RBloomberg Manual

Ana Nelson

September 30, 2010

Contents

1 About RBloomberg **1**

2 Installation and Requirements **1**
2.1 Prerequisites . . . 1
2.2 Installation . . . 2
2.3 Hello, World . . . 2
2.4 blpConnect . . . 3

3 Requesting Data **4**
3.1 Reference Data . . . 4
3.2 Bulk Data . . . 5
3.3 Historical Data . . . 7
3.4 Tick Data . . . 12
3.5 Bar Data . . . 13

4 Overrides, Options, Tips and Tricks **13**
4.1 Ticker Format . . . 13
4.2 Using Overrides . . . 13
4.3 Setting Options . . . 14
4.4 Ignoring Ticker Errors . . . 15
4.5 Field Lookup . . . 16

5 Troubleshooting **16**

1 About RBloomberg

RBloomberg is an R package which handles fetching data from the Bloomberg financial data application. RBloomberg was written by Robert Sams, see the package README for additional contributors and acknowledgements. RBloomberg is released under a GPL open source license.

This documentation refers to RBloomberg version 0.4-144.

To download the latest version of this document, please visit the RBloomberg home page.

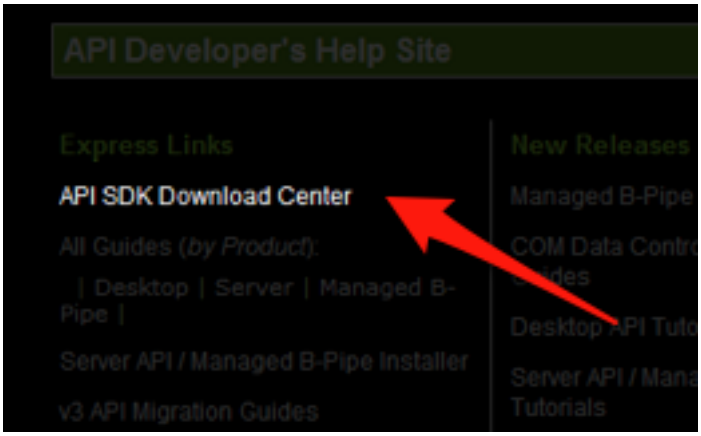
Please download the Bloomberg API Developer Guide PDF via WAPI or DOCS 2041121 as this has additional useful documentation. Appendix A provides details on options that can be set for data requests.

2 Installation and Requirements

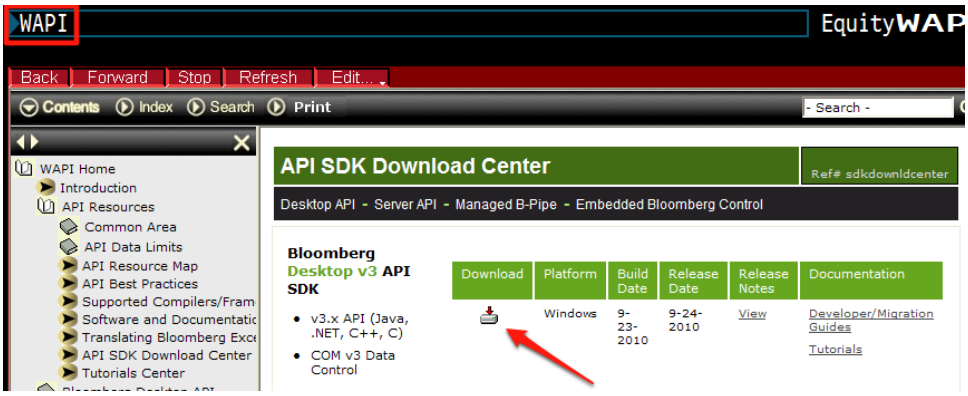
2.1 Prerequisites

RBloomberg will only work on a Bloomberg workstation. RBloomberg uses the Java Version 3 API, and depends on the rJava package. Java 1.5 or higher must also be installed.

