

Package ‘rPublic’

July 9, 2026

Title 'Public Trading API'

Version 1.2.0

Description The 'Public Trading API' <<https://public.com/api/docs>> allows clients to access their brokerage accounts, request market data, and place stock/etf/option orders.

License GPL-3

Language en-US

Encoding UTF-8

RoxygenNote 7.3.2

Imports http, jsonlite, data.table, lubridate, stringr, uuid

VignetteBuilder knitr

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

NeedsCompilation no

Author Jason Guevara [aut, cre]

Maintainer Jason Guevara <Jason.guevara.yt@gmail.com>

Repository CRAN

Date/Publication 2026-07-09 17:30:02 UTC

Contents

.rp_checkAccessToken	2
.rp_env	3
.rp_make_multileg_payload	3
.rp_make_opt_symbol	4
.rp_make_ord_payload	5
.rp_make_qte_payload	6
.rp_read_tokens	7
rp_bars	8
rp_cancel_order	9
rp_fix_bars	9
rp_getAccHist	10

rp_getAccToken	11
rp_getAccts	12
rp_getAcctsPort	12
rp_getAllInstruments	13
rp_getInstrument	14
rp_getOptChains	14
rp_getOptExp	15
rp_getOrderId	16
rp_getQuote	16
rp_get_greeks	17
rp_get_order	18
rp_histBars	18
rp_order_multi	20
rp_order_single	21
rp_preOrder_multiLeg	23
rp_preOrder_singleLeg	25
Index	27

.rp_checkAccessToken	<i>Check & Auto-Renew Bearer Tokens (Internal)</i>
----------------------	--

Description

Check & Auto-Renew Bearer Tokens (Internal)

Usage

.rp_checkAccessToken(printMsg = FALSE, mins = 120)

Arguments

- printMsg = (bool) Should outcome messages be printed out? defaults to FALSE
- mins = (int) The number of minutes that the bearer token will be valid for.

Value

Checks validity of Bearer Token & auto-updates if needed. Assigns the new tokens in 'rp' environment and 'rp_tokens.rds' file

Examples

```
## Not run:  
# For Internal Use Prior to Making API Requests  
.rp_checkAccessToken(printMsg=FALSE, mins=120)  
  
## End(Not run)
```

.rp_env	<i>temporary working environment</i>
---------	--------------------------------------

Description

temporary working environment

Usage

```
.rp_env
```

Format

An object of class environment of length 0.

Value

An auto generated environment to store our tokens

Examples

```
## Not run:  
.rp_env <- new.env(parent = emptyenv())  
  
## End(Not run)
```

.rp_make_multileg_payload	<i>Build Multi-Leg Order Payload (Internal)</i>
---------------------------	---

Description

Build Multi-Leg Order Payload (Internal)

Usage

```
.rp_make_multileg_payload(  
  orderType,  
  qty,  
  orderId = NULL,  
  leg_symbols,  
  leg_types,  
  leg_sides,  
  leg_indicator,  
  leg_ratios,  
  lmtPrc = NULL,
```

```

    tif,
    expTime = NULL
)

```

Arguments

orderType	= (string) The Type of order: 'MARKET', 'LIMIT', 'STOP', 'STOP_LIMIT'
qty	= (string) leg_ratio multiple: ex. '2' multiples the leg_ratios by 2X
orderId	= (string) The order ID
leg_symbols	= (string) Symbols: ex. c("SPY250815C00631000", "SPY250815C00631000")
leg_types	= (string) Symbol types: ex. c("OPTION", "OPTION")
leg_sides	= (string) The side for each leg: ex. c("BUY", "SELL")
leg_indicator	= (string) Indicates if this is BUY to OPEN/CLOSE ex. c("OPEN", "OPEN")
leg_ratios	= (string) The number of contracts to BUY/SELL: ex. c('5','5')
lmtPrc	= (string) The limit price. Used when orderType = LIMIT or orderType = STOP_LIMIT
tif	= (string) The time in for the order: 'DAY' or 'GTD'
expTime	= (string) The expiration date. Only used when timeInForce is GTD, cannot be more than 90 days in the future

Value

Returns an appropriate payload list for a multiple-leg order

Examples

```

## Not run:
# Return the proper order payload for multiple-leg orders
.rp_make_multileg_payload(orderType="LIMIT", qty="2", orderId=rp_getOrderId(),
    leg_symbols = c("SPY250815C00631000", "SPY250815C00631000"),
    leg_types = c("OPTION", "OPTION"), leg_sides = c("BUY", "SELL"),
    leg_indicator = c("OPEN", "OPEN"), leg_ratios=c('5','5'),
    lmtPrc='0.25', tif="DAY")

## End(Not run)

