

# IBrokers

Automated Trading with R and Interactive Brokers

Jeffrey A. Ryan

jeffrey.ryan @ insightalgo.com

R/Finance 2010 Workshop

# The Official API

[www.interactivebrokers.com](http://www.interactivebrokers.com)



Available in JAVA, C++, Excel, Visual Basic  
Data, Orders, and Account Requests



# The Official API

## Exchanges

Toronto Stock Exchange

ISE

NYSE AMEX

CME

ArcaEdge

LAVA

EUREX

BOX

Hong Kong Stock Exchange

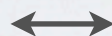
CBOE

OneChicago

London Stock Exchange

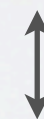
LIFFE

EURONEXT



Underlying	Exchange	Descrptn	Position	Av...	Mk...	P&L	Unr...	Bid Size	Bid	Ask	Ask Size	Last	Change
			TIF	Actn	Qty	Type	Lmt...	Status					
TOTAL	USD				3...	-5...	5...						
AAPL	SMART	Stock (N...	130	2...	3...	-5...	5...	3	244.75	245.25	2	245.25	-0.44
DRYS	SMART	Stock (N...	20	7...	1...	0.00	-1...	4	6.58	6.70	21	6.58	
NQ	GLOBEX	JUN10 Fu...						4	2022...	2022.25	29	2022.00	-4.50
USD	IDEALPRO	USD.CHF...						1.000	1.05	1.05935	1.000	1.05	+0.00

TWS



Interactive Brokers **API**

# The Official API



# The Official API

Safari File Edit View History Bookmarks Window Help 08:10 06:10:10 (100%) Wed 16:58:57

Electronic Access Trading, Stocks, Options, Futures, Forex, Bonds, Funds

http://www.interactivebrokers.com/en/main.php Google

Creating two...tex Matters Coding Horr...dward Tufte Install and ... tablespoons Apple Yahoo! Google Maps YouTube Wikipedia News (978) Popular

**Interactive Brokers**  
The Professional's Gateway to the World's Markets

Contract Search | Site Map | Help & Contacts

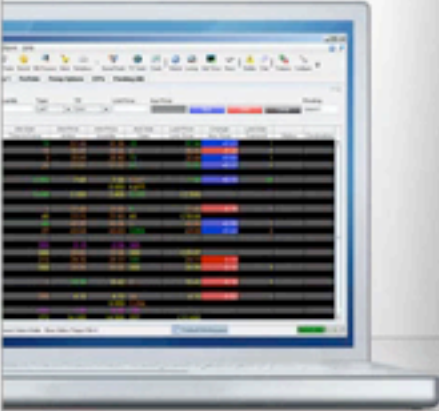
Type Website Search Text Search

Home Why IB Fees Trading Accounts Software Education About IB Login Open an Account

Stocks • Options • Futures • Forex • Bonds — Over 80 Markets Worldwide in a single IB Universal Account<sup>SM</sup>


IB News Headlines | Reporting Dates for 2009 Tax Year Now Available...

### Demos



Try a Demo

### Education



Education Center

### Low Cost

**BARRON'S**  
LOW COST  
★★★★★

**TAG**  
BEST PRICE EXECUTION  
BEST PRICE EXECUTION

See how we compare to other brokers

### US Products Commission Schedule

**US** CA Europe Asia Interest

**Stocks & ETFs (all-in):**  
\$0.005 or less per share

**Options (plus exchange fees):**  
\$0.15 to \$0.70 per contract

**Futures (plus exchange, regulatory and carry fees):**  
\$0.25 to 0.85 per contract

**Bonds (all-in):**  
\$1.00 per \$1,000 face value (<=10,000)  
\$0.25 per \$1,000 face value (>10,000)

No extra ticket charges, \$1.00 order minimum

### Real-time Forex Quotes

<b>EUR.USD</b> BID 1.3850% ASK 1.3852	<b>USD.JPY</b> BID 93.25 ASK 93.26
<b>GBP.USD</b> BID 1.5467% ASK 1.5468	<b>USD.CAD</b> BID 0.9989 ASK 0.9989
<b>USD.CHF</b> BID 1.0517 ASK 1.0519	

