_side = c("BUY", "SELL"),
pending_above_type = c("STOP_LOSS_LIMIT", "STOP_LOSS", "LIMIT_MAKER", "TAKE_PROFIT",
    "TAKE_PROFIT_LIMIT"),
pending_above_client_order_id = NULL,
pending_above_price = NULL,
pending_above_stop_price = NULL,
pending_above_trailing_delta = NULL,
pending_above_iceberg_qty = NULL,
pending_above_time_in_force = NULL,
pending_above_strategy_id = NULL,
pending_above_strategy_type = NULL,
pending_above_peg_price_type = NULL,
pending_above_peg_offset_type = NULL,
pending_above_peg_offset_value = NULL,
pending_below_type = c("STOP_LOSS", "STOP_LOSS_LIMIT", "TAKE_PROFIT",
    "TAKE_PROFIT_LIMIT"),
pending_below_client_order_id = NULL,
pending_below_price = NULL,
pending_below_stop_price = NULL,
pending_below_trailing_delta = NULL,
pending_below_iceberg_qty = NULL,
pending_below_time_in_force = NULL,
pending_below_strategy_id = NULL,
pending_below_strategy_type = NULL,
pending_below_peg_price_type = NULL,
pending_below_peg_offset_type = NULL,
pending_below_peg_offset_value = NULL,
config = config_spot()
)

```

Arguments

symbol Trading pair symbol, for example "BTCUSDT".
list_client_order_id Optional order-list client ID.
new_order_resp_type Optional response type. One of "ACK", "RESULT", or "FULL".

self_trade_prevention_mode
Optional self-trade prevention mode.

working_type Working leg type. One of "LIMIT" or "LIMIT_MAKER".

working_side Working leg side. One of "BUY" or "SELL".

working_client_order_id
Optional working-leg client ID.

working_price Working leg price.

working_quantity
Working leg quantity.

working_iceberg_qty
Optional working-leg iceberg quantity.

working_time_in_force
Optional working-leg time-in-force.

working_strategy_id
Optional working-leg strategy ID.

working_strategy_type
Optional working-leg strategy type.

working_peg_price_type
Optional working-leg peg price type.

working_peg_offset_type
Optional working-leg peg offset type.

working_peg_offset_value
Optional working-leg peg offset value.

pending_side Shared side for the pending OCO legs.

pending_above_type
Type for the pending-above leg.

pending_above_client_order_id
Optional pending-above client ID.

pending_above_price
Optional pending-above price.

pending_above_stop_price
Optional pending-above stop price.

pending_above_trailing_delta
Optional pending-above trailing delta.

pending_above_iceberg_qty
Optional pending-above iceberg quantity.

pending_above_time_in_force
Optional pending-above time-in-force.

pending_above_strategy_id
Optional pending-above strategy ID.

pending_above_strategy_type
Optional pending-above strategy type.

pending_above_peg_price_type
Optional pending-above peg price type.

pending_above_peg_offset_type	Optional pending-above peg offset type.
pending_above_peg_offset_value	Optional pending-above peg offset value.
pending_below_type	Type for the pending-below leg.
pending_below_client_order_id	Optional pending-below client ID.
pending_below_price	Optional pending-below price.
pending_below_stop_price	Optional pending-below stop price.
pending_below_trailing_delta	Optional pending-below trailing delta.
pending_below_iceberg_qty	Optional pending-below iceberg quantity.
pending_below_time_in_force	Optional pending-below time-in-force.
pending_below_strategy_id	Optional pending-below strategy ID.
pending_below_strategy_type	Optional pending-below strategy type.
pending_below_peg_price_type	Optional pending-below peg price type.
pending_below_peg_offset_type	Optional pending-below peg offset type.
pending_below_peg_offset_value	Optional pending-below peg offset value.
config	A spot configuration created by <code>config_spot()</code> .

Value

A parsed list.

spot_place_order_list_oto

Place a Binance Spot OTO order list

Description

Place a Binance Spot OTO order list

Usage

