Package ‘Rblpapi’

April 7, 2019

Title R Interface to 'Bloomberg'

Version 0.3.10

Date 2019-04-02

Maintainer Dirk Eddelbuettel <edd@debian.org>

Author Whit Armstrong, Dirk Eddelbuettel and John Laing

Imports Rcpp (>= 0.11.0), utils

Suggests fts, xts, zoo, data.table, knitr, RUnit

VignetteBuilder knitr

LazyLoad yes

StagedInstall no

LinkingTo Rcpp, BH

Description An R Interface to 'Bloomberg' is provided via the 'Blp API'.

SystemRequirements A valid Bloomberg installation. The API headers and
dynamic library are downloaded from
<https://github.com/Rblp/blp> during the build step. See
<https://bloomberg.github.io/blpapi-docs/cpp/3.8> as well as
<https://www.bloomberg.com/professional/support/api-library/>
for API documentation. A compiler recent enough for (at least
partial) C++11 support is required; g++-4.6.* or later should
be sufficient and g++-4.9.* or later is preferred.

URL http://dirk.eddelbuettel.com/code/rblpapi.html,
https://github.com/Rblp/Rblpapi

BugReports https://github.com/Rblp/Rblpapi/issues

License file LICENSE

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-04-07 10:32:43 UTC

R topics documented:

`bdh` *Run 'Bloomberg Data History' Queries*

Description

This function uses the Bloomberg API to retrieve 'bdh' (Bloomberg Data History) queries

Usage

```
bdh(securities, fields, start.date, end.date = NULL,
  include.non.trading.days = FALSE, options = NULL, overrides = NULL,
  verbose = FALSE, identity = NULL, con = defaultConnection(),
  int.as.double = getOption("blpIntAsDouble", FALSE))
```

Arguments

`securities`	A character vector with security symbols in Bloomberg notation.
`fields`	A character vector with Bloomberg query fields.
`start.date`	A Date variable with the query start date.
`end.date`	An optional Date variable with the query end date; if omitted the most recent available date is used.
`include.non.trading.days`	An optional logical variable indicating whether non-trading days should be included.

`options`	An optional named character vector with option values. Each field must have both a name (designating the option being set) as well as a value.
`overrides`	An optional named character vector with override values. Each field must have both a name (designating the override being set) as well as a value.
`verbose`	A boolean indicating whether verbose operation is desired, defaults to 'FALSE'
`identity`	An optional identity object.
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.
`int.as.double`	A boolean indicating whether integer fields should be retrieved as doubles instead. This option is a workaround for very large values which would overflow int32. Defaults to 'FALSE'

Value

A list with as a many entries as there are entries in `securities`; each list contains a data.frame with one row per observations and as many columns as entries in `fields`. If the list is of length one, it is collapsed into a single data frame. Note that the order of securities returned is determined by the backend and may be different from the order of securities in the `securities` field.

Author(s)

Whit Armstrong and Dirk Eddelbuettel

See Also

For historical futures series, see 'DOCS #2072138 <GO>' on the Bloomberg terminal about selecting different rolling conventions.

Examples

```
## Not run:
  bdh("SPY US Equity", c("PX_LAST", "VOLUME"), start.date=Sys.Date()-31)

  ## example for an options field: request monthly data; see section A.2.4 of
 ##  http://www.bloomberglabs.com/content/uploads/sites/2/2014/07/blpapi-developers-guide-2.54.pdf
  ## for more
  opt <- c("periodicitySelection"="MONTHLY")
  bdh("SPY US Equity", c("PX_LAST", "VOLUME"),
      start.date=Sys.Date()-31*6, options=opt)

  ## example for non-date start
  bdh("SPY US Equity", c("PX_LAST", "VOLUME"),
      start.date="-6CM", options=opt)

  ## example for options and overrides
  opt <- c("periodicitySelection" = "QUARTERLY")
  ovrd <- c("BEST_FPERIOD_OVERRIDE"="1GQ")
  bdh("IBM US Equity", "BEST_SALES", start.date=Sys.Date()-365.25*4,
      options=opt, overrides=ovrd)
 ## example for returnRelativeDate option
 opt <- c(periodicitySelection="YEARLY", periodicityAdjustment="FISCAL", returnRelativeDate=TRUE)
 bdh("GLB ID Equity", "CUR_MKT_CAP", as.Date("1997-12-31"), as.Date("2017-12-31"), options=opt)

## End(Not run)
```

bdp *Run 'Bloomberg Data Point' Queries*

Description

This function uses the Bloomberg API to retrieve 'bdp' (Bloomberg Data Point) queries

