Package 'charlesschwabapi'

November 26, 2025

Type Package

Title Wrapper Functions Around 'Charles Schwab Individual Trader API'

Version 1.0.5

Description For those wishing to interact with the 'Charles Schwab Individual Trader API' (<https://developer.schwab.com/products/trader-api--individual>) with R in a simplified manner, this package offers wrapper functions around authentication and the available API calls to streamline the process.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Imports httr, lubridate, stringr, purrr, tidyr, anytime, dplyr, openssl, tibble

Suggests testthat (>= 3.0.0), covr

Config/testthat/edition 3

NeedsCompilation no

Author Nick Bultman [aut, cre, cph], Todd Sykes [ctb]

Maintainer Nick Bultman <njbultman74@gmail.com>

Repository CRAN

Date/Publication 2025-11-26 17:40:02 UTC

Contents

`cancel_order` *Cancel Specific Order by ID for Account*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function, the encrypted account ID, and the order ID, cancel the specific order.

Usage

```
cancel_order(tokens, encrypted_account_id, order_id)
```

Arguments

`tokens`	token object from ‘get_authentication_tokens‘ function (list).
`encrypted_account_id`	encrypted ID of the account from ‘get_account_numbers‘ function (string).
`order_id`	order ID for the order (numeric).

Value

Returns a message informing the user if cancellation was successful or if there was an error.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

get_account *Get Account Information for Specific Account*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function and the encrypted account ID, return the account information (positions, fundamentals, and general account information). The encrypted account ID can be found using the ‘get_account_numbers‘ function.

Usage

```
get_account(tokens, encrypted_account_id, fields = NULL)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).
encrypted_account_id	encrypted ID of the account from ‘get_account_numbers‘ function (string).
fields	specific fields to be returned, one example being "positions" (string).

Value

Returns a data frame containing the account information. This includes position information, a day trader flag, the account number, and more.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

get_accounts *Get Accounts Information*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function, return the account(s) information associated with the authenticated user. By default, this includes positions, fundamentals, and general account information. However, one can use the ‘fields‘ argument to get more specific as to the information returned.

Usage

```
get_accounts(tokens, fields = NULL)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).
fields	specific fields to be returned, an example being "positions" (string).

Value

Returns a data frame containing the account(s) information.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

---

get_account_numbers *Get Account Numbers*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function, return the account number(s) information, including the actual account number(s) and the encrypted ID(s) of the user that were granted when authenticating. The encrypted IDs are used in other functions (like orders and transactions) that require a specific encrypted account ID to be specified.

Usage

```
get_account_numbers(tokens)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).

Value

Returns a data frame containing the account numbers and their encrypted values.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

`get_authentication_tokens` *Get Authentication Tokens*

Description

Given the app key, redirect uri, and app secret, this function will walk the user through the process to gather the appropriate authentication tokens (access token and refresh token) and store them in the user's specified location (if no location is specified, it will store the tokens in the user's working directory). Note that the tokens are saved in an RDS file via a list since additional metadata is captured in addition to the tokens (such as the expiration of those tokens to help with knowing when to refresh). After this function is initially called, be sure use the same path to the token for future calls along with maintaining the default name that is used for the RDS file to avoid manual reauthentication whenever possible. This function will always first look to see if an RDS file exists at the specified path and with the default name to check if tokens are valid or expired. Authentication requires no user intervention when the refresh token is valid. User intervention (via the login method through a separate browser) is only required when both the access token and the refresh token are expired.

Usage

```
get_authentication_tokens(
  app_key,
  redirect_uri,
  app_secret,
  token_save_path = getwd()
)
```

Arguments

`app_key`	application key generated by Charles Schwab (character)
`redirect_uri`	redirect URI specified when registering application (character)
`app_secret`	application secret generated by Charles Schwab (character)
`token_save_path`	path to current token or where you would like token stored. Default is the working directory (character)

Value

Returns a message on whether the authentication was successful or not along with token information (if successful, NULL otherwise), including the path to where the token RDS object is saved.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

get_instruments *Get Instruments*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function, the symbol(s) of interest, and the search type, return a data frame with information about the securities matching the search.

Usage

```
get_instruments(tokens, symbol, projection)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).
symbol	symbol(s) of interest to search for (string or character vector).
projection	type of search to be performed. Valid values are "symbol-search", "symbol-regex", "desc-search", "desc-regex", "search", and "fundamental" (string).