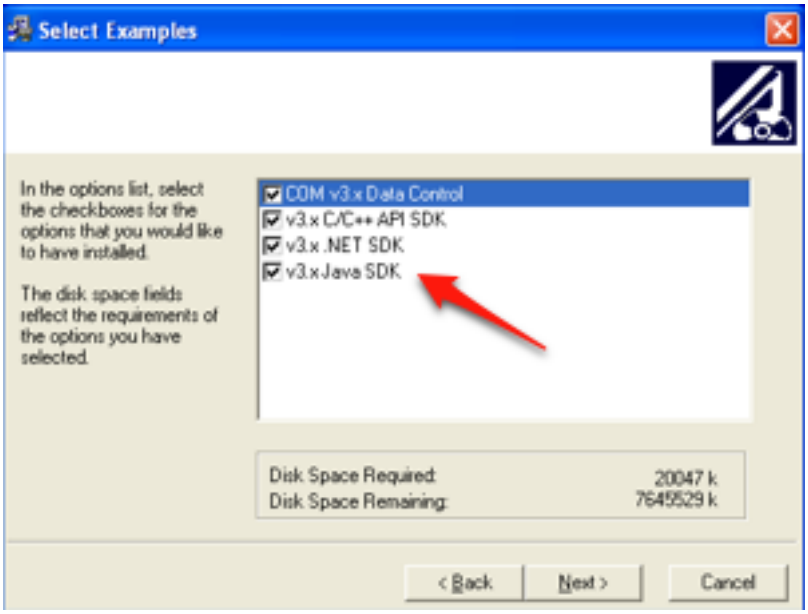
To check if you have the Version 3 API installed, go to `C:\blp\API`. You should see a folder named `APIv3`. To make sure it's working, you can navigate to `C:blp\API\bin` and run the `BBAPIDemo` app. If you don't see this folder, or if you want to make sure you have the latest version of the API installed, then go to the WAPI page within your Bloomberg terminal to install the latest version.



You probably want the Desktop API.



Ensure that the Java API is included in the installation.



To check whether you have Java installed, you can open a command prompt (Start; All Programs; Accessories; Command Prompt) and type `java -version`.

2.2 Installation

To install RBloomberg:

```
install.packages("RBloomberg", repos="http://r.findata.org")
```

This should automatically install the rJava dependency for you.

2.3 Hello, World

Once you have RBloomberg installed, load the library just like any other. The blpConnect() function initializes a connection to Bloomberg and returns a connection object which will be used in all subsequent calls. The bdp() function fetches a basic data call.

```
library(RBloomberg)
conn <- blpConnect()
bdp(conn, "AMZN US Equity", "NAME")
```

The result of running these three commands should be something like this:

```
> library(RBloomberg)
Loading required package: rJava
> conn <- blpConnect()
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
Bloomberg API Version 3.3.3.3
> bdp(conn, "AMZN US Equity", "NAME")
                        NAME
AMZN US Equity AMAZON.COM INC
```

```
>
>
> proc.time()
   user  system elapsed
   1.20    0.75    2.61
```

2.4 blpConnect

```
blpConnect <- function(iface="Java", log.level = "warning",
    blpapi.jar.file = NULL, throw.ticker.errors = TRUE,
    jvm.params = NULL)
```

You can pass parameters for the JVM via jvm.params, which should be a vector of strings, each of which contains a single directive. Consult the documentation for the .jinit function in the rJava package for further details. You can list available parameters via `java -X`.

```
    -Xmixed           mixed mode execution (default)
    -Xint             interpreted mode execution only
    -Xbootclasspath:<directories and zip/jar files separated by :>
                      set search path for bootstrap classes and resources
    -Xbootclasspath/a:<directories and zip/jar files separated by :>
                      append to end of bootstrap class path
    -Xbootclasspath/p:<directories and zip/jar files separated by :>
                      prepend in front of bootstrap class path
    -Xnoclassgc       disable class garbage collection
    -Xloggc:<file>    log GC status to a file with time stamps
    -Xbatch           disable background compilation
    -Xms<size>        set initial Java heap size
    -Xmx<size>        set maximum Java heap size
    -Xss<size>        set java thread stack size
    -Xprof            output cpu profiling data
    -Xfuture          enable strictest checks, anticipating future default
    -Xrs              reduce use of OS signals by Java/VM (see documentation)
    -Xdock:name=<application name>
                      override default application name displayed in dock
    -Xdock:icon=<path to icon file>
                      override default icon displayed in dock
    -Xcheck:jni       perform additional checks for JNI functions
    -Xshare:off              do not attempt to use shared class data
    -Xshare:auto      use shared class data if possible (default)
    -Xshare:on              require using shared class data, otherwise fail.

The -X options are non-standard and subject to change without notice.
```

Here is an example of configuring for verbose garbage collection information:

```
library(RBloomberg)

add.equity.label <- function(ticker) {
  paste(ticker, "Equity")
}

conn <- blpConnect(jvm.params = c("-Xmx256m", "-Xloggc:rbloomberg.gc", "-XX:+PrintGCDetails"))

tickers <- bds(conn, "UKX Index", "INDX_MEMBERS")[,1]
tickers <- add.equity.label(tickers)
```

```
x <- bdp(conn, tickers, "PX_LAST")
blpDisconnect(conn)
```

The resulting .gc file looks like:

```
0.474: [GC 0.475: [DefNew: 3328K->281K(3712K), 0.0070881 secs] 3328K->281K(7808K), 0.0072199 secs] [
1.289: [GC 1.289: [DefNew: 3609K->384K(3712K), 0.0084491 secs] 3609K->722K(7808K), 0.0085550 secs] [
1.862: [GC 1.862: [DefNew: 3711K->57K(3712K), 0.0038647 secs] 4050K->672K(7808K), 0.0039701 secs] [T
```

and can be viewed with a GC viewer. An open source GC viewer is available from here.

