Package: rCoinbase (via r-universe)

June 5, 2026

Title 'Coinbase Advance Trade API Interface'

Version 1.0.0

Description The 'Coinbase Advanced Trade API'
<https://docs.cdp.coinbase.com/api-reference/advanced-trade-api/rest-api/introduction>
lets you manage orders, portfolios, products, and fees with the
new v3 endpoints.

License GPL-3

Language en-US

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports httr, jose, openssl, uuid, purrr, tidyr, httr2, lubridate,
data.table, tibble, dplyr

VignetteBuilder knitr

Suggests testthat, knitr, rmarkdown

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev libssl-dev

Repository https://jgquantscripts.r-universe.dev

Date/Publication 2025-08-09 02:54:28 UTC

RemoteUrl https://github.com/jgquantscripts/rcoinbase

RemoteRef HEAD

RemoteSha fa8ebc168ad4ddd84a56fc38fbec5a399455e2f7

Contents

`assign_tokens` *Assign working tokens*

Description

Assign working tokens

Usage

```
assign_tokens()
```

Value

creates working environment that reads/writes binary tokens and assigns variable in working environment

Examples

```
## Not run:
  assign_tokens()

## End(Not run)
```

`build_jwt` *Gets Bearer Token*

Description

Gets Bearer Token

Usage

```
build_jwt(key_var, secret_var, method, endpoint)
```

Arguments

key_var	= your personal API key
secret_var	= your personal secret token
method	= GET, POST, etc.
endpoint	= endpoint to use

Value

returns JWT token to make API requests

`cb_bars` *Get OHLCV Bars (short-term)*

Description

Get OHLCV Bars (short-term)

Usage

```
cb_bars(product_id, start_time, end_time, bar_size)
```

Arguments

product_id	= The trading pair (e.g. 'BTC-USD').
start_time	= The UNIX timestamp indicating the start of the time interval.
end_time	= The UNIX timestamp indicating the end of the time interval.
bar_size	= The timeframe each candle represents. Examples: ONE_MINUTE, FIVE_MINUTE, FIFTEEN_MINUTE, THIRTY_MINUTE, ONE_HOUR, TWO_HOUR, SIX_HOUR, ONE_DAY

Value

Get a data.frame with rates for a single product by product ID, grouped in buckets.

returns OHLCV for cryptocurrencies

Examples

```
## Not run:
  cb_bars(product_id = "ETH-USD", start_time = Sys.time()-hours(1),
          end_time = Sys.time(), bar_size = 'FIVE_MINUTE')

## End(Not run)
```

---

cb_cancel_futures_sweep

Futures: Cancel Sweep

Description

Futures: Cancel Sweep

Usage

```
cb_cancel_futures_sweep()
```

Value

A data.frame detailing the pending sweep of funds from FCM wallet to USD Spot wallet

Examples

```
## Not run:
  cb_cancel_futures_sweep()

## End(Not run)
```

cb_cancel_order *Spot: Cancel Order*

Description

Spot: Cancel Order

Usage

```
cb_cancel_order(order_ids)
```

Arguments

order_ids	= (string) enter the order id that you wish to cancel

Value

returns order details as a data.frame for cancelled orders

Examples

```
## Not run:
  cb_cancel_order(order_ids='ASDGF123-SDVSA123-SAEF123')

## End(Not run)
```

cb_candles *Get OHLCV Bars (long-term)*

Description

Get OHLCV Bars (long-term)

Usage

```
cb_candles(product_id, start, end, bar_size)
```

Arguments

product_id	= The trading pair (e.g. 'BTC-USD').
start	= start date to get data. Ex. Sys.Date()-60
end	= End date to get data. Ex. Sys.Date()
bar_size	= The timeframe each candle represents. Examples: ONE_MINUTE, FIVE_MINUTE, FIFTEEN_MINUTE, THIRTY_MINUTE, ONE_HOUR, TWO_HOUR, SIX_HOUR, ONE_DAY

Value

Get a `data.frame` with rates for a single product by product ID, grouped in buckets for more than 350 bars.

