Package: schwabr (via r-universe)

February 12, 2025

Type Package

Title 'Schwab API' Interface

Version 0.1.3

Author Anthony Trevisan [aut, cre]

Maintainer Anthony Trevisan <anthonytrevisan@gmail.com>

URL https://altanalytics.github.io/schwabr/

BugReports https://github.com/altanalytics/schwabr/issues

Description Use R to interface with the 'Charles Schwab Trade API' <https://developer.schwab.com/>. Functions include authentication, trading, price requests, account information, and option chains. A user will need a Schwab brokerage account and Schwab Individual Developer app. See README for authentication process and examples.

License GPL-3

Encoding UTF-8

LazyData true

Imports httr, urltools (>= 1.7.3), base64enc, lubridate, dplyr, jsonlite, magrittr, methods

RoxygenNote 7.3.2

Suggests testthat

Config/pak/sysreqs libssl-dev

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

RemoteUrl https://github.com/altanalytics/schwabr

RemoteRef HEAD

RemoteSha e3bdeecb13b3a576eb5539e781aaa2c12dfd8be9

Contents

`schwab_accountData` *Get account balances, positions, and account numbers returned as a list*

Description

Retrieves account data for all accounts linked to the Access Token or a specific account

Usage

```
schwab_accountData(
  output = "df",
  account_number = "",
  value_pull = c("all", "bal", "pos", "acts"),
  accessTokenList = NULL
)
```

Arguments

`output`	Use 'df' for a list of 3 data frames containing balances, positions, and orders. Otherwise the data will be returned as a list of lists
`account_number`	The account number as shown on the Schwab website
`value_pull`	Can be one of 'all','bal','pos','acts' depending on what you want to pull back
`accessTokenList`	A valid Access Token must be set using the output from `schwab_auth3_accessToken`. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

The output will be either a list of three data frames or a list of three lists that contain balances, positions, and account numbers for Schwab accounts linked to the access token or specified. For historical orders, see schwab_orderSearch. The default is for a data frame output which is much cleaner.

Value

a list of requested account details

Examples

```
## Not run:

# Get stored refresh token
refreshToken = readRDS('/secure/location/')

# Generate a new access token
accessTokenList = schwab_auth3_accessToken(appKey, appSecret, refreshToken)

# Passing the accessTokenList is optional. The default will return balances
asDF = schwab_accountData()
asList = schwab_accountData('list',account_number = '', accessTokenList)


## End(Not run)
```

---

schwab_act_hash *Get account hashed value*

Description

Retrieves the Hashed account value for a specific account

Usage

```
schwab_act_hash(account_number = "", accessTokenList = NULL)
```

Arguments

account_number	A Standard Schwab Account number
accessTokenList	A valid Access Token must be set using schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

A hashed account number

Examples

```
## Not run:

# Get stored refresh token
refreshToken = readRDS('/secure/location/')

# Generate a new access token
accessTokenList = schwab_auth3_accessToken(appKey, appSecret, refreshToken)

# Passing the accessTokenList is optional. The default will return balances
act_hash = schwab_act_hash(account_number = '123456789')



## End(Not run)
```

---

schwab_auth1_loginURL *Auth Step 1: Generate LogIn URL*

Description

Create URL to grant App access to Charles Schwab accounts

Usage

```
schwab_auth1_loginURL(appKey, callbackURL)
```

Arguments

appKey	'Schwab API' generated App Key for the registered app.
callbackURL	Users Callback URL for the registered app

Details

To use the 'Schwab API', both an account and a registered developer app are required. The developer app functions as a middle layer between the brokerage account and the API. A developer app should be registered on the Schwab Developer site. Once logged in to the developer site, use My Apps to register an application. An App will have a key and secret provided. The Key/Secret is auto generated and can be found under Dashboard > View Details at the bottom. The user must also create a Callback URL. The Callback URL must be a valid URL. The example below assumes the Callback URL is https://127.0.0.1. The Application should be in a "Ready to Use" state before attempting to login.