```

spot_place_order_list_oto(
    symbol,
    list_client_order_id = NULL,
    new_order_resp_type = NULL,
    self_trade_prevention_mode = NULL,
    working_type = c("LIMIT", "LIMIT_MAKER"),
    working_side = c("BUY", "SELL"),
    working_client_order_id = NULL,
    working_price,
    working_quantity,
    working_iceberg_qty = NULL,
    working_time_in_force = NULL,
    working_strategy_id = NULL,
    working_strategy_type = NULL,
    working_peg_price_type = NULL,
    working_peg_offset_type = NULL,
    working_peg_offset_value = NULL,
    pending_type = c("LIMIT", "MARKET", "STOP_LOSS", "STOP_LOSS_LIMIT", "TAKE_PROFIT",
        "TAKE_PROFIT_LIMIT", "LIMIT_MAKER"),
    pending_side = c("BUY", "SELL"),
    pending_client_order_id = NULL,
    pending_price = NULL,
    pending_stop_price = NULL,
    pending_trailing_delta = NULL,
    pending_quantity,
    pending_iceberg_qty = NULL,
    pending_time_in_force = NULL,
    pending_strategy_id = NULL,
    pending_strategy_type = NULL,
    pending_peg_price_type = NULL,
    pending_peg_offset_type = NULL,
    pending_peg_offset_value = NULL,
    config = config_spot()
)

```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
list_client_order_id	Optional order-list client ID.
new_order_resp_type	Optional response type. One of "ACK", "RESULT", or "FULL".
self_trade_prevention_mode	Optional self-trade prevention mode.
working_type	Working leg type. One of "LIMIT" or "LIMIT_MAKER".
working_side	Working leg side. One of "BUY" or "SELL".

working_client_order_id
Optional working-leg client ID.

working_price Working leg price.

working_quantity
Working leg quantity.

working_iceberg_qty
Optional working-leg iceberg quantity.

working_time_in_force
Optional working-leg time-in-force.

working_strategy_id
Optional working-leg strategy ID.

working_strategy_type
Optional working-leg strategy type.

working_peg_price_type
Optional working-leg peg price type.

working_peg_offset_type
Optional working-leg peg offset type.

working_peg_offset_value
Optional working-leg peg offset value.

pending_type Pending leg order type.

pending_side Pending leg side. One of "BUY" or "SELL".

pending_client_order_id
Optional pending-leg client ID.

pending_price Optional pending-leg price.

pending_stop_price
Optional pending-leg stop price.

pending_trailing_delta
Optional pending-leg trailing delta.

pending_quantity
Pending leg quantity.

pending_iceberg_qty
Optional pending-leg iceberg quantity.

pending_time_in_force
Optional pending-leg time-in-force.

pending_strategy_id
Optional pending-leg strategy ID.

pending_strategy_type
Optional pending-leg strategy type.

pending_peg_price_type
Optional pending-leg peg price type.

pending_peg_offset_type
Optional pending-leg peg offset type.

pending_peg_offset_value
Optional pending-leg peg offset value.

config A spot configuration created by `config_spot()`.

Value

A parsed list.

spot_place_order_list_otoco

Place a Binance Spot OTOCO order list

Description

Place a Binance Spot OTOCO order list

Usage

```
spot_place_order_list_otoco(
    symbol,
    list_client_order_id = NULL,
    new_order_resp_type = NULL,
    self_trade_prevention_mode = NULL,
    working_type = c("LIMIT", "LIMIT_MAKER"),
    working_side = c("BUY", "SELL"),
    working_client_order_id = NULL,
    working_price,
    working_quantity,
    working_iceberg_qty = NULL,
    working_time_in_force = NULL,
    working_strategy_id = NULL,
    working_strategy_type = NULL,
    working_peg_price_type = NULL,
    working_peg_offset_type = NULL,
    working_peg_offset_value = NULL,
    pending_side = c("BUY", "SELL"),
    pending_quantity,
    pending_above_type = c("STOP_LOSS_LIMIT", "STOP_LOSS", "LIMIT_MAKER", "TAKE_PROFIT",
        "TAKE_PROFIT_LIMIT"),
    pending_above_client_order_id = NULL,
    pending_above_price = NULL,
    pending_above_stop_price = NULL,
    pending_above_trailing_delta = NULL,
    pending_above_iceberg_qty = NULL,
    pending_above_time_in_force = NULL,
    pending_above_strategy_id = NULL,
    pending_above_strategy_type = NULL,
    pending_above_peg_price_type = NULL,
    pending_above_peg_offset_type = NULL,
    pending_above_peg_offset_value = NULL,
    pending_below_type = c("STOP_LOSS", "STOP_LOSS_LIMIT", "TAKE_PROFIT",
```