Examples

```
## Not run:
  cb_candles(product_id="BTC-USD", start=Sys.Date()-30, end=Sys.Date(),
             bar_size= "FIFTEEN_MINUTE")

## End(Not run)
```

`cb_close_order` *Futures: Cancel Order*

Description

Futures: Cancel Order

Usage

```
cb_close_order(client_order_id = cb_get_order_id(), product_id, size = NULL)
```

Arguments

`client_order_id`	= defaults to random id via *cb_get_order_id()*
`product_id`	= futures contract to close
`size`	= number of contracts to close, defaults to closing all available

Value

Cancel response `data.frame` status for a futures order

Examples

```
## Not run:
  cb_close_order(product_id = "BIT-28JUL23-CDE")

## End(Not run)
```

cb_commit_convert_trade
Commit Convert Trade

Description

Commit Convert Trade

Usage

```
cb_commit_convert_trade(trade_id, from_account, to_account)
```

Arguments

trade_id	= The ID of the trade to commit.
from_account	= The currency of the account to convert from (e.g. USD).
to_account	= The currency of the account to convert to (e.g. USDC).

Value

Commits a convert trade with a specified trade id, source account, and target account and returns a data.frame response

Examples

```
## Not run:
  qte = cb_create_convert_quote(amount = 100,
                                from_account = "USD",
                                to_account = "USDC")
  ord = cb_commit_convert_trade(trade_id = qte$id,
                                from_account = "USD",
                                to_account = "USDC")
 stat = cb_get_convert_trade(trade_id = qte$id,
                             from_account = "USD",
                             to_account = "USDC")

## End(Not run)
```

cb_create_convert_quote
Create Convert Quote

Description

Create Convert Quote

Usage

```
cb_create_convert_quote(amount, from_account, to_account)
```

Arguments

amount	= The ID of the trade to commit.
from_account	= The currency of the account to convert from (e.g. USD).
to_account	= The currency of the account to convert to (e.g. USDC).

Value

A data.frame with details regarding creating a convert quote with a specified source account,target account, and amount. Convert is applicable for USDC-USD, EURC-EUR, and PYUSD-USD conversion

Examples

```
## Not run:
  qte = cb_create_convert_quote(amount = 100,
                                from_account = "USD",
                                to_account = "USDC")
  ord = cb_commit_convert_trade(trade_id = qte$id,
                                from_account = "USD",
                                to_account = "USDC")
 stat = cb_get_convert_trade(trade_id = qte$id,
                             from_account = "USD",
                             to_account = "USDC")

## End(Not run)
```

---

cb_get_convert_trade *Get Convert Trade*

Description

Get Convert Trade

Usage

```
cb_get_convert_trade(trade_id, from_account, to_account)
```

Arguments

trade_id	= The ID of the trade to commit.
from_account	= The currency of the account to convert from (e.g. USD).
to_account	= The currency of the account to convert to (e.g. USDC).

Value

A data.frame with account information about a convert trade with a specified trade id, source account, and target account

Examples

```
## Not run:
  qte = cb_create_convert_quote(amount = 100,
                                from_account = "USD",
                                to_account = "USDC")
  ord = cb_commit_convert_trade(trade_id = qte$id,
                                from_account = "USD",
                                to_account = "USDC")
 stat = cb_get_convert_trade(trade_id = qte$id,
                             from_account = "USD",
                             to_account = "USDC")

## End(Not run)
```

---

cb_get_current_margin_window

Futures: Get Margin Window

Description

Futures: Get Margin Window

Usage

```
cb_get_current_margin_window(
  margin_profile_type = "MARGIN_PROFILE_TYPE_RETAIL_REGULAR"
)
```

Arguments

margin_profile_type	= The margin profile type for your account: *MARGIN_PROFILE_TYPE_UNSPECIFIED*

Value

Get the futures current margin window as a data.frame

Examples

```
## Not run:
  cb_get_current_margin_window()

## End(Not run)
```

cb_get_fees *Account Fees*

Description

Account Fees

Usage

```
cb_get_fees(
  product_type = "UNKNOWN_PRODUCT_TYPE",
  contract_expiry_type = "UNKNOWN_CONTRACT_EXPIRY_TYPE",
  product_venue = "UNKNOWN_VENUE_TYPE"
)
```