3 Requesting Data

3.1 Reference Data

This section covers getting current, i.e. non-historical, data from Bloomberg. This may be live (or delayed as per your availability) market data, or static descriptive data. All such data is called using the bdp() function, as defined below:

```
bdp <- function(conn, securities, fields,
    override_fields = NULL, override_values = NULL,
    option_names = NULL, option_values = NULL)
```

```
> library(RBloomberg)
Loading required package: rJava
> conn <- blpConnect()
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
Bloomberg API Version 3.3.3.3
>
> bdp(conn, "AMZN US Equity", "NAME")
                          NAME
AMZN US Equity AMAZON.COM INC
>
> securities <- c("AMZN US Equity", "OCN US Equity")
> fields <- c("NAME", "PX_LAST", "TIME", "SETTLE_DT", "HAS_CONVERTIBLES") # Demo different return da
> bdp(conn, securities, fields)
                             NAME  PX_LAST     TIME  SETTLE_DT
AMZN US Equity       AMAZON.COM INC 160.0199 15:46:04 2010-09-30
OCN US Equity  OCWEN FINANCIAL CORP   9.9600 15:43:32 2010-09-30
               HAS_CONVERTIBLES
AMZN US Equity             TRUE
OCN US Equity              TRUE
>
> securities <- c("AMZN US Equity", "OCN US Equity")
> fields <- c("CUST_TRR_RETURN_HOLDING_PER")
> override_fields <- c("CUST_TRR_START_DT", "CUST_TRR_END_DT", "CUST_TRR_CRNCY")
> overrides <- c("20090601", "20091231", "PRC")
> bdp(conn, securities, fields, override_fields, overrides)
               CUST_TRR_RETURN_HOLDING_PER
AMZN US Equity                    61.97471
OCN US Equity                     25.40992
>
> securities <- c("RYA ID EQUITY", "OCN US EQUITY", "YHOO US EQUITY")
```

```
> fields <- c("LT_DEBT_TO_COM_EQY")
> override_fields <- c("EQY_FUND_DT")
> overrides <- c("20051231")
> bdp(conn, securities, fields, override_fields, overrides)
               LT_DEBT_TO_COM_EQY
RYA ID EQUITY           76.527530
OCN US EQUITY           44.423110
YHOO US EQUITY           8.755063
>
> override_fields <- c("EQY_FUND_DT")
> overrides <- c("20061231")
> bdp(conn, securities, fields, override_fields, overrides)
               LT_DEBT_TO_COM_EQY
RYA ID EQUITY            66.27159
OCN US EQUITY            26.94169
YHOO US EQUITY            8.18630
>
> bdp(conn, "/SEDOL1/2292612 EQUITY", "NAME")
                                                        NAME
/SEDOL1/2292612 EQUITY TELE NORTE CELULAR PART-PREF
>
> blpDisconnect(conn)
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 224451  6.0     407500 10.9   350000  9.4
Vcells 112910  0.9     786432  6.0   390780  3.0
>
>
> proc.time()
   user  system elapsed
   1.78    0.67    3.93
```

3.2 Bulk Data

Bulk data fields can return multiple fields and rows in response to a single requested field. Dividend history data (DVD_HIST) and getting a list of members in an index (INDX_MEMBERS) are examples of bulk data fields. If more than one security is requested, then the tickers are shown in the first column of the returned data frame. Multiple fields can be requested at a time if they return data with the same column names, for example INDX_MEMBERS, INDX_MEMBERS2 and INDX_MEMBERS3.

```
bds <- function(conn, securities, fields,
    override_fields = NULL, override_values = NULL,
    option_names = NULL, option_values = NULL)
```

```
> library(RBloomberg)
Loading required package: rJava
> conn <- blpConnect(log.level = "finest")
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
Bloomberg API Version 3.3.3.3
>
>
> security <- c("BKIR ID Equity")
> field <- c("DVD_HIST")
```

```
>
> bds(conn, security, field)[1:5,]
  Declared Date    Ex-Date Record Date Payable Date Dividend Amount
1    2009-11-04 2009-11-18                                      NA
2    2009-05-19 2009-05-27                                      NA
3    2008-11-13 2008-11-19                                      NA
4    2008-05-21 2008-05-28  2008-05-30   2008-07-23        0.250471
5    2007-11-14 2007-11-21  2007-11-23   2008-01-15        0.153843
  Dividend Frequency Dividend Type
1            Suspend  Discontinued
2          Semi-Anl       Omitted
3          Semi-Anl       Omitted
4          Semi-Anl         Final
5          Semi-Anl       Interim
>
> security <- "TYA Comdty"
> field <- "FUT_DELIVERABLE_BONDS"
>
> bds(conn, security, field)[1:5,]
  Ticker, Coupon, Maturity of Deliverable Bonds Conversion Factor
1                        T 1.875 08/31/17 Govt            0.7807
2                        T 2.375 07/31/17 Govt            0.8072
3                          T 2.5 06/30/17 Govt            0.8139
4                        T 2.625 08/15/20 Govt            0.7583
5                         T 2.75 02/15/19 Govt            0.7959
>
> security <- "UKX Index"
> field <- "INDX_MEMBERS"
>
> bds(conn, security, field)[1:5,]
[1] "AAL LN" "ABF LN" "ABG LN" "ADM LN" "AGK LN"
>
> securities <- c("UKX Index", "SPX Index")
> fields <- c("INDX_MEMBERS", "INDX_MEMBERS2", "INDX_MEMBERS3")
>
> bds(conn, securities, fields)[c(1:5, 350:355),]
       ticker Member.Ticker.and.Exchange.Code
1   UKX Index                          AAL LN
2   UKX Index                          ABF LN
3   UKX Index                          ABG LN
4   UKX Index                          ADM LN
5   UKX Index                          AGK LN
350 SPX Index                          JWN UN
351 SPX Index                            K UN
352 SPX Index                          KEY UN
353 SPX Index                          KFT UN
354 SPX Index                           KG UN
355 SPX Index                          KIM UN
>
>
>
> proc.time()
   user  system elapsed
   1.73    0.82    4.45
```