Value

Returns a data frame containing information surrounding the symbol(s) of interest in the search.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

get_instruments_cusip *Get Instruments by Cusip*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function and a cusip, return a data frame with information about the security matching the search.

Usage

```
get_instruments_cusip(tokens, cusip_id)
```

Arguments

tokens	tokens object from ‘get_authentication_tokens‘ function (list).
cusip_id	cusip of the security to be searched on (string).

Value

Returns a data frame containing information surrounding the cusip of interest in the search.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

`get_market_hours` *Get Market Hours*

Description

Given the tokens object from the 'get_authentication_tokens' function, return a data frame containing information about the market(s) of interest and the specific hours of operation. By default, all of the markets are returned for today's date, but both the specific markets returned and the date can be tweaked. Please see the parameters for more specifics related to what can be specified for the function.

Usage

```
get_market_hours(
  tokens,
  markets = c("equity", "option", "bond", "future", "forex"),
  date = NULL
)
```

Arguments

`tokens`	token object from 'get_authentication_tokens' function (list).
`markets`	markets of interest that are "equity", "option", "bond", "future", and/or "forex". Default is all markets (string or character vector).
`date`	date you would like to get the hours of operation. Valid dates are today through one year from now. Default is NULL, which is today (date).

Value

Returns a data frame containing information surrounding the market of interest and its specific hours of operation.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

get_market_hours_single *Get Market Hours for Single Market*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function and the market of interest, return specific information about that market’s hours of operation. Note that a data parameter can also be specified between now and one year from now to obtain specific hours of operation for the market related to that date.

Usage

```
get_market_hours_single(tokens, market, date = NULL)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).
market	market of interest that is either "equity", "option", "bond", "future", or "forex" (string).
date	date you would like to get the hours of operation. Valid dates are today through one year from now. Default is NULL, which is today (date).

Value

Returns a data frame containing information surrounding the market’s hours of operation.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

get_movers *Get Movers*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function and the symbol of interest, return the top 10 securities movement for a specific index in a data frame. By default, it is sorted by volume and the frequency is 0, but these can be tweaked.

Usage

```
get_movers(tokens, symbol_id, sort = NULL, frequency = NULL)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).
symbol_id	symbol of interest to get movers from. Valid values are "$DJI", "$COMPX", "$SPX", "NYSE", "NASDAQ", "OTCBB", "INDEX_ALL", "EQUITY_ALL", "OPTION_ALL", "OPTION_PUT", and "OPTION_CALL" (string).
sort	the attribute that you would like sorted by. Valid values are "VOLUME", "TRADES", "PERCENT_CHANGE_UP", and "PERCENT_CHANGE_DOWN". Default is NULL, which is VOLUME (string).
frequency	to return movers with hthe specified directions of up or down. Valid values are 0, 1, 5, 10, 30, or 60. Default is NULL, which is zero (numeric).

Value

Returns a data frame containing information surrounding the top 10 securities movement for the symbol specified.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

---

get_option_chains *Get Option Chains*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function and the symbol of interest, return a data frame containing information about the option chain. Note that there are many ways to customize the data returned, including being able to specify the strategy as analytical and providing custom input assumptions. Refer to the parameters below for more detailed information.

