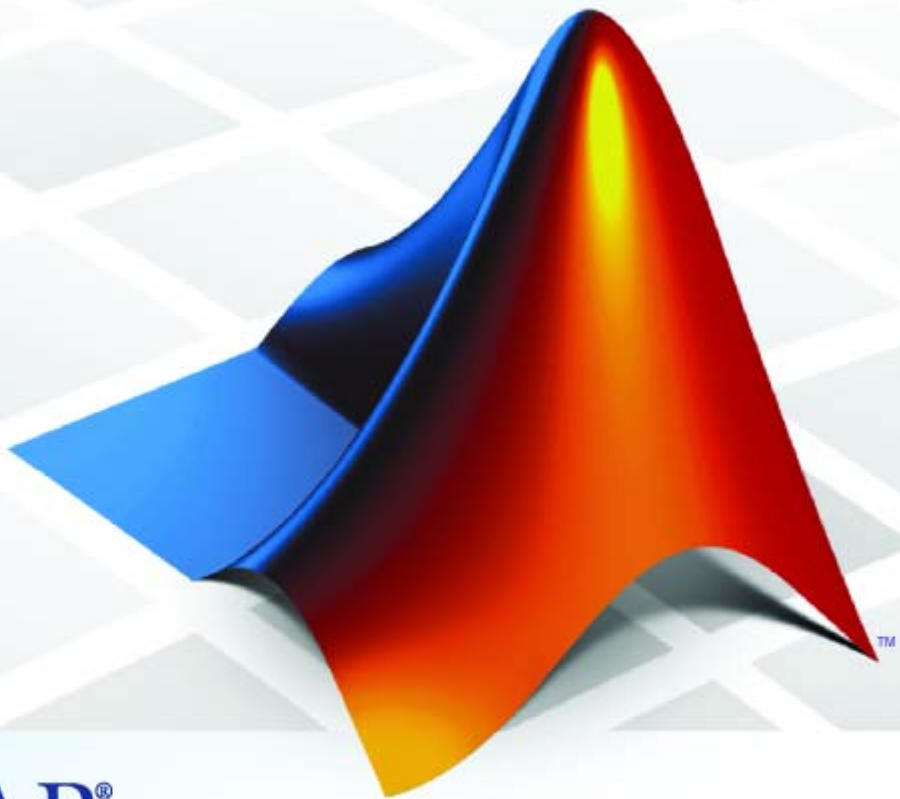


# Datafeed Toolbox™ 3

## User's Guide



**MATLAB®**

## How to Contact The MathWorks



[www.mathworks.com](http://www.mathworks.com) Web  
[comp.soft-sys.matlab](mailto:comp.soft-sys.matlab) Newsgroup  
[www.mathworks.com/contact\\_TS.html](http://www.mathworks.com/contact_TS.html) Technical Support



[suggest@mathworks.com](mailto:suggest@mathworks.com) Product enhancement suggestions  
[bugs@mathworks.com](mailto:bugs@mathworks.com) Bug reports  
[doc@mathworks.com](mailto:doc@mathworks.com) Documentation error reports  
[service@mathworks.com](mailto:service@mathworks.com) Order status, license renewals, passcodes  
[info@mathworks.com](mailto:info@mathworks.com) Sales, pricing, and general information



508-647-7000 (Phone)



508-647-7001 (Fax)



The MathWorks, Inc.  
3 Apple Hill Drive  
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

*Datafeed Toolbox™ User's Guide*

© COPYRIGHT 1999–2008 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### Patents

The MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

## Revision History

|                |                 |   |
|----------------|-----------------|---|
| December 1999  | First printing  | New for MATLAB® 5.3 (Release 11)        |
| June 2000      | Online only     | Revised for Version 1.2                 |
| December 2000  | Online only     | Revised for Version 1.3                 |
| February 2003  | Online only     | Revised for Version 1.4                 |
| June 2004      | Online only     | Revised for Version 1.5 (Release 14)    |
| August 2004    | Online only     | Revised for Version 1.6 (Release 14+)   |
| September 2005 | Second printing | Revised for Version 1.7 (Release 14SP3) |
| March 2006     | Online only     | Revised for Version 1.8 (Release 2006a) |
| September 2006 | Online only     | Revised for Version 1.9 (Release 2006b) |
| March 2007     | Third printing  | Revised for Version 2.0 (Release 2007a) |
| September 2007 | Online only     | Revised for Version 3.0 (Release 2007b) |
| March 2008     | Online only     | Revised for Version 3.1 (Release 2008a) |
| October 2008   | Online only     | Revised for Version 3.2 (Release 2008b) |



## Getting Started

### 1

|  |     |
|--|-----|
| <b>Product Overview</b> .....                              | 1-2 |
| <b>About Data Servers and Data Service Providers</b> ..... | 1-3 |
| Supported Data Service Providers .....                     | 1-3 |
| Data Server Connection Requirements .....                  | 1-3 |

## Communicating with Financial Data Servers

### 2

|   |     |
|---|-----|
| <b>Communication Management</b> .....                               | 2-2 |
| Communicating with Data Servers .....                               | 2-2 |
| Core Functions .....  | 2-2 |
| Connecting to the Bloomberg Data Server .....                       | 2-3 |
| <b>Verifying Connections</b> .....                                  | 2-4 |
| <b>Retrieving Connection Properties</b> .....                       | 2-5 |
| How to Retrieve Connection Properties .....                         | 2-5 |
| Example: Retrieving Bloomberg Connection Object<br>Properties ..... | 2-5 |
| <b>Disconnecting from Data Servers</b> .....                        | 2-7 |

## Retrieving Data

### 3

|  |     |
|--|-----|
| <b>Using the Fetch Function to Retrieve Data</b> ..... | 3-2 |
|--|-----|

|   |            |
|---|------------|
| <b>Example: Retrieving Bloomberg Data</b> ..... | <b>3-3</b> |
| About This Example .....                        | <b>3-3</b> |
| Retrieving Header Data .....                    | <b>3-3</b> |
| Retrieving Field Data .....                     | <b>3-6</b> |
| Retrieving Time Series Data .....               | <b>3-7</b> |
| Retrieving Historical Data .....                | <b>3-8</b> |
| Finding Ticker Symbols .....                    | <b>3-9</b> |

## Datafeed Toolbox Graphical User Interface

# 4

|   |            |
|---|------------|
| <b>Introduction</b> .....                             | <b>4-2</b> |
| <br>  |            |
| <b>Using the Datafeed Dialog Box</b> .....            | <b>4-3</b> |
| About the Datafeed Dialog Box .....                   | <b>4-3</b> |
| Connecting to Data Servers .....                      | <b>4-4</b> |
| Retrieving Data .....                                 | <b>4-5</b> |
| Using the Datafeed Securities Lookup Dialog Box ..... | <b>4-6</b> |
| Setting Overrides .....                               | <b>4-8</b> |

## Function Reference

# 5

|  |            |
|--|------------|
| <b>Bloomberg</b> .....                               | <b>5-2</b> |
| <br>   |            |
| <b>Datastream</b> .....                              | <b>5-3</b> |
| <br>   |            |
| <b>FactSet</b> .....                                 | <b>5-4</b> |
| <br>   |            |
| <b>FRED</b> .....                                    | <b>5-5</b> |
| <br>   |            |
| <b>Haver Analytics</b> .....                         | <b>5-6</b> |
| <br>   |            |
| <b>Interactive Data Pricing and RemotePlus</b> ..... | <b>5-7</b> |

|   |             |
|---|-------------|
| <b>Kx Systems</b> .....                     | <b>5-8</b>  |
| <b>Reuters</b> .....                        | <b>5-9</b>  |
| <b>Reuters Datascope Tick History</b> ..... | <b>5-10</b> |
| <b>Reuters Knowledge Direct</b> .....       | <b>5-11</b> |
| <b>Reuters Newscope</b> .....               | <b>5-12</b> |
| <b>Yahoo!</b> .....                         | <b>5-13</b> |

## Functions — Alphabetical List

# 6

### Examples

# A

|  |            |
|--|------------|
| <b>Communicating with Financial Data Servers</b> ..... | <b>A-2</b> |
| <b>Retrieving Connection Properties</b> .....          | <b>A-2</b> |
| <b>Retrieving Data</b> .....                           | <b>A-2</b> |

### Index





# Getting Started

---

- “Product Overview” on page 1-2
- “About Data Servers and Data Service Providers” on page 1-3

## Product Overview

This toolbox, used with the MATLAB® product, effectively turns your MATLAB workstation into a financial data acquisition terminal. The toolbox enables you to:

- Retrieve and analyze a wide variety of security data from financial data servers in MATLAB.
- Access market, time-series, and historical market data in MATLAB.
- Monitor the status and history of each connection to a supported data service provider.
- Fetch data fields for multiple securities in a single call.
- Look up security ticker symbols from the toolbox GUI or the MATLAB command line.

## About Data Servers and Data Service Providers

| In this section...                                |
|---|
| “Supported Data Service Providers” on page 1-3    |
| “Data Server Connection Requirements” on page 1-3 |

### Supported Data Service Providers

This toolbox supports connections to financial data servers that the following corporations provide:

- Bloomberg L.P. (<http://www.bloomberg.com>)
- FactSet Research Systems, Inc. (<http://www.factset.com>)
- Federal Reserve Economic Data (FRED) (<http://research.stlouisfed.org/fred2/>)
- Haver Analytics (<http://www.haver.com>)
- Interactive Data Pricing and Reference Data (<http://www.interactivedata-prd.com/>)
- Kx Systems, Inc. (<http://www.kx.com>)
- Reuters (<http://about.reuters.com/>)
- Thomson <http://www.thomson.com>)
- Yahoo!, Inc. (<http://finance.yahoo.com>)

### Data Server Connection Requirements

To connect to some of these data servers, additional requirements apply.

### Additional Software Requirements

The following data service providers require you to install proprietary software on your PC:

- Bloomberg

---

**Note** You must have a Bloomberg® software license for the host on which the Datafeed Toolbox™ and MATLAB software are running.

---

- Interactive Data Pricing and Reference Data's RemotePlus™
- Haver Analytics
- Kx Systems. Inc.
- Reuters

You must have a valid license for required client software on your machine. If you do not, the following error message appears when you try to connect to a data server:

```
Invalid MEX-file
```

For more information about how to obtain required software, contact your data server sales representative.

## **Proxy Information Requirements**

The following data service providers may require you to specify a proxy host and proxy port plus a username and password if the user's site requires proxy authentication:

- FactSet
- Federal Reserve Economic Data
- Thomson Datastream
- Yahoo!

For information on how to specify these settings, see “Internet Connection and Fonts for Web Browser — Web Preferences” in the MATLAB documentation.

## **FactSet Data Service Requirements**

You need a license to use FactSet® FAST technology. For more information, see the FactSet Web site at <http://www.factset.com>.