Arguments

product_type	= Only returns the orders matching this product type. By default, returns all product types. 'UNKNOWN_PRODUCT_TYPE', 'SPOT','FUTURE'
contract_expiry_type	= Only returns the orders matching this contract expiry type. Only applicable if product_type is set to FUTURE. 'UNKNOWN_CONTRACT_EXPIRY_TYPE','EXPIRING', 'PERPETUAL'
product_venue	= Venue for product 'UNKNOWN_VENUE_TYPE','CBE','FCM','INTX'

Value

A data.frame with a summary of transactions with fee tiers, total volume, and fees.

Examples

```
## Not run:
  cb_get_fees()

## End(Not run)
```

cb_get_futures_balance *Futures: Get Balance*

Description

Futures: Get Balance

Usage

```
cb_get_futures_balance()
```

Value

Get Futures Balance Summary as a `data.frame`

Examples

```
## Not run:
  cb_get_futures_balance()

## End(Not run)
```

`cb_get_futures_position` *Futures: Get Position*

Description

Futures: Get Position

Usage

```
cb_get_futures_position(product_id)
```

Arguments

`product_id`	= The ticker symbol (e.g. 'BIT-28JUL25-CDE')

Value

Get Futures Position as a `data.frame`

Get positions for a specific CFM product

Examples

```
## Not run:
  cb_get_futures_position(product_id = 'BIT-28JUL25-CDE')

## End(Not run)
```

cb_get_intraday_margin *Futures: Get Intraday Margin*

Description

Futures: Get Intraday Margin

Usage

```
cb_get_intraday_margin()
```

Value

Get intraday margin Setting as a `data.frame`

Examples

```
## Not run:
  cb_get_intraday_margin()

## End(Not run)
```

cb_get_order_id *Order ID*

Description

Order ID

Usage

```
cb_get_order_id()
```

Value

An auto generated `character` string to use for placing orders

Examples

```
## Not run:
  cb_get_order_id()

## End(Not run)
```

cb_getAccount *Get Accounts*

Description

Get Accounts

Usage

```
cb_getAccount(acct_uuid)
```

Arguments

acct_uuid	= The account's UUID.

Value

Get a data.frame of information about an account, given an account UUID.

Examples

```
## Not run:
  cb_getAccount(acct_uuid = 'f412dr89-01d0-576d-g457-ea0b52a13716')

## End(Not run)
```

cb_getAccounts *List Accounts*

Description

List Accounts

Usage

```
cb_getAccounts(lmt)
```

Arguments

lmt	= The number of accounts to display per page. By default, displays 49 (max 250)

Value

Get a data.frame of authenticated Advanced Trade accounts for the current user.

Examples

```
## Not run:
  cb_getAccounts(lmt = 100)

## End(Not run)
```

`cb_getCryptoList` *Get Crypto List*

Description

Get Crypto List

Usage

```
cb_getCryptoList()
```

Value

Get a `data.frame` with all crypto currency pairs

Examples

```
## Not run:
  cb_getCryptoList()

## End(Not run)
```

`cb_getOrder` *Get Order*

Description

Get Order

Usage

```
cb_getOrder(id)
```

Arguments

`id`	= The ID of the order

Value

Get a detailed `data.frame` for the order requested.

Examples

```
## Not run:
  cb_getOrder(id='1234')

## End(Not run)
```

`cb_list_futures_positions`
Futures: List All Positions

Description

Futures: List All Positions

Usage

```
cb_list_futures_positions()
```

Value

Get a list of positions in CFM products as a `data.frame`

Examples

```
## Not run:
  cb_list_futures_positions()

## End(Not run)
```

`cb_list_futures_sweeps`
Futures: List Sweeps

Description

Futures: List Sweeps

Usage

```
cb_list_futures_sweeps()
```

Value

Gets `data.frame` for pending and processing sweeps of funds from FCM wallet to USD Spot wallet