```

.rp_make_opt_symbol *Build Option Symbol (Internal)*

Description

Build Option Symbol (Internal)

Usage

```
.rp_make_opt_symbol(under_sym, exp, type, strike)
```

Arguments

<code>under_sym</code>	= (string) Underlying symbol for the option: ex. 'SPY'
<code>exp</code>	= (string) The option expiration: ex. "2025-08-15"
<code>type</code>	= (string) The option type: 'C' for Call & 'P' for Put
<code>strike</code>	= (double/int) The option strike price: 631 or 631.00

Value

Returns a valid symbol string for the option contract of interest

Examples

```
## Not run:  
# Return the proper option symbol of interest: "TSLA250808C00325000"  
.rp_make_opt_symbol(under_sym="TSLA", exp="2025-08-08", type="C", strike=325)  
  
## End(Not run)
```

`.rp_make_ord_payload` *Build Single-Leg Order Payload (Internal)*

Description

Build Single-Leg Order Payload (Internal)

Usage

```
.rp_make_ord_payload(  
  ticker,  
  symType,  
  orderId = NULL,  
  side = NULL,  
  ordType = NULL,  
  timeInForce = NULL,  
  expirationTime = NULL,  
  qty = NULL,  
  amt = NULL,  
  lmtPrc = NULL,  
  stopPrc = NULL,  
  openCloseIndicator = NULL  
)
```

Arguments

ticker	= (string) Ticker symbol: ex. 'SPY'
symType	= (string) Symbol type: ex. 'EQUITY'
orderId	= (string) The order ID
side	= (string) The Order Side BUY/SELL. For Options also include the openCloseIndicator
ordType	= (string) The Type of order: 'MARKET', 'LIMIT', 'STOP', 'STOP_LIMIT'
timeInForce	= (string) The time in for the order: 'DAY' or 'GTD'
expirationTime	= (string) The expiration date. Only used when timeInForce is GTD, cannot be more than 90 days in the future
qty	= (string) The order quantity. Used when buying/selling whole shares and when selling fractional. Mutually exclusive with amount
amt	= (string) The order amount. Used when buying/selling shares for a specific notional value
lmtPrc	= (string) The limit price. Used when orderType = LIMIT or orderType = STOP_LIMIT
stopPrc	= (string) The stop price. Used when orderType = STOP or orderType = STOP_LIMIT
openCloseIndicator	= (string) Used for options only. Indicates if this is BUY to OPEN/CLOSE

Value

Returns an appropriate payload list for a single-leg order

Examples

```
## Not run:
# Return the proper order payload for single-leg orders
.rp_make_ord_payload(under_sym="IWM", symType = "EQUITY", orderId = rp_getOrderId(),
  side="BUY", orderType="LIMIT", timeInForce="GTD",
  expirationTime = "2023-11-07T05:31:56Z", qty=1.735, lmtPrc="200.00")

## End(Not run)
```

.rp_make_qte_payload *Build Dynamic Payload For rp_getQuote (Internal)*

Description

Build Dynamic Payload For rp_getQuote (Internal)

Usage

```
.rp_make_qte_payload(symbols, types)
```

Arguments

`symbols` = (string) Equity/ETF/Option symbol(s)
`types` = (string) The product type (ex. 'EQUITY' or 'OPTION')

Value

Returns a list in the appropriate payload format in case the user needs multiple symbols for quotes

Examples

```
## Not run:  
# Create the correct quote payload for AAPL and a SPY 631 Call 8/15/25 Expiration  
.rp_make_qte_payload(symbols=c("AAPL", "SPY250815C00631000"), types=c("EQUITY", "OPTION"))  
  
## End(Not run)
```

<code>.rp_read_tokens</code>	<i>Request token file (Internal)</i>
------------------------------	--------------------------------------

Description

Request token file (Internal)

Usage

```
.rp_read_tokens()
```

Value

Requests your token file 'rp_tokens.rds' from working directory & assigns a working environment if it exists

Examples

```
## Not run:  
# For Internal Use (assigns tokens inside of the 'rp' environment)  
.rp_read_tokens()  
  
## End(Not run)
```

rp_bars

*Get Bars V2***Description**

Get Bars V2

Usage

```
rp_bars(
  symbol,
  period = "DAY",
  purchaseDate = NULL,
  type = "EQUITY",
  include_ah = FALSE
)
```