## Reuters Data Service Requirements

**Configuring Reuters® Connections Using the Reuters Configuration Editor software.** You must use the Reuters Configuration Editor to configure your connections as follows:

- 1 In a DOS prompt, set your CLASSPATH environment variable:

```
set CLASSPATH=%CLASSPATH%; ...  
$MATLAB/toolbox/datafeed/datafeed/config_editor.jar
```

- 2 Navigate to the Datafeed Toolbox directory:

```
cd %MATLAB%\toolbox\datafeed\datafeed
```

- 3 Enter the following command to run the Reuters Configuration Editor:

```
starteditor
```

- 4 Load the sample configuration file.

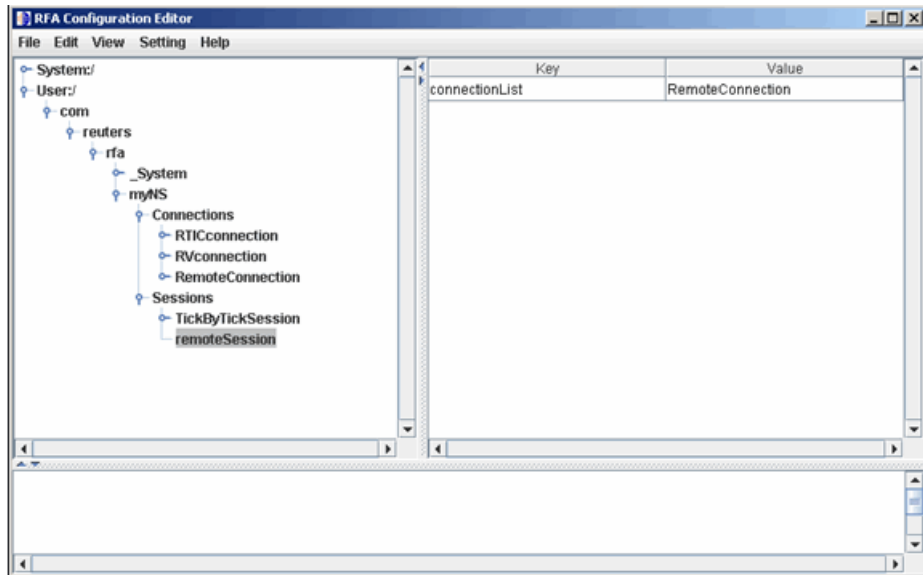
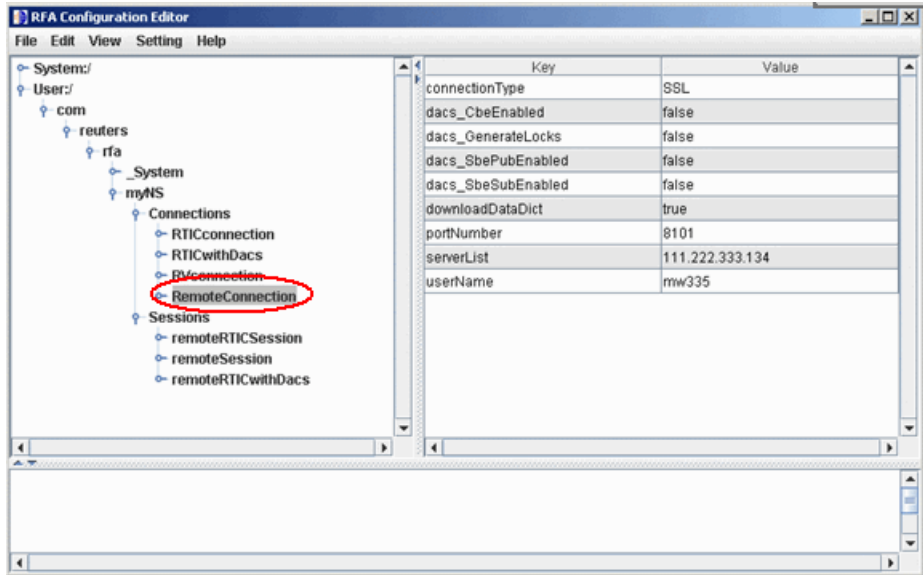
- a Click **File > Import > File**.

- b Select the file

```
%MATLAB%\toolbox\datafeed\datafeed\sampleconfig.xml.
```

- 5 Modify `sampleconfig.xml` based on the site-specific settings that you obtain from Reuters.

- 6 Define a namespace, a connection, and a session associated with the connection. The following example adds the session `remoteSession` with the namespace `MyNS` to the connection list for the connection `remoteConnection`.



7 If you are not DACS enabled, disable DACS.

- a Add the following to your connection configuration:

```
dacs_CbeEnabled=false
dacs_SbePubEnabled=false
dacs_SbeSubEnabled=false
```

- b If you are running an SSL connection, add the following to your connection configuration:

```
dacs_GenerateLocks=false
```

For more information, see the `reuters` function reference page.

**Troubleshooting Issues with Reuters Configuration Editor.** These errors occur when you attempt to use the Reuters Configuration Editor to configure connections on a machine on which an XML Parser is not installed.

```
java com.reuters.rfa.tools.config.editor.ConfigEditor
org.xml.sax.SAXException: System property
org.xml.sax.driver not specified
at org.xml.sax.helpers.XMLReaderFactory.createXMLReader(Unknown
Source)
at com.reuters.rfa.tools.config.editor.rfaConfigRuleDB.rfaConfig
RuleDB.java:56)
at com.reuters.rfa.tools.config.editor.ConfigEditor.init
(ConfigEditor.java:86)
at (com.reuters.rfa.tools.config.editor.ConfigEditor.
(ConfigEditor.java:61) at
com.reuters.rfa.tools.config.editor.ConfigEditor.main
(ConfigEditor.java:1303)
```

To address this problem, download an XML parser file, and then include a path to this file in your CLASSPATH environment variable.

The following example shows how to set your CLASSPATH environment variable to include the XML parser file `C:\xerces.jar` (downloaded from <http://xerces.apache.org/xerces-j/index.html>):

```
set CLASSPATH=%CLASSPATH%;...
matlabroot\toolbox\datafeed\datafeed\config_editor.jar;...
c:\xerces.jar
```

## **Thomson Data Service Requirements**

You need the following to connect to Thomson® data servers:

- A license for Thomson® DataWorks®.
- To connect to the Thomson® Datastream® API from the Web, you need a user name, password, and URL provided by Thomson.

For more information, see the Thomson Web site at <http://www.thomson.com>.



# Communicating with Financial Data Servers

---

- “Communication Management” on page 2-2
- “Verifying Connections” on page 2-4
- “Retrieving Connection Properties” on page 2-5
- “Disconnecting from Data Servers” on page 2-7

# Communication Management

| In this section...                                    |
|---|
| “Communicating with Data Servers” on page 2-2         |
| “Core Functions” on page 2-2                          |
| “Connecting to the Bloomberg Data Server” on page 2-3 |

## Communicating with Data Servers

This section uses the Bloomberg financial data server as an example of how to retrieve data with the Datafeed Toolbox software. You can communicate with other supported data servers using a similar set of toolbox functions.

## Core Functions

The following set of core functions manage communication with each supported financial data server.

- To establish a connection to the appropriate data server, use:
  - bloomberg
  - datastream
  - factset
  - fred
  - haver
  - idc
  - kx
  - reuters
  - yahoo
- To verify that a connection is working, use `isconnection`.
- To retrieve connection properties, use `get`.
- To terminate a connection, use `close`.

- To retrieve data from the data server and transfer it to your computer, use `fetch`.

## Connecting to the Bloomberg Data Server

This example shows how to use the `bloomberg` function to connect to the Bloomberg data server.

The syntax for this function is:

```
Connect = bloomberg
```

This function returns a Bloomberg connection object `c`:

```
c =  
  
  connection: 84554360  
  ipaddress: '123.456.54.123'  
  port: 8194
```

The `connection` field within the object `c` contains the Bloomberg connection handle that the `bloomberg` function uses to process future data requests.

### Verifying Connections

To verify that a connection to a data server is valid and open, use the data server's `isconnection` function. For a connection object `c` previously created with a connection function, the following command:

```
x = isconnection(c)
```

Returns `x = 1` if the connection is valid and open, or `x = 0` if the connection is closed or invalid.

## Retrieving Connection Properties

### In this section...

“How to Retrieve Connection Properties” on page 2-5

“Example: Retrieving Bloomberg Connection Object Properties” on page 2-5

### How to Retrieve Connection Properties

To retrieve the properties of a connection object, use the `get` function. This function returns different values depending upon which data server you are using.

### Example: Retrieving Bloomberg Connection Object Properties

Establish a connection, `c`, to a Bloomberg data server:

```
c = bloomberg
```

The following command returns the list of all valid connection properties of `c`, and their values:

```
p = get(c)
p =
  connection: 84554360
  ipaddress:  '123.456.54.123'
  port: 8194
  socket: 248
  version: 1.8000
```

The `get` function can return specific properties of a connection object. For example, to obtain the port number and version of the connection object `c` run the command:

```
p = get(c,{'Port';'Version'})
p =
  port: 8194
  version: 1.8000
```

Use the following command to return a single property, in this case, the connection handle:

```
p = get(c, 'Connection')  
p =  
    84554360
```

---

**Note** A single property is not returned as a structure.

---

## Disconnecting from Data Servers

To close a data server connection and disconnect, use the `close` function with the format:

```
close(Connect)
```

You must have previously created the connection object with one of the connection functions. For more information, see Chapter 5, “Function Reference”.





# Retrieving Data

---

- “Using the Fetch Function to Retrieve Data” on page 3-2
- “Example: Retrieving Bloomberg Data” on page 3-3

### Using the Fetch Function to Retrieve Data

The `fetch` function controls data retrieval from a data server connection. `fetch` returns different information depending upon which data server is being accessed. For further information, see the version of `fetch` appropriate for your data server.

The following example shows how to use the `fetch` function to retrieve data from a Bloomberg data server.

## Example: Retrieving Bloomberg Data

### In this section...

“About This Example” on page 3-3  
“Retrieving Header Data” on page 3-3  
“Retrieving Field Data” on page 3-6  
“Retrieving Time Series Data” on page 3-7  
“Retrieving Historical Data” on page 3-8  
“Finding Ticker Symbols” on page 3-9

### About This Example

The following example illustrates the use of the `bloomberg.fetch` function to retrieve data from a Bloomberg data server. Versions of the `fetch` function that retrieve data from other data servers work similarly.

### Retrieving Header Data

A header (default) data request to a Bloomberg data server returns a fixed set of field data. Not all fields in the header data are relevant for a specific security.

### Determining Header Fields

The list of valid header fields is stored in the file `@bloomberg/bbfields.mat`. To load this file, use the MATLAB load command:

```
load @bloomberg/bbfields
```

The variable `headerfieldnames` contains the list of header field names.