### IB Market Brief

IB Market Brief

**Fx** BRIEF

**iR** BRIEF

# The IBrokers API

## Design and Motivation

Provide native R Access to IB API --- *no dependencies*

Keep official documentation *THE* documentation

Bring the power of R syntax into the equation



# The IBrokers API

Connections

Contracts

Client Methods

# The IBrokers API

Connections

Contracts

Client Methods

eWrapper

CALLBACK

processMsg



# The IBrokers API

Connections

Contracts

Client Methods

eWrapper

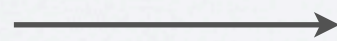
CALLBACK

processMsg

# The IBrokers API



eWrapper + processMsg





# The IBrokers API

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
  eW <- eWrapper(NULL)
  eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
    7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
    "Low", "Close", "Volume", "WAP", "Count")))), n))
  eW$realtimeBars <- function(curMsg, msg, timestamp, file,
    ...) {
    id <- as.numeric(msg[2])
    file <- file[[id]]
    data <- eW$get.Data("data")
    attr(data[[id]], "index") <- as.numeric(msg[3])
    nr.data <- NROW(data[[id]])
    cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
      msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
    data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
    eW$assign.Data("data", data)
    c(curMsg, msg)
  }
  return(eW)
}
<environment: namespace:IBrokers>
```

# The IBrokers API

first, create a set of methods that do nothing with the incoming data and assign to `eW`

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
  eW <- eWrapper(NULL)
  eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
    7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
    "Low", "Close", "Volume", "WAP", "Count")))), n))
  eW$realtimeBars <- function(curMsg, msg, timestamp, file,
    ...) {
    id <- as.numeric(msg[2])
    file <- file[[id]]
    data <- eW$get.Data("data")
    attr(data[[id]], "index") <- as.numeric(msg[3])
    nr.data <- NROW(data[[id]])
    cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
      msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
    data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
    eW$assign.Data("data", data)
    c(curMsg, msg)
  }
  return(eW)
}
<environment: namespace:IBrokers>
```



# The IBrokers API

use eWrapper  
method  
assign.Data to  
create a list of xts  
objects to hold our  
bars -- one for  
each contract  
watched

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
  eW <- eWrapper(NULL)
  eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
    7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
    "Low", "Close", "Volume", "WAP", "Count")))), n))
  eW$realtimeBars <- function(curMsg, msg, timestamp, file,
    ...) {
    id <- as.numeric(msg[2])
    file <- file[[id]]
    data <- eW$get.Data("data")
    attr(data[[id]], "index") <- as.numeric(msg[3])
    nr.data <- NROW(data[[id]])
    cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
      msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
    data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
    eW$assign.Data("data", data)
    c(curMsg, msg)
  }
  return(eW)
}
<environment: namespace:IBrokers>
```

# The IBrokers API

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
  eW <- eWrapper(NULL)
  eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
    7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
    "Low", "Close", "Volume", "WAP", "Count")))), n))
  eW$realtimeBars <- function(curMsg, msg, timestamp, file,
    ...) {
    id <- as.numeric(msg[2])
    file <- file[[id]]
    data <- eW$get.Data("data")
    attr(data[[id]], "index") <- as.numeric(msg[3])
    nr.data <- NROW(data[[id]])
    cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
      msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
    data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
    eW$assign.Data("data", data)
    c(curMsg, msg)
  }
  return(eW)
}
<environment: namespace:IBrokers>
```



# The IBrokers API

create a new method `realtimeBars` to do what we want. In this case capture and print each new message in a format that makes sense