3.3 Historical Data

For historical data, fetching a single security at a time is currently supported. Dates can be in string form using Bloomberg's YYYYMMDD format, or using any date/time class which responds to strftime. End dates are optional, if omitted all data from start time to current time will be requested.

If multiple securities are requested, then another column will be included containing the security ticker. If you wish to include this column when requesting a single security, which may be necessary if you don't know in advance whether your query will be for one or multiple securities, then pass `always.display.tickers = TRUE`. By default, the dates will be used as row names unless multiple tickers are requested. To disable this, pass `dates.as.row.names = FALSE`. Setting `dates.as.row.names=TRUE` when requesting multiple tickers will result in an error.

You can specify that a row be returned for all dates in the requested period, even when markets or closed or otherwise no data is available, by specifying `include.non.trading.days=TRUE`. If you request multiple securities, this is automatically set to TRUE. You can use `na.omit` or `na.exclude` to remove these rows. See examples below for more fine-grained control available by passing options directly to API.

```
bdh <- function(conn, securities, fields, start_date, end_date = NULL,
    override_fields = NULL, override_values = NULL,
    option_names = NULL, option_values = NULL,
    always.display.tickers = FALSE, dates.as.row.names = (length(securities) == 1),
    include.non.trading.days = NULL)
```

```
> library(RBloomberg)
Loading required package: rJava
> conn <- blpConnect()
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
Bloomberg API Version 3.3.3.3
>
> bdh(conn, "GOLDS Comdty", "PX_LAST", "20090101", "20090107")
                date PX_LAST
2009-01-01 2009-01-01  879.45
2009-01-02 2009-01-02  875.40
2009-01-05 2009-01-05  859.50
2009-01-06 2009-01-06  863.90
2009-01-07 2009-01-07  842.95
>
> Sys.setenv(TZ="GMT")
> start.date <- as.POSIXct("2009-01-01")
> end.date <- as.POSIXct("2009-01-07")
>
> bdh(conn, "GOLDS Comdty", "PX_LAST", start.date, end.date)
                date PX_LAST
2009-01-01 2009-01-01  879.45
2009-01-02 2009-01-02  875.40
2009-01-05 2009-01-05  859.50
2009-01-06 2009-01-06  863.90
2009-01-07 2009-01-07  842.95
>
> bdh(conn, "GOLDS Comdty", "PX_LAST", Sys.Date() - 10)
                date PX_LAST
2010-09-17 2010-09-17 1274.30
2010-09-20 2010-09-20 1278.35
```

```
2010-09-21 2010-09-21 1287.15
2010-09-22 2010-09-22 1291.35
2010-09-23 2010-09-23 1292.45
2010-09-24 2010-09-24 1295.95
2010-09-27 2010-09-27 1296.55
>
> library(zoo)
> result <- bdh(conn, "GOLDS Comdty", "PX_LAST", Sys.Date() - 10)
> zoo(result, order.by = rownames(result))
           date       PX_LAST
2010-09-17 2010-09-17 1274.30
2010-09-20 2010-09-20 1278.35
2010-09-21 2010-09-21 1287.15
2010-09-22 2010-09-22 1291.35
2010-09-23 2010-09-23 1292.45
2010-09-24 2010-09-24 1295.95
2010-09-27 2010-09-27 1296.55
>
> bdh(conn, "GOLDS Comdty", "PX_LAST", Sys.Date() - 366,
+     option_names = "periodicitySelection", option_values = "MONTHLY")
                date PX_LAST
2009-09-30 2009-09-30 1007.70
2009-10-30 2009-10-30 1045.40
2009-11-30 2009-11-30 1179.60
2009-12-31 2009-12-31 1096.95
2010-01-29 2010-01-29 1080.85
2010-02-26 2010-02-26 1117.60
2010-03-31 2010-03-31 1113.25
2010-04-30 2010-04-30 1179.20
2010-05-31 2010-05-31 1216.20
2010-06-30 2010-06-30 1242.25
2010-07-30 2010-07-30 1181.00
2010-08-31 2010-08-31 1247.45
>
> df <- bdh(conn, c("AMZN US Equity", "GOOG US Equity", "MSFT US Equity"),
+     c("PX_LAST", "BID"), start.date, end.date)
> df
           ticker       date PX_LAST    BID
1  AMZN US Equity 2009-01-01      NA     NA
2  AMZN US Equity 2009-01-02   54.36  54.19
3  AMZN US Equity 2009-01-03      NA     NA