### Obtaining Data

To retrieve header data using a connection to a Bloomberg data server, use the `fetch` function with the following syntax:

```
data = fetch(Connect, 'Security', 'HEADER', 'Flag')
```

Where:

- Connect is a Bloomberg connection object established with the `bloomberg` function.
- Security is the list of securities for which you request data.

---

**Note** Security names are case sensitive for the Bloomberg `fetch` function.

---

- The 'HEADER' argument is entered literally.
- 'Flag' denotes dates for which you can retrieve data. Flag has three possible values:
  - 'DEFAULT' fills all fields with data from the most recent date with a bid, ask, or trade.
  - 'TODAY' fills the fields with data from today only.
  - 'ENHANCED' fills the fields with data for the most recent event for each individual field. In this case, for example, the bid and ask group fields could come from different dates.

Commands of the form:

```
data = fetch(Connection, Security)
data = fetch(Connection, Security, 'HEADER')
data = fetch(Connection, Security, 'HEADER', 'DEFAULT')
```

Are equivalent.

The returned data has a fixed set of fields. For example, a header inquiry for the security IBM US Equity returns data of the form:

```
Status:0
      Open:93
TodaysOpenPrice:93
      HighPrice:93.1875
TodaysHighPrice:93.1875
      LowPrice:89
```

```
TodaysLowPrice:89
  LastPrice:90.9375
TodaysLastPrice:0
  SettlePrice:NaN
  BidPrice:0
TodaysBidPrice:NaN
  AskPrice:0
TodaysAskPrice:NaN
  YieldBid:NaN
TodaysYieldBid:NaN
  YieldAsk:NaN
TodaysYieldAsk:NaN
  LimitUp:NaN
  LimitDown:NaN
  OpenInterest:3359000
LastPriceYesterday:95
  Scale:1
  LastPriceTime:0.4993
LastTradeExchange:7
  TickDirection:-1
  BidSize:0
  TodaysBidSize:NaN
  AskSize:NaN
  TodaysAskSize:0
  BidCondition:NaN
  AskCondition:NaN
LastTradeCondition:NaN
LastMarketCondition:NaN
  Monitorable:1
  TotalVolume:60018500
  TodaysTotalVolume:0
  TotalNumberOfTicks:63318
TodaysTotalNumberOfTicks:63318
  SessionStartTime:0.3958
  SessionEndTime:0.6875
  Currency:538989397
  Format:0
  SecurityKey:{'IBM US Equity'}
  AsOfDate:730441
  TodaysAsOfDate:730441
```

## Retrieving Field Data

The `fetch` function with the `GETDATA` argument obtains Bloomberg field data. The entire set of field data provides statistics for all possible securities, but it does not apply universally to any one security.

## Determining Field Names

The file `@bloomberg/bbfields.mat` stores the complete list of valid field names. To load this file, use the function:

```
load @bloomberg/bbfields
```

This command returns the following list of variables:

```
bbcategories  
bbcurrency  
bbdatamask  
bbfieldids  
bbfieldnames  
bbfieldtypes  
bbhelpfields  
bboverrides  
bbsectype  
headerfieldnames
```

The variable `bbfieldnames` contains a list of field names. This list includes the header field names plus many others. The other variables loaded extend the list of field names.

## Obtaining Data

To obtain data for specific fields of a given security, use the `fetch` function with the following syntax:

```
d = fetch(Connect, Security, 'GETDATA', Fields)
```

For example, use the Bloomberg connection object `c` to retrieve the values of the fields `Open` and `Last_Price`:

```
d = fetch(c, 'IBM US Equity', 'GETDATA', {'Open'; 'Last_Price'})
```

```
d =
  Open: 126.2500
  Last_Price: 125.1250
```

## Retrieving Time Series Data

The `fetch` function called with the 'TIMESERIES' argument returns price and volume data for a particular security on a specified date. Use the following command to return time-series data for a given security and a specific date:

```
data = fetch(Connection, Security, 'TIMESERIES', Date)
```

Date can be a MATLAB date string or serial date number.

To obtain time-series data for the current day, use the alternate form of the function:

```
data = fetch(Connection, Security, 'TIMESERIES', now)
```

To obtain time-series data for IBM using an existing connection `c1`, enter the function:

```
data = fetch(c1, 'IBM US Equity', 'TIMESERIES', '11/16/99')
data =
 31.00    730440.31    130.00    1000.00
 32.00    730440.31    130.00     200.00
 32.00    730440.35    129.50   10000.00
 31.00    730440.35    129.50    100.00
 32.00    730440.35    129.50    100.00
  1.00    730440.56    129.25   4000.00
 31.00    730440.56    129.38   1500.00
 32.00    730440.56    129.50    500.00
  1.00    730440.56    129.63   5000.00
 31.00    730440.56    129.63    400.00
 32.00    730440.56    129.63    200.00
  1.00    730440.56    129.69   5000.00
 31.00    730440.56    129.69    500.00
 32.00    730440.56    129.69    500.00
 31.00    730440.56    129.75    100.00
 32.00    730440.56    130.00    100.00
  1.00    730440.56    130.00   5000.00
```

|       |           |        |         |
|-------|-----------|--------|---------|
| 1.00  | 730440.56 | 129.88 | 5000.00 |
| 31.00 | 730440.56 | 129.88 | 300.00  |

Column 1 contains the tick type flag. Column 2 contains the time stamp in MATLAB serial date number format. Column 3 contains the tick value. Column 4 contains the number of shares in the transaction.

## Retrieving Historical Data

Use the `fetch` function with the 'HISTORY' argument to obtain historical data for a specific security.

To obtain historical data for a specified field of a particular security, run:

```
d = fetch(Connect,Security,'HISTORY',Field,FromDate,ToDate)
```

`fetch` returns data for the date range from `FromDate` to `ToDate`.

For instructions on determining valid field names, see “Determining Field Names” on page 3-6.

For example, to obtain the closing price for IBM for the dates July 15, 1999 to August 2, 1999 using the connection `c1`, enter:

```
data = fetch(c1, 'IBM US Equity', 'HISTORY', 'Last_Price',...  
'07/15/99', '08/02/99')  
data =  
730316.00      136.31  
730317.00      136.25  
730320.00      134.63  
730321.00      128.25  
730322.00      129.00  
730323.00      123.88  
730324.00      124.81  
730327.00      123.00  
730328.00      126.25  
730329.00      128.38  
730330.00      125.38  
730331.00      125.69  
730334.00      122.25
```



Column 1 contains the date represented as a MATLAB date number, and column 2 contains the last price.

## Finding Ticker Symbols

You can use the `fetch` function with the `'LOOKUP'` argument to find a ticker symbol when you are not sure what the symbol is. To locate a specific ticker symbol, use the following syntax:

```
data = fetch(Connect, SearchString, 'LOOKUP', Market)
```

The `SearchString` argument is the comparison string used in the lookup operation. `Market` indicates the market in which the security trades. Allowable values for `Market` are:

- `'Comdty'` (Commodities)
- `'Corp'` (Corporate bonds)
- `'Curncy'` (Currencies)
- `'Equity'` (Equities)
- `'Govt'` (Government bonds)
- `'Index'` (Indexes)
- `'M-Mkt'` (Money Market securities)
- `'Mtge'` (Mortgage-backed securities)
- `'Muni'` (Municipal bonds)
- `'Pfd'` (Preferred stocks)

For example, using `fetch` with the connection `c1` to look up the ticker symbol for New Zealand government bonds returns a list of possible values:

```
data = fetch(c1, 'New', 'LOOKUP', 'Govt')
data =
'NZTB      New Zealand Treasury Bill NZGB      New Zealand Governme'
'NZGB      New Zealand Government Bond NZ      New Zealand Govern'
'NZ        New Zealand Government International Bond HCNZ  Hous'
'ECNZ      Electric Corporation of New Zealand Bond ...
```

NZTB NZGB NZ H'

# Datafeed Toolbox Graphical User Interface

---

- “Introduction” on page 4-2
- “Using the Datafeed Dialog Box” on page 4-3

# Introduction

You can use the Datafeed Toolbox Graphical User Interface (GUI) to connect to and retrieve information from some supported data service providers.

This GUI consists of two dialog boxes:

- The Datafeed dialog box. Use this dialog box to connect to and retrieve data from the following service providers:
  - Bloomberg
  - Interactive Data Pricing and Reference Data's RemotePlus
  - Yahoo!®

For more information on how to use this dialog box, see “Using the Datafeed Dialog Box” on page 4-3.

- The Securities Lookup dialog box. You can use this dialog box to find the ticker symbol for a security when you know part of the security name. Use this dialog box with connections to the following service providers:
  - Bloomberg
  - Interactive Data Pricing and Reference Data's RemotePlus

For more information on how to use this dialog box, see “Using the Datafeed Securities Lookup Dialog Box” on page 4-6.

## Using the Datafeed Dialog Box

### In this section...

“About the Datafeed Dialog Box” on page 4-3

“Connecting to Data Servers” on page 4-4

“Retrieving Data” on page 4-5

“Using the Datafeed Securities Lookup Dialog Box” on page 4-6

“Setting Overrides” on page 4-8

### About the Datafeed Dialog Box

The Datafeed dialog box establishes the connection with the data server and manages data retrieval. To display this dialog box, enter the `dftool` command in the MATLAB Command Window.

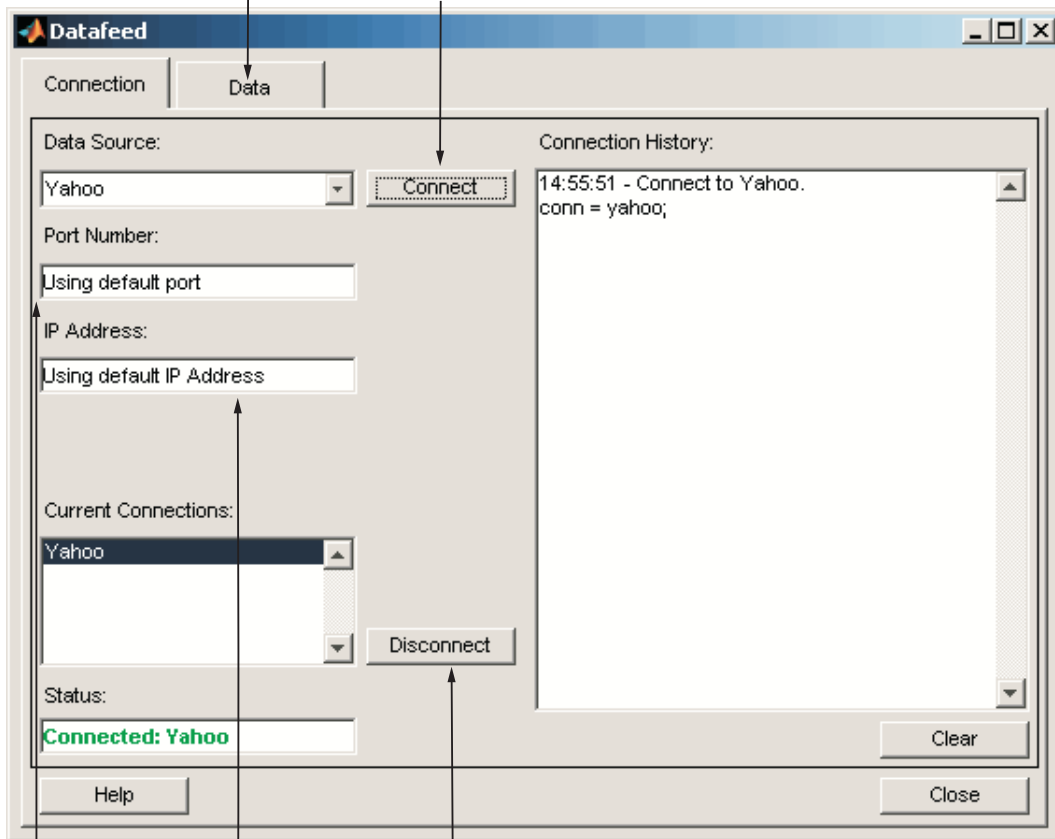
The Datafeed dialog box consists of two tabs:

- The **Connection** tab establishes communication with a data server. For more information, see “Connecting to Data Servers” on page 4-4.
- The **Data** tab specifies the data request. For more information, see “Retrieving Data” on page 4-5.
- You can also set overrides for the data you retrieve. For more information, see “Setting Overrides” on page 4-8.

