IBrokers Reference Card

IBrokers 0.2-7; TWS API 9.62

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.

connect	`twsConnect`
disconnect	`twsDisconnect, close`
is a tws connection	`is.twsConnection`
set logging level	`setServerLogLevel`
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.

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`

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*.

request contract(s) description	`reqContractDetails`
extract *twsContract* from details	`as.twsContract`

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.

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

eWrapper methods:
tickPrice, tickSize, tickOptionComputation, tickGeneric tickString, tickEFP, tickSnapshotEnd

Market Depth

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

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

eWrapper methods:
updateMktDepth, updateMktDepthL2

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.

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).

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

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 order id in the connection object.

next valid order id	`reqIds`
create order object	`twsOrder`
place order and process	`placeOrder`
place order (only)	`.placeOrder`
cancel order	`cancelOrder`

eWrapper methods:
orderStatus, openOrder, nextValidId, execDetails

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`.

subscribe and process	`reqAccountUpdates`
subscribe (only)	`.reqAccountUpdates`

eWrapper methods:
updateAccountValue, updatePortfolio, updateAccountTime

Executions

Returns execution details in a *twsExecution* object. This method is currently only implemented as a request, with no built-in mechanism to manage response data apart from it being discarded.

request execution data	`reqExecutions`
filter argument	`reqExecutionFilter`

eWrapper methods:
execDetails, execDetailsEnd

DISCLAIMER