Arguments

symbol	= The symbol to fetch bar data for: Ex. "SPY"
period	= one of DAY, WEEK, MONTH, QUARTER, HALF_YEAR, YEAR, FIVE_YEARS, YTD, SINCE_PURCHASE
purchaseDate	= Required for SINCE_PURCHASE. YYYY-MM-DD
type	= Ticker Type: Ex. 'EQUITY', 'OPTION', 'CRYPTO', 'INDEX'
include_ah	= Should after-hours data (if available) be included? Default is FALSE

Value

Fetch bar data for a given symbol and period and return as a `data.frame`.

Examples

```
## Not run:
# Fetches Bars for the DAY - include after-hours data
df <- rp_bars(symbol = "PLTR", period = "DAY", include_ah=TRUE)
# Fetches Bars using SINCE_PURCHASE
df <- rp_bars(symbol = "DASH", period = "SINCE_PURCHASE",
purchaseDate = "2026-05-14")
# Fetches Bars for Options
df <- rp_bars(symbol = "SPY260515C00730000", period = "WEEK", type="OPTION")
# Get BTC Bars
df <- rp_bars(symbol = "BTC", period = "DAY", type="CRYPTO")

## End(Not run)
```

rp_cancel_order	<i>Cancel Order</i>
-----------------	---------------------

Description

Cancel Order

Usage

```
rp_cancel_order(accountId, orderId)
```

Arguments

accountId	= Public Brokerage Account Number
orderId	= The order ID

Value

Request order cancellation & return as a `data.frame`.

Examples

```
## Not run:  
# Cancels Specific Order  
my_acc <- rp_getAccts()  
rp_cancel_order(accountId = my_acc$accountId,  
                 orderId = "c99be1dd-bb87-4f7a-803f-ec47226bf64e")  
  
## End(Not run)
```

rp_fixBars	<i>Convert bars to data.frame</i>
------------	-----------------------------------

Description

Convert bars to data.frame

Usage

```
rp_fixBars(ls2fix)
```

Arguments

ls2fix	= List to convert
--------	-------------------

Value

Pass in list to convert it to a data.frame.

Examples

```
## Not run:
# Fetches Bars for the Week

# Fetch Bars
df <- rp_histBars("BTC", period = "WEEK", type = "CRYPTO",
  aggregation = "FIFTEEN_MINUTES")
# Extract the list you need, ex. regularMarket bars
ls2fix <- df$regularMarket$bars
# Pass it into function
rp_fix_bars(ls2fix)

## End(Not run)
```

rp_getAccHist

Get History

Description

Get History

Usage

```
rp_getAccHist(accountId, start = NULL, end = NULL, pageSize = NULL)
```

Arguments

accountId	= Public Brokerage Account Number
start	= (Optional) Start timestamp in ISO 8601 format with timezone. Ex. "YYYY-MM-DDTHH:MM:SSZ"
end	= (Optional) End timestamp in ISO 8601 format with timezone. Ex. "YYYY-MM-DDTHH:MM:SSZ"
pageSize	= (Optional) Maximum number of records to return.

Value

Fetches a paginated data.frame of historical events for the specified account.

Examples

```
## Not run:
# Return Public Brokerage Account History
my_acc <- rp_getAccts()

# using only accountId
my_hist <- rp_getAccHist(accountId = my_acc$accountId)

# using some parameters
my_hist <- rp_getAccHist(accountId = my_acc$accountId,
                        start = format(Sys.time()-days(30), format="%Y-%m-%dT%H:%M:%SZ"),
                        pageSize = 20
                        )

## End(Not run)
```

rp_getAccToken

Get New Access/Bearer Token From Secret Key

Description

Get New Access/Bearer Token From Secret Key

Usage

```
rp_getAccToken(exp_in_mins)
```

Arguments

exp_in_mins = (int) The number of minutes that the bearer token will be valid for.