Usage

```
get_option_chains(
  tokens,
  symbol,
  contract_type = NULL,
  strike_count = NULL,
  include_underlying_quote = NULL,
  strategy = NULL,
  interval = NULL,
  strike = NULL,
  range = NULL,
  from_date = NULL,
  to_date = NULL,
  volatility = NULL,
  underlying_price = NULL,
```

```
  interest_rate = NULL,
  days_to_expiration = NULL,
  exp_month = NULL,
  option_type = NULL,
  entitlement = NULL
)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).
symbol	symbol of interest (string).
contract_type	string containing "ALL", "PUT", or "CALL". Default is NULL, which is "ALL" (string).
strike_count	number of strikes to return above or below the at-the-monty price (numeric). Default is NULL, which will return all.
include_underlying_quote	flag to indicate whether underlying quotes to be included. Default is NULL, which is TRUE (boolean).
strategy	type of strategy to return, can be "SINGLE", "ANALYTICAL", "COVERED", "VERTICAL", "CALENDAR", "STRANGLE", "STRADDLE", "BUTTERFLY", "CONDOR", "DIAGONAL", "COLLAR", or "ROLL". Default is NULL, which is "SINGLE" (string).
interval	strike interval for the spread strategy chains (see strategy parameter). Default is NULL (numeric).
strike	strike price - default is NULL (which will not return a specific strike price) (numeric).
range	range (ITM, NTM, OTM, etc.). Default is NULL, which will return all (string).
from_date	from date - default is NULL, which will return all (date).
to_date	to date - default is NULL, which will return all (date).
volatility	volatility to use in calculation and only applies to ANALYTICAL strategy (see strategy parameter). Default is NULL. (numeric).
underlying_price	underlying price to use in calculation and only applies to ANALYTICAL strategy (see strategy parameter). Default is NULL (numeric).
interest_rate	interest rate to use in calculation and only applies to ANALYTICAL strategy (see strategy parameter). Default is NULL (numeric).
days_to_expiration	days to expiration to use in calculation and only applies to ANALYTICAL strategy (see strategy parameter). Default is NULL (numeric).
exp_month	expiration month - valid values are "ALL", "JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", or "DEC". Default is NULL, which is "ALL" (string).
option_type	option type - default is NULL, which will return everything (string).
entitlement	applicable only for retail token, entitlement of client PP-PayingPro, NP-NonPro, and PN-NonPayingPro. Valid values are "PN", "NP", "PP" - default is NULL, which will return everything (string).

Value

Returns a data frame containing information about the given symbol's option chain.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

`get_option_expiration_chain`
Get Option Expiration Chain

Description

Given the tokens object from the 'get_authentication_tokens' function and the symbol of interest, return the option expiration chain information related to the symbol. This includes expiration dates, expiration types, settlement types, and more.

Usage

```
get_option_expiration_chain(tokens, symbol)
```

Arguments

`tokens`	token object from 'get_authentication_tokens' function (list).
`symbol`	symbol for option expiration chain (string).

Value

Returns a data frame containing information surrounding the option expiration chain.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

get_orders *Get Orders*

Description

Given the tokens object from the 'get_authentication_tokens' function, return a data frame containing the orders for all accounts related to the authenticated user. By default, it will return all orders (default max is 3000) from the last 60 days. This can be adjusted through additional function parameters, but the maximum date range is one year.

Usage

```
get_orders(
  tokens,
  from_entered_datetime = strftime(Sys.time() - lubridate::days(60), format =
    "%Y-%m-%dT%H:%M:%S.000Z"),
  to_entered_datetime = strftime(Sys.time(), format = "%Y-%m-%dT%H:%M:%S.000Z"),
  max_results = NULL,
  status = NULL
)
```

Arguments

tokens	token object from 'get_authentication_tokens' function (list).
from_entered_datetime	start of time range for orders that should be gathered - default is current datetime less 60 days (string).
to_entered_datetime	end of time range for orders that should be gathered - default is current datetime (string).
max_results	maximum number of results to be returned - default is NULL, which is 3000 (numeric).
status	only orders of this status should be returned. Default is NULL, which is all statuses. Valid values are "AWAITING_PARENT_ORDER", "AWAITING_CONDITION", "AWAITING_STOP_CONDITION", "AWAITING_MANUAL_REVIEW", "ACCEPTED", "AWAITING_UR_OUT", "PENDING_ACTIVATION", "QUEUED", "WORKING", "REJECTED", "PENDING_CANCEL", "CANCELED", "PENDING_REPLACE", "REPLACED", "FILLED", "EXPIRED", "NEW", "AWAITING_RELEASE_TIME", "PENDING_ACKNOWLEDGEMENT", "PENDING_RECALL", and "UNKNOWN" (string).

Details

The easiest way to enter the right strings in the start/end times is to begin with 'Sys.time()', use the 'lubridate' package to adjust it, and then format it as ' date less 30 days, you can use: 'strftime(Sys.time() - lubridate::days(30), format = " The defaults for this function illustrate the appropriate usage for the API.

Value

Returns a data frame containing order information for all accounts affiliated with authorized user.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

`get_orders_account` *Get Orders for Specific Account*

Description

Given the tokens object from the 'get_authentication_tokens' function and the encrypted account ID, return a data frame containing the orders for the specific account. By default, it will return all orders (default max is 3000) from the last 60 days. This can be adjusted through additional function parameters.

Usage

