# Package 'rRofex'

July 31, 2020

**Type** Package

**Title** Interface to 'Matba Rofex' Trading API

**Version** 2.0.3

**Description** Execute API calls to the 'Matba Rofex' <https://apihub.primary.com.ar> trading platform. Functionality includes accessing account data and current holdings, retrieving investment quotes, placing and canceling orders, and getting reference data for instruments.

**Depends** R (>= 3.1.0)

**License** MIT + file LICENSE

**URL** https://matbarofex.github.io/rRofex, https://github.com/matbarofex/rRofex

**BugReports** https://github.com/matbarofex/rRofex/issues

**Encoding** UTF-8

**LazyData** true

**Imports** dplyr (>= 1.0.0),
httr,
jsonlite,
magrittr,
tibble (>= 3.0.0),
tidyr,
rlang,
purrr,
glue,
methods,
websocket,
later,
lifecycle

**RoxygenNote** 7.1.1

**Collate** 'attach.R'
's4_object.R'
'functions.R'
'functions_helpers.R'
'functions_websocket.R'
'globals.R'
'rRofex.R'

**RdMacros** lifecycle

**Suggests** rmarkdown

# R topics documented:



---

`rRofex-package` *rRofex: Interface to 'Matba Rofex' Trading API*

---

## Description

Execute API calls to the 'Matba Rofex' <https://apihub.primary.com.ar> trading platform. Functionality includes accessing account data and current holdings, retrieving investment quotes, placing and canceling orders, and getting reference data for instruments.

## Author(s)

**Maintainer**: Augusto Hassel `<mpi-augusto@primary.com.ar>`

Other contributors:

- Juan Francisco Gomez [contributor]
- Matba Rofex [copyright holder]

## See Also

Useful links:

- https://matbarofex.github.io/rRofex
- https://github.com/matbarofex/rRofex
- Report bugs at https://github.com/matbarofex/rRofex/issues

---

`.validate_fecha` *Helper: Date validation*

---

## Description

**Questioning** Validate date

## Usage

```
.validate_fecha(date)
```

## Arguments

| | |
|---|---|
| `date` | Date |

## Value

TRUE if date has a correct format.

---

`agent` *See Agent*

---

## Description

Shows information about the agent set with `trading_login`

## Usage

```
agent(x)

## S4 method for signature 'rRofexConnection'
agent(x)
```

## Arguments

| | |
|---|---|
| `x` | S4 Class. rRofexConnection object |

## Value

Scalar with the 'agent'

---

`base_url` *See Base URL*

---

## Description

Shows information about the 'base url' where the user has been connected with `trading_login`

## Usage

```
base_url(x)

## S4 method for signature 'rRofexConnection'
base_url(x)
```

## Arguments

| | |
|---|---|
| `x` | S4 Class. rRofexConnection object |

## Value

Scalar with the 'base url'

---

`login_date_time` *See Log-in Timestamp*

---

## Description

Shows information about the connection timestamp when calling `trading_login`

## Usage

```
login_date_time(x)

## S4 method for signature 'rRofexConnection'
login_date_time(x)
```

## Arguments

| | |
|---|---|
| `x` | S4 Class. rRofexConnection object |

## Value

Scalar with the 'log-in timestamp'

`rRofexConnection-class` *Connection Class: rRofexConnection*

## Description

**Stable** Creates an rRofex connection object that contains a summary from the `trading_login` function.

## Value

S4 rRofexConnection object.

## Slots

`token` character. Obtained from login method

`base_url` character. Connected environment

`login_date_time` character. Log-in date time. The connection object is only valid for a day.

`agent` character. User Agent to pass to the API. Format: 'rRofex-<environment>-user_name'

`user_name` character. User Name.

`rRofex_connection` *Create rRofex Connection Object*

## Description

**Stable** `rRofex_connection` creates a New Connection Object.

## Usage

```
rRofex_connection(token, base_url, user_name)
```

## Arguments

| | |
|---|---|
| `token` | String. **Mandatory** Obtained with `trading_login` |
| `base_url` | String. **Mandatory** URL given by `trading_login` or known by the client. |
| `user_name` | character. User Name |

## Value

S4 rRofexConnection object.