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
  eW <- eWrapper(NULL)
  eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
    7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
    "Low", "Close", "Volume", "WAP", "Count")))), n))
  eW$realtimeBars <- function(curMsg, msg, timestamp, file,
    ...) {
    id <- as.numeric(msg[2])
    file <- file[[id]]
    data <- eW$get.Data("data")
    attr(data[[id]], "index") <- as.numeric(msg[3])
    nr.data <- NROW(data[[id]])
    cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
      msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
    data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
    eW$assign.Data("data", data)
    c(curMsg, msg)
  }
  return(eW)
}
<environment: namespace:IBrokers>
```

# The IBrokers API

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
  eW <- eWrapper(NULL)
  eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
    7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
    "Low", "Close", "Volume", "WAP", "Count")))), n))
  eW$realtimeBars <- function(curMsg, msg, timestamp, file,
    ...) {
    id <- as.numeric(msg[2])
    file <- file[[id]]
    data <- eW$get.Data("data")
    attr(data[[id]], "index") <- as.numeric(msg[3])
    nr.data <- NROW(data[[id]])
    cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
      msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
    data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
    eW$assign.Data("data", data)
    c(curMsg, msg)
  }
  return(eW)
}
<environment: namespace:IBrokers>
```

return the  
modified  
eWrapper object



# The IBrokers API

Connections

Contracts

Client Methods

eWrapper

CALLBACK

processMsg

# The IBrokers API

Connections

Contracts

Client Methods

eWrapper

CALLBACK

processMsg

# The IBrokers API

“snapshot” market data



# The IBrokers API

“snapshot” market data

reqMktData **snapshot=TRUE** is *slow* and *wrong*

# The IBrokers API

“snapshot” market data

reqMktData **snapshot=TRUE** is *slow* and *wrong*

Solution: re-implement with a custom CALLBACK

# The IBrokers API

“snapshot” market data

reqMktData snapshot=TRUE is *slow* and *wrong*

Solution: re-implement with a custom CALLBACK

3 Lines of Code!



# The IBrokers API

```
> snapShot
function (twCon, eWrapper, timestamp, file, playback = 1, ...)
{
  if (missing(eWrapper))
    eWrapper <- eWrapper()
  names(eWrapper$.Data$data) <- eWrapper$.Data$symbols
  con <- twCon[[1]]
  while (TRUE) {
    socketSelect(list(con), FALSE, NULL)
    curMsg <- .Internal(readBin(con, "character", 1L,
      NA_integer_, TRUE, FALSE))
    if (!is.null(timestamp)) {
      processMsg(curMsg, con, eWrapper, format(Sys.time(),
        timestamp), file, ...)
    }
    else {
      processMsg(curMsg, con, eWrapper, timestamp,
        file, ...)
    }
  }
  if (!any(sapply(eWrapper$.Data$data, is.na)))
    return(do.call(rbind,lapply(eWrapper$.Data$data,as.data.frame)))
}
}
```

# The IBrokers API

```
> snapShot
function (twCon, eWrapper, timestamp, file, playback = 1, ...)
{
  if (missing(eWrapper))
    eWrapper <- eWrapper()
  names(eWrapper$.Data$data) <- eWrapper$.Data$symbols
  con <- twCon[[1]]
  while (TRUE) {
    socketSelect(list(con), FALSE, NULL)
    curMsg <- .Internal(readBin(con, "character", 1L,
      NA_integer_, TRUE, FALSE))
    if (!is.null(timestamp)) {
      processMsg(curMsg, con, eWrapper, format(Sys.time(),
        timestamp), file, ...)
    }
    else {
      processMsg(curMsg, con, eWrapper, timestamp,
        file, ...)
    }
  }
  if(!any(sapply(eWrapper$.Data$data, is.na)))
    return(do.call(rbind,lapply(eWrapper$.Data$data,as.data.frame)))
}
}
```

# Putting it all together



## Putting it all together

Extend CALLBACK and eWrapper objects  
to implement a simple strategy:

Buy when ES crosses a specific threshold

Additional things to do...

Aggregate market data to OHLC bars with built in  
eWrapper function

Add filter interface to allow for CEP  
style queries

Complete the API:  
fundamental data, scanner data

# IBrokers

Automated Trading with R and Interactive Brokers

## **Thank You**

Jeffrey A. Ryan  
jeffrey.ryan @ insightalgo.com

R/Finance 2010 Workshop