```

    "TAKE_PROFIT_LIMIT"),
    pending_below_client_order_id = NULL,
    pending_below_price = NULL,
    pending_below_stop_price = NULL,
    pending_below_trailing_delta = NULL,
    pending_below_iceberg_qty = NULL,
    pending_below_time_in_force = NULL,
    pending_below_strategy_id = NULL,
    pending_below_strategy_type = NULL,
    pending_below_peg_price_type = NULL,
    pending_below_peg_offset_type = NULL,
    pending_below_peg_offset_value = NULL,
    config = config_spot()
)

```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
list_client_order_id	Optional order-list client ID.
new_order_resp_type	Optional response type. One of "ACK", "RESULT", or "FULL".
self_trade_prevention_mode	Optional self-trade prevention mode.
working_type	Working leg type. One of "LIMIT" or "LIMIT_MAKER".
working_side	Working leg side. One of "BUY" or "SELL".
working_client_order_id	Optional working-leg client ID.
working_price	Working leg price.
working_quantity	Working leg quantity.
working_iceberg_qty	Optional working-leg iceberg quantity.
working_time_in_force	Optional working-leg time-in-force.
working_strategy_id	Optional working-leg strategy ID.
working_strategy_type	Optional working-leg strategy type.
working_peg_price_type	Optional working-leg peg price type.
working_peg_offset_type	Optional working-leg peg offset type.
working_peg_offset_value	Optional working-leg peg offset value.

pending_side Shared side for the pending OCO legs.
pending_quantity
 Pending leg quantity.
pending_above_type
 Type for the pending-above leg.
pending_above_client_order_id
 Optional pending-above client ID.
pending_above_price
 Optional pending-above price.
pending_above_stop_price
 Optional pending-above stop price.
pending_above_trailing_delta
 Optional pending-above trailing delta.
pending_above_iceberg_qty
 Optional pending-above iceberg quantity.
pending_above_time_in_force
 Optional pending-above time-in-force.
pending_above_strategy_id
 Optional pending-above strategy ID.
pending_above_strategy_type
 Optional pending-above strategy type.
pending_above_peg_price_type
 Optional pending-above peg price type.
pending_above_peg_offset_type
 Optional pending-above peg offset type.
pending_above_peg_offset_value
 Optional pending-above peg offset value.
pending_below_type
 Type for the pending-below leg.
pending_below_client_order_id
 Optional pending-below client ID.
pending_below_price
 Optional pending-below price.
pending_below_stop_price
 Optional pending-below stop price.
pending_below_trailing_delta
 Optional pending-below trailing delta.
pending_below_iceberg_qty
 Optional pending-below iceberg quantity.
pending_below_time_in_force
 Optional pending-below time-in-force.
pending_below_strategy_id
 Optional pending-below strategy ID.

pending_below_strategy_type
 Optional pending-below strategy type.
 pending_below_peg_price_type
 Optional pending-below peg price type.
 pending_below_peg_offset_type
 Optional pending-below peg offset type.
 pending_below_peg_offset_value
 Optional pending-below peg offset value.
 config
 A spot configuration created by `config_spot()`.

Value

A parsed list.

spot_place_sor_order *Place a Binance Spot SOR order*

Description

Place a Binance Spot SOR order

Usage

```

spot_place_sor_order(
  symbol,
  side = c("BUY", "SELL"),
  type = c("LIMIT", "MARKET"),
  quantity,
  time_in_force = NULL,
  price = NULL,
  new_client_order_id = NULL,
  strategy_id = NULL,
  strategy_type = NULL,
  iceberg_qty = NULL,
  new_order_resp_type = NULL,
  self_trade_prevention_mode = NULL,
  config = config_spot()
)

```

Arguments

symbol Trading pair symbol, for example "BTCUSDT".
 side One of "BUY" or "SELL".
 type SOR order type. One of "LIMIT" or "MARKET".
 quantity Order quantity.

- time_in_force Optional time-in-force value.
- price Optional limit price.
- new_client_order_id Optional client order identifier.
- strategy_id Optional numeric strategy identifier.
- strategy_type Optional numeric strategy type. Must be at least 1000000.
- iceberg_qty Optional iceberg quantity.
- new_order_resp_type Optional response type. One of "ACK", "RESULT", or "FULL".
- self_trade_prevention_mode Optional self-trade prevention mode.
- config A spot configuration created by `config_spot()`.

Value

A parsed list.

spot_test_order	<i>Test a Binance Spot order</i>
-----------------	----------------------------------

Description

Test a Binance Spot order

Usage