A valid rRofexConeccion object.

### Note

You can use accessors to get information about the Object by using:

- `token(conn)`
- `base_url(conn)`
- `login_date_time(conn)`
- `agent(conn)`
- `user_name(conn)`

---

## `show,rRofexConnection-method` *Show summary of rRofexConnection*

---

### Description

Shows a summary about the rRofexConnection object created with `trading_login`

### Usage

```
## S4 method for signature 'rRofexConnection'
show(object)
```

### Arguments

| | |
|---|---|
| `object` | S4 Class. rRofexConnection object |

### Value

Summary text with User, Environment and Timestamp

---

## `token` *See Token*

---

### Description

Shows information about the token thas has been generated with `trading_login`

### Usage

```
token(x)

## S4 method for signature 'rRofexConnection'
token(x)
```

### Arguments

| | |
|---|---|
| `x` | S4 Class. rRofexConnection object |

### Value

Scalar with token

## trading_account *Account Information*

### Description

**Maturing** Access information about the trading account.

### Usage

```
trading_account(connection, account, detailed = FALSE)
```

### Arguments

| | |
|---|---|
| connection | S4. **Mandatory** Formal rRofexConnection class object |
| account | String. **Mandatory** Account Number |
| detailed | Logical. Expanded information. |

### Value

If correct, it will load a tibble.

### See Also

Other account functions: `trading_account_report()`

## trading_account_report *Account Report*

### Description

**Maturing** Access report about your trading account.

### Usage

```
trading_account_report(connection, account)
```

### Arguments

| | |
|---|---|
| connection | S4. **Mandatory** Formal rRofexConnection class object |
| account | String. **Mandatory** Account Number |

### Value

If correct, it will load a tibble.

### Note

To access nested data is strongly recommended the use of ‘pluck‘.

## See Also

Other account functions: `trading_account()`

## Examples

```
## Not run:
data %>% pluck("detailedAccountReports", 1, "availableToOperate", 1, "cash")

## End(Not run)
```

---

`trading_cancel_order` *Cancel Order Sent to the Market*

---

## Description

**Maturing** The method `trading_cancel_order` should be use to cancel orders that are open on the market.

## Usage

```
trading_cancel_order(connection, id, proprietary)
```

## Arguments

| | |
|---|---|
| `connection` | S4. **Mandatory** Formal rRofexConnection class object |
| `id` | String. **Mandatory** clOrdId given by the `trading_orders` method. |
| `proprietary` | String. **Mandatory** ID given by the `trading_orders` method.<br>• **PBCP** |

## Value

If correct, it will load a tibble.

## See Also

Other order placements functions: `trading_new_order()`

## trading_currencies *Currencies*

### Description

**Stable** Access currencies prices.

### Usage

```
trading_currencies(connection)
```

### Arguments

| | |
|---|---|
| connection | S4. **Mandatory** Formal rRofexConnection class object |

### Value

If correct, it will load a data frame.

### See Also

Other market data functions: `trading_mdh()`, `trading_md()`

## trading_instruments *List of Instruments*

### Description

**Stable** Method to list segments and instruments currently available through the Trading API.

### Usage

```
trading_instruments(
  connection,
  request,
  sec_detailed = FALSE,
  market_id = "ROFX",
  segment_id,
  cfi_code,
  sec_type
)
```

### Arguments

| | |
|---|---|
| connection | S4. **Mandatory** Formal rRofexConnection class object |
| request | String. **Mandatory** The type of request that you are making:<br>• **segments**: List available market segments<br>• **securities**: List available instruments listed on Matba Rofex. *Depends on 'sec_detailed'*. |

- **by_segment**: List available instruments searching by market segment. *Depends on 'market_id' and 'segment_id'*
- **by_cfi_code**: List available instruments searching by CFI Code. *Depends on 'cfi_code'*
- **by_type**: List available instruments searching by Instrument Type. See section Instrument Types. *Depends on 'sec_detailed' and 'sec_type'*.

`sec_detailed` Logical. Optional for request='securities'. Brings additional information like segment, price, minimal/maximal trading quantity, settlement date, etc.