This function will use these inputs to generate a URL where the user can log in to their standard Charles Schwab Access Page and grant the application access to the specific accounts, enabling the API. The URL Authorization Code generated at the end of the log in process will feed into schwab_auth2_refreshToken. For questions, please reference the Schwab Docs or see the examples in the 'schwabr' readme.

Value

login url to grant app permission to Schwab accounts

Examples

```
# Visit the URL generated from the function below to log in accept terms and
# select the accounts you want to have API permissions.

# This assumes you set the callback to 'https://127.0.0.1'
appKey = 'ALPHANUM1234KEY'
loginURL = schwab_auth1_loginURL(appKey, 'https://127.0.0.1')
```

---

schwab_auth2_refreshToken

Auth Step 2: Obtain Refresh Token

Description

Get a Refresh Token using the Authorization Code

Usage

```
schwab_auth2_refreshToken(appKey, appSecret, callbackURL, codeToken)
```

Arguments

appKey	'Schwab API' generated App Key for the registered app.
appSecret	'Schwab API' generated Secret for the registered app.
callbackURL	Users Callback URL for the registered app
codeToken	Will be the URL at the end of Auth Step 1. Somewhere in the URL you should see code=CO.xxx. Paste the entire URL into the function.

Details

Once a URL has been generated using schwab_auth1_loginURL, a user can visit that URL to grant access to Schwab accounts. Once the button "Done" at the end of the process is pressed, the user will be redirected, potentially to "This site can't be reached". This indicates a successful log in. The URL of this page contains the Authorization Code. Paste the entire URL, not just the Authorization Code, into schwab_auth2_refreshToken. The authorization code will be a long alpha numeric string starting with 'https' and having 'code=' embedded.

The output of schwab_auth2_refreshToken will be a Refresh Token which will be used to gain access to the Schwab account(s) going forward. The Refresh Token will be valid for 7 days. Be sure to save the Refresh Token to a safe location.

The Refresh Token is needed to generate an Access Token using schwab_auth3_accessToken, which is used for general account access. The Access Token expires after 30 minutes but the

Refresh Token remains active for 7 days. You want to store your refresh token somewhere safe so that you can reference it later to regenerate an authorization token. After 7 days you have to manually log in again. The 'Schwab API' team indicated this might change in the future, but no set timeline.

Value

Refresh Token that is valid for 7 days

See Also

schwab_auth1_loginURL to generate a login url which leads to an authorization code, and more importantly generated a Refresh Token, you can feed the refresh token into schwab_auth3_accessToken to generate a new Access Token

Examples

```
## Not run:

# Initial access will require manually logging in to the URL from schwab_auth1_loginURL
# After a successful log in, the URL authorization code can be fed with a callbackURL
tok = schwab_auth2_refreshToken(appKey = 'schwab_APP_KEY',
                                appSecret = 'schwab_SECRET',
                                callbackURL = 'https://127.0.0.1',
                                codeToken = 'https://127.0.0.1?code=Auhtorizationcode')


# Save the Refresh Token somewhere safe where it can be retrieved
saveRDS(tok$refresh_token,'/secure/location/')


## End(Not run)
```

---

schwab_auth3_accessToken

Auth Step 3: Get Access Token

Description

Get a new Access Token using a valid Refresh Token

Usage

```
schwab_auth3_accessToken(appKey, appSecret, refreshToken)
```

Arguments

appKey	'Schwab API' generated App Key for the registered app.
appSecret	'Schwab API' generated Secret for the registered app.
refreshToken	An existing Refresh Token generated using schwab_auth2_refreshToken. Only pass the refresh_token, not the entire list

Details

An Access Token is required for the functions within 'schwabr' It serves as a user login to your accounts. The token is valid for 30 minutes and allows the user to place trades, get account information, get order history, pull historical stock prices, etc. A Refresh Token is required to generate an Access Token. schwab_auth2_refreshToken can be used to generate a Refresh Token which stays valid for 7 days. The appKey is generated automatically when an App is registered on the Schwab Developer site. By default, the Access Token is stored into options and will automatically be passed to downstream functions. However, the user can also submit an Access Token manually if multiple tokens are in use (for example: when managing more than one log in.)