Examples

```
## Not run:
  cb_list_futures_sweeps()

## End(Not run)
```

`cb_lmt_fok_order` *Spot: Place Limit FOK Order*

Description

Spot: Place Limit FOK Order

Usage

```
cb_lmt_fok_order(
  client_order_id = cb_get_order_id(),
  product_id,
  side,
  base_size,
  limit_price
)
```

Arguments

`client_order_id`	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
`product_id`	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
`side`	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
`base_size`	= (string) The amount of the first Asset in the Trading Pair. Example: 0.001
`limit_price`	= (string) The specified price, or better, that the Order should be executed at. A Buy Order will execute at or lower than the limit price. A Sell Order will execute at or higher than the limit price. Example: 10000.00

Value

returns order details as a `data.frame` for limit FOK orders

Examples

```
## Not run:
  cb_lmt_fok_order(product_id = "BTC-USD", side = "BUY",
                   base_size = '0.00004', limit_price = '100000')

## End(Not run)
```

cb_lmt_gtc_order *Spot: Place Limit GTC Order*

Description

Spot: Place Limit GTC Order

Usage

```
cb_lmt_gtc_order(
  client_order_id = cb_get_order_id(),
  product_id,
  side,
  base_size,
  limit_price
)
```

Arguments

client_order_id	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
product_id	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
side	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
base_size	= (string) The amount of the first Asset in the Trading Pair. Example: 0.001
limit_price	= (string) The specified price, or better, that the Order should be executed at. A Buy Order will execute at or lower than the limit price. A Sell Order will execute at or higher than the limit price. Example: 10000.00

Value

returns order details as a `data.frame` for limit GTC orders

Examples

```
## Not run:
  cb_lmt_gtc_order(product_id = "BTC-USD", side = "BUY",
                   base_size = '0.00004', limit_price = '100000')

## End(Not run)
```

cb_lmt_gtd_order *Spot: Place Limit GTD Order*

Description

Spot: Place Limit GTD Order

Usage

```
cb_lmt_gtd_order(
  client_order_id = cb_get_order_id(),
  product_id,
  side,
  base_size,
  limit_price,
  order_exp
)
```

Arguments

client_order_id	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
product_id	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
side	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
base_size	= (string) The amount of the first Asset in the Trading Pair. Example: 0.001
limit_price	= (string) The specified price, or better, that the Order should be executed at. A Buy Order will execute at or lower than the limit price. A Sell Order will execute at or higher than the limit price. Example: 10000.00
order_exp	= (TimeStamp) Enter the time you wish to cancel if not filled: Ex. Sys.time()+minutes(5)

Value

returns order details as a `data.frame` for limit GTD orders

Examples

```
## Not run:
  cb_lmt_fok_order(product_id = "BTC-USD", side = "BUY",
                   base_size = '0.00004', limit_price = '100000',
                   order_exp = Sys.time()+minutes(33))

## End(Not run)
```

`cb_lmt_twap_gtd_order` *Spot: Place Limit TWAP Order*

Description

Spot: Place Limit TWAP Order

Usage

```
cb_lmt_twap_gtd_order(
  client_order_id = cb_get_order_id(),
  product_id,
  side,
  base_size,
  order_start,
  order_exp,
  limit_price,
  number_buckets,
  bucket_duration
)
```

Arguments

`client_order_id`	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
`product_id`	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
`side`	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
`base_size`	= (string) The amount of the first Asset in the Trading Pair. Example: 0.001
`order_start`	= (TimeStamp) Enter the time you wish to cancel if not filled: Ex. Sys.time()+minutes(5)
`order_exp`	= (TimeStamp) Enter the time you wish to cancel if not filled: Ex. Sys.time()+minutes(10)
`limit_price`	= (string) The specified price, or better, that the Order should be executed at. A Buy Order will execute at or lower than the limit price. A Sell Order will execute at or higher than the limit price. Example: 10000.00
`number_buckets`	= (string) The number of smaller buckets/suborders over which the entire order will be broken into. Each suborder will be executed over a duration calculated based on the end_time. Example: 5
`bucket_duration`	= (string) The duration over which each sub order was executed. Example: 300s