4  AMZN US Equity 2009-01-04      NA     NA
5  AMZN US Equity 2009-01-05   54.06  54.03
6  AMZN US Equity 2009-01-06   57.36  57.35
7  AMZN US Equity 2009-01-07   56.20  56.19
8  GOOG US Equity 2009-01-01      NA     NA
9  GOOG US Equity 2009-01-02  321.32 320.94
10 GOOG US Equity 2009-01-03      NA     NA
11 GOOG US Equity 2009-01-04      NA     NA
12 GOOG US Equity 2009-01-05  328.05 327.99
13 GOOG US Equity 2009-01-06  334.06 333.58
14 GOOG US Equity 2009-01-07  322.01 321.73
15 MSFT US Equity 2009-01-01      NA     NA
16 MSFT US Equity 2009-01-02   20.33  20.30
17 MSFT US Equity 2009-01-03      NA     NA
18 MSFT US Equity 2009-01-04      NA     NA
```

```
19 MSFT US Equity 2009-01-05   20.52  20.51
20 MSFT US Equity 2009-01-06   20.76  20.75
21 MSFT US Equity 2009-01-07   19.51  19.50
> na.omit(df)
           ticker       date PX_LAST    BID
2  AMZN US Equity 2009-01-02   54.36  54.19
5  AMZN US Equity 2009-01-05   54.06  54.03
6  AMZN US Equity 2009-01-06   57.36  57.35
7  AMZN US Equity 2009-01-07   56.20  56.19
9  GOOG US Equity 2009-01-02  321.32 320.94
12 GOOG US Equity 2009-01-05  328.05 327.99
13 GOOG US Equity 2009-01-06  334.06 333.58
14 GOOG US Equity 2009-01-07  322.01 321.73
16 MSFT US Equity 2009-01-02   20.33  20.30
19 MSFT US Equity 2009-01-05   20.52  20.51
20 MSFT US Equity 2009-01-06   20.76  20.75
21 MSFT US Equity 2009-01-07   19.51  19.50
>
> bdh(conn, c("AMZN US Equity"), c("PX_LAST", "BID"), start.date, end.date,
+     always.display.tickers = TRUE)
                  ticker       date PX_LAST   BID
2009-01-02 AMZN US Equity 2009-01-02   54.36 54.19
2009-01-05 AMZN US Equity 2009-01-05   54.06 54.03
2009-01-06 AMZN US Equity 2009-01-06   57.36 57.35
2009-01-07 AMZN US Equity 2009-01-07   56.20 56.19
>
> bdh(conn, c("AMZN US Equity"), c("PX_LAST", "BID"), start.date, end.date,
+     always.display.tickers = TRUE, dates.as.row.names = FALSE)
          ticker       date PX_LAST   BID
1 AMZN US Equity 2009-01-02   54.36 54.19
2 AMZN US Equity 2009-01-05   54.06 54.03
3 AMZN US Equity 2009-01-06   57.36 57.35
4 AMZN US Equity 2009-01-07   56.20 56.19
>
> bdh(conn, "/SEDOL1/2292612 EQUITY", c("PX_LAST", "BID"), "20090401", "20090410")
                date PX_LAST  BID
2009-04-01 2009-04-01      NA 0.01
2009-04-02 2009-04-02      NA 0.01
2009-04-03 2009-04-03      NA 0.05
2009-04-06 2009-04-06      NA 1.00
2009-04-07 2009-04-07      NA 0.05
2009-04-08 2009-04-08      32 0.05
2009-04-09 2009-04-09      NA 0.05
>
> # We should get NULL back when there's no data...
> bdh(conn, "/SEDOL1/2292612 EQUITY", c("PX_LAST", "BID"), "20090405", "20090405")
NULL
>
> # To return rows for all requested dates, even when they have no data...
> bdh(conn, "/SEDOL1/2292612 EQUITY", c("PX_LAST", "BID"), "20090405", "20090405",
+     include.non.trading.days = TRUE)
                date PX_LAST  BID
2009-04-05 2009-04-05    <NA> <NA>
>
> # This is equivalent to...
> bdh(conn, "/SEDOL1/2292612 EQUITY", c("PX_LAST", "BID"), "20090405", "20090405",
```

```
+     option_names = c("nonTradingDayFillOption", "nonTradingDayFillMethod"),
+     option_values = c("ALL_CALENDAR_DAYS", "NIL_VALUE"))
                date PX_LAST  BID
2009-04-05 2009-04-05    <NA> <NA>
>
> # Consult API documentation for other available option values.
> bdh(conn, "/SEDOL1/2292612 EQUITY", c("PX_LAST", "BID"), "20090405", "20090405",
+     option_names = c("nonTradingDayFillOption", "nonTradingDayFillMethod"),
+     option_values = c("ALL_CALENDAR_DAYS", "PREVIOUS_VALUE"))
                date PX_LAST  BID
2009-04-05 2009-04-05      28 0.05
>
> blpDisconnect(conn)
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 256146  6.9     407500 10.9   350000  9.4
Vcells 131342  1.1     786432  6.0   390780  3.0
>
> proc.time()
   user  system elapsed
   2.64    0.84    6.48
```