Value

Update Bearer Token from secret key & returns working environment and saves updated tokens in 'rp_tokens.rds'

Examples

```
## Not run:
# Request New Bearer Token that expires in 120 minutes
rp_getAccToken(exp_in_mins=120)

## End(Not run)
```

`rp_getAccts`*Get Public Account Info*

Description

Get Public Account Info

Usage

```
rp_getAccts()
```

Value

Returns a `data.frame` for the user's Public Brokerage Account

Examples

```
## Not run:  
# Return Public Brokerage Account Information  
rp_getAccts()  
  
## End(Not run)
```

`rp_getAcctsPort`*Get Account Portfolio V2*

Description

Get Account Portfolio V2

Usage

```
rp_getAcctsPort(accountId)
```

Arguments

`accountId` = Public Brokerage Account Number

Value

Returns a list for the user's specific Public Brokerage Account

Examples

```
## Not run:
# Return Public Brokerage Account Information
my_acc <- rp_getAccts()
my_port <- rp_getAcctsPort(accountId = my_acc$accountId)

## End(Not run)
```

rp_getAllInstruments *Get All Instruments*

Description

Get All Instruments

Usage

```
rp_getAllInstruments(
  typeFilter = NULL,
  tradingFilter = NULL,
  fractionalTradingFilter = NULL,
  optionTradingFilter = NULL,
  optionSpreadTradingFilter = NULL
)
```

Arguments

```
typeFilter      = (Optional) Ex. "BOND","EQUITY","CRYPTO","INDEX","ALT"
tradingFilter   = (Optional) Ex. "BUY_AND_SELL","DISABLED","LIQUIDATION_ONLY"
fractionalTradingFilter
                = (Optional) Ex. "DISABLED","BUY_AND_SELL","LIQUIDATION_ONLY"
optionTradingFilter
                = (Optional) Ex. "DISABLED","BUY_AND_SELL","LIQUIDATION_ONLY"
optionSpreadTradingFilter
                = (Optional) Ex. "DISABLED","BUY_AND_SELL","LIQUIDATION_ONLY"
```

Value

Retrieves all available trading instruments with optional filtering capabilities as a data.frame.

Examples

```
## Not run:
# Fetches All Instruments From Public
all_inst <- rp_getAllInstruments()

# Fetches All equities enabled for trading fractional shares
```

```

all_frac <- rp_getAllInstruments(typeFilter = "EQUITY",
                                tradingFilter = 'BUY_AND_SELL',
                                fractionalTradingFilter = 'BUY_AND_SELL')

## End(Not run)

```

rp_getInstrument	<i>Get Specific Instrument Information</i>
------------------	--

Description

Get Specific Instrument Information

Usage

```
rp_getInstrument(symbol, type)
```

Arguments

symbol	= Trading Symbol Type: Ex. "AAPL"
type	= Symbol Type Ex. "EQUITY", "OPTION", "MULTI_LEG_INSTRUMENT", "CRYPTO", "ALT", "TREASURY", "BOND", "INDEX"

Value

Retrieves specific trading instrument with optional filtering capabilities as a data.frame.

Examples

```

## Not run:
# Fetches AAPL instrument trading information
this_ins <- rp_getInstrument(symbol = "AAPL", type="EQUITY")

## End(Not run)

```

rp_getOptChains	<i>Get Option Chains</i>
-----------------	--------------------------

Description

Get Option Chains

Usage

```
rp_getOptChains(accountId, ticker, type, exp)
```

Arguments

accountId = Public Brokerage Account Number
ticker = Ticker symbol: Ex. "SPY"
type = Ticker Type: Ex. 'EQUITY','OPTION','MULTI_LEG_INSTRUMENT','CRYPTO',
 'ALT','TREASURY','BOND','INDEX'
exp = Option Expiration Date: Ex. "2025-08-08"

Value

Retrieve option chains by symbol and return as a `data.frame`.

Examples

```
## Not run:  
# Fetches Option Chains for Ticker Symbol  
my_acc <- rp_getAccts()  
rp_getOptChains(accountId = my_acc$accountId, ticker = 'SPY', type = "EQUITY", exp="2025-08-15")  
  
## End(Not run)
```

`rp_getOptExp`*Get Option Expiration Dates*

Description

Get Option Expiration Dates

Usage

```
rp_getOptExp(accountId, ticker, type)
```

Arguments

accountId = Public Brokerage Account Number
ticker = Ticker symbol: Ex. "SPY"
type = Ticker Type: Ex. 'EQUITY','OPTION','MULTI_LEG_INSTRUMENT','CRYPTO',
 'ALT','TREASURY','BOND','INDEX'

Value

Retrieve option expiration dates for a specific ticker symbol as a `data.frame`.