Usage

```
bdp(securities, fields, options = NULL, overrides = NULL, verbose = FALSE,
  identity = NULL, con = defaultConnection())
```

Arguments

`securities`	A character vector with security symbols in Bloomberg notation.
`fields`	A character vector with Bloomberg query fields.
`options`	An optional named character vector with option values. Each field must have both a name (designating the option being set) as well as a value.
`overrides`	An optional named character vector with override values. Each field must have both a name (designating the override being set) as well as a value.
`verbose`	A boolean indicating whether verbose operation is desired, defaults to 'FALSE'
`identity`	An optional identity object.
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

A data frame with as a many rows as entries in `securities` and columns as entries in `fields`.

Author(s)

Whit Armstrong and Dirk Eddelbuettel

Examples

```
## Not run:
  bdp(c("ESA Index", "SPY US Equity"), c("PX_LAST", "VOLUME"))

  ##  using overrides (cf https://github.com/Rblp/Rblpapi/issues/67)
  bdp("EN00 Index", "MLI_OAS", overrides=c(MLI_DATE="20150831"))

  ##  another override example (cf http://stackoverflow.com/a/39373019/143305)
  ovrd <- c("CALC_INTERVAL"="10Y", "MARKET_DATA_OVERRIDE"="PE_RATIO")
  bdp("SPX Index", "INTERVAL_AVG", overrides=ovrd)

## End(Not run)
```

`bds` *Run 'Bloomberg Data Set' Queries*

Description

This function uses the Bloomberg API to retrieve 'bds' (Bloomberg Data Set) queries

Usage

```
bds(security, field, options = NULL, overrides = NULL, verbose = FALSE,
  identity = NULL, con = defaultConnection())
```

Arguments

`security`	A character value with a single security symbol in Bloomberg notation.
`field`	A character string with a single Bloomberg query field.
`options`	An optional named character vector with option values. Each field must have both a name (designating the option being set) as well as a value.
`overrides`	An optional named character vector with override values. Each field must have both a name (designating the override being set) as well as a value.
`verbose`	A boolean indicating whether verbose operation is desired, defaults to 'FALSE'
`identity`	An optional identity object.
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

A list with as many entries as there are entries in `securities`; each list contains a data.frame with one row per observations and as many columns as entries in `fields`. If the list is of length one, it is collapsed into a single data frame.

Author(s)

Whit Armstrong and Dirk Eddelbuettel

Examples

```
## Not run:
  ## simple query
  bds("GOOG US Equity", "TOP_20_HOLDERS_PUBLIC_FILINGS")
  ## example of using overrides
  overrd <- c("START_DT"="20150101", "END_DT"="20160101")
  bds("CPI YOY Index","ECO_RELEASE_DT_LIST", overrides = overrd)

## End(Not run)
```

`beqs` *Run 'Bloomberg EQS' Queries*

Description

This function uses the Bloomberg API to retrieve 'beqs' (Bloomberg EQS Data) queries

Usage

```
beqs(screenName, screenType = "GLOBAL", language = "", group = "",
  date = NULL, verbose = FALSE, con = defaultConnection())
```

Arguments

`screenName`	A character string with the name of the screen to execute. It can be a user defined EQS screen or one of the Bloomberg Example screens on EQS
`screenType`	A character string of value PRIVATE or GLOBAL Use PRIVATE for user-defined EQS screen. Use GLOBAL for Bloomberg EQS screen.
`language`	An optional character string with the EQS language
`group`	An optional character string with the Screen folder name as defined in EQS
`date`	An optional Date object with the 'point in time' date of the screen to execute.
`verbose`	A boolean indicating whether verbose operation is desired, defaults to 'FALSE'.
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

A data frame object with the date in the first column and and the requested EQS data in the remaining columns.