If you wish to have a data frame with tickers as row names and dates as column names, and data for a single field displayed in this grid (or vice versa), then see the examples below.

```
> library(RBloomberg)
Loading required package: rJava
> conn <- blpConnect()
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
Bloomberg API Version 3.3.3.3
>
> Sys.setenv(TZ="GMT")
> start.date <- as.POSIXct("2009-01-01")
> end.date <- as.POSIXct("2009-01-07")
>
> df <- bdh(conn, c("AMZN US Equity", "OCN US Equity"), c("PX_LAST", "BID"), start.date, end.date)
> df
           ticker       date PX_LAST     BID
1  AMZN US Equity 2009-01-01      NA      NA
2  AMZN US Equity 2009-01-02 54.3600 54.1900
3  AMZN US Equity 2009-01-03      NA      NA
4  AMZN US Equity 2009-01-04      NA      NA
5  AMZN US Equity 2009-01-05 54.0600 54.0300
6  AMZN US Equity 2009-01-06 57.3600 57.3500
7  AMZN US Equity 2009-01-07 56.2000 56.1900
8   OCN US Equity 2009-01-01      NA      NA
9   OCN US Equity 2009-01-02  5.4868  5.4627
10  OCN US Equity 2009-01-03      NA      NA
11  OCN US Equity 2009-01-04      NA      NA
12  OCN US Equity 2009-01-05  5.5109  5.4989
13  OCN US Equity 2009-01-06  5.5109  5.4989
14  OCN US Equity 2009-01-07  5.5049  5.4989
>
```

```
> t <- unstack(na.omit(df), PX_LAST~ticker)
> rownames(t) <- unique(na.omit(df)$date)
> t
           AMZN.US.Equity OCN.US.Equity
2009-01-02          54.36        5.4868
2009-01-05          54.06        5.5109
2009-01-06          57.36        5.5109
2009-01-07          56.20        5.5049
>
> t <- unstack(df, BID~ticker)
> rownames(t) <- unique(df$date)
> t
           AMZN.US.Equity OCN.US.Equity
2009-01-01             NA            NA
2009-01-02          54.19        5.4627
2009-01-03             NA            NA
2009-01-04             NA            NA
2009-01-05          54.03        5.4989
2009-01-06          57.35        5.4989
2009-01-07          56.19        5.4989
>
> reshape(df, direction="wide", timevar="date", idvar="ticker")
          ticker PX_LAST.2009-01-01 BID.2009-01-01 PX_LAST.2009-01-02
1 AMZN US Equity                 NA             NA            54.3600
8  OCN US Equity                 NA             NA             5.4868
  BID.2009-01-02 PX_LAST.2009-01-03 BID.2009-01-03 PX_LAST.2009-01-04
1        54.1900                 NA             NA                 NA
8         5.4627                 NA             NA                 NA
  BID.2009-01-04 PX_LAST.2009-01-05 BID.2009-01-05 PX_LAST.2009-01-06
1             NA            54.0600        54.0300            57.3600
8             NA             5.5109         5.4989             5.5109
  BID.2009-01-06 PX_LAST.2009-01-07 BID.2009-01-07
1        57.3500            56.2000        56.1900
8         5.4989             5.5049         5.4989
> reshape(df, direction="wide", timevar="date", idvar="ticker", drop="BID", new.row.names=unique(df$
                     ticker PX_LAST.2009-01-01 PX_LAST.2009-01-02
AMZN US Equity AMZN US Equity                NA            54.3600
OCN US Equity   OCN US Equity                NA             5.4868
               PX_LAST.2009-01-03 PX_LAST.2009-01-04 PX_LAST.2009-01-05
AMZN US Equity                 NA                 NA            54.0600
OCN US Equity                  NA                 NA             5.5109
               PX_LAST.2009-01-06 PX_LAST.2009-01-07
AMZN US Equity            57.3600            56.2000
OCN US Equity              5.5109             5.5049
>
> blpDisconnect(conn)
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 229680  6.2     407500 10.9   350000  9.4
Vcells 113766  0.9     786432  6.0   390780  3.0
>
> proc.time()
   user  system elapsed
   1.51    0.64    2.92
```