```
spot_test_order(  
  symbol,  
  side = c("BUY", "SELL"),  
  type = c("LIMIT", "MARKET", "STOP_LOSS", "STOP_LOSS_LIMIT", "TAKE_PROFIT",  
    "TAKE_PROFIT_LIMIT", "LIMIT_MAKER"),  
  time_in_force = NULL,  
  quantity = NULL,  
  quote_order_qty = NULL,  
  price = NULL,  
  new_client_order_id = NULL,  
  strategy_id = NULL,  
  strategy_type = NULL,  
  stop_price = NULL,  
  trailing_delta = NULL,  
  iceberg_qty = NULL,  
  new_order_resp_type = NULL,  
  self_trade_prevention_mode = NULL,  
  peg_price_type = NULL,  
  peg_offset_value = NULL,
```

```

    peg_offset_type = NULL,
    compute_commission_rates = FALSE,
    config = config_spot()
)

```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
side	One of "BUY" or "SELL".
type	Order type.
time_in_force	Optional time-in-force value.
quantity	Optional order quantity.
quote_order_qty	Optional quote-order quantity for MARKET orders.
price	Optional limit price.
new_client_order_id	Optional client order identifier.
strategy_id	Optional numeric strategy identifier.
strategy_type	Optional numeric strategy type. Must be at least 1000000.
stop_price	Optional trigger price.
trailing_delta	Optional trailing delta value.
iceberg_qty	Optional iceberg quantity.
new_order_resp_type	Optional response type. One of "ACK", "RESULT", or "FULL".
self_trade_prevention_mode	Optional self-trade prevention mode.
peg_price_type	Optional pegged price type. One of "PRIMARY_PEG" or "MARKET_PEG".
peg_offset_value	Optional peg offset value.
peg_offset_type	Optional peg offset type. Only "PRICE_LEVEL" is supported.
compute_commission_rates	Whether to return commission-rate details.
config	A spot configuration created by config_spot() .

Value

A parsed list.

spot_test_sor_order	<i>Test a Binance Spot SOR order</i>
---------------------	--------------------------------------

Description

Test a Binance Spot SOR order

Usage