Author(s)

Rademeyer Vermaak and Dirk Eddelbuettel

Examples

```
## Not run:
head(beqs("Global Oil Companies YTD Return"), 20)
head(beqs("Global Oil Companies YTD Return", "GLOBAL"), 20)
head(beqs("Global Oil Companies YTD Return", "GLOBAL", "GERMAN"), 20)
head(beqs("Global Oil Companies YTD Return", "GLOBAL", "GERMAN", "GENERAL"), 20)
head(beqs("Global Oil Companies YTD Return", "GLOBAL", "ENGLISH", "GENERAL",
          as.Date("2015-09-30")), 20)

## End(Not run)
```

`blpAuthenticate` *Authenticate Bloomberg API access*

Description

This function authenticates against the the Bloomberg API

Usage

```
blpAuthenticate(uuid, host = "localhost", ip.address,
  con = defaultConnection())
```

Arguments

`uuid`	A character variable with a unique user id token. If this is missing the function will attempt to connect to bpipe using the connection. It is assumed that an app_name was set. See blpConnect() for app_name information
`host`	A character variable with a hostname, defaults to 'localhost'
`ip.address`	An optional character variable with an IP address
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function. This is the only required argument to authenticate a bpipe connection with a appName. `defaultConnection`.

Value

The returned object should be passed to subsequent data calls via bdp(), bds(), etc.

Author(s)

Whit Armstrong and Dirk Eddelbuettel

Examples

```
## Not run:
blpConnect(host=blpHost, port=blpPort)
blpid <- blpAuthenticate(uuid=blpUUID, ip=blpIP_address)
bdp("IBM US Equity", "NAME", identity=blpid)

## End(Not run)
```

`blpConnect` *Establish connection to Bloomberg service*

Description

This function connects to the Bloomberg API

Usage

```
blpConnect(host = getOption("blpHost", "localhost"),
  port = getOption("blpPort", 8194L), default = TRUE,
  appName = getOption("blpAppName", NULL))
```

Arguments

`host`	A character option with either a machine name that is resolvable by DNS, or an IP address. Defaults to 'localhost'.
`port`	An integer variable with the connection port. Default to `8194L`.
`default`	A logical indicating whether this connection should be saved as the default, as opposed to returned to the user. Default to `TRUE`.
`appName`	the name of an application that is authorized to connect to bpipe. If this is NULL Rblpapi connects to the Bloomberg API but cannot authenticate with an app name. This requires the user to authenticate with a user uuid.

Details

For both `host` and `port` argument, default values can also be specified via `options` using, respectively, the named entries `blpHost` and `blpConnect`.

If an additional option `blpAutoConnect` is set to 'TRUE', a connection is established in the `.onAttach()` function and stored in the package environment. This effectively frees users from having to explicitly create such an object.

Value

In the `default=TRUE` case nothing is returned, and this connection is automatically used for all future calls which omit the con argument. Otherwise a connection object is returned which is required by all the accessor functions in the package.

Author(s)

Whit Armstrong and Dirk Eddelbuettel

See Also

Many SAPI and bPipe connections require authentication via `blpAuthenticate` after `blpConnect`.

Examples

```
## Not run:
  con <- blpConnect()   # adjust as needed

## End(Not run)
```

`blpDisconnect` *Placeholder function for disconnection from Bloomberg*

Description

This function provides an empty stub and does not really disconnect.

Usage

```
blpDisconnect(con)
```

Arguments

`con`	A connection object

Details

The internal connection object is managed via finalizers. As such the connection is only destroyed, and the connection removed, once the packaged is unloaded or the session is otherwise terminated.

Value

A boolean is returned; it simply states whether the connection object was small or large relative to an arbitrary cutoff of 1000 bytes.

Author(s)

Whit Armstrong and Dirk Eddelbuettel

Examples

```
## Not run:
  blpDisconnect(con)

## End(Not run)
```

`bsrch` *Run 'Bloomberg SRCH' Queries*

Description

This function uses the Bloomberg API to retrieve 'bsrcb' (Bloomberg SRCH Data) queries

Usage

```
bsrch(domain, limit = "", verbose = FALSE, con = defaultConnection())
```

Arguments

`domain`	A character string with the name of the domain to execute. It can be a user defined SRCH screen, commodity screen or one of the variety of Bloomberg examples. All domains are in the format <domain>:<search_name>.
`limit`	A character string containing a value by which to limit the search length – NOT YET IMPLEMENTED
`verbose`	A boolean indicating whether verbose operation is desired, defaults to 'FALSE'.
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

A data frame object with the requested SRCH data.