`market_id` String. Needed for request='by_segment'. Market ID.

- **ROFX**: Matba Rofex

`segment_id` String. Needed for request='by_segment'. Market Segment ID.

- **DDF**: Financial Derivatives
- **DDA**: Agricultural Derivatives
- **DUAL**: Other Derivatives
- **MERV**: S&P Merval

`cfi_code` String. Needed for request='by_cfi_code'. CFI Code. See https://www.quotemedia.com/apifeeds/cfi_code

`sec_type` String. Needed for request='by_type'.

- **E**: Equities
- **D**: Debt
- **C**: Collective Investment Vehicles
- **R**: Entitlements (Rights)
- **O**: Listed Options
- **F**: Futures
- **T**: Referential Instruments
- **M**: Others

## Value

If correct, it will load a tibble data frame

## See Also

Other reference data functions: `trading_instruments_fronts()`

---

`trading_instruments_fronts` *Front Month of Futures*

---

## Description

**Stable** List all front month contracts for futures.

## Usage

```
trading_instruments_fronts(connection)
```

### Arguments

| | |
|---|---|
| connection | S4. **Mandatory** Formal rRofexConnection class object |

### Value

If correct, it will load a tibble data frame

### See Also

Other reference data functions: `trading_instruments()`

---

## trading_login *API Log-in*

---

### Description

**Stable** Function that it is use to log-in into Primary trading API

### Usage

```
trading_login(username, password, base_url)
```

### Arguments

| | |
|---|---|
| username | String. User Name |
| password | String. Password |
| base_url | String. Which environment are you going to connect:<br>• reMarkets: 'https://api.remarkets.primary.com.ar'<br>• production: 'https://api.primary.com.ar'<br>• xOMS: 'https://api.<BROKER>.xoms.com.ar' |

### Value

S4 rRofexConnection object.

### Note

- reMarkets: Testing environment. For credentials go to https://remarkets.primary.ventures
- production: Production environment. For credentials send an email to <mpi@primary.com.ar>
- xOMS: Ask your broker about it.

Accessors: You can use accessors to get information about the Object by using:

- `token(conn)`
- `base_url(conn)`
- `login_date_time(conn)`
- `agent(conn)`
- `user_name(conn)`

## Examples

```
## Not run:
conn <- trading_login(
username = "pepe",
password = "pepino",
base_url = "https://api.remarkets.primary.com.ar"
)

## End(Not run)
```

---

`trading_lookup` *Lookup Order Status*

---

## Description

**Stable** The method `trading_lookup` is used to check the status of an order.

## Usage

```
trading_lookup(connection, lookup_type, id, proprietary)
```

## Arguments

| | |
|---|---|
| `connection` | S4. **Mandatory** Formal rRofexConnection class object |
| `lookup_type` | String. **Mandatory**. Look-up by:<br>• **COID** - Client Order ID.<br>• **OID** - Order ID. |
| `id` | String. **Mandatory**. ID given by the `trading_orders` method. Depends on 'lookup_type'. |
| `proprietary` | String. ID given by the `trading_orders` method. Only for 'lookup_type=COID' In most cases:<br>• **PBCP** |

## Value

If correct, it will load a tibble.

## See Also

Other order management functions: `trading_orders()`

`trading_md` *Market Data Real Time*

## Description

**Stable** This method brings Market Data in Real Time.

## Usage

```
trading_md(
  connection,
  symbol,
  entries = c("BI", "OF", "LA", "OP", "CL", "SE", "OI", "HI", "LO", "TV", "IV", "EV",
    "NV", "TC"),
  depth = 1L,
  market_id = "ROFX",
  tidy = TRUE
)
```