Examples

```
## Not run:
# Fetches Option Expiry Dates Available
my_acc <- rp_getAccts()
rp_getOptExp(accountId = my_acc$accountId, ticker = "TSLA", type="EQUITY")

## End(Not run)
```

rp_getOrderId	<i>Order ID</i>
---------------	-----------------

Description

Order ID

Usage

```
rp_getOrderId()
```

Value

An auto generated character string to use for placing orders

Examples

```
## Not run:
rp_getOrderId()

## End(Not run)
```

rp_getQuote	<i>Get Trading Quotes</i>
-------------	---------------------------

Description

Get Trading Quotes

Usage

```
rp_getQuote(accountId, ticker, type)
```

Arguments

- accountId = Public Brokerage Account Number
- ticker = Ticker symbol: Ex. "SPY"
- type = Ticker Type: Ex. 'EQUITY','OPTION','MULTI_LEG_INSTRUMENT','CRYPTO','ALT','TREASURY','BOND','INDEX'

Value

Retrieve real-time quotes as a `data.frame`.

Examples

```
## Not run:
# Fetches Multiple Real-Time Quotes
my_acc <- rp_getAccts()
rp_getQuote(accountId = my_acc$accountId, ticker = "TSLA", type="EQUITY")
rp_getQuote(accountId = my_acc$accountId, ticker = 'SPY250807C00633000', type = "OPTION")
rp_getQuote(accountId = my_acc$accountId,
             ticker = c("AAPL", 'SPY250807C00633000'),
             type = c("EQUITY", "OPTION"))

## End(Not run)
```

rp_get_greeks

Get Option Greeks

Description

Get Option Greeks

Usage

```
rp_get_greeks(accountId, osiOptionSymbol)
```

Arguments

```
accountId      = Public Brokerage Account Number
osiOptionSymbol
               = option symbol
```

Value

Request order cancellation & return as a `data.frame`.

Examples

```
## Not run:
# get account number
my_acc <- rp_getAccts()

# build option symbol
this_op = .rp_make_opt_symbol(under_sym = "SPY", exp = "2025-08-22",
                              type = "P", strike = 600)

# get greeks
rp_get_greeks(accountId = my_acc$accountId, osiOptionSymbol = this_op)
```

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

rp_get_order	Get Order Details
--------------	-------------------

Description

Get Order Details

Usage

```
rp_get_order(accountId, orderId)
```

Arguments

accountId = Public Brokerage Account Number
orderId = The order ID

Value

Retrieve order details & return as a data.frame.

Examples

```
## Not run:  
# Fetches Specific Order  
my_acc <- rp_getAccts()  
rp_get_order(accountId = my_acc$accountId,  
              orderId = "c99be1dd-bb87-4f7a-803f-ec47226bf64e")  
  
## End(Not run)
```

rp_histBars	Get bars v2 with aggregation
-------------	------------------------------

Description

Get bars v2 with aggregation

Usage

```
rp_hist_bars(
  symbol,
  period = "DAY",
  type = "EQUITY",
  aggregation,
  purchaseDate = NULL,
  include_ah = FALSE
)
```

Arguments

symbol	= The symbol to fetch bar data for: Ex. "SPY"
period	= one of DAY, WEEK, MONTH, QUARTER, HALF_YEAR, YEAR, FIVE_YEARS, YTD, SINCE_PURCHASE
type	= Ticker Type: Ex. 'EQUITY', 'OPTION', 'CRYPTO', 'INDEX'
aggregation	= Aggregation Ex. ONE_MINUTE, FIVE_MINUTES, TEN_MINUTES, FIFTEEN_MINUTES, THIRTY_MINUTES, ONE_HOUR, ONE_DAY, ONE_WEEK, ONE_MONTH, THREE_MONTHS, SIX_MONTH, SONE_YEAR
purchaseDate	= Required for SINCE_PURCHASE. YYYY-MM-DD
include_ah	= Should after-hours data (if available) be included? Default is FALSE

Value

Fetch bar data for a given symbol and period and return as a `data.frame`.

Examples

```
## Not run:
# Fetches Bars for the DAY - include after-hours data
df <- rp_hist_bars("TSLA", period = "DAY", type = "EQUITY",
  aggregation = "FIVE_MINUTES", include_ah = TRUE)
# Fetches Bars using SINCE_PURCHASE
df <- rp_hist_bars("UBER", period = "MONTH", type = "EQUITY",
  aggregation = "ONE_DAY", purchaseDate = "2026-04-01")
# Fetches Bars for Options
df <- rp_hist_bars("SPY260522P00739000", period = "DAY",
  type = "OPTION", aggregation = "ONE_MINUTE")
# Fetches Bars for Crypto
df <- rp_hist_bars("BTC", period = "WEEK", type = "CRYPTO",
  aggregation = "FIFTEEN_MINUTES")

## End(Not run)
```

rp_order_multi	<i>Multi-Leg Live Order</i>
----------------	-----------------------------