Author(s)

Morgan Williams and Dirk Eddelbuettel

Examples

```
## Not run:
head(bsrch("COMDTY:NGFLOW"), 20)
head(bsrch("COMDTY:VESSEL"), 20)

## End(Not run)
```

`defaultConnection` *Return the default connection object*

Description

This function return the default connection object from the package environment. If no default connection has been established yet, an error message is shown,

Usage

```
defaultConnection()
```

Details

For the connection object, the required arguments `host` and `port` argument can be set via `options`. In addition, if an additional option `blpAutoConnect` is set to 'TRUE', a connection is established in the `.onAttach()` function and stored in the package environment. This effectively frees users from having to explicitly create such an object. Of course, the user can also call `blpConnect` explicitly and store the connection object. This helper function looks up the stored connection object and returns it. In case no connection has been established, and error message is shown.

Author(s)

Whit Armstrong and Dirk Eddelbuettel

Examples

```
## Not run:
  con <- defaultConnection()

## End(Not run)
```

`fieldInfo` *Run 'Bloomberg Field Data' Queries*

Description

This function uses the Bloomberg API to retrieve fieldInfo

Usage

```
fieldInfo(fields, con = defaultConnection())
```

Arguments

`fields`	A character vector with Bloomberg query fields.
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

A data frame with as a many rows as entries in `fields`

Author(s)

Whit Armstrong and Dirk Eddelbuettel

Examples

```
## Not run:
  fieldInfo(c("PX_LAST", "VOLUME"))

## End(Not run)
```

`fieldSearch` *Search for matching data fields*

Description

This function searches for matching Bloomberg data fields given a search term.

Usage

```
fieldSearch(searchterm, excludeterm = "Static", con = defaultConnection())
```

Arguments

`searchterm`	A string with the term to search for
`excludeterm`	A string with an expression for matches to excludes, defaults to “Static”
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

A `data.frame` with three columns of the id, mnenemonic and description of each match.

Author(s)

Dirk Eddelbuettel

Examples

```
## Not run:
  head(fieldSearch("vwap"), 20)

## End(Not run)
```

---

getBars *Get Open/High/Low/Close/Volume Bars from Bloomberg*

Description

This function uses the Bloomberg API to retrieve bars for the requested security.

Usage

```
getBars(security, eventType = "TRADE", barInterval = 60,
  startTime = Sys.time() - 60 * 60 * 6, endTime = Sys.time(),
  options = NULL, verbose = FALSE, returnAs = getOption("blpType",
  "matrix"), tz = Sys.getenv("TZ", unset = "UTC"),
  con = defaultConnection())
```

Arguments

`security`	A character variable describing a valid security ticker
`eventType`	A character variable describing an event type; default is 'TRADE'
`barInterval`	A integer denoting the number of minutes for each bar
`startTime`	A Datetime object with the start time, defaults to one hour before current time
`endTime`	A Datetime object with the end time, defaults to current time
`options`	An optional named character vector with option values. Each field must have both a name (designating the option being set) as well as a value.
`verbose`	A boolean indicating whether verbose operation is desired, defaults to 'FALSE'
`returnAs`	A character variable describing the type of return object; currently supported are 'matrix' (also the default), 'fts', 'xts', 'zoo' and 'data.table'
`tz`	A character variable with the desired local timezone, defaulting to the value 'TZ' environment variable, and 'UTC' if unset
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

A numeric matrix with elements 'time' (as a 'POSIXct' object), 'open', 'high', 'low', 'close', 'numEvents', 'volume', 'value' or an object of the type selected in `returnAs`. Note that the 'time' value is adjusted: Bloomberg returns the *opening* time of the bar interval, whereas financial studies typically refer to the most recent timestamp. Therefore, if one wants the timestamp associated with the end of the bar interval one should add the length of the bar interval to time value returned from Bloomberg to obtain the time at the end of the interval.