Value

returns order details as a `data.frame` for limit TWAP orders

Examples

```
## Not run:
  cb_lmt_twap_gtd_order(product_id = "BTC-USD",
                        side = "BUY",
                        base_size = '0.00004',
                        order_start = Sys.time() + minutes(1),
                        order_exp = Sys.time()+minutes(6),
                        limit_price = '100000',
                        number_buckets = 2,
                        bucket_duration = "300")

## End(Not run)
```

`cb_mkt_order` *Spot: Place Market Order*

Description

Spot: Place Market Order

Usage

```
cb_mkt_order(client_order_id = cb_get_order_id(), product_id, side, base_size)
```

Arguments

`client_order_id`	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
`product_id`	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
`side`	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
`base_size`	= (string) The amount of the first Asset in the Trading Pair. Example: 0.001

Value

returns order details as a `data.frame` for market orders

Examples

```
## Not run:
  cb_mkt_order(product_id = "BTC-USD", side = "BUY", base_size = '0.00001')

## End(Not run)
```

cb_order_builder *Order Builder*

Description

Order Builder

Usage

```
cb_order_builder(
  order_type,
  client_order_id,
  product_id,
  side,
  leverage = 1,
  margin_type = "CROSS",
  preview_id = NULL,
  base_size = NULL,
  quote_size = NULL,
  start_time = NULL,
  end_time = NULL,
  limit_price = NULL,
  number_buckets = NULL,
  bucket_duration = NULL,
  bucket_size = NULL,
  post_only = FALSE,
  stop_price = NULL,
  stop_direction = NULL,
  stop_trigger_price = NULL
)
```

Arguments

order_type	= (string) type of order : "market_market_ioc"
client_order_id	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
product_id	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
side	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
leverage	= (string) The amount of leverage for the order (default is 1.0). Example: 2.0
margin_type	= (string) Margin Type for this order (default is CROSS). Possible values: CROSS, ISOLATED
preview_id	= (string) Preview ID for this order, to associate this order with a preview request. Example: b40bbff9-17ce-4726-8b64-9de7ae57ad26

base_size = (string) The amount of the first Asset in the Trading Pair. Example: 0.001

quote_size = (string) The amount of the second Asset in the Trading Pair. Example: 10.00

start_time = (RFC3339 Timestamp) Time at which the order should begin executing. Example: 2021-05-31T07:59:59Z

end_time = (RFC3339 Timestamp) The time at which the order will be canceled if it is not Filled. Example: 2021-05-31T09:59:59Z

limit_price = (string) The specified price, or better, that the Order should be executed at. A Buy Order will execute at or lower than the limit price. A Sell Order will execute at or higher than the limit price. Example: 10000.00

number_buckets = (string) The number of smaller buckets/suborders over which the entire order will be broken into. Each suborder will be executed over a duration calculated based on the end_time. Example: 5

bucket_duration = (string) The duration over which each sub order was executed. Example: 300s

bucket_size = (string) The size of each suborder. bucket_size multiplied by number_buckets should match the size of the entire twap order)

post_only = (boolean) Enable or disable Post-only Mode. When enabled, only Maker Orders will be posted to the Order Book. Orders that will be posted as a Taker Order will be rejected.

stop_price = (string) The specified price that will trigger the placement of the Order. Example: 20000.00

stop_direction = (string) The direction of the stop limit Order. Possible values: STOP_DIRECTION_STOP_UP, STOP_DIRECTION_STOP_DOWN

stop_trigger_price = (string) The price level (in quote currency) where the position will be exited. When triggered, a stop limit order is automatically placed with a limit price 5% higher for BUYS and 5% lower for SELLS. Example: 20000.00

Value

returns a `list` for the order configuration depending on the order type

Examples

```
## Not run:
  cb_order_builder(order_type="market_market_ioc", client_order_id='1234',
                   product_id="BTC-USD", side="BUY", leverage = 1.0,
                   margin_type='CROSS',preview_id=NULL,
                   base_size="0.00001", quote_size=NULL,
                   start_time=NULL, end_time=NULL, limit_price=NULL,
                   number_buckets=NULL, bucket_duration=NULL,
                   bucket_size = NULL, post_only=FALSE,
                   stop_price=NULL, stop_direction=NULL,
                   stop_trigger_price=NULL)

## End(Not run)
```

cb_quote *Crypto Pair Bid/Ask Quotes*

Description

Crypto Pair Bid/Ask Quotes

Usage

```
cb_quote(ids)
```

Arguments

ids	= vector of product id(s) Example: "BTC-USD" OR c("BTC-USD","ETH-USD")

Value

Get a `data.frame` for the best bid/ask for all products.