3.4 Tick Data

Period start and end times must be expressed as strings in UTC time.

```
tick <- function(conn, security, fields, start_date_time, end_date_time,
    option_names = NULL, option_values = NULL)
```

```
> library(RBloomberg)
Loading required package: rJava
> conn <- blpConnect(log.level = "finest")
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
Bloomberg API Version 3.3.3.3
> tick(conn, "RYA ID Equity", "TRADE", "2010-09-21 09:00:00.000", "2010-09-21 09:10:00.000")
                  time  type  value  size
1 2010-09-21T09:06:17.000 TRADE 3.9100 35000
2 2010-09-21T09:07:08.000 TRADE 3.9111  2000
>
> tick(conn, "RYA ID Equity", c("TRADE", "BID_BEST"), "2010-09-21 09:00:00.000", "2010-09-21 09:10:0
                   time     type  value  size
1  2010-09-21T09:00:02.000 BID_BEST 3.9070  1000
2  2010-09-21T09:00:02.000 BID_BEST 3.9070  2000
3  2010-09-21T09:00:02.000 BID_BEST 3.9070  1000
4  2010-09-21T09:00:02.000 BID_BEST 3.9060  2000
5  2010-09-21T09:00:14.000 BID_BEST 3.9070  2000
6  2010-09-21T09:00:55.000 BID_BEST 3.9080  2000
7  2010-09-21T09:00:55.000 BID_BEST 3.9090  2000
8  2010-09-21T09:01:34.000 BID_BEST 3.9100   243
9  2010-09-21T09:01:54.000 BID_BEST 3.9090  2000
10 2010-09-21T09:02:04.000 BID_BEST 3.9100   243
11 2010-09-21T09:02:15.000 BID_BEST 3.9090  2000
12 2010-09-21T09:03:07.000 BID_BEST 3.9080  2000
13 2010-09-21T09:03:13.000 BID_BEST 3.9100  6650
14 2010-09-21T09:06:17.000    TRADE 3.9100 35000
15 2010-09-21T09:07:08.000    TRADE 3.9111  2000
16 2010-09-21T09:08:27.000 BID_BEST 3.9100  8150
17 2010-09-21T09:08:49.000 BID_BEST 3.9100  8650
>
>
> proc.time()
   user  system elapsed
   1.45    0.78    3.48
```

3.5 Bar Data

Period start and end times must be expressed as strings in UTC time.

```
bar <- function(conn, security, field, start_date_time, end_date_time, interval)
```

```
> library(RBloomberg)
Loading required package: rJava
> conn <- blpConnect()
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
```

```
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
Bloomberg API Version 3.3.3.3
> bar(conn, "RYA ID Equity", "TRADE", "2010-09-21 09:00:00.000", "2010-09-21 15:00:00.000", "60")
                                          time   open  high    low close
2010-09-21T09:00:00.000 2010-09-21T09:00:00.000 3.9100 3.920 3.9090 3.920
2010-09-21T10:00:00.000 2010-09-21T10:00:00.000 3.9200 3.920 3.9000 3.903
2010-09-21T11:00:00.000 2010-09-21T11:00:00.000 3.9000 3.900 3.8950 3.900
2010-09-21T12:00:00.000 2010-09-21T12:00:00.000 3.9021 3.905 3.9021 3.905
2010-09-21T13:00:00.000 2010-09-21T13:00:00.000 3.9150 3.915 3.9000 3.900
2010-09-21T14:00:00.000 2010-09-21T14:00:00.000 3.9000 3.906 3.8930 3.895
                        numEvents volume
2010-09-21T09:00:00.000        28 852072
2010-09-21T10:00:00.000        15 247946
2010-09-21T11:00:00.000         6  74187
2010-09-21T12:00:00.000         2  30000
2010-09-21T13:00:00.000        10 270940
2010-09-21T14:00:00.000        20 543604
>
>
> proc.time()
   user  system elapsed
   1.35    0.71    2.70
```

4 Overrides, Options, Tips and Tricks

4.1 Ticker Format

To use a SEDOL or CUSIP to identify a security, you need to prefix the identifier with /SEDOL1/ or /CUSIP/. Bloomberg tickers themselves have an implicit /ticker/ before them. It is possible, though not yet implemented for RBloomberg, to change the default from /ticker/ to /CUSIP/ (or any other prefix) at the session level.

4.2 Using Overrides

The Bloomberg API allows you to make certain overrides to change the value that a function returns. We give a few examples here, to determine whether a field you wish to request can be overridden, consult the FLDS screen in your Bloomberg terminal (look for What Overrides Me).