DISCLOSURE: This software is in no way affiliated, endorsed, or approved by Charles Schwab or any of its affiliates. It comes with absolutely no warranty and should not be used in actual trading unless the user can read and understand the source code. The functions within this package have been tested under basic scenarios. There may be bugs or issues that could prevent a user from executing trades or canceling trades. It is also possible trades could be submitted in error. The user will use this package at their own risk.

Value

Access Token that is valid for 30 minutes. By default it is stored in options.This is a list of objects that also shows when the access token expires

See Also

schwab_auth1_loginURL to generate a login url which leads to an authorization code, then use schwab_auth2_refreshToken to generate Refresh Token with the authorization code

Examples

```
## Not run:

# A valid Refresh Token can be fed into the function below for a new Access Token
refreshToken = readRDS('/secure/location/')
accessTokenList = schwab_auth3_refreshToken('schwab_APPKey', 'schwab_AppSecret', refreshToken)


## End(Not run)
```

schwab_cancelOrder *Cancel an Open Order*

Description

Pass an Order ID and Account number to cancel an existing open order

Usage

```
schwab_cancelOrder(orderId, account_number, accessTokenList = NULL)
```

Arguments

orderId	A valid Schwab Order ID
account_number	A Schwab account number associated with the Access Token
accessTokenList	A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

order API URL. Message confirming cancellation

Examples

```
 ## Not run:

schwab_cancelOrder(orderId = 123456789, account_number = 987654321)


## End(Not run)
```

schwab_marketHours *Get Market Hours*

Description

Returns a list output for current day and specified market that details the trading window

Usage

```
schwab_marketHours(
  marketType = c("EQUITY", "OPTION", "BOND", "FUTURE", "FOREX"),
  date = Sys.Date(),
  accessTokenList = NULL
)
```

Arguments

marketType	The asset class to pull: 'EQUITY','OPTION','BOND','FUTURE','FOREX'. Default is EQUITY
date	Current or future date to check hours
accessTokenList	A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

List output of times and if the current date is a trading day

Examples

```
## Not run:

# Access Token must be set using schwab_auth_accessToken
# Market hours for the current date
schwab_marketHours()
schwab_marketHours('2020-06-24', 'OPTION')


## End(Not run)
```

---

schwab_optionChain *Get Options Chain*

Description

Search an Option Chain for a specific ticker

Usage

```
schwab_optionChain(
  ticker,
  strikes = 10,
  inclQuote = TRUE,
  startDate = Sys.Date() + 1,
  endDate = Sys.Date() + 360,
  contractType = c("ALL", "CALL", "PUT"),
  accessTokenList = NULL
)
```

Arguments

ticker	underlying ticker for the options chain
strikes	the number of strikes above and below the current strike
inclQuote	set TRUE to include pricing details (will be delayed if account is set for delayed quotes)
startDate	the start date for expiration (should be greater than or equal to today). Format: yyyy-mm-dd
endDate	the end date for expiration (should be greater than or equal to today). Format: yyyy-mm-dd
contractType	can be 'ALL', 'CALL', or 'PUT'
accessTokenList	A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

Return a list containing two data frames. The first is the underlying data for the symbol. The second item in the list is a data frame that contains the options chain for the specified ticker.

Value

a list of 2 data frames - underlying and options chain

Examples

```
## Not run:

# Pull all option contracts expiring over the next 6 months
# with 5 strikes above and below the at-the-money price
schwab_optionChain(ticker = 'SPY',
             strikes = 5,
             endDate = Sys.Date() + 180)


## End(Not run)
```

---

schwab_optionExpiration

Get Options Expiration Chain

Description

Search an Option Chain for a specific ticker

Usage

```
schwab_optionExpiration(ticker, accessTokenList = NULL)
```

Arguments

ticker	underlying ticker for the options chain
accessTokenList	A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

Return a list containing two data frames. The first is the underlying data for the symbol. The second item in the list is a data frame that contains the options chain for the specified ticker.