## Arguments

| | |
|---|---|
| `connection` | S4. **Mandatory**. Formal rRofexConnection class object |
| `symbol` | String. **Mandatory**. Use `trading_instruments` to see which symbols are available. |
| `entries` | Vector of Strings. When nothing is set, then **all entries are the default**. It contains the information to be queried:<br>• **BI** - Bid.<br>• **OF** - Offer.<br>• **LA** - Last Available Price.<br>• **OP** - Open Price.<br>• **CL** - Close Price.<br>• **SE** - Settlement Price.<br>• **OI** - Open Interest.<br>• **HI** - Trading Session High Price<br>• **LO** - Trading Session Low Price<br>• **TV** - Trading Volume<br>• **IV** - Index Value<br>• **EV** - Trading Effective Volume<br>• **NV** - Nominal Volume<br>• **TC** - Trade Count |
| `depth` | Integer. Depth of the book. Default is **1L**. |
| `market_id` | String. Market to which you are going to connect. Default is **ROFX**.<br>• **ROFX** - Matba Rofex |
| `tidy` | Logical. Data arranged on a tidy format. Default is **TRUE**. |

## Value

If correct, it will load a tibble data frame

## See Also

Other market data functions: trading_currencies(), trading_mdh()

## Examples

```
# If you want to query many products at once,
# I recommend you to use "purrr::map" family like this:

## Not run:
purrr::map_df(
list('MERV - XMEV - GGAL - 48hs','MERV - XMEV - BYMA - 48hs'),
~trading_md(connection = conn, symbol = .x, entries = c("LA","OP", "NV"), tidy = T)
)

## End(Not run)
```

---

# trading_mdh *Historical Market Data*

---

## Description

**Stable** Access Historical Trades for a given instrument.

## Usage

```
trading_mdh(
  connection,
  market_id = "ROFX",
  symbol,
  date,
  date_from,
  date_to,
  tidy = TRUE
)
```

## Arguments

| | |
|---|---|
| connection | S4. **Mandatory** Formal rRofexConnection class object |
| market_id | String. Market to which we are going to connect.<br>• **ROFX** - Matba Rofex.<br>• **MERV** - S&P Merval. |
| symbol | String. Use trading_instruments to see which symbols are available. |
| date | String. Date to be queried. With format '%Y-%m-%d'. |
| date_from | String. Used together with 'date_to'. |
| date_to | String. Userd together with 'date_from'. |
| tidy | Logical. Data arranged on a tidy format. |

## Value

If correct, it will load a data frame.

## See Also

Other market data functions: `trading_currencies()`, `trading_md()`

---

`trading_new_order` *Send Order to the Market*

---

## Description

**Maturing** The method `trading_new_order` is use to send orders.

## Usage

```
trading_new_order(
  connection,
  account,
  symbol,
  side,
  quantity,
  price,
  order_type = "Limit",
  time_in_force = "Day",
  iceberg = FALSE,
  expire_date = NULL,
  display_quantity = NULL,
  cancel_previous = FALSE
)
```

## Arguments

| | |
|---|---|
| `connection` | S4. **Mandatory** Formal rRofexConnection class object |
| `account` | String. **Mandatory** Account Number |
| `symbol` | String. Use `trading_instruments` to see which symbols are available. |
| `side` | String. **Mandatory** Either:<br>• **Buy**<br>• **Sell** |
| `quantity` | Numeric. **Mandatory** Quantity of the order. |
| `price` | Numeric. **Mandatory** Price of the order. |
| `order_type` | String. Type of order.<br>• **Limit** - Default. Limit order sets the maximum or minimum price at which you are willing to buy or sell. |
| `time_in_force` | String. Specifies how long the order remains in effect. Absence of this field is interpreted as 'Day':<br>• **Day** - Day or session. |

- **IOC** - Immediate or Cancel.
- **FOK** - Fill or Kill.
- **GTD** - Good Till Date.

`iceberg` Logical. If TRUE, then the order is 'iceberg'. FALSE as default.

`expire_date` String. **Only for GDT orders**. Maturity date of the order, With format '%Y-%m-%d'.

`display_quantity` Numeric. **Only for Iceberg orders**. Indicate the disclosed quantity for the 'iceberg' order.

`cancel_previous` Logigal. Optional parameter only valid for Matba Rofex instruments. By default it's FALSE.

## Value

If correct, it will load a tibble.