Examples

```
## Not run:
  cb_quote(ids=c("BTC-USD","ETH-USD"))

## End(Not run)
```

cb_schedule_futures_sweeps *Futures: Schedule Sweeps*

Description

Futures: Schedule Sweeps

Usage

```
cb_schedule_futures_sweeps(usd_amount)
```

Arguments

usd_amount	= The amount of USD to be swept. By default, sweeps all available excess funds.

Value

Gets `data.frame` for scheduling a sweep of funds from FCM wallet to USD Spot wallet

Examples

```
## Not run:
  cb_schedule_futures_sweeps(usd_amount = 100.00)

## End(Not run)
```

`cb_set_intraday_margin` *Futures: Set Intraday Margin*

Description

Futures: Set Intraday Margin

Usage

```
cb_set_intraday_margin(setting)
```

Arguments

`setting`	= The amount of USD to be swept. By default, sweeps all available excess funds.

Value

Gets `data.frame` with details for setting intraday margin

Examples

```
## Not run:
  cb_set_intraday_margin(setting = 100.00)

## End(Not run)
```

`cb_sor_lmt_ioc_order` *Spot: Place Limit IOC Order*

Description

Spot: Place Limit IOC Order

Usage

```
cb_sor_lmt_ioc_order(
  client_order_id = cb_get_order_id(),
  product_id,
  side,
  base_size,
  limit_price
)
```

Arguments

client_order_id	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
product_id	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
side	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
base_size	= (string) The amount of the first Asset in the Trading Pair. Example: 0.001
limit_price	= (string) The specified price, or better, that the Order should be executed at. A Buy Order will execute at or lower than the limit price. A Sell Order will execute at or higher than the limit price. Example: 10000.00

Value

returns order details as a `data.frame` for limit IOC orders

Examples

```
## Not run:
  cb_sor_lmt_ioc_order(product_id = "BTC-USD", side = "BUY",
                       base_size = '0.00004', limit_price = '100000')

## End(Not run)
```

---

cb_stp_lmt_gtc_order *Spot: Place Stop-Limit GTC Order*

Description

Spot: Place Stop-Limit GTC Order

Usage

```
cb_stp_lmt_gtc_order(
  client_order_id = cb_get_order_id(),
  product_id,
  side,
  base_size,
  limit_price,
  stop_price,
  stop_direction
)
```

Arguments

client_order_id	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
product_id	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
side	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
base_size	= (string) The amount of the first Asset in the Trading Pair. Example: 0.001
limit_price	= (string) The specified price, or better, that the Order should be executed at. A Buy Order will execute at or lower than the limit price. A Sell Order will execute at or higher than the limit price. Example: 10000.00
stop_price	= (string) The specified price that will trigger the placement of the Order. Example: 20000.00
stop_direction	= (string) The direction of the stop limit Order. Possible values: STOP_DIRECTION_STOP_UP, STOP_DIRECTION_STOP_DOWN

Value

returns order details as a `data.frame` for stop-limit GTC orders

Examples

```
## Not run:
  cb_stp_lmt_gtc_order(product_id="BTC-USD",
                       side="BUY",
                       base_size="0.00001",
                       limit_price="100000",
                       stop_price="85000",
                       stop_direction='STOP_DIRECTION_STOP_DOWN')

## End(Not run)
```

cb_stp_lmt_gtd_order *Spot: Place Stop-Limit GTD Order*

Description

Spot: Place Stop-Limit GTD Order

Usage

```
cb_stp_lmt_gtd_order(
  client_order_id = cb_get_order_id(),
  product_id,
  side,
  base_size,
  limit_price,
  stop_price,
  order_exp,
  stop_direction
)
```