```
get_orders_account(
  tokens,
  encrypted_account_id,
  from_entered_datetime = strftime(Sys.time() - lubridate::days(60), format =
    "%Y-%m-%dT%H:%M:%S.000Z"),
  to_entered_datetime = strftime(Sys.time(), format = "%Y-%m-%dT%H:%M:%S.000Z"),
  max_results = NULL,
  status = NULL
)
```

Arguments

`tokens`	token object from 'get_authentication_tokens' function (list).
`encrypted_account_id`	encrypted ID of the account from 'get_account_numbers' function (string).
`from_entered_datetime`	start of time range for orders that should be gathered - default is current datetime less 60 days (string).
`to_entered_datetime`	end of time range for orders that should be gathered - default is current datetime (string).
`max_results`	maximum number of results to be returned - default is NULL, which is 3000 (numeric).

`status` only orders of this status should be returned. Default is NULL, which is all statuses. Valid values are "AWAITING_PARENT_ORDER", "AWAITING_CONDITION", "AWAITING_STOP_CONDITION", "AWAITING_MANUAL_REVIEW", "ACCEPTED", "AWAITING_UR_OUT", "PENDING_ACTIVATION", "QUEUED", "WORKING", "REJECTED", "PENDING_CANCEL", "CANCELED", "PENDING_REPLACE", "REPLACED", "FILLED", "EXPIRED", "NEW", "AWAITING_RELEASE_TIME", "PENDING_ACKNOWLEDGEMENT", "PENDING_RECALL", and "UNKNOWN" (string).

Details

The easiest way to enter the right strings in the start/end times is to begin with ‘Sys.time()‘, use the ‘lubridate‘ package to adjust it, and then format it as ‘ date less 30 days, you can use: ‘strftime(Sys.time() - lubridate::days(30), format = " The defaults for this function illustrate the appropriate usage for the API.

Value

Returns a data frame containing order information for the specified account.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

---

get_order_id *Get Specific Order by ID for Account*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function, the encrypted account ID, and the order ID, return a data frame containing information about the specific order.

Usage

```
get_order_id(tokens, encrypted_account_id, order_id)
```

Arguments

`tokens` token object from ‘get_authentication_tokens‘ function (list).

`encrypted_account_id` encrypted ID of the account from ‘get_account_numbers‘ function (string).

`order_id` order ID for the order (numeric).

Value

Returns a data frame containing the information about the specific order.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

`get_price_history` *Get Price History*

Description

Given the tokens object from the 'get_authentication_tokens' function and the equity symbol of interest, return the recent price history for the symbol. There are additional parameters that can be specified to customize the call, but specific information about the default value information is given below (both in this description and the specifics on each parameter).

Usage

