Package 'IButils'

May 19, 2024

Type Package

Title Utility Functions for the Interactive Brokers API

Version 0.4-0

Date 2024-03-12

Maintainer Enrico Schumann <es@enricoschumann.net>

Description Utility functions for the Interactive Brokers
API, based on packages 'IBrokers' and 'rib'.
The package provides tools, for instance, for
downloading historical data and reports from
IB.

License GPL-3

Imports IBrokers, textutils, zoo

Suggests rib, uuid

Additional_repositories http://enricoschumann.net/R/

URL http://enricoschumann.net/R/packages/IButils/,
https://github.com/enricoschumann/IButils

NeedsCompilation no

Author Enrico Schumann [aut, cre] (<https://orcid.org/0000-0001-7601-6576>)

Contents

IButils-package   *Utility Functions for the Interactive Brokers API*

Description

Utility functions for the Interactive Brokers API, based on packages 'IBrokers' and 'rib'. The package provides tools, for instance, for downloading historical data and reports from IB.

Details

The package contains functions to download historical data from IB.

Author(s)

Enrico Schumann [aut, cre] (<https://orcid.org/0000-0001-7601-6576>)

Maintainer: Enrico Schumann <es@enricoschumann.net>

---

combine_files   *Combine Data Files*

Description

Combine files of downloaded price data

Usage

```
combine_files(directory, max.rows = -1, pattern = NULL,
              verbose = TRUE, prefix = "processed___",
              delete.processed = FALSE, actual.timestamp = FALSE,
              sep = ",")
```

Arguments

directory	character
max.rows	numeric
pattern	character
verbose	logical
prefix	character
delete.processed	
	logical
actual.timestamp	
	logical
sep	character: the column separator

Details

Read CSV files and combine them into one large file per symbol.

Value

a data.frame

Author(s)

Enrico Schumann

See Also

[ib_hist_data](#)

Examples

```
## TODO add examples
```

---

flex_web_service   *Flex Web Service*

Description

Retrieve queries via the Flex Web Service.

Usage

```
flex_web_service(file, token, query, version = 3, delay = 2,
                 no.write.msg = TRUE, no.write.warn = TRUE,
                 verbose = TRUE)
```

Arguments

file	character: filename for the downloaded report
token	character (not numeric!)
query	integer
version	integer; currently only 3 is supported
delay	integer: number of seconds to wait between sending the token and retrieving the flex-data report
no.write.msg	logical: do not write file if it contains a message
no.write.warn	logical: do not write file if it contains a warning
verbose	logical: sets the 'quiet' argument of [download.file](#)

Details

Retrieve flex queries via `download.file`.

The function also checks whether the downloaded file contains messages from IB (lines that start with "MSG").

Value

The function is called for its side effect: downloading and storing the flex query.

The function will first send `token` and `query` to IB. If the response does not contain a

```
<Status>Success</Status>
```

, the complete response will be printed and a value of 1 is invisibly returned.

Otherwise, an attempt to download the report is made, and the return value of `download.file` will be invisibly returned.

However, even if the download succeeds, the file may contain error messages or warnings. Even mere 'messages' often indicate that something did not work (e.g. particular accounts could not be included). `flex_web_service` will print messages or warnings that it finds in the files, and the return value will be invisible 1. The downloaded file will not be written in such a case, unless `no.write.msg` and `no.write.warn` are set to FALSE; both arguments default to `TRUE`.

Author(s)

Enrico Schumann

References

https://ibkrcampus.com/ibkr-api-page/flex-web-service/

See Also

`ib_hist_data`, `download.file`

Examples

```
## Not run:
flex_web_service(file = "~/my_files/my_report.csv",
                 token = "12345678901234567890", ## character!
                 query = 123)

file <- "~/Downloads/Net_Asset_Value_NAV_in_Base"
ans <- read_flex_report(file)


## End(Not run)
```

`ib_hist_data`	*Download Historical Data*

Description

Download historical data from IB API and store them as text files.