Arguments

client_order_id	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
product_id	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
side	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
base_size	= (string) The amount of the first Asset in the Trading Pair. Example: 0.001
limit_price	= (string) The specified price, or better, that the Order should be executed at. A Buy Order will execute at or lower than the limit price. A Sell Order will execute at or higher than the limit price. Example: 10000.00
stop_price	= (string) The specified price that will trigger the placement of the Order. Example: 20000.00
order_exp	= (TimeStamp) Enter the time you wish to cancel if not filled: Ex. Sys.time()+minutes(10)
stop_direction	= (string) The direction of the stop limit Order. Possible values: STOP_DIRECTION_STOP_UP, STOP_DIRECTION_STOP_DOWN

Value

returns order details as a `data.frame` for stop-limit GTD orders

Examples

```
## Not run:
  cb_stp_lmt_gtd_order(product_id="BTC-USD",
                       side="BUY",
                       base_size="0.00001",
                       limit_price="100000",
                       stop_price="85000",
                       order_exp=Sys.time()+minutes(15),
                       stop_direction='STOP_DIRECTION_STOP_DOWN')

## End(Not run)
```

---

cb_trig_gtc_order *Spot: Place Trigger Bracket GTC Order*

Description

Spot: Place Trigger Bracket GTC Order

Usage

```
cb_trig_gtc_order(
  client_order_id = cb_get_order_id(),
  product_id,
  side,
  base_size,
  limit_price,
  stop_trigger_price
)
```

Arguments

client_order_id	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
product_id	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
side	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
base_size	= (string) The amount of the first Asset in the Trading Pair. Example: 0.001
limit_price	= (string) The specified price, or better, that the Order should be executed at. A Buy Order will execute at or lower than the limit price. A Sell Order will execute at or higher than the limit price. Example: 10000.00
stop_trigger_price	= (string) The price level (in quote currency) where the position will be exited. When triggered, a stop limit order is automatically placed with a limit price 5% higher for BUYS and 5% lower for SELLS. Example: 20000.00

Value

returns order details as a `data.frame` for trigger GTC orders

Examples

```
## Not run:
  cb_trig_gtc_order(product_id="BTC-USD", side="BUY",
                    base_size="0.00001", limit_price="100000",
                    stop_trigger_price="115000")

## End(Not run)
```

---

cb_trig_gtd_order *Spot: Place Trigger Bracket GTD Order*

Description

Spot: Place Trigger Bracket GTD Order

Usage

```
cb_trig_gtd_order(
  client_order_id = cb_get_order_id(),
  product_id,
  side,
  base_size,
  limit_price,
  stop_trigger_price,
  order_exp
)
```

Arguments

client_order_id	= (string) A unique ID provided for the order (used for identification purposes) Example: 0000-00000-000000
product_id	= (string) The trading pair (e.g. 'BTC-USD'). Example: BTC-USD
side	= (string) The side of the market that the order is on (e.g. 'BUY', 'SELL'). Possible values: BUY, SELL
base_size	= (string) The amount of the first Asset in the Trading Pair. Example: 0.001
limit_price	= (string) The specified price, or better, that the Order should be executed at. A Buy Order will execute at or lower than the limit price. A Sell Order will execute at or higher than the limit price. Example: 10000.00
stop_trigger_price	= (string) The price level (in quote currency) where the position will be exited. When triggered, a stop limit order is automatically placed with a limit price 5% higher for BUYS and 5% lower for SELLS. Example: 20000.00
order_exp	= (TimeStamp) Enter the time you wish to cancel if not filled: Ex. Sys.time()+minutes(10)

Value

returns order details as a `data.frame` for trigger GTD orders

Examples

```
## Not run:
  cb_trig_gtd_order(product_id="BTC-USD",
                    side="BUY",
                    base_size="0.00001",
                    limit_price="100000",
                    stop_trigger_price="115000",
                    order_exp = Sys.time()+minutes(5))

## End(Not run)
```

Index