Value

a list of 2 data frames - underlying and options chain

Examples

```
## Not run:

# Pull all option contracts expiring over the next 6 months
# with 5 strikes above and below the at-the-money price
schwab_optionChain(ticker = 'SPY',
             strikes = 5,
             endDate = Sys.Date() + 180)


## End(Not run)
```

---

schwab_orderDetail *Get Details for a Single Order*

Description

Pass an order ID and Account number to get details such as status, quantity, ticker, executions (if applicable), account, etc.

Usage

```
schwab_orderDetail(orderId, account_number, accessTokenList = NULL)
```

Arguments

orderId	A valid Schwab Order ID
account_number	A Schwab account number associated with the Access Token
accessTokenList	A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

list of order details

Examples

```
## Not run:

# Get stored refresh token
refreshToken = readRDS('/secure/location/')

# generate a new access token
accessTokenList = schwab_auth3_accessToken('AppKey', 'AppSecret', refreshToken)

# Get order details for a single order
# Passing Access Token is optional once it's been set
schwab_orderDetail(orderId = 123456789, account_number = 987654321)


## End(Not run)
```

---

schwab_orderSearch *Search for orders by date*

Description

Search for orders associated with a Schwab account over the previous 60 days. The result is a list of three objects:

1. jsonlite formatted extract of all orders
2. all entered orders with details
3. a data frame of all executed orders with the executions

Usage

```
schwab_orderSearch(
  account_number,
  startDate = Sys.Date() - 30,
  endDate = Sys.Date(),
```

```
  maxResults = 50,
  orderStatus = "",
  accessTokenList = NULL
)
```

Arguments

account_number	A Schwab account number associated with the Access Token
startDate	Orders from a certain date with. Format yyyy-mm-dd.
endDate	Filter orders that occurred before a certain date. Format yyyy-mm-dd
maxResults	the max results to return in the query
orderStatus	search by order status (ACCEPTED, FILLED, EXPIRED, CANCELED, REJECTED, etc). This can be left blank for all orders. See documentation for full list
accessTokenList	A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

a list of three objects: a jsonlite formatted extract of all orders, all entered orders with details, a data frame of all executed orders with the executions

Examples

```
## Not run:

# Get all orders run over the last 50 days (up to 500)
schwab_orderSearch(account_number = account_number,
            startDate = Sys.Date()-50,
            maxResults = 500, orderStatus = '')


## End(Not run)
```

---

schwab_placeOrder *Place Order for a specific account*

Description

Place trades through the SchwabAPI using a range of parameters

Usage

```
schwab_placeOrder(
  account_number,
  ticker,
  quantity,
  instruction,
  orderType = "MARKET",
  limitPrice = NULL,
  stopPrice = NULL,
  assetType = c("EQUITY", "OPTION"),
  session = "NORMAL",
  duration = "DAY",
  stopPriceBasis = NULL,
  stopPriceType = NULL,
  stopPriceOffset = NULL,
  accessTokenList = NULL
)
```

Arguments

account_number	A Schwab account number associated with the Access Token
ticker	a valid Equity/ETF or option. If needed, use schwab_symbolDetail to confirm. This should be a ticker/symbol, not a CUSIP
quantity	the number of shares to be bought or sold. Must be an integer.
instruction	Equity instructions include 'BUY', 'SELL', 'BUY_TO_COVER', or 'SELL_SHORT'. Options instructions include 'BUY_TO_OPEN', 'BUY_TO_CLOSE', 'SELL_TO_OPEN', or 'SELL_TO_CLOSE'
orderType	MARKET, LIMIT (requiring limitPrice), STOP (requiring stopPrice), STOP_LIMIT, TRAILING_STOP (requiring stopPriceBasis, stopPriceType, stopPriceOffset)
limitPrice	the limit price for a LIMIT or STOP_LIMIT order
stopPrice	the stop price for a STOP or STOP_LIMIT order
assetType	EQUITY or OPTION. No other asset types are available at this time. EQUITY is the default.
session	NORMAL for normal market hours, AM or PM for extended market hours
duration	how long will the trade stay open without a fill: DAY, GOOD_TILL_CANCEL, FILL_OR_KILL
stopPriceBasis	LAST, BID, or ASK which is the basis for a STOP, STOP_LIMIT, or TRAILING_STOP
stopPriceType	the link to the stopPriceBasis. VALUE for dollar difference or PERCENT for a percentage offset from the price basis
stopPriceOffset	an integer that indicates the offset used for the stopPriceType, 10 and PERCENT is a 10 percent offset from the current price basis. 5 and VALUE is a 5 dollar offset from the current price basis