```
get_price_history(
  tokens,
  symbol,
  period_type = NULL,
  period = NULL,
  frequency_type = NULL,
  frequency = NULL,
  start_datetime = NULL,
  end_datetime = NULL,
  need_extended_hours_data = NULL,
  need_previous_close = NULL
)
```

Arguments

`tokens`	token object from 'get_authentication_tokens' function (list).
`symbol`	equity symbol (string).
`period_type`	chart period being requested. Can be "day", "month", "year", or "ytd". Default is NULL, which is "day" (string).
`period`	number of chart period types. Default is NULL, which will then be dependent on the period type (numeric).
`frequency_type`	the time frequency type. Default is NULL, which will then be dependent on the period type (string).
`frequency`	the time frequency duration. Default is NULL, which is 1 (numeric).
`start_datetime`	start datetime of request. Default is NULL, which is the end date less the period (datetime).
`end_datetime`	end datetime of request. Default is NULL, which is the market close of the previous business day (datetime).

need_extended_hours_data
: indicator for whether extended hours data is needed. Default is NULL, which is TRUE (boolean).

need_previous_close
: indicator for whether the previous close price/date is needed. Default is NULL, which is FALSE (boolean).

Details

While the parameter defaults for this function are NULL, the values that ultimately feed into the API call are used behind-the-scenes. For example, the period type parameter's default in the function is NULL, but behind-the-scenes the API is using "day" to grab the appropriate data.

Additionally, there are defaults for parameters that are dependent on other parameters. For the period parameter, if the period type is "day", the default period is 10, otherwise it is 1. For the frequency type parameter, if it is "day" then the default value is "minute", otherwise it is "monthly" for year and "weekly" for the others.

Value

Returns a data frame containing information surrounding the price history of the symbol of interest.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

---

get_quotes *Get Quotes*

Description

Given the tokens object from the 'get_authentication_tokens' function and the symbols of interest, return a data frame containing information about those symbols. Note that this function can return information that goes beyond a standard quote (for example, fundamental information can be returned). By default, everything is returned, but the specific information returned can be customized through the 'fields' argument below.

Usage

```
get_quotes(tokens, symbols, fields = NULL, indicative = NULL)
```

Arguments

tokens	token object from 'get_authentication_tokens' function (list).
symbols	symbols for quotes (vector).
fields	request for subset of data. Possible values are NULL, "all", "quote", "fundamental", "extended", "reference", or "regular". Note these can be combined in a vector or a string with a value can be used (string or vector).
indicative	include indicative symbol quotes or not for ETF requests. Possible values are "true" and "false" (string).

Value

Returns a data frame containing information about the given symbols of interest.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

`get_quotes_single_symbol`
Get Quotes for Single Symbol

Description

Given the tokens object from the 'get_authentication_tokens' function and the symbol ID, return a data frame containing information about that symbol. Note that this function can return information that goes beyond a standard quote (for example, fundamental information can be returned). By default, everything is returned, but the specific information returned can be customized through the 'fields' argument below.

Usage

```
get_quotes_single_symbol(tokens, symbol_id, fields = NULL)
```

Arguments

`tokens`	token object from 'get_authentication_tokens' function (list).
`symbol_id`	symbol for the security of interest
`fields`	request for subset of data. Possible values are NULL, "all", "quote", "fundamental", "extended", "reference", or "regular". Note these can be combined in a vector or a string with a value can be used. The default is NULL, which is everything (string or vector).

Value

Returns a data frame containing information about the given symbol of interest.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

get_transaction *Get Specific Transaction Information*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function, the encrypted account ID of interest, and the transaction ID of interest, return information corresponding to that transaction. Note that the transaction ID can be obtained by first calling the ‘get_transactions‘ function, finding the transaction of interest, and then finding the transaction ID on that data frame.

Usage

```
get_transaction(tokens, encrypted_account_id, transaction_id)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).
encrypted_account_id	encrypted ID of the account from ‘get_account_numbers‘ function (string).
transaction_id	transaction ID of interest (numeric).

Value

Returns a data frame containing the transaction information.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

get_transactions *Get Account Transactions*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function and the encrypted ID for the account number of interest, get the transactions associated with that account number. By default, the last 180 days worth of transactions are returned. However, this can be tweaked according to the date parameters along with the types of transactions using the types parameter, but the maximum date range is one year.