The following figure summarizes how to connect to data servers and retrieve data using the Datafeed dialog box.

4. After the connection is made, click the Data tab to begin data retrieval.

3. Click to establish a connection to the data server.



5. Click to close the highlighted connection.

2. Enter IP address of data server or use the default values (Bloomberg data servers only).

1. Enter port number on data server (Bloomberg data servers only).

### The Datafeed Dialog Box

## Connecting to Data Servers

1 Click the **Connect** button to establish a connection.

- 2** When the **Connected** message appears in the **Status** field, click the **Data** tab to begin the process of retrieving data from the data server. For more information, see “Retrieving Data” on page 4-5.
- 3** Click the **Disconnect** button to terminate the session highlighted in the **Current Connections** box.

For Bloomberg data servers, you must also specify the port number and IP address of the server:

- 1** Enter the port number on the data server in the **Port Number** field.
- 2** Enter the IP address of the data server in the **IP Address** field.
- 3** To establish a connection to the Bloomberg data server, follow steps 1 through 3 above.

---

**Tip** You can also connect to the Bloomberg data server by selecting the **Connect** button and accepting the default values.

---

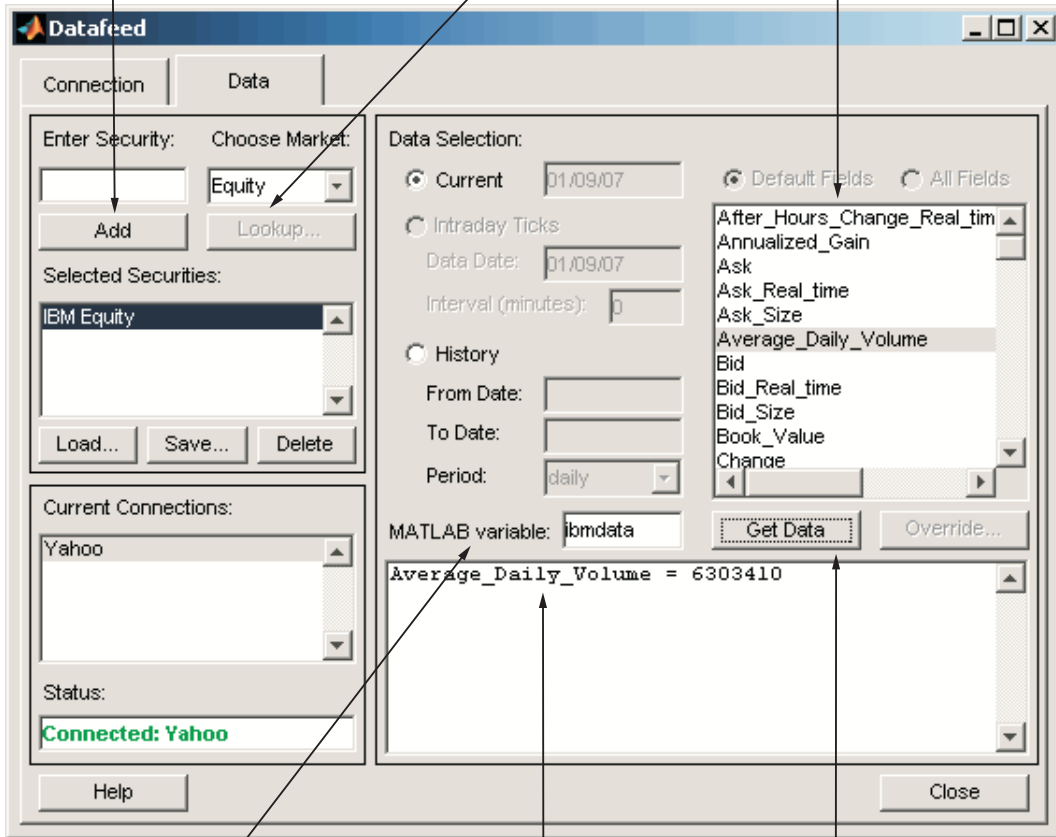
## Retrieving Data

The **Data** tab allows you to retrieve data from the data server as follows:

- 1** Enter the security symbol in the **Enter Security** field.
- 2** Indicate the type of data to retrieve in the **Data Selection** field.
- 3** Specify whether you want the default set of data, or the full set:
  - Select the **Default fields** button for the default set of data.
  - Select the **All fields** button for the full set of data.
- 4** Click the **Get Data** button to retrieve the data from the data server.
- 5** (Optional) Click the **Override** button if you want to set overrides on the data you request from the data server. For more information, see “Setting Overrides” on page 4-8.

The following figure summarizes these steps.

- 2. Enter security symbol if known, or click **Add** button to add security to **Selected Securities** list.
- 2a. Use to find security symbol, if unknown. (For Bloomberg and Interactive Data Pricing and Reference Data data servers only)
- Security fields.



## Using the Datafeed Securities Lookup Dialog Box

When requesting data from Bloomberg or Interactive Data Pricing and Reference Data's RemotePlus servers, you can use the Datafeed Securities

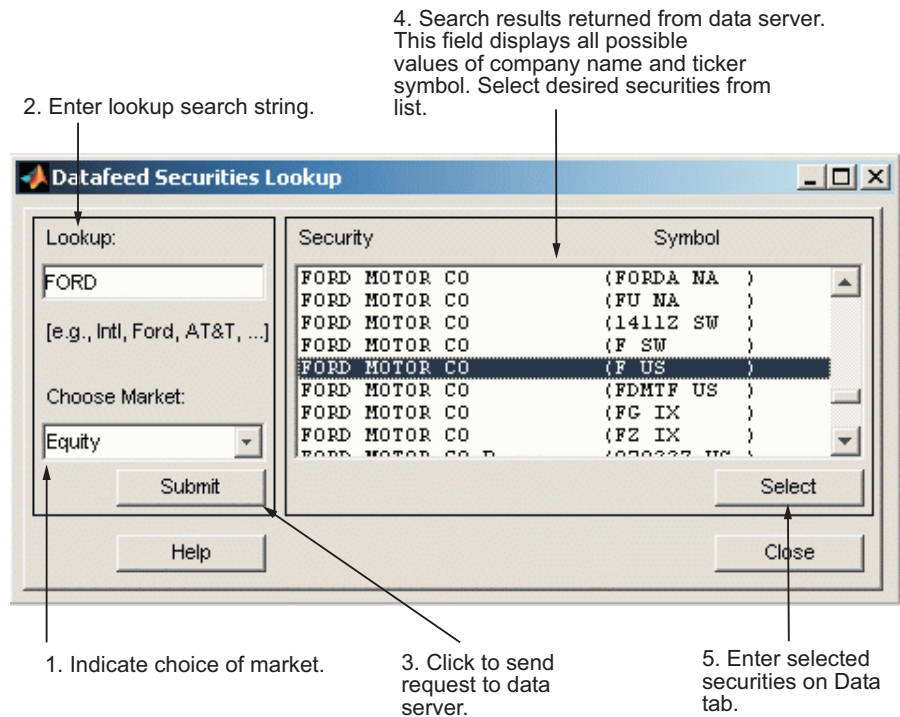


Lookup dialog box to obtain the ticker symbol for a given security if you know only part of the security name.

- 1** Click the **Lookup** button on the Datafeed dialog box **Data** tab. The Securities Lookup dialog box opens.
- 2** Specify your choice of market in the **Choose Market** field.
- 3** Enter the known part of the security name in the **Lookup** field.
- 4** Click **Submit**. All possible values of the company name and ticker symbol corresponding to the security name you specified display in the **Security** and **Symbol** list.
- 5** Select one or more securities from the list, and then click **Select**.

The selected securities are added to the **Selected Securities** list on the **Data** tab.

The following figure summarizes these steps.



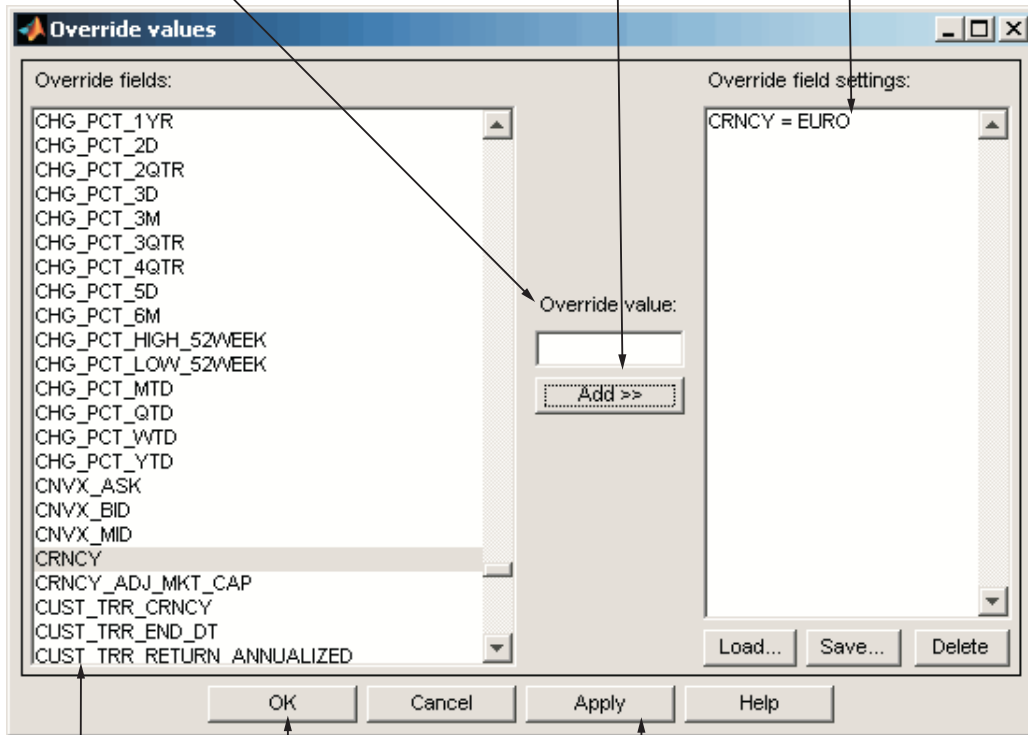
## Setting Overrides

To set overrides on retrieved data:

- 1 Click the **Override** button. The Override values dialog box opens.
- 2 Select the field to override from the **Override fields** selection list.
- 3 Enter the desired override value in the **Override value** field.
- 4 Click **Add** to add the field to override to the **Override field settings** list.
- 5 Click **Apply** to apply overrides to the current session and keep the Override values dialog box open, or click **OK** to apply the overrides and close the dialog box.