Note that dates must be passed as strings in the form "YYYYMMDD". If you pass an invalid override field, or one that doesn't apply to the main field you are requesting, this will be silently ignored. So, make sure you know that your request is valid before you count on the returned value being correct. This is a feature rather than a bug since it makes it possible to specify override fields which only apply to 1 field requested. So, you can ask for CRNCY_ADJ_PX_LAST and NAME in the same request, and you don't get an error saying that EQY_FUND_CRNCY doesn't apply to the NAME field.

4.3 Setting Options

The new Bloomberg API has per-request options. See Historical Data for an example of periodicitySelection.

All options need to be provided as strings, where necessary they will be converted to the correct data type within the Java code. The following named options will be converted to Datetime and Boolean types respectively, all other options will be sent to the Bloomberg API as strings. If you are aware of an option that should be converted to a different data type which is not on this list, please report this issue.

```
> library(RBloomberg)
Loading required package: rJava
> conn <- blpConnect()
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
Bloomberg API Version 3.3.3.3
>
> for (name in conn$DATETIME_OPTION_NAMES) {
+   print(name)
+ }
[1] "startDateTime"
[1] "endDateTime"
>
> for (name in conn$BOOLEAN_OPTION_NAMES) {
+   print(name)
+ }
[1] "useUTCTime"
[1] "returnRelativeDate"
[1] "adjustmentNormal"
[1] "adjustmentAbnormal"
[1] "adjustmentSplit"
[1] "adjustmentFollowDPDF"
[1] "returnEids"
[1] "includeConditionCodes"
[1] "includeNonPlottableEvents"
[1] "includeExchangeCodes"
>
> blpDisconnect(conn)
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 216145  5.8     407500 10.9   350000  9.4
Vcells 111783  0.9     786432  6.0   390780  3.0
>
>
> proc.time()
   user  system elapsed
   1.23    0.68    2.21
```

4.4 Ignoring Ticker Errors

By default, an error will be raised if you request data for an invalid ticker. If you would prefer to have NA values returned silently when you pass an invalid ticker, then connect to RBloomberg as follows:

```
> library(RBloomberg)
Loading required package: rJava
>
> conn <- blpConnect(throw.ticker.errors = FALSE)
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
Bloomberg API Version 3.3.3.3
> bdp(conn, "THIS IS NOT A VALID TICKER", "NAME")
```

```
                          NAME
THIS IS NOT A VALID TICKER <NA>
>
> securities <- c("AMZN US Equity", "OCN US Equity", "123456 XX Equity")
> fields <- c("NAME", "PX_LAST", "TIME", "SETTLE_DT")
> bdp(conn, securities, fields)
                               NAME PX_LAST     TIME  SETTLE_DT
AMZN US Equity         AMAZON.COM INC  160.00 15:46:01 2010-09-30
OCN US Equity    OCWEN FINANCIAL CORP    9.96 15:43:32 2010-09-30
123456 XX Equity                <NA>      NA     <NA>       <NA>
>
> blpDisconnect(conn)
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 224527  6.0     407500 10.9   350000  9.4
Vcells 113000  0.9     786432  6.0   390780  3.0
>
>
>
> conn <- blpConnect(throw.ticker.errors = TRUE)
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
Bloomberg API Version 3.3.3.3
> try(bdp(conn, "THIS IS NOT A VALID TICKER", "NAME"))
Error in .jcall("RJavaTools", "Ljava/lang/Object;", "invokeMethod", cl,  :
  org.findata.blpwrapper.WrapperException: invalid security THIS IS NOT A VALID TICKER
> blpDisconnect(conn)
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 224701  6.1     467875 12.5   396841 10.6
Vcells 113059  0.9     786432  6.0   390780  3.0
>
>
>
> proc.time()
   user  system elapsed
   1.48    0.87    3.37
```

4.5 Field Lookup

5 Troubleshooting

If you are having difficulty with RBloomberg, check the log files (located in your home directory, probably Documents and Settings/username) for possible error messages. Bloomberg creates its own very basic log files called blpjavaapi, and RBloomberg logs to org.findata.blpwrapper. You can make RBloomberg's logging much more verbose if you initialize it as follows:

```
> library(RBloomberg)
Loading required package: rJava
> conn <- blpConnect(log.level = "finest")
R version 2.11.1 (2010-05-31)
rJava Version 0.8-7
RBloomberg Version 0.4-143
Java environment initialized successfully.
Looking for most recent blpapi3.jar file...
```

```
Bloomberg API Version 3.3.3.3
>
>
> proc.time()
   user  system elapsed
   1.17    0.67    2.09
```

You might have a typo in your ticker or in the fields you are requesting, or the fields might not be applicable to the ticker. Try making the same request from within your Bloomberg terminal. The FLDS page gives you access to all the same fields from RBloomberg, so you can re-create the request you are trying to make (for one security at a time). You could also use the Excel API to double-check your request.

Also, check any override fields you are using.