Description

Multi-Leg Live Order

Usage

```
rp_order_multi(
  accountId,
  orderType,
  orderId,
  qty,
  leg_symbols,
  leg_types,
  leg_sides,
  leg_indicator,
  leg_ratios,
  tif,
  mins = NULL,
  lmtPrc = NULL
)
```

Arguments

accountId	= Public Brokerage Account Number
orderType	= The Type of order: Ex. 'MARKET', 'LIMIT', 'STOP', 'STOP_LIMIT'
orderId	= The order ID: use rp_getOrderId()
qty	= leg_ratio multiple: ex. '2' multiplies the leg_ratios by 2X
leg_symbols	= Symbols: ex. c("SPY250815C00631000", "SPY250815C00631000")
leg_types	= Symbol types: ex. c("OPTION", "OPTION")
leg_sides	= The side for each leg: ex. c("BUY", "SELL")
leg_indicator	= Indicates if this is BUY to OPEN/CLOSE ex. c("OPEN", "OPEN")
leg_ratios	= The number of contracts to BUY/SELL: ex. c('5','5')
tif	= The time in for the order: 'DAY' or 'GTD'
mins	= Minutes till order expires.
lmtPrc	= The limit price. Used when orderType = LIMIT or orderType = STOP_LIMIT

Value

Place a new multi-leg order and returns order id as a data.frame.

Examples

```
## Not run:
# Fetches costs associated with the type of order being placed
my_acc <- rp_getAccts()

# open bull-call spread for 0.25 (buy 2, sell 2)
rp_order_multi(accountId = my_acc$accountId, orderType = "LIMIT", qty = 2,
  leg_symbols = c("SPY250815C00630000", "SPY250815C00632000"),
  leg_types = c("OPTION", "OPTION"), leg_sides = c("BUY", "SELL"),
  leg_indicator = c("OPEN", "OPEN"), leg_ratios = c(1, 1),
  tif = "DAY", lmtPrc = 0.25, orderId = rp_getOrderId())

# open long butterfly for 0.05
rp_order_multi(accountId = my_acc$accountId, orderType = "LIMIT", qty = 1,
  leg_symbols = c("SPY250815C00630000",
    "SPY250815C00631000",
    "SPY250815C00632000"),
  leg_types = c("OPTION", "OPTION", "OPTION"),
  leg_sides = c("BUY", "SELL", "BUY"),
  leg_indicator = c("OPEN", "OPEN", "OPEN"),
  leg_ratios = c(1, 2, 1), tif = "DAY", lmtPrc = 0.05,
  orderId = rp_getOrderId())

# open iron-condor
rp_order_multi(accountId = my_acc$accountId, orderType = "LIMIT", qty = 1,
  leg_symbols = c("SPY250815C00631000", "SPY250815C00630000",
    "SPY250815C00625000", "SPY250815C00624000"),
  leg_types = c("OPTION", "OPTION", "OPTION", "OPTION"),
  leg_sides = c("SELL", "BUY", "SELL", "BUY"),
  leg_indicator = c("OPEN", "OPEN", "OPEN", "OPEN"),
  leg_ratios = c(1, 1, 1, 1), tif = "DAY", lmtPrc = 0.30,
  orderId = rp_getOrderId())

# covered call
rp_order_multi(accountId = my_acc$accountId, orderType = "LIMIT", qty = 1,
  leg_symbols = c("RIVN", "RIVN250815C00012000"),
  leg_types = c("EQUITY", "OPTION"),
  leg_sides = c("BUY", "SELL"),
  leg_indicator = c("OPEN", "OPEN"),
  leg_ratios = c(100, 1),
  tif = "DAY", lmtPrc = 11.75,
  orderId = rp_getOrderId())

## End(Not run)
```

Description

Single-Leg Live Order

Usage

```
rp_order_single(
    accountId,
    ticker,
    symType,
    orderId,
    side = NULL,
    ordType = NULL,
    timeInForce = NULL,
    expirationTime = NULL,
    qty = NULL,
    amt = NULL,
    lmtPrc = NULL,
    stopPrc = NULL,
    openCloseIndicator = NULL
)
```