Author(s)

Dirk Eddelbuettel

Examples

```
## Not run:
  getBars("ES1 Index")

## End(Not run)
```

---

getHeaderVersion — *Get Bloomberg library header version*

Description

This function retrieves the version of Bloomberg API headers.

Usage

```
getHeaderVersion()
```

Value

A string with four dot-separated values for major, minor, pathch and build version of the headers.

Author(s)

Dirk Eddelbuettel

See Also

`getRuntimeVersion`

Examples

```
## Not run:
   getHeaderVersion()

## End(Not run)
```

`getMultipleTicks` *Get Multiple Ticks from Bloomberg*

Description

This function uses the Bloomberg API to retrieve multiple ticks for the requested security.

Usage

```
getMultipleTicks(security, eventType = c("TRADE", "BID", "ASK"),
  startTime = Sys.time() - 60 * 60, endTime = Sys.time(), verbose = FALSE,
  returnAs = getOption("blpType", "data.frame"), tz = Sys.getenv("TZ", unset
  = "UTC"), con = defaultConnection())
```

Arguments

`security`	A character variable describing a valid security ticker
`eventType`	A character vector describing event types, default is `c("TRADE", "BID", "ASK")`
`startTime`	A Datetime object with the start time, defaults to one hour before current time
`endTime`	A Datetime object with the end time, defaults to current time
`verbose`	A boolean indicating whether verbose operation is desired, defaults to 'FALSE'
`returnAs`	A character variable describing the type of return object; currently supported are 'data.frame' (also the default) and 'data.table'
`tz`	A character variable with the desired local timezone, defaulting to the value 'TZ' environment variable, and 'UTC' if unset
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

A numeric matrix with elements 'time', (as a 'POSIXct' object), 'values' and 'sizes', or an object of the type selected in `returnAs`.

Author(s)

Dirk Eddelbuettel

getPortfolio *Run 'Portfolio Data' Queries*

Description

This function uses the Bloomberg API to retrieve 'portfolio' queries

Usage

```
getPortfolio(security, field, options = NULL, overrides = NULL,
  verbose = FALSE, identity = NULL, con = defaultConnection())
```

Arguments

`security`	A character value with a single security symbol in Bloomberg notation.
`field`	A character string with a single Bloomberg query field.
`options`	An optional named character vector with option values. Each field must have both a name (designating the option being set) as well as a value.
`overrides`	An optional named character vector with override values. Each field must have both a name (designating the override being set) as well as a value.
`verbose`	A boolean indicating whether verbose operation is desired, defaults to 'FALSE'
`identity`	An optional identity object.
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

A list with as many entries as there are entries in `securities`; each list contains a data.frame with one row per observations and as many columns as entries in `fields`. If the list is of length one, it is collapsed into a single data frame.

Author(s)

John Laing

`getRuntimeVersion` *Get Bloomberg library run-time version*

Description

This function retrieves the version of Bloomberg API run-time.

Usage

```
getRuntimeVersion()
```

Value

A string with four dot-separated values for major, minor, pathch and build version of the run-time library.

Author(s)

Dirk Eddelbuettel

See Also

`getHeaderVersion`

Examples

```
## Not run:
   getRuntimeVersion()

## End(Not run)
```

`getTicks` *Get Ticks from Bloomberg*

Description

This function uses the Bloomberg API to retrieve ticks for the requested security.

Usage

```
getTicks(security, eventType = "TRADE", startTime = Sys.time() - 60 * 60,
  endTime = Sys.time(), verbose = FALSE, returnAs = getOption("blpType",
  "data.frame"), tz = Sys.getenv("TZ", unset = "UTC"),
  con = defaultConnection())
```

Arguments

`security`	A character variable describing a valid security ticker
`eventType`	A character variable describing an event, default is 'TRADE'.
`startTime`	A Datetime object with the start time, defaults to one hour before current time
`endTime`	A Datetime object with the end time, defaults to current time
`verbose`	A boolean indicating whether verbose operation is desired, defaults to 'FALSE'
`returnAs`	A character variable describing the type of return object; currently supported are 'data.frame' (also the default), 'data.table', 'fts', 'xts' and 'zoo'
`tz`	A character variable with the desired local timezone, defaulting to the value 'TZ' environment variable, and 'UTC' if unset
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

Depending on the value of 'returnAs', either a 'data.frame' or 'data.table' object also containing non-numerical information such as condition codes, or a time-indexed container of type 'fts', 'xts' and 'zoo' with a numeric matrix containing only 'value' and 'size'.