```
spot_test_sor_order(
    symbol,
    side = c("BUY", "SELL"),
    type = c("LIMIT", "MARKET"),
    quantity,
    time_in_force = NULL,
    price = NULL,
    new_client_order_id = NULL,
    strategy_id = NULL,
    strategy_type = NULL,
    iceberg_qty = NULL,
    new_order_resp_type = NULL,
    self_trade_prevention_mode = NULL,
    compute_commission_rates = FALSE,
    config = config_spot()
)
```

Arguments

symbol	Trading pair symbol, for example "BTCUSDT".
side	One of "BUY" or "SELL".
type	SOR order type. One of "LIMIT" or "MARKET".
quantity	Order quantity.
time_in_force	Optional time-in-force value.
price	Optional limit price.
new_client_order_id	Optional client order identifier.
strategy_id	Optional numeric strategy identifier.
strategy_type	Optional numeric strategy type. Must be at least 1000000.
iceberg_qty	Optional iceberg quantity.
new_order_resp_type	Optional response type. One of "ACK", "RESULT", or "FULL".
self_trade_prevention_mode	Optional self-trade prevention mode.
compute_commission_rates	Whether to return commission-rate details.
config	A spot configuration created by <code>config_spot()</code> .

spot_test_sor_order

111

Value

A parsed list.

Index

binxr_config_futures, 5
binxr_config_options, 5
binxr_config_spot, 6

cancel_fapi_trade_order
 (futures_cancel_order), 11
cancel_fapi_trade_orders_all
 (futures_cancel_all_orders), 11
coerce_account_frames, 7
config_futures, 7
config_futures(), 10–41, 43–46
config_options, 8
config_options(), 47–62
config_spot, 9
config_spot(), 64–66, 68–90, 92, 93, 96, 98,
 101, 103, 107–110

futures_cancel_algo_order, 10
futures_cancel_all_algo_orders, 10
futures_cancel_all_orders, 11
futures_cancel_order, 11
futures_countdown_cancel_all, 12
futures_get_24hr_ticker, 13
futures_get_account, 13
futures_get_account(), 14, 33
futures_get_account_summary, 14
futures_get_account_trades, 14
futures_get_aggregate_trades, 15
futures_get_algo_order, 16
futures_get_algo_orders, 17
futures_get_balance, 17
futures_get_book_ticker, 18
futures_get_commission_rate, 18
futures_get_continuous_klines, 19
futures_get_exchange_info, 20
futures_get_exchange_info(), 63
futures_get_force_orders, 20
futures_get_funding_info, 21
futures_get_funding_rate_history, 22
futures_get_historical_trades, 22
futures_get_index_constituents, 23
futures_get_index_price_klines, 24
futures_get_insurance_balance, 24
futures_get_klines, 25
futures_get_mark_price, 26
futures_get_mark_price_klines, 27
futures_get_multi_assets_mode, 27
futures_get_open_algo_orders, 28
futures_get_open_interest, 29
futures_get_open_orders, 29
futures_get_order, 30
futures_get_order_amendments, 31
futures_get_order_book, 32
futures_get_order_rate_limit, 33
futures_get_orders, 31
futures_get_position_mode, 34
futures_get_position_risk, 34
futures_get_positions, 33
futures_get_premium_index_klines, 35
futures_get_recent_trades, 35
futures_get_rpi_depth, 36
futures_get_server_time, 37
futures_get_symbol_adl_risk, 37
futures_get_ticker_price, 38
futures_get_trading_schedule, 38
futures_modify_order, 39
futures_ping, 40
futures_place_algo_order, 40
futures_place_order, 41
futures_set_leverage, 43
futures_set_margin_type, 44
futures_set_multi_assets_mode, 44
futures_set_position_mode, 45
futures_test_order, 46

get_fapi_account (futures_get_account),
 13
get_fapi_account_position_risk
 (futures_get_position_risk), 34

get_fapi_account_positions
 (futures_get_positions), 33
get_fapi_account_summary
 (futures_get_account_summary),
 14
get_fapi_exchange_info
 (futures_get_exchange_info), 20
get_fapi_klines (futures_get_klines), 25
get_fapi_mark_price
 (futures_get_mark_price), 26
get_fapi_system_time
 (futures_get_server_time), 37
get_fapi_trade_open_orders
 (futures_get_open_orders), 29
get_fapi_trade_order
 (futures_get_order), 30
get_fapi_trade_orders
 (futures_get_orders), 31
get_spot_mark_price
 (spot_get_ticker_price), 87

options_cancel_all_orders, 47
options_cancel_all_orders_by_underlying,
 47
options_cancel_order, 48
options_get_24hr_ticker, 48
options_get_account_trades, 49
options_get_commission, 50
options_get_exchange_info, 50
options_get_exercise_history, 51
options_get_exercise_records, 51
options_get_funding_flow, 52
options_get_index_price, 53
options_get_klines, 54
options_get_margin_account, 54
options_get_mark_price, 55
options_get_open_interest, 55
options_get_open_orders, 56
options_get_order, 57
options_get_order_book, 57
options_get_order_history, 58
options_get_positions, 59
options_get_recent_block_trades, 59
options_get_recent_trades, 60
options_get_server_time, 61
options_ping, 61
options_place_order, 62
place_fapi_trade_order
 (futures_place_order), 41
round_price_qty, 63
set_fapi_account_leverage
 (futures_set_leverage), 43
set_fapi_account_margin_type
 (futures_set_margin_type), 44
set_fapi_account_position_side
 (futures_set_position_mode), 45
spot_amend_order_keep_priority, 64
spot_cancel_all_orders, 65
spot_cancel_order, 65
spot_cancel_order_list, 66
spot_cancel_replace_order, 67
spot_get_24hr_ticker, 69
spot_get_account, 69
spot_get_account_trades, 70
spot_get_aggregate_trades, 71
spot_get_all_order_lists, 72
spot_get_allocations, 71
spot_get_average_price, 73
spot_get_book_ticker, 74
spot_get_commission_rates, 74
spot_get_exchange_info, 75
spot_get_execution_rules, 76
spot_get_historical_block_trades, 76
spot_get_historical_trades, 77
spot_get_klines, 78
spot_get_open_order_lists, 79
spot_get_open_orders, 79
spot_get_order, 80
spot_get_order_amendments, 81
spot_get_order_book, 82
spot_get_order_list, 82
spot_get_orders, 80
spot_get_prevented_matches, 83
spot_get_recent_trades, 84
spot_get_reference_price, 84
spot_get_reference_price_calculation,
 85
spot_get_relevant_filters, 85
spot_get_rolling_window_ticker, 86
spot_get_server_time, 87
spot_get_ticker_price, 87
spot_get_trading_day_ticker, 88
spot_get_ui_klines, 89
spot_get_unfilled_order_count, 90
spot_ping, 90

spot_place_oco_order, [91](#)
spot_place_order, [92](#)
spot_place_order_list_oco, [94](#)
spot_place_order_list_opo, [96](#)
spot_place_order_list_opoco, [98](#)
spot_place_order_list_oto, [101](#)
spot_place_order_list_otoco, [104](#)
spot_place_sor_order, [107](#)
spot_test_order, [108](#)
spot_test_sor_order, [110](#)

util_round_price_qty (round_price_qty),
 [63](#)