Arguments

accountId	= Public Brokerage Account Number
ticker	= Ticker symbol: Ex. "SPY"
symType	= Ticker Type: Ex. 'EQUITY', 'OPTION', 'MULTI_LEG_INSTRUMENT', 'CRYPTO', 'ALT', 'TREASURY', 'BOND', 'INDEX'
orderId	= The order ID: use rp_getOrderId()
side	= The Order Side BUY/SELL. For Options also include the openCloseIndicator. Ex. 'BUY' OR 'SELL'
ordType	= The Type of order: Ex. 'MARKET', 'LIMIT', 'STOP', 'STOP_LIMIT'
timeInForce	= The time in for the order: Ex. 'DAY' OR 'GTD'
expirationTime	= The expiration date. Only used when timeInForce is GTD, cannot be more than 90 days in the future
qty	= The order quantity. Used when buying/selling whole shares and when selling fractional. Mutually exclusive with amount
amt	= The order amount. Used when buying/selling shares for a specific notional value
lmtPrc	= The limit price. Used when orderType = LIMIT or orderType = STOP_LIMIT
stopPrc	= The stop price. Used when orderType = STOP or orderType = STOP_LIMIT
openCloseIndicator	= Used for options only. Indicates if this is BUY to OPEN/CLOSE

Value

Submit a live single-leg order and returns the order ID as a `data.frame`.

Examples

```
## Not run:
# Submit a live single-leg order to your Public Brokerage Account
my_acc <- rp_getAccts()

# Option Order
rp_order_singleLeg(accountId = my_acc$accountId, ticker = "SPY250815C00633000", symType = "OPTION",
                    orderId = rp_getOrderId(), side = "BUY", ordType = "LIMIT", lmtPrc = 1.50,
                    timeInForce = "DAY", qty = 1, openCloseIndicator = "OPEN")

# Equity Fraction Share Order
rp_preOrder_singleLeg(accountId = my_acc$accountId, ticker = "TSLA", symType = "EQUITY",
                      side = "BUY", ordType = "MARKET", timeInForce = "DAY", qty = 0.50,
                      openCloseIndicator = "OPEN")

## End(Not run)
```

rp_preOrder_multiLeg *Preflight Multiple-Leg*

Description

Preflight Multiple-Leg

Usage

```
rp_preOrder_multiLeg(
  accountId,
  orderType,
  qty,
  leg_symbols,
  leg_types,
  leg_sides,
  leg_indicator,
  leg_ratios,
  tif,
  mins = NULL,
  lmtPrc = NULL
)
```

Arguments

accountId	= Public Brokerage Account Number
orderType	= The Type of order: Ex. 'MARKET', 'LIMIT', 'STOP', 'STOP_LIMIT'
qty	= leg_ratio multiple: ex. '2' multiples the leg_ratios by 2X
leg_symbols	= Symbols: ex. c("SPY250815C00631000", "SPY250815C00631000")
leg_types	= Symbol types: ex. c("OPTION", "OPTION")

leg_sides = The side for each leg: ex. c("BUY", "SELL")
 leg_indicator = Indicates if this is BUY to OPEN/CLOSE ex. c("OPEN", "OPEN")
 leg_ratios = The number of contracts to BUY/SELL: ex. c('5','5')
 tif = The time in for the order: 'DAY' or 'GTD'
 mins = Minutes till order expires.
 lmtPrc = The limit price. Used when orderType = LIMIT or orderType = STOP_LIMIT

Value

Calculates the estimated financial impact of a complex multi-leg trade before execution and returns as a data.frame.

Examples

```
## Not run:
# Fetches costs associated with the type of order being placed
my_acc <- rp_getAccts()

# open bull-call spread for 0.25 (buy 2, sell 2)
rp_preOrder_multiLeg(accountId = my_acc$accountId, orderType = "LIMIT", qty = 2,
  leg_symbols = c("SPY250815C00630000", "SPY250815C00632000"),
  leg_types = c("OPTION", "OPTION"), leg_sides = c("BUY", "SELL"),
  leg_indicator = c("OPEN", "OPEN"), leg_ratios = c(1, 1),
  tif = "DAY", lmtPrc = 0.25)

# open long butterfly for 0.05
rp_preOrder_multiLeg(accountId = my_acc$accountId, orderType = "LIMIT", qty = 1,
  leg_symbols = c("SPY250815C00630000",
    "SPY250815C00631000",
    "SPY250815C00632000"),
  leg_types = c("OPTION", "OPTION", "OPTION"),
  leg_sides = c("BUY", "SELL", "BUY"),
  leg_indicator = c("OPEN", "OPEN", "OPEN"),
  leg_ratios = c(1, 2, 1), tif = "DAY", lmtPrc = 0.05)

# open iron-condor
rp_preOrder_multiLeg(accountId = my_acc$accountId, orderType = "LIMIT", qty = 1,
  leg_symbols = c("SPY250815C00631000", "SPY250815C00630000",
    "SPY250815C00625000", "SPY250815C00624000"),
  leg_types = c("OPTION", "OPTION", "OPTION", "OPTION"),
  leg_sides = c("SELL", "BUY", "SELL", "BUY"),
  leg_indicator = c("OPEN", "OPEN", "OPEN", "OPEN"),
  leg_ratios = c(1, 1, 1, 1), tif = "DAY", lmtPrc = 0.30)

## End(Not run)
```