Note

Bloomberg returns condition codes as well, and may return *multiple observations for the same trade*. Eg for ES we can get 'AS' or 'AB' for aggressor buy or sell, 'OR' for an order participating in the matching event, or a 'TSUM' trade summary. Note that this implies double-counting. There may be an option for this in the API.

The Bloomberg API allows to retrieve up to 140 days of intra-day history relative to the current date.

Author(s)

Dirk Eddelbuettel

Examples

```
## Not run:
  res <- getTicks("ES1 Index")
  str(res)
  head(res, 20)
  res <- getTicks("ES1 Index", returnAs="data.table")
  str(res)
  head(res, 20)

## End(Not run)
```

`lookupSecurity` *Look up symbol from Bloomberg*

Description

This function uses the Bloomberg API to look up tickers and descriptions given the name of a company.

Usage

```
lookupSecurity(query, yellowkey = c("none", "cmdt", "eqty", "muni", "prfd",
  "clnt", "mmkt", "govt", "corp", "indx", "curr", "mtge"),
  language = c("none", "english", "kanji", "french", "german", "spanish",
  "portuguese", "italian", "chinese_trad", "korean", "chinese_simp", "none_1",
  "none_2", "none_3", "none_4", "none_5", "russian"), maxResults = 20,
  verbose = FALSE, con = defaultConnection())
```

Arguments

`query`	A character variable describing the name of the company; for certain queries a trailing space may help.
`yellowkey`	A character variable that restricts the asset classes to search in; one of “none”, “cmdt”, “eqty”, “muni”, “prfd”, “clnt”, “mmkt”, “govt”, “corp”, “indx”, “curr”, “mtge”.
`language`	A character variable denoting the language that the results will be translated in; one of “NONE”, “english”, “kanji”, “french”, “german”, “spanish”, “portuguese”, “italian”, “chinese_trad”, “korean”, “chinese_simp”, “none_1”, “none_2”, “none_3”, “none_4”, “none_5”, “russian”
`maxResults`	A integer variable containing a value by which to limit the search length
`verbose`	A boolean indicating whether verbose operation is desired, defaults to ‘FALSE’
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Value

A data.frame with two columns of the ticker and description of each match.

Author(s)

Kevin Jin and Dirk Eddelbuettel

Examples

```
## Not run:
  lookupSecurity("IBM")
  lookupSecurity("IBM", maxResuls=1000)    # appears to be capped at 1000
  lookupSecurity("IBM", "mtge")
  lookupSecurity("IBM ", "mtge")           # trailing space affects query

  ## modify the symbol column (cf issue ticket 215 at GitHub)
  res <- lookupSecurity("IBM")
  res[, "symbol"] <- sub(pattern="^(.+)<(.)(.+)>$", "\\1 \\U\\2\\E\\3",
                         perl=TRUE, res[, "security"])
  res

## End(Not run)
```

subscribe   *Subscribe to streaming market data*

Description

This function uses the Bloomberg API to stream live market data

Usage

```
subscribe(securities, fields, fun, options = NULL, identity = NULL,
  con = defaultConnection())
```

Arguments

`securities`	A character vector with security symbols in Bloomberg notation.
`fields`	A character vector with Bloomberg query fields.
`fun`	An R function to be called on the subscription data.
`options`	An optional named character vector with option values. Each field must have both a name (designating the option being set) as well as a value.
`identity`	An optional identity object.
`con`	A connection object as created by a `blpConnect` call, and retrieved via the internal function `defaultConnection`.

Details

The subscribe function allows one to subscribe to streaming market quotes.

Full detials of the subscription string can be found in the header file blpapi_subscriptionlist.h.

Value

This function always returns NULL.

Author(s)

Whit Armstrong

References

http://bloomberg.github.io/blpapi-docs/cpp/3.8

Examples

```
## Not run:
  subscribe(securities=c("TYZ5 Comdty","/cusip/912810RE0@BGN"),
            fields=c("LAST_PRICE","BID","ASK"),
            fun=function(x) print(str(x$data)))

## End(Not run)
```

Index