Usage

```
get_transactions(
  tokens,
  encrypted_account_id,
  start_datetime = strftime(Sys.time() - lubridate::days(180), format =
    "%Y-%m-%dT%H:%M:%OS3Z"),
  end_datetime = strftime(Sys.time(), format = "%Y-%m-%dT%H:%M:%OS3Z"),
  symbol = NULL,
  types = NULL
)
```

Arguments

tokens	token object from 'get_authentication_tokens' function (list).
encrypted_account_id	encrypted ID of the account from 'get_account_numbers' function (string).
start_datetime	datetime that you would like to start gathering transactions from, in yyyy-mm-dd'T'HH:mm:ss.SSSZ format. Default is current datetime less 180 days (string).
end_datetime	datetime that you would like to end gathering transactions from, in yyyy-mm-dd'T'HH:mm:ss.SSSZ format. Default is current datetime (string).
symbol	filter for transactions based on this symbol. Default is NULL, which means no filtering (string).
types	filter for transactions based on its type. Default is NULL, which means no filtering. Available values are 'TRADE', 'RECEIVE_AND_DELIVER', 'DIVIDEND_OR_INTEREST', 'ACH_RECEIPT', 'ACH_DISBURSEMENT', 'CASH_RECEIPT', 'CASH_DISBURSEMENT', 'ELECTRONIC_FUND', 'WIRE_OUT', 'WIRE_IN', 'JOURNAL', 'MEMORANDUM', 'MARGIN_CALL', 'MONEY_MARKET', or 'SMA_ADJUSTMENT' (string or character vector).

Details

The easiest way to enter the right strings in the start/end times is to begin with 'Sys.time()', use the 'lubridate' package to adjust it, and then format it as ' date less 30 days, you can use: 'strftime(Sys.time() - lubridate::days(30), format = " The defaults for this function illustrate the appropriate usage for the API.

Value

Returns a data frame containing the account's transaction information.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

get_user_preferences *Get User Preferences*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function, returns the user preferences associated with the account that was authenticated.

Usage

```
get_user_preferences(tokens)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).

Value

Returns a data frame containing the user's preferences

Author(s)

Nick Bultman, <njbultman74@gmail.com>, June 2024

place_order *Place Order for Specific Account*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function, the encrypted account ID, and the request body (JSON), place the specific order. Note that due to the complexity of the orders that can be created, currently this function allows maximum flexibility by not cultivating an easier solution to building the request body and assumes the user passes the appropriate JSON. As a result, it is strongly encouraged to look at the documentation for how to build the proper orders for programmatic execution and do robust testing outside of market hours to ensure that when a live trade comes it will be just as the user intended. The user of this function assumes all risk that trades could not be executed exactly as intended.

Usage

```
place_order(tokens, encrypted_account_id, request_body)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).
encrypted_account_id	encrypted ID of the account from ‘get_account_numbers‘ function (string).
request_body	Valid request to API for placing an order (JSON).

Value

Returns a numeric order number and a message informing the user if the order was successfully placed/created or if there was an error.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

---

replace_order *Replace Order for Specific Account*

Description

Given the tokens object from the ‘get_authentication_tokens‘ function, the encrypted account ID, and the order ID and the request body, replace the specific order. Due to the complexity of the orders that can be created/replaced, currently this function allows maximum flexibility by not cultivating an easier solution to building the request body and assumes the user passes the appropriate JSON. Much like the ‘place_order‘ function, it is strongly encouraged to look at the documentation (in this package and on the Charles Schwab developer site) for how to build proper orders before attempting to replace any. The user of this function assumes all risk that trades could not be replaced (and then executed) exactly as intended.

Usage

```
replace_order(tokens, encrypted_account_id, order_id, request_body)
```

Arguments

tokens	token object from ‘get_authentication_tokens‘ function (list).
encrypted_account_id	encrypted ID of the account from ‘get_account_numbers‘ function (string).
order_id	order ID to be replaced (numeric).
request_body	valid request to API for replacing an order (JSON).

Value

Returns a numeric order number and a message informing the user if the order was successfully placed/created or if there was an error.

Author(s)

Nick Bultman, <njbultman74@gmail.com>, July 2024

Index