Usage

```
ib_hist_data(Symbol, Security_Type, Exchange, Currency, id = NULL,
             directory, barSize, durationStr = NULL, whatToShow,
             start = as.POSIXct(Sys.Date() - 30), end = Sys.time(),
             useRTH = FALSE,
             skip.from, skip.until, skip.tz = "",
             verbose = TRUE, trim = TRUE, accumulate = FALSE,
             port = 7496, sep = ",",
             filename = "%id%_%start%_%end%",
             backend = NULL,
             clientId = NULL)
```

Arguments

`Symbol`	character
`Security_Type`	character
`Exchange`	character
`Currency`	character
`id`	character; optional: the prefix for text files. If NULL, Symbol etc. are pasted together
`directory`	character: the directory in which the files are stored
`barSize`	character
`durationStr`	character. If NULL, the function attempts to set a long duration. (This argument may be removed in the future since it rarely makes sense to set it explicitly.)
`whatToShow`	character: TRADES, MIDPOINT, BID or ASK
`start`	POSIXct
`end`	POSIXct
`useRTH`	logical
`skip.from`	character
`skip.until`	character
`skip.tz`	character
`verbose`	logical
`accumulate`	logical
`trim`	logical

sep	character: the column separator
port	the TWS port
clientId	either NULL or an integer. Specifying an integer is supported only for backend rib. If NULL, a random id is chosen.
filename	character: pattern for the filenames of the stored files. May include special patterns such as '%id%'.
backend	a lowercase string or NULL. Supported are ibrokers or rib. Default is NULL, which translates into ibrokers.

Details

See IB API documentation <https://interactivebrokers.github.io/>

The package uses functionality of either package **IBrokers** or **rib**, depending on the setting of backend.

Value

A character vector: the names of the file (including the directory) that have been saved.

Author(s)

Enrico Schumann

See Also

[reqHistoricalData](#)

Examples

```
## Not run:
require("IButils")

Symbol <- "ZQ   SEP 16"
Security_Type  <- "FUT"
Exchange <- "ECBOT"
Currency <- "USD"

id <- "zq201609"

download_dir <- "~/Trading/Data/IB_downloads/"

barSize <- "5 mins"
whatToShow <- "MIDPOINT"

start <- structure(1423526400, class = c("POSIXct", "POSIXt"))
end <- start + 86400*20

ib_hist_data(Symbol = Symbol,
             Security_Type = Security_Type,
             Exchange = Exchange,
```

```
               Currency = Currency,
               id = id,
               directory = download_dir,
               barSize = barSize,
               whatToShow = whatToShow,
               start = start,
               end = start+86400*40)

## End(Not run)
```

---

latest_timestamp   *Latest Timestamp*

Description

Get the latest timestamp from the files in a directory.

Usage

```
latest_timestamp(directory, id)
```

Arguments

directory   character

id   character

Details

Get the latest (i.e. largest) timestamp for particular id.

Value

numeric

Author(s)

Enrico Schumann

See Also

ib_hist_data

Examples

```
## TODO
```

---

message_codes   *Error Messages*

Description

A table of message and error codes and their descriptions

Usage

```
data("message_codes")
```

Format

A data frame with 306 observations on the following 3 variables.

Code a character vector

'TWS message' a character vector

'Additional notes' a character vector

Details

Scraped from the official API documentation.

Source

["https://interactivebrokers.github.io/tws-api/message_codes.html"](https://interactivebrokers.github.io/tws-api/message_codes.html)

Examples

```
data(message_codes)
```

---

positions   *Helper Functions for TWS*

Description

Several convenience functions for connecting synchronously to the TWS.

Usage

```
executions(port = 7496, clientId = 1)
order_status(port = 7496, clientId = 1)
positions(port = 7496, clientId = 1)
```

Arguments

```
port              integer
clientId          integer
```

Details

The functions follow the same pattern: connect to the TWS, send requests, receive and process results, and disconnect. Results are typically arranged as data frames.

Warning: these functions are very experimental, and returned results may change.

Value

A `data.frame`. If there are no executions, orders or positions, the functions return `NULL` invisibly.

Author(s)

Enrico Schumann

Examples

```
positions()
```

---

read_flex_report   *Read Flex Report*

Description

Read reports returned by the Flex Web Service.

Usage

```
read_flex_report(file,
                 date.format = "yyyy-MM-dd",
                 time.format = "HH:mm:ss",
                 date.time.separator = ",",
                 ...)
```

Arguments

```
file              character: filename for the downloaded report
date.format       string
time.format       string
date.time.separator
                  string
...               additional settings
```

Details

Reads CSV reports from Flex Queries.

Value

a list of data.frames

Author(s)

Enrico Schumann

References

https://www.interactivebrokers.com/en/software/am/am/reports/flex_web_service_version_
3.htm

See Also

flex_web_service

Examples

```
## Not run:
file <- "~/Downloads/Net_Asset_Value_NAV_in_Base"
ans <- read_flex_report(file)

## End(Not run)
```

Index