rp_preOrder_singleLeg *Preflight Single-Leg*

Description

Preflight Single-Leg

Usage

```
rp_preOrder_singleLeg(
    accountId,
    ticker,
    symType,
    side = NULL,
    ordType = NULL,
    timeInForce = NULL,
    expirationTime = NULL,
    qty = NULL,
    amt = NULL,
    lmtPrc = NULL,
    stopPrc = NULL,
    openCloseIndicator = NULL
)
```

Arguments

accountId	= Public Brokerage Account Number
ticker	= Ticker symbol: Ex. "SPY"
symType	= Ticker Type: Ex. 'EQUITY', 'OPTION', 'MULTI_LEG_INSTRUMENT', 'CRYPTO', 'ALT', 'TREASURY', 'BOND', 'INDEX'
side	= The Order Side BUY/SELL. For Options also include the openCloseIndicator. Ex. 'BUY' OR 'SELL'
ordType	= The Type of order: Ex. 'MARKET', 'LIMIT', 'STOP', 'STOP_LIMIT'
timeInForce	= The time in for the order: Ex. 'DAY' OR 'GTD'
expirationTime	= The expiration date. Only used when timeInForce is GTD, cannot be more than 90 days in the future
qty	= The order quantity. Used when buying/selling whole shares and when selling fractional. Mutually exclusive with amount
amt	= The order amount. Used when buying/selling shares for a specific notional value
lmtPrc	= The limit price. Used when orderType = LIMIT or orderType = STOP_LIMIT
stopPrc	= The stop price. Used when orderType = STOP or orderType = STOP_LIMIT
openCloseIndicator	= Used for options only. Indicates if this is BUY to OPEN/CLOSE

Value

Calculates the estimated financial impact of a potential trade before execution and returns as a `data.frame`.

Examples

```
## Not run:
# Fetches costs associated with the type of order being placed
my_acc <- rp_getAccts()
rp_preOrder_singleLeg(accountId = my_acc$accountId, ticker = "SPY250815C00633000",
                      symType = "OPTION", side = "BUY", ordType = "MARKET",
                      timeInForce = "DAY", qty = 1, openCloseIndicator = "OPEN")

rp_preOrder_singleLeg(accountId = my_acc$accountId, ticker = "TSLA",
                      symType = "EQUITY", side = "BUY", ordType = "MARKET",
                      timeInForce = "DAY", qty = 0.50,
                      openCloseIndicator = "OPEN")

## End(Not run)
```

Index

* datasets

- [.rp_env, 3](#)
- [.rp_checkAccessToken, 2](#)
- [.rp_env, 3](#)
- [.rp_make_multileg_payload, 3](#)
- [.rp_make_opt_symbol, 4](#)
- [.rp_make_ord_payload, 5](#)
- [.rp_make_qte_payload, 6](#)
- [.rp_read_tokens, 7](#)
- [rpBars, 8](#)
- [rpCancelOrder, 9](#)
- [rpFixBars, 9](#)
- [rpGetGreeks, 17](#)
- [rpGetOrder, 18](#)
- [rpGetAccHist, 10](#)
- [rpGetAccToken, 11](#)
- [rpGetAccts, 12](#)
- [rpGetAcctsPort, 12](#)
- [rpGetAllInstruments, 13](#)
- [rpGetInstrument, 14](#)
- [rpGetOptChains, 14](#)
- [rpGetOptExp, 15](#)
- [rpGetOrderId, 16](#)
- [rpGetQuote, 16](#)
- [rpHistBars, 18](#)
- [rpOrderMulti, 20](#)
- [rpOrderSingle, 21](#)
- [rpPreOrderMultiLeg, 23](#)
- [rpPreOrderSingleLeg, 25](#)