The following figure summarizes these steps.

2. Enter desired override value.
3. Click **Add** to add the field to the **Override field settings** list.
- Lists data to override.



1. Select field to override.
- 4a. Apply overrides and close dialog. Return to previous dialog box.
4. Apply overrides to current session.



# Function Reference

---

Bloomberg (p. 5-2)

Get Bloomberg financial data

Datastream (p. 5-3)

Get Thomson Datastream financial data

FactSet (p. 5-4)

Get FactSet financial data

FRED (p. 5-5)

Get Federal Reserve Economic Data (FRED®) financial data

Haver Analytics (p. 5-6)

Get Haver Analytics financial data

Interactive Data Pricing and RemotePlus (p. 5-7)

Get Interactive Data Pricing and Reference Data's RemotePlus financial data

Kx Systems (p. 5-8)

Get Kx Systems®, Inc. kdb+ financial data

Reuters (p. 5-9)

Get Reuters financial data

Reuters Datascope Tick History (p. 5-10)

Retrieve data from Reuters Datascope Tick History file

Reuters Knowledge Direct (p. 5-11)

Establish Reuters Knowledge Direct connection

Reuters Newscope (p. 5-12)

Retrieve data from Reuters Newscope sentiment archive file

Yahoo! (p. 5-13)

Get Yahoo! financial data

## **Bloomberg**

|                                     |  |
|-------------------------------------|--|
| <code>bloomberg</code>              | Connect to Bloomberg data servers                              |
| <code>bloomberg.close</code>        | Close connections to Bloomberg data servers                    |
| <code>bloomberg.fetch</code>        | Request data from Bloomberg data servers                       |
| <code>bloomberg.get</code>          | Retrieve Bloomberg connection object properties                |
| <code>bloomberg.isconnection</code> | Verify whether connections to Bloomberg data servers are valid |
| <code>bloomberg.pricevol</code>     | Price and volume (demonstration)                               |
| <code>bloomberg.showtrades</code>   | Recent trade data (demonstration)                              |
| <code>bloomberg.stockticker</code>  | Trades with volumes (demonstration)                            |

## Datastream

|                                      |   |
|--------------------------------------|---|
| <code>datastream</code>              | Establish connections to Thomson Datastream API                         |
| <code>datastream.close</code>        | Close connections to Thomson Datastream data servers                    |
| <code>datastream.fetch</code>        | Request data from Thomson Datastream data servers                       |
| <code>datastream.get</code>          | Retrieve properties of Thomson Datastream connection objects            |
| <code>datastream.isconnection</code> | Verify whether connections to Thomson Datastream data servers are valid |

## FactSet

|                                   |  |
|-----------------------------------|--|
| <code>factset</code>              | Establish connections to FactSet data servers                |
| <code>factset.close</code>        | Close connections to FactSet data servers                    |
| <code>factset.fetch</code>        | Request data from FactSet data servers                       |
| <code>factset.get</code>          | Retrieve properties of FactSet connection objects            |
| <code>factset.isconnection</code> | Verify whether connections to FactSet data servers are valid |



## FRED

|                                |   |
|--------------------------------|---|
| <code>fred</code>              | Connect to FRED data servers                              |
| <code>fred.close</code>        | Close connections to FRED data servers                    |
| <code>fred.fetch</code>        | Request data from FRED data servers                       |
| <code>fred.get</code>          | Retrieve properties of FRED connection objects            |
| <code>fred.isconnection</code> | Verify whether connections to FRED data servers are valid |

## Haver Analytics

|                                 |  |
|---------------------------------|--|
| <code>haver</code>              | Connect to local Haver Analytics database                            |
| <code>haver.aggregation</code>  | Set Haver Analytics aggregation mode                                 |
| <code>haver.close</code>        | Close Haver Analytics database                                       |
| <code>haver.fetch</code>        | Request data from Haver Analytics database                           |
| <code>haver.get</code>          | Retrieve properties from Haver Analytics connection objects          |
| <code>haver.info</code>         | Retrieve information about Haver Analytics variables                 |
| <code>haver.isconnection</code> | Verify whether connections to Haver Analytics data servers are valid |
| <code>haver.nextinfo</code>     | Retrieve information about next Haver Analytics variable             |
| <code>havertool</code>          | Run Haver Analytics graphical user interface (GUI)                   |

## Interactive Data Pricing and RemotePlus

|                               |   |
|-------------------------------|---|
| <code>idc</code>              | Connect to Interactive Data Pricing and Reference Data's RemotePlus data servers                              |
| <code>idc.close</code>        | Close connections to Interactive Data Pricing and Reference Data's RemotePlus data servers                    |
| <code>idc.fetch</code>        | Request data from Interactive Data Pricing and Reference Data's RemotePlus data servers                       |
| <code>idc.get</code>          | Retrieve properties of Interactive Data Pricing and Reference Data's RemotePlus connection objects            |
| <code>idc.isconnection</code> | Verify whether connections to Interactive Data Pricing and Reference Data's RemotePlus data servers are valid |

## Kx Systems

|                              |   |
|------------------------------|---|
| <code>kx</code>              | Connect to Kx Systems, Inc. kdb+ databases                              |
| <code>kx.close</code>        | Close connections to Kx Systems, Inc. kdb+ databases                    |
| <code>kx.exec</code>         | Run Kx Systems, Inc. kdb+ commands                                      |
| <code>kx.fetch</code>        | Request data from Kx Systems, Inc. kdb+ databases                       |
| <code>kx.get</code>          | Retrieve Kx Systems, Inc. kdb+ connection object properties             |
| <code>kx.insert</code>       | Write data to Kx Systems, Inc. kdb+ databases                           |
| <code>kx.isconnection</code> | Verify whether connections to Kx Systems, Inc. kdb+ databases are valid |
| <code>kx.tables</code>       | Retrieve table names from Kx Systems, Inc. kdb+ databases               |

## Reuters

|                            |  |
|----------------------------|--|
| <code>reuters</code>       | Create Reuters sessions                        |
| <code>reuters.close</code> | Release connections to Reuters data servers    |
| <code>reuters.fetch</code> | Request data from Reuters data servers         |
| <code>reuters.get</code>   | Retrieve properties of Reuters session objects |
| <code>reuters.stop</code>  | Unsubscribe securities                         |

## **Reuters Datascope Tick History**

`rdthloader`

Retrieve data from Reuters  
Datascope Tick History file

## Reuters Knowledge Direct

|                               |   |
|-------------------------------|---|
| <code>rkd</code>              | Establish Reuters Knowledge Direct connection               |
| <code>rkd.close</code>        | Close Reuters Knowledge Direct connection                   |
| <code>rkd.fetch</code>        | Request Reuters Knowledge Direct data                       |
| <code>rkd.get</code>          | Get properties of Reuters Knowledge Data connection         |
| <code>rkd.isconnection</code> | Verify whether Reuters Knowledge Data connections are valid |

## **Reuters Newscope**

`rnsloader`

Retrieve data from Reuters  
Newscope sentiment archive file



## Yahoo!

|                                 |   |
|---------------------------------|---|
| <code>yahoo</code>              | Connect to Yahoo! data servers                              |
| <code>yahoo.close</code>        | Close connections to Yahoo! data servers                    |
| <code>yahoo.fetch</code>        | Request data from Yahoo! data servers                       |
| <code>yahoo.get</code>          | Retrieve properties of Yahoo! connection objects            |
| <code>yahoo.isconnection</code> | Verify whether connections to Yahoo! data servers are valid |



# Functions — Alphabetical List

---

# bloomberg

---

**Purpose** Connect to Bloomberg data servers

**Syntax** `Connect = bloomberg`

**Description** `Connect = bloomberg` establishes a connection, `Connect`, to a Bloomberg data server. It uses port number 8194 and the default internet address provided when you installed the Bloomberg software on your machine.

**Examples** Establish a connection, `c`, to a Bloomberg data server:

```
c = bloomberg
```

**See Also** `bloomberg.close`, `bloomberg.fetch`, `bloomberg.get`, `bloomberg.isconnection`

**Purpose** Close connections to Bloomberg data servers

**Syntax** `close(Connect)`

**Arguments**

|                      |   |
|----------------------|---|
| <code>Connect</code> | Bloomberg connection object created with the <code>bloomberg</code> function. |
|----------------------|---|

**Description** `close(Connect)` closes the connection to the Bloomberg data server.

**Examples** Establish a Bloomberg connection `c`:

```
c = bloomberg
```

Close this connection:

```
close(c)
```

**See Also** `bloomberg`

# bloomberg.fetch

---

**Purpose** Request data from Bloomberg data servers

**Syntax**

```
data = fetch(Connect, 'Security')
data = fetch(Connect, 'Security', 'HEADER', 'Flag', 'Ident')
data = fetch(Connect, 'Security', 'GETDATA', 'Fields',
            'Override', 'Ident', 'Values')
data = fetch(Connect, 'Security', 'TIMESERIES', 'Date',
            'Minutes', 'TickField')
data = fetch(Connect, 'Security', 'HISTORY', 'Fields',
            'FromDate', 'ToDate', 'Period', 'Currency', 'Ident')
ticker = fetch(Connect, 'SearchString', 'LOOKUP', 'Market')
data = fetch(Connect, 'Security', 'REALTIME', 'Fields',
            'MATLABProg')
data = fetch(Connect, 'Security', 'STOP')
```

## Arguments

**Connect** Bloomberg connection object created with the `bloomberg` function.

**'Security'** A MATLAB string containing the name of a security, or a cell array of strings containing a list of securities, specified in a format recognizable by the Bloomberg server. You can substitute a CUSIP number for a security name as needed.