accessTokenList
: A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

A valid account and access token must be passed. An access token will be passed by default when schwab_auth3_accessToken is executed successfully and the token has not expired, which occurs after 30 minutes. Only simple orders using equities and options can be traded at through this function at this time. This function is built to allow a single trade submission. More complex trades can be executed through the API, but a custom function or submission will need to be constructed. To build more custom trading strategies, reference the 'Schwab API' examples. A full list of the input parameters and details can be found in the documentation. TEST ALL ORDERS FIRST WITH SMALL DOLLAR AMOUNTS!!!

A minimum of four parameters are required for submission: ticker, instruction, quantity, and account number associated with the Access Token. The following parameters default: session - NORMAL, duration - DAY, asset type - EQUITY, and order type - MARKET

Value

the trade id, account id, and other order details

Warning

TRADES THAT ARE SUCCESSFULLY ENTERED WILL BE SUBMITTED IMMEDIATELY THERE IS NO REVIEW PROCESS. THIS FUNCTION HAS HUNDREDS OF POTENTIAL COMBINATIONS AND ONLY A HANDFUL HAVE BEEN TESTED. IT IS STRONGLY RECOMMENDED TO TEST THE DESIRED ORDER ON A VERY SMALL QUANTITY WITH LITTLE MONEY AT STAKE. ANOTHER OPTION IS TO USE LIMIT ORDERS FAR FROM THE CURRENT PRICE. TD AMERITRADE HAS THEIR OWN ERROR HANDLING BUT IF A SUCCESSFUL COMBINATION IS ENTERED IT COULD BE EXECUTED IMMEDIATELY. DOUBLE CHECK ALL ENTRIES BEFORE SUBMITTING.

Examples

```
## Not run:

# Get stored refresh token
refreshToken = readRDS('/secure/location/')

# generate a new access token
accessTokenList = schwab_auth3_accessToken('AppKey', 'AppSecret', refreshToken)

# Set Account Number
account_number = 1234567890

# Standard market buy order
# Every order must have at least these 4 paramters
schwab_placeOrder(account_number = account_number,
```

```
                  ticker = 'AAPL',
                  quantity = 1,
                  instruction = 'buy')

# Stop limit order - good until canceled
schwab_placeOrder(account_number = account_number,
            ticker = 'AAPL',
            quantity = 1,
            instruction = 'sell',
            duration = 'good_till_cancel',
            orderType = 'stop_limit',
            limitPrice = 98,
            stopPrice = 100)

# Trailing Stop Order
schwab_placeOrder(account_number = account_number,
            ticker='AAPL',
            quantity = 1,
            instruction='sell',
            orderType = 'trailing_stop',
            stopPriceBasis = 'BID',
            stopPriceType = 'percent',
            stopPriceOffset = 10)

# Option Order with a limit price


## End(Not run)
```

---

schwab_priceHistory *Get price history for a multiple securities*

Description

Open, Close, High, Low, and Volume for one or more securities

Usage

```
schwab_priceHistory(
  tickers = c("AAPL", "MSFT"),
  startDate = Sys.Date() - 30,
  endDate = Sys.Date(),
  freq = c("daily", "1min", "5min", "10min", "15min", "30min"),
  accessTokenList = NULL
)
```

Arguments

tickers	a vector of tickers - no more than 15 will be pulled. for bigger requests, split up the request or use the 'Tiingo API', 'FMP Cloud API', or other free data providers

startDate	the Starting point of the data
endDate	the Ending point of the data
freq	the frequency of the interval. Can be daily, 1min, 5min, 10min, 15min, or 30min
accessTokenList	A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

Pulls price history for a list of security based on the parameters that include a date range and frequency of the interval. Depending on the frequency interval, data can only be pulled back to a certain date. For example, at a one minute interval, data can only be pulled for 30-35 days. Prices are adjusted for splits but not dividends.

