IBrokers R API Overview

The IBrokers API parallels the official Java API provided by Interactive Brokers, LLC to access data and execution services provided to IB clients. Commands can be run interactively or automated.

The official API documentation is grouped by EClientSocket methods, EWrapper methods, and SocketClient objects. This document combines all related objects and methods into groups by functionality.

Where appropriate, EWrapper methods for processing incoming messages from related calls are listed.

Connection and Server

Connecting to either the TWS or IB Gateway requires setting connection parameters external to IBrokers. Once enabled, the following commands can be used for connections and details.

```java
connect twsConnect, ibgConnect
disconnect twsDisconnect, close
set logging level is.twsConnection, isConnected
check server version serverVersion
request current time reqCurrentTime
request connection time twsConnectionTime
```

Contracts

All requests require validly constructed `twsContract` objects. The basic function to create a valid object is `twsContract`, though IBrokers implements wrapper functions to simplify commonly requested types such as equity, cash, and futures. Depending on the context the constructors may need more or less detail.

```java
create any contract twsContract
create equity contract twsEquity, twsSTK
create equity option contract twsOption, twsOPT
create future contract twsFuture, twsFUT
create future option contract twsFutureOpt, twsFOP
create currency contract twsCurrency, twsCASH
create cash contract twsCASH
create combo contract twsCombolog
create contract for difference twsCPD
```

Contract Details

Given a full or partial `twsContract`, returns a list of `twsContractDetails` objects; named lists containing contract details including a `contract` element of class `twsContract`. Many IBrokers calls will accept `Contract` arguments of `twsContract` or `twsContractDetails`.

```java
request contract(s) description `twsContractDetails`
eWrapper methods:
contractDetails, bondContractDetails, contractDetailsEnd
```

Market Data

Market Data provides for nearly real-time data from Interactive Brokers. Data is actually aggregated into one-third second ‘snapshot’ data from the exchange, and subsequently passed along to the client.

```java
request market data and process reqMktData
request market data (only) .reqMktData
cancel market data cancelMktData
```

eWrapper methods:
`mktPrice`, `mktSize`, `mktOptionComputation`, `mktGeneric`

```java
request markert depth data reqMktDepth
request markert depth data cancel market depth data cancelMktDepth
```

```java
eWrapper methods:
updateMktDepth, updateMktDepthL2
```

Market Depth

Depth of book varies according to contract, and may not be available for all security types.

```java
request market depth data reqMktDepth
```

```java
eWrapper methods:
cancelMktDepth
```

Real Time Bars

Real-time bars are limited to 5-second bars by the official API. All other `barSize` values will fail. Realtime bars may not be available for all security types.

```java
request real-time bars reqRealTimeBars
cancel real-time bars cancelRealTimeBars
```

eWrapper methods:
`realtimeBars`

Historical Data

Depending on the contract, only specific combinations of `barSize` and `duration` arguments are valid, and some security types have no historical data. `reqHistory` is an IBrokers only call, allowing for one year of 1 minute bars, respecting IB timeouts (10 seconds) and maximum bars per request (2000).

```java
request historical data reqHistoricalData
request maximum history reqHistory
```

```java
cancel historical request cancelHistoricalData
```

Valid `barSize` values include: 1 secs, 15 secs, 1 min, 2 mins, 3 mins, 5 mins, 15 mins, 30 mins, 1 hour, 1 day, 1 week, 1 month, 3 months, 1 year.

Valid `duration` form is ‘n S’, where n is the number of periods of S. The second argument may be S (seconds), D (days), W (weeks), M (months), Y (year). Year requests are limited to 1 year.

```java
```

Fundamental Data

Reuters fundamental data

```java
request fundamental data reqFundamentalData
```

```java
cancel fundamental data cancelFundamentalData
```

eWrapper methods:
`fundamentalData`

News Bulletins

Subscribe to news bulletins from Interactive Brokers.

```java
subscribe reqNewsBulletins
```

```java
unsubscribe cancelNewsBulletins
```

eWrapper methods:
`newsBulletins`

Pricing

Calculate option values, price and implied volatility, via the TWS engine.

```java
calculate option price calculateOptionPrice
```

```java
calculate option volatility calculateImpliedVolatility
```

eWrapper methods:
`tickOptionCalculation`
Orders
Orders via the IB API, and the IBrokers API, require three primary components: A `twsContract` object, a `twsOrder` object, and a `placeOrder` call. Additionally, a valid `orderId` is required to the `twsOrder` object. This is found by calling `reqIds` on the `twsConnection` object. `reqIds` operates directly on the connection object by retrieving and then incrementing the next valid `orderId` in the connection object.

```r
nextValidOrderId <- reqIds(tws)
createOrder <- twsOrder(reqIds(tws),
  'BUY', 10, 'MKT'))
```

Account
Account data is requested on a subscription basis. The user subscribes to a continuously updated feed from the TWS by passing the connection object and the `subscribe` argument set to `TRUE`; unsubscribe with `FALSE`. The `.reqAccountUpdates` function will return immediately and will begin or end a subscription; account messages must be handled by the user. `reqAccountUpdates` (without the prepended `dot`) will subscribe, collect data, and unsubscribe — returning an `AccountUpdate` object which may be processed with `twsPortfolioValue`.

```r
cancelAccountUpdates <- reqAccountUpdates(canceled = TRUE)
getAccountData <- reqAccountUpdates(accountUpdates = TRUE)
viewPortfolio <- reqAccountUpdates(accountUpdates = TRUE)
```

eWrapper methods:
- `orderStatus`, `openOrder`, `nextValidId`, `execDetails`