## See Also

Other order placements functions: `trading_cancel_order()`

---

`trading_orders` *View Orders*

---

## Description

**Stable** The method `trading_orders` is used to see each order sent by Account.

## Usage

```
trading_orders(connection, account)
```

## Arguments

`connection` S4. **Mandatory** Formal rRofexConnection class object

`account` String. **Mandatory** Account Number

## Value

If correct, it will load a tibble.

## See Also

Other order management functions: `trading_lookup()`

## trading_ws_close *Web Sockets: Close connection*

### Description

**Maturing** This method it is use to close open Websocket connections.

### Usage

```
trading_ws_close(close_all = TRUE, selection, where_is_env = .GlobalEnv)
```

### Arguments

| | |
|---|---|
| close_all | Logical. Should all connections be closed or only the selected ones. |
| selection | List. Is the same name that you have chosen for destination in trading_ws_md |
| where_is_env | Environment. **Only for advance users**. |

### Value

If correct, it will show a message saying that the connection has been closed.

### See Also

Other websocket functions: trading_ws_md(), trading_ws_orders()

### Examples

```
# To close all connections at once

## Not run:
trading_ws_close(close_all = TRUE)

## End(Not run)
```

## trading_ws_md *Web Sockets: Market Data Real Time*

### Description

**Experimental** This method brings Market Data in Real Time using web socket protocol.

## Usage

```
trading_ws_md(
  connection,
  destination,
  symbol,
  entries = list("BI", "OF", "LA", "OP", "CL", "SE", "OI", "HI", "LO", "TV", "IV",
    "EV", "NV", "TC"),
  listen_to = NA,
  market_id = "ROFX",
  where_is_env = .GlobalEnv
)
```

## Arguments

| | |
|---|---|
| connection | S4. **Mandatory** Formal rRofexConnection class object |
| destination | String. Name of the tibble where the data is going to be stored. |
| symbol | String. **Mandatory**. Use trading_instruments to see which symbols are available. |
| entries | List of Strings. It contains the information to be queried:<br>• **BI** - Bid.<br>• **OF** - Offer.<br>• **LA** - Last Available Price.<br>• **OP** - Open Price.<br>• **CL** - Close Price.<br>• **SE** - Settlement Price.<br>• **OI** - Open Interest.<br>• **HI** - Trading Session High Price<br>• **LO** - Trading Session Low Price<br>• **TV** - Trading Volume<br>• **IV** - Index Value<br>• **EV** - Trading Effective Volume<br>• **NV** - Nominal Volume<br>• **TC** - Trade Count |
| listen_to | List. Column names from the tibble that you are going to listen to. This is not the same as entries names. |
| market_id | String. Market to which you are going to connect. |
| where_is_env | Environment. **Only for advance users**. |

## Value

If correct, it will load a tibble.

## See Also

Other websocket functions: trading_ws_close(), trading_ws_orders()

## Examples

```
# To create simultaneously many connections

## Not run:
purrr::walk2(
.x = symbols,
.y = tickers,
.f = ~ trading_ws_md(connection = conn, destination = .y, symbol = .x)
)

## End(Not run)
```

---

`trading_ws_orders` *Web Sockets: Orders Lookup*

---

## Description

**Experimental** This method brings orders states in real time using web socket protocol.

## Usage

```
trading_ws_orders(
  connection,
  destination,
  account = NA,
  only_active = FALSE,
  where_is_env = .GlobalEnv
)
```

## Arguments

| | |
|---|---|
| `connection` | S4. **Mandatory** Formal rRofexConnection class object |
| `destination` | String. Name of the tibble where the data is going to be stored. |
| `account` | List. List of accounts to be listeting |
| `only_active` | Logical. Wheater or not to listen to only active orders |
| `where_is_env` | Environment. **Only for advance users**. |

## Value

If correct, it will load a tibble.

## See Also

Other websocket functions: `trading_ws_close()`, `trading_ws_md()`

`user_name` *See User Name*

## Description

Shows information about the user name connected using `trading_login`

## Usage

```
user_name(x)

## S4 method for signature 'rRofexConnection'
user_name(x)
```

## Arguments

| | |
|---|---|
| x | S4 Class. rRofexConnection object |

## Value

Scalar with the 'user_name'

# Index