PLEASE NOTE: Large data requests will take time to pull back because of the looping nature. The 'Schwab API' does not allow bulk ticker request, so this is simply running each ticker individually. For faster and better historical data pulls, try the 'Tiingo API' or 'FMP Cloud API'

Value

a tibble of historical price data

Examples

```
## Not run:

# Set the access token and a provide a vector of one or more tickers
refreshToken = readRDS('/secure/location/')
accessToken = schwab_auth_accessToken(refreshToken, 'consumerKey')
tickHist5min = schwab_priceHistory(c('TSLA','AAPL'), freq='5min')

# The default is daily. Access token is optional once it's been set
tickHistDay = schwab_priceHistory(c('SPY','IWM'), startDate = '1990-01-01')


## End(Not run)
```

---

schwab_priceQuote *Get Quotes for specified tickers in List form*

Description

Enter tickers for real time or delayed quotes returned as a list

Usage

```
schwab_priceQuote(
  tickers = c("AAPL", "MSFT"),
  output = "df",
  accessTokenList = NULL
)
```

Arguments

tickers	One or more tickers
output	indication on whether the data should be returned as a list or df. The default is 'df' for data frame, anything else would be a list.
accessTokenList	A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

Quotes may be delayed depending on agreement with Schwab. If the account is set up for real-time quotes then this will return real-time. Otherwise the quotes will be delayed.

Value

a list or data frame with quote details for each valid ticker submitted

Examples

```
## Not run:

# Get stored refresh token
refreshToken = readRDS('/secure/location/')

# generate a new access token
accessToken = schwab_auth_accessToken('consumerKey', refreshToken)

# Pass one or more tickers as a vector
# accessToken is optional once it is set
quoteSPY = schwab_priceQuote('SPY')
quoteList = schwab_priceQuote(c('GOOG','TSLA'), output = 'list', accessToken)


## End(Not run)
```

schwab_symbolDetail *Get ticker details*

Description

Get identifiers and fundamental data for a specific ticker

Usage

```
schwab_symbolDetail(tickers = c("AAPL", "SPY"), accessTokenList = NULL)
```

Arguments

tickers	valid ticker(s) or symbol(s)
accessTokenList	A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

data frame of ticker details

Examples

```
## Not run:

# Details for Apple
schwab_symbolDetail('AAPL')


## End(Not run)
```

schwab_transactSearch *Search for all Transaction types*

Description

Can pull trades as well as transfers, dividend reinvestment, interest, etc. Any activity associated with the account.

Usage

```
schwab_transactSearch(
  account_number,
  startDate = Sys.Date() - 30,
  endDate = Sys.Date(),
  transType = "TRADE",
  accessTokenList = NULL
)
```

Arguments

account_number	A Schwab account number associated with the Access Token
startDate	Transactions after a certain date. Will not pull back transactions older than 1 year. format yyyy-mm-dd
endDate	Filter transactions that occurred before a certain date. format yyyy-mm-dd
transType	Filter for a specific Transaction type. No entry will return all types. For example: TRADE, CASH_IN_OR_CASH_OUT, CHECKING, DIVIDEND, INTEREST, OTHER
accessTokenList	A valid Access Token must be set using the output from schwab_auth3_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

a jsonlite data frame of transactions

Examples

```
## Not run:

# Access Token must be set using schwab_auth_accessToken
# Transactions for the last 5 days
schwab_transactSearch(account_number = 987654321,
                startDate = Sys.Date()-days(5))


## End(Not run)
```

Index