---

**Note** This argument is case sensitive.

---

**'Flag'** A MATLAB string indicating the dates for which to retrieve data. Possible values are:

- **DEFAULT**: Data from most recent bid, ask, or trade. If you do not specify a Flag value, `fetch` uses the default value of 'DEFAULT'.
- **TODAY**: Today's data only.
- **ENHANCED**: Data from most recent date of each individual field.

|             |  |
|-------------|--|
| 'Currency'  | (Optional) Currency in which the <code>fetch</code> function returns historical data. A list of valid currencies appears in the file <code>@bloomberg/bbfields.mat</code> . Default = [].  |
| 'Ident'     | (Optional) Security type identifier. A list of valid security type identifiers appears in the file <code>@bloomberg/bbfields.mat</code> . Default = [].  |
| 'Fields'    | A MATLAB string or cell array of strings specifying specific fields for which you request data. A list of valid field names appears in the file <code>@bloomberg/bbfields.mat</code> . The variable <code>bbfieldnames</code> contains the list of field names. Default = [].                    |
| 'Override'  | (Optional) String or cell array of strings containing override field list. Default = [].   |
| 'Values'    | (Optional) String or cell array of strings containing override field values.   |
| 'Date'      | Date string, serial date number, or cell array of dates that specifies dates for the time-series data. Specify <code>now</code> to retrieve today's time-series data.  |
| 'Minutes'   | (Optional) Tick interval in minutes.   |
| 'TickField' | (Optional) You can specify a string or numeric value for this field. For example, <code>TickField = 'Trade'</code> or <code>TickField = 1</code> return data for ticks of type <code>Trade</code> . Use the command <code>dftool('ticktypes')</code> to return the list of intraday tick fields. |
| 'FromDate'  | Beginning date for historical data.  |

---

**Note** You can specify dates in any of the formats supported by `datestr` and `datenum` that display a year, month, and day.

---

|          |                               |
|----------|-------------------------------|
| 'ToDate' | End date for historical data. |
|----------|-------------------------------|

'Period' (Optional) Period of the data. A MATLAB three-part string with the format:

'Frequency Days Data'

Frequency Values:

- d: Daily (default)
- w: Weekly
- m: Monthly
- q: Quarterly
- s: Semiannually
- y: Yearly

Days Values:

- o: Omit all days for which there is no data (default)
- i: Include all trading days
- a: Include all calendar days

Data Values:

- b: Report missing data using Bloomberg (default)
- s: Show missing data as last found value
- n: Report missing data as NaN

For example, 'dan' returns daily data for all calendar days, reporting missing values as NaN. If a value is unspecified, fetch returns a default value.

---

**Note** If you do not specify a value for Period, fetch uses default values.

---



- 'Currency' (Optional) Currency type. The file @bloomberg/bbfields.mat lists supported currencies.
- 'Market' A MATLAB string indicating the market in which a particular security trades. Possible values are:
- Comdty: (Commodities)
  - Corp: (Corporate bonds)
  - Equity: (Equities)
  - Govt: (Government bonds)
  - Index: (Indexes)
  - M-Mkt: (Money Market securities)
  - Mtge: Mortgage-backed securities)
  - Muni: (Municipal bonds)
  - Pfd: (Preferred stocks)
- 'MATLABProg' A string that is the name of any valid MATLAB program.

## Description

For a given security, `fetch` returns header (default), current, time-series, real time, and historical data via a connection to a Bloomberg data server.

`data = fetch(Connect, 'Security')` fills the header fields with data from the most recent date with a bid, ask, or trade.

`data = fetch(Connect, 'Security', 'HEADER', 'Flag', 'Ident')` returns data for the most recent date of each individual field for the specified security type identifiers, based upon the value of `Flag`.

- If 'Flag' is 'DEFAULT', `fetch` fills the header fields with data from the most recent date with a bid, ask, or trade. Alternatively, you could use the command `data = fetch(Connect, 'Security')`.

- If 'Flag' is 'TODAY', fetch returns the header field data with data from today only.
- If 'Flag' is 'ENHANCED', fetch returns the header field data for the most recent date of each individual field. In this case, for example, the bid and ask group fields could come from different dates.

`data = fetch(Connect, 'Security', 'GETDATA', 'Fields', 'Override', 'Ident', 'Values')` returns the current market data for the specified fields of the indicated security. You can further specify the data with the optional `Override`, `Values` and `Ident` arguments.

---

**Note** If a call to the `fetch` function with the `GETDATA` argument encounters an invalid security in a list of securities to retrieve, it returns NaN data for the invalid security's fields.

---

`data = fetch(Connect, 'Security', 'TIMESERIES', 'Date', 'Minutes', 'TickField')` returns the tick data for a security for the specified date. You can further specify data with the optional `Minutes` and `TickField` arguments. If there is no data found in the specified range, `fetch` returns an empty matrix.

You can specify `TickField` as a string or numeric value. For example, `TickField = 'Trade'` or `TickField = 1` returns data for ticks of type Trade. The function `dftool('ticktypes')` returns the list of intraday tick fields. `fetch` returns intraday tick data requested with an interval with the following columns:

- Time
- Open
- High
- Low
- Value of last tick

- Volume total value of ticks
- Total value of ticks for the time range
- Number of ticks

The `fetch` function returns columns 7 and 8 only if they make sense for the requested field.

For today's tick data, enter the command:

```
data = fetch(Connect, 'Security', 'TIMESERIES', now)
```

For today's trade time series aggregated into five-minute intervals, enter:

```
data = fetch(Connect, 'Security', 'TIMESERIES', ...  
now, 5, 'Trade')
```

```
data = fetch(Connect, 'Security', 'HISTORY', 'Fields',  
'FromDate', 'ToDate', 'Period', 'Currency', 'Ident')
```

 returns historical data for the specified field for the date range `FromDate` to `ToDate`. You can set the time period with the optional `Period` argument to return a more specific data set. You can further specify returned data by appending the `Currency` or `Ident` argument.

---

**Note** If a call to the `fetch` function with the `HISTORY` argument encounters an invalid security in a list of securities to retrieve, it returns no data for any securities in the list.

---

```
ticker = fetch(Connect, 'SearchString', 'LOOKUP', 'Market')
```

 uses `SearchString` to find the ticker symbol for a security trading in a designated market. The output `ticker` is a column vector of possible ticker values.

---

**Note** If you supply `Ident` without a period or currency, enter `[]` for the missing values.

---

`data = fetch(Connect, 'Security', 'REALTIME', 'Fields', 'MATLABProg')` subscribes to a given security or list of securities, requesting the indicated fields, and runs any specified MATLAB function. See `pricevol`, `showtrades`, or `stockticker` for information on the data returned by asynchronous Bloomberg events.

`data = fetch(Connect, 'Security', 'STOP')` unsubscribes the list of securities from processing Bloomberg real-time events.

## Examples

### Retrieving Header Data

Retrieve header data for a United States equity with ticker ABC:

```
D = fetch(C, 'ABC US Equity')
```

### Retrieving Opening and Closing Prices

Retrieve the opening and closing prices:

```
D = fetch(C, 'ABC US Equity', 'GETDATA', ...  
{'Last_Price'; 'Open'})
```

### Retrieving Override Fields

Retrieve the requested fields, given override fields and values:

```
D = fetch(C, '3358ABCD4 Corp', 'GETDATA', ...  
{'YLD_YTM_ASK', 'ASK', 'OAS_SPREAD_ASK', 'OAS_VOL_ASK'}, ...  
{'PX_ASK', 'OAS_VOL_ASK'}, {'99.125000', '14.000000'})
```

### Retrieving Time Series Data

Retrieve today's time series:

```
D = fetch(C, 'ABC US Equity', 'TIMESERIES', now)
```

### **Retrieving Time Series Data, Aggregated into Time Intervals**

Retrieve today's trade time series for the given security, aggregated into five-minute intervals:

```
D = fetch(C, 'ABC US Equity', 'TIMESERIES', now, 5, 'Trade')
```

### **Retrieving Time Series Default Closing Price**

Retrieve the closing price for the given dates, using the default period of the data:

```
D = fetch(C, 'ABC US Equity', 'HISTORY', 'Last_Price', ...  
         '8/01/99', '8/10/99')
```

### **Retrieving Monthly Closing Price**

Retrieve the monthly closing price for the specified dates:

```
D = fetch(C, 'ABC US Equity', 'HISTORY', 'Last_Price', ...  
         '8/01/99', '9/30/00', 'm')
```

### **See Also**

bloomberg, bloomberg.close, bloomberg.get,  
bloomberg.isconnection

# bloomberg.get

---

**Purpose** Retrieve Bloomberg connection object properties

**Syntax**  
`value = get(Connect, 'PropertyName')`  
`value = get(Connect)`

## Arguments

|              |   |
|--------------|---|
| Connect      | Bloomberg connection object created with the <code>bloomberg</code> function.   |
| PropertyName | (Optional) A MATLAB string or cell array of strings containing property names. Property names are: <ul style="list-style-type: none"><li>• 'Connection'</li><li>• 'IPAddress'</li><li>• 'Port'</li><li>• 'Socket'</li><li>• 'Version'</li></ul> |

**Description** `value = get(Connect, 'PropertyName')` returns a MATLAB structure containing the value of the specified properties for the Bloomberg connection object.

`value = get(Connect)` returns the value for all properties.

## Examples

Establish a connection, `c`, to a Bloomberg data server:

```
c = bloomberg
```

Retrieve this connection's properties:

```
p = get(c, {'Port', 'IPAddress'})  
p =  
    port: 8194  
    ipaddress: 111.222.33.444
```

## **See Also**

`bloomberg`, `bloomberg.close`, `bloomberg.fetch`,  
`bloomberg.isconnection`

# bloomberg.isconnection

---

**Purpose** Verify whether connections to Bloomberg data servers are valid

**Syntax** `x = isconnection(Connect)`

**Arguments**

|                      |   |
|----------------------|---|
| <code>Connect</code> | Bloomberg connection object created with the <code>bloomberg</code> function. |
|----------------------|---|

**Description** `x = isconnection(Connect)` returns `x = 1` if the connection to the Bloomberg data server is valid, and `x = 0` otherwise.

**Examples** Establish a connection, `c`, to a Bloomberg data server:

```
c = bloomberg
```

Verify that `c` is a valid connection:

```
x = isconnection(c)
x = 1
```

**See Also** `bloomberg`, `bloomberg.close`, `bloomberg.fetch`, `bloomberg.get`



**Purpose** Price and volume (demonstration)

**Syntax** pricevol(InputList)

**Arguments**

|           |   |
|-----------|---|
| InputList | Fields from which you request real-time data. |
|-----------|---|

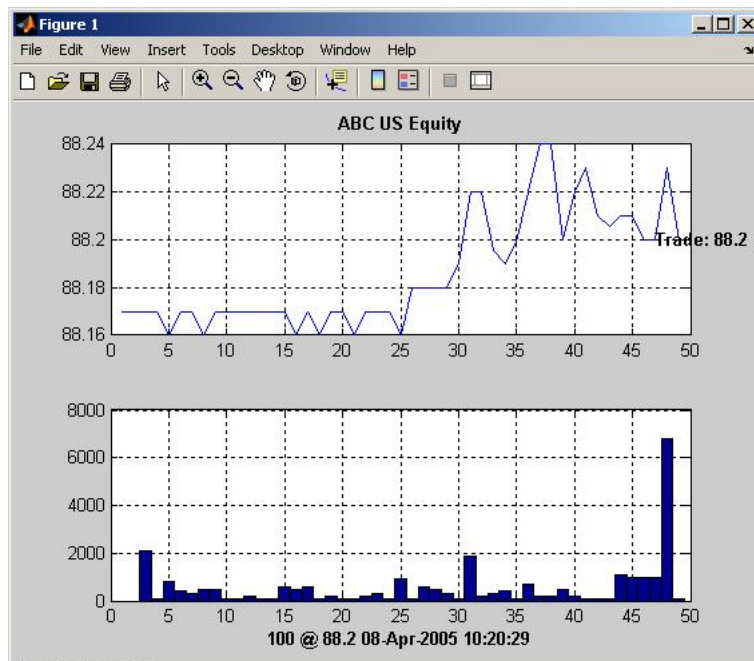
**Description** pricevol(InputList) demonstrates the Bloomberg real-time data import functionality, where InputList is an input list of elements as described in the following table.

|                                     |                                       |
|-------------------------------------|---------------------------------------|
| InputList(1) = COM.Bloomberg.Data.1 | Bloomberg handle                      |
| InputList(2) = 1                    | Event ID                              |
| InputList(3) = ('Security')         | Security string                       |
| InputList(4) = 1                    | Cookie                                |
| InputList(5) = 2                    | Field number ID                       |
| InputList(6) = {[43.58]}            | Return data for the given tick        |
| InputList(7) = 0                    | Status                                |
| InputList(8)                        | Structure containing the above fields |
| InputList(9) = 'Data'               | Event type                            |

The input argument InputList(8) contains the information required to process real-time events.

**Examples** Display the most recent Trade and Volume values in a figure window and show the most recent trade with volumes:

```
b = bloomberg;  
d = fetch(b, 'ABC US Equity', 'REALTIME', ...  
{ 'Last_Trade', 'Volume'}, 'pricevol');
```



## See Also

`bloomberg.showtrades`, `bloomberg.stockticker`

**Purpose** Recent trade data (demonstration)

**Syntax** `showtrades(InputList)`

**Arguments**

`InputList` Fields from which you request real-time data.

**Description**

`showtrades(InputList)` demonstrates the Bloomberg real-time data import functionality, where `InputList` is an input list of elements as described in the following table.

|  |                                       |
|--|---------------------------------------|
| <code>InputList(1) = COM.Bloomberg.Data.1</code> | Bloomberg handle                      |
| <code>InputList(2) = 1</code>                    | Event ID                              |
| <code>InputList(3) = ('Security')</code>         | Security string                       |
| <code>InputList(4) = 1</code>                    | Cookie                                |
| <code>InputList(5) = 2</code>                    | Field number ID                       |
| <code>InputList(6) = {[43.58]}</code>            | Return data for the given tick        |
| <code>InputList(7) = 0</code>                    | Status                                |
| <code>InputList(8)</code>                        | Structure containing the above fields |
| <code>InputList(9) = 'Data'</code>               | Event type                            |

The input argument `InputList(8)` contains the information required to process real-time events.

**Examples**

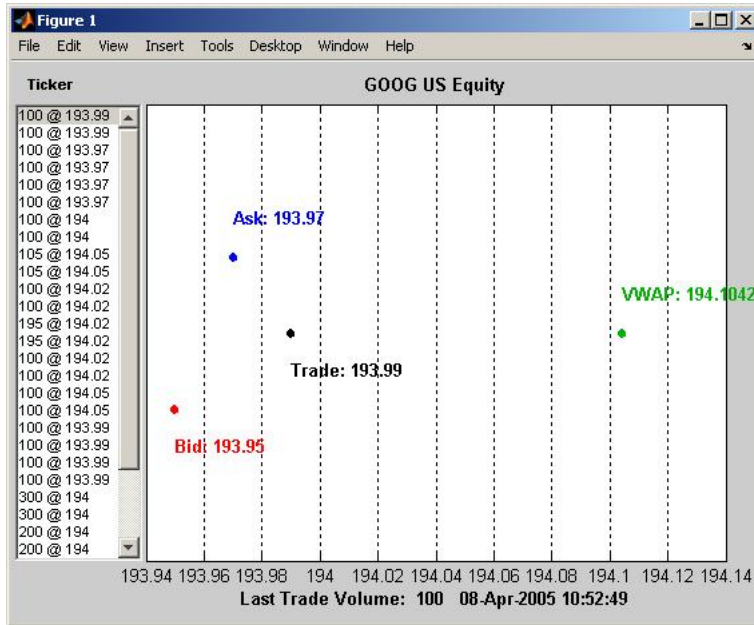
Establish a connection, `c`, to a Bloomberg data server:

```
c = bloomberg;
```

Display the most recent Trade, Bid, Ask, and VWAP (volume-weighted adjusted price), and a list of the most recent trades with volumes:

# bloomberg.showtrades

```
d = fetch(c, 'GOOG US Equity', 'REALTIME', ...  
{ 'Last_Trade', 'Bid', 'Ask', 'Volume', 'VWAP'}, 'showtrades');
```



## See Also

`bloomberg.pricevol`, `bloomberg.stockticker`

**Purpose** Trades with volumes (demonstration)

**Syntax** stockticker(InputList)

## Arguments

InputList Fields from which you request real-time data.

## Description

stockticker(InputList) demonstrates the Bloomberg real-time data import functionality, where InputList is an input list of elements as described in the following table.

|                                     |                                       |
|-------------------------------------|---------------------------------------|
| InputList(1) = COM.Bloomberg.Data.1 | Bloomberg handle                      |
| InputList(2) = 1                    | Event ID                              |
| InputList(3) = ('Security')         | Security string                       |
| InputList(4) = 1                    | Cookie                                |
| InputList(5) = 2                    | Field number ID                       |
| InputList(6) = {[43.58]}            | Return data for the given tick        |
| InputList(7) = 0                    | Status                                |
| InputList(8)                        | Structure containing the above fields |
| InputList(9) = 'Data'               | Event type                            |

The input argument InputList(8) contains the information required to process real-time events.

## Examples

Retrieve a list of trades with volumes for each requested security:

```
b = bloomberg;  
d = fetch(b, {'IBM US Equity', 'EMC US Equity', 'NTAP US Equity'}, ...  
          'REALTIME', {'Last_Trade', 'Volume'}, 'stockticker');  
** EMC US Equity ** 0 @ 12.65 08-Apr-2005 10:24:57
```

# bloomberg.stockticker

---

```
** IBM US Equity ** 0 @ 88.17 08-Apr-2005 10:24:57
** NTAP US Equity ** 0 @ 29.02 08-Apr-2005 10:24:57
** EMC US Equity ** 200 @ 12.66 08-Apr-2005 10:24:58
** EMC US Equity ** 1400 @ 12.65 08-Apr-2005 10:24:58
** EMC US Equity ** 3100 @ 12.66 08-Apr-2005 10:25:00
** IBM US Equity ** 1300 @ 88.17 08-Apr-2005 10:25:00
.
.
.
```

## **See Also**

bloomberg.pricevol, bloomberg.showtrades

**Purpose** Establish connections to Thomson Datastream API

**Syntax** `Connect = datastream('UserName', 'Password', 'Source', 'URL')`

**Arguments**

|            |   |
|------------|---|
| 'UserName' | User name.  |
| 'Password' | User password.  |
| 'Source'   | To connect to the Thomson Datastream API, enter 'Datastream' in this field. |
| 'URL'      | Web URL.  |

---

**Note** Thomson assigns the values for you to enter for each argument. Enter all arguments as MATLAB strings.

---

**Description** `Connect = datastream('UserName', 'Password', 'Source', 'URL')` makes a connection to the Thomson Datastream API, which provides access to Thomson Datastream software content.

**Examples** Establish a connection to the Thomson Datastream API:

```
Connect = datastream('User1', 'Pass1', 'Datastream', ...  
    'http://dataworks.thomson.com/Dataworks/Enterprise/1.0')
```

**See Also** `datastream.close`, `datastream.fetch`, `datastream.get`, `datastream.isconnection`

# datastream.close

---

|                    |  |
|--------------------|--|
| <b>Purpose</b>     | Close connections to Thomson Datastream data servers   |
| <b>Syntax</b>      | <code>close(Connect)</code>  |
| <b>Arguments</b>   | <code>Connect</code> Thomson Datastream connection object created with the <code>datastream</code> function. |
| <b>Description</b> | <code>close(Connect)</code> closes a connection to a Thomson Datastream data server.                         |
| <b>See Also</b>    | <code>datastream</code>  |



## Purpose

Request data from Thomson Datastream data servers

## Syntax

```
data = fetch(Connect, 'Security')
data = fetch(Connect, 'Security', 'Fields')
data = fetch(Connect, 'Security', 'Fields', 'Date')
data = fetch(Connect, 'Security', 'Fields', 'FromDate',
            'ToDate')
data = fetch(Connect, 'Security', 'Fields', 'FromDate',
            'ToDate', 'Period')
data = fetch(Connect, 'Security', 'Fields', 'FromDate',
            'ToDate', 'Period', 'Currency')
```

## Arguments

|            |   |
|------------|---|
| Connect    | Thomson Datastream connection object created with the <code>datastream</code> function.   |
| 'Security' | MATLAB string containing the name of a security, or cell array of strings containing names of multiple securities. This data is in a format recognizable by the Thomson Datastream data server. |
| 'Fields'   | (Optional) MATLAB string or cell array of strings indicating the data fields for which to retrieve data.  |
| 'Date'     | (Optional) MATLAB string indicating a specific calendar date for which you request data.  |
| 'FromDate' | (Optional) Start date for historical data.  |

# datastream.fetch

---

'ToDate' (Optional) End date for historical data. If you specify a value for 'ToDate', 'FromDate' cannot be an empty value.

---

**Note** You can specify dates in any of the formats supported by `datestr` and `datenum` that show a year, month, and day.

---

'Period' (Optional) Period within a date range. Period values are:

- 'd': daily values
- 'w': weekly values
- 'm': monthly values

'Currency' (Optional) Currency in which `fetch` returns the data.

---

**Note** You can enter the optional arguments 'Fields', 'FromDate', 'ToDate', 'Period', and 'Currency' as MATLAB strings or empty arrays ([ ]).

---

## Description

`data = fetch(Connect, 'Security')` returns the default time series for the indicated security.

`data = fetch(Connect, 'Security', 'Fields')` returns data for the specified security and fields.

`data = fetch(Connect, 'Security', 'Fields', 'Date')` returns data for the specified security and fields on a particular date.

`data = fetch(Connect, 'Security', 'Fields', 'FromDate', 'ToDate')` returns data for the specified security and fields for the indicated date range.

`data = fetch(Connect, 'Security', 'Fields', 'FromDate', 'ToDate', 'Period')` returns instrument data for the given range with the indicated period.

`data = fetch(Connect, 'Security', 'Fields', 'FromDate', 'ToDate', 'Period', 'Currency')` also specifies the currency in which the data is reported.

---

**Note** The Thomson Datastream interface returns all data as strings. For example, it returns Price data to the MATLAB workspace as a cell array of strings within the structure. There is no way to determine the data type from the Datastream® interface.

---

## Examples

### Retrieving Time Series Data

Return the trailing one-year price time series for the instrument 'P', which is the default value for the 'Fields' argument using the command:

```
data = fetch(Connect, 'ICI')
```

Or the command:

```
data = fetch(Connect, 'ICI', 'P')
```

### Retrieving Opening and Closing Prices

Return the closing and opening prices for the instruments P and PO on the date September 1, 2007.

```
data = fetch(Connect, 'ICI', {'P', 'PO'}, '09/01/2007')
```

# datastream.fetch

---

## Retrieving Monthly Opening and Closing Prices for a Specified Date Range

Return the monthly closing and opening prices for the securities ICI and IBM from 09/01/2005 to 09/01/2007:

```
data = fetch(Connect, {'ICI', 'IBM'}, {'P', 'PO'}, ...  
            '09/01/2005', '09/01/2007', 'M')
```

### See Also

`datastream.close`, `datastream`, `datastream.get`,  
`datastream.isconnection`

**Purpose** Retrieve properties of Thomson Datastream connection objects

**Syntax**  
`value = get(Connect, 'PropertyName')`  
`value = get(Connect)`

## Arguments

**Connect** Thomson Datastream connection object created with the `datastream` function.

**PropertyName** (Optional) A MATLAB string or cell array of strings containing property names. Valid property names include:

- `user`
- `datasource`
- `endpoint`
- `wSDL`
- `sources`
- `systeminfo`
- `version`

**Description**  
`value = get(Connect, 'PropertyName')` returns the value of the specified properties for the Thomson Datastream connection object.  
`value = get(Connect)` returns a MATLAB structure where each field name is the name of a property of `Connect`. Each field contains the value of the property.

**See Also** `datastream.close`, `datastream`, `datastream.fetch`, `datastream.isconnection`

# datastream.isconnection

---

**Purpose** Verify whether connections to Thomson Datastream data servers are valid

**Syntax** `x = isconnection(Connect)`

**Arguments**

|                      |   |
|----------------------|---|
| <code>Connect</code> | Thomson Datastream connection object created with the <code>datastream</code> function. |
|----------------------|---|

**Description** `x = isconnection(Connect)` returns `x = 1` if the connection is a valid Thomson Datastream connection, and `x = 0` otherwise.

**Examples** Establish a connection to the Thomson Datastream API:

```
c = datastream
```

Verify that `c` is a valid connection:

```
x = isconnection(c)
x = 1
```

**See Also** `datastream.close`, `datastream`, `datastream.fetch`, `datastream.get`

**Purpose** Establish connections to FactSet data servers

**Syntax** `Connect = factset('UserName', 'SerialNumber', 'Password', 'ID')`

### Arguments

|              |   |
|--------------|---|
| UserName     | User login name.                        |
| SerialNumber | User serial number.                     |
| Password     | User password.                          |
| ID           | FactSet customer identification number. |

---

**Note** FactSet assigns values to all input arguments.

---

**Description** `Connect = factset('UserName', 'SerialNumber', 'Password', 'ID')` connects to the FactSet FAST interface.

**Examples** Establish a connection to a FactSet server:

```
Connect = factset('username', '1234', 'password', 'fsid')
Connect =
    user: 'username'
    serial: '1234'
    password: 'password'
    cid: 'fsid'
```

**See Also** `factset.close`, `factset.fetch`, `factset.get`, `factset.isconnection`

# factset.close

---

**Purpose** Close connections to FactSet data servers

**Syntax** `close(Connect)`

**Arguments**

|         |   |
|---------|---|
| Connect | FactSet connection object created with the <code>factset</code> function. |
|---------|---|

**Description** `close(Connect)` closes the connection to the FactSet data server.

**See Also** `factset`



**Purpose** Request data from FactSet data servers

**Syntax**

```
data = fetch(Connect)
data = fetch(Connect, 'Library')
data = fetch(Connect, 'Security', 'Fields')
data = fetch(Connect, 'Security', 'Fields', 'FromDate',
            'ToDate')
data = fetch(Connect, 'Security', 'FromDate', 'ToDate',
            'Period')
```

### Arguments

|          |  |
|----------|--|
| Connect  | FactSet connection object created with the factset function.   |
| Library  | FactSet formula library.   |
| Security | A MATLAB string or cell array of strings containing the names of securities in a format recognizable by the FactSet server.          |
| Fields   | A MATLAB string or cell array of strings indicating the data fields for which to retrieve data.                                      |
| Date     | Date string or serial date number indicating date for the requested data. If you enter today's date, fetch returns yesterday's data. |
| FromDate | Beginning date for date range.   |

---

**Note** You can specify dates in any of the formats supported by `datestr` and `datenum` that display a year, month, and day.

---

|                     |  |
|---------------------|--|
| <code>ToDate</code> | End date for date range.   |
| <code>Period</code> | Period within date range. Period values are: <ul style="list-style-type: none"><li>• <code>'d'</code>: daily values</li><li>• <code>'b'</code>: business day daily values</li><li>• <code>'m'</code>: monthly values</li><li>• <code>'mb'</code>: beginning monthly values</li><li>• <code>'me'</code>: ending monthly values</li><li>• <code>'q'</code>: quarterly values</li><li>• <code>'qb'</code>: beginning quarterly values</li><li>• <code>'qe'</code>: ending quarterly values</li><li>• <code>'y'</code>: annual values</li><li>• <code>'yb'</code>: beginning annual values</li><li>• <code>'ye'</code>: ending annual values</li></ul> |

## Description

`data = fetch(Connect)` returns the names of all available formula libraries.

`data = fetch(Connect, 'Library')` returns the valid field names for a given formula library.

`data = fetch(Connect, 'Security', 'Fields')` returns data for the specified security and fields.

`data = fetch(Connect, 'Security', 'Fields', 'Date')` returns security data for the specified fields on the requested date.

`data = fetch(Connect, 'Security', 'Fields', 'FromDate', 'ToDate')` returns security data for the specified fields for the date range `FromDate` to `ToDate`.

`data = fetch(Connect, 'Security', 'FromDate', 'ToDate', 'Period')` returns security data for the date range `FromDate` to `ToDate` with the specified period.

## Examples

### Retrieving Names of Available Formula Libraries

Obtain the names of available formula libraries:

```
D = fetch(Connect)
```

### Retrieving Valid Field Names of a Specified Library

Obtain valid field names of the `FactSetSecurityCalcs` library:

```
D = fetch(Connect, 'fs')
```

### Retrieving the Closing Price of a Specified Security

Obtain the closing price of the security `IBM`:

```
D = fetch(Connect, 'IBM', 'price')
```

### Retrieving the Closing Price of a Specified Security Using Default Date Period

Obtain the closing price for `IBM` using the default period of the data:

```
D = fetch(C, 'IBM', 'price', '09/01/07', '09/10/07')
```

### Retrieving the Monthly Closing Prices of a Specified Security for a Given Date Range

Obtain the monthly closing prices for `IBM` from `09/01/05` to `09/10/07`:

```
D = fetch(C, 'IBM', 'price', '09/01/05', '09/10/07', 'm')
```

## See Also

`factset.close`, `factset`, `factset.isconnection`

**Purpose** Retrieve properties of FactSet connection objects

**Syntax**  
`value = get(Connect, 'PropertyName')`  
`value = get(Connect)`

**Arguments**

|              |   |
|--------------|---|
| Connect      | FactSet connection object created with the factset function.  |
| PropertyName | (Optional) A MATLAB string or cell array of strings containing property names. Property names are: <ul style="list-style-type: none"><li>• user</li><li>• serial</li><li>• password</li><li>• cid</li></ul> |

**Description**

`value = get(Connect, 'PropertyName')` returns the value of the specified properties for the FactSet connection object.

`value = get(Connect)` returns a MATLAB structure where each field name is the name of a property of Connect, and each field contains the value of that property.

**Examples**

Establish a connection to a FactSet data server:

```
Connect = factset('Fast_User', '1234', 'Fast_Pass', 'userid')
```

Retrieve the connection property value:

```
h = get(Connect)
h=
    user: 'Fast_User'
  serial: '1234'
password: 'Fast_Pass'
```

```
cid: 'userid'
```

Retrieve the value of the connection's user property:

```
get(Connect, 'user')  
ans =  
Fast_User
```

**See Also**

`factset.close`, `factset.fetch`, `factset`, `factset.isconnection`

# factset.isconnection

---

**Purpose** Verify whether connections to FactSet data servers are valid

**Syntax** `x = isconnection(Connect)`

**Arguments**

Connect                      FactSet connection object created with the factset function.

**Description** `x = isconnection(Connect)` returns `x = 1` if the connection to the FactSet data server is valid, and `x = 0` otherwise.

**Examples** Establish a connection, `c`, to a FactSet data server:

```
c = factset
```

Verify that `c` is a valid connection:

```
x = isconnection(c);  
x = 1
```

**See Also** `factset.close`, `factset.fetch`, `factset`, `factset.get`

---

|                    |  |
|--------------------|--|
| <b>Purpose</b>     | Connect to FRED data servers   |
| <b>Syntax</b>      | <code>Connect = fred(URL)</code><br><code>Connect = fred</code>  |
| <b>Arguments</b>   | URL                      Create a connection using a specified URL.  |
| <b>Description</b> | <code>Connect = fred(URL)</code> establishes a connection to a FRED data server.<br><code>Connect = fred</code> verifies that the URL <code>http://research.stlouisfed.org/fred2/</code> is accessible and creates a connection. |
| <b>Examples</b>    | Connect to the FRED data server at the URL <code>http://research.stlouisfed.org/fred2/</code> :<br><pre>c = fred('http://research.stlouisfed.org/fred2/')</pre>  |
| <b>See Also</b>    | <code>fred.close</code> , <code>fred.fetch</code> , <code>fred.get</code> , <code>fred.isconnection</code>   |

# fred.close

---

|                      |   |                      |   |
|----------------------|---|----------------------|---|
| <b>Purpose</b>       | Close connections to FRED data servers  |                      |   |
| <b>Syntax</b>        | <code>close(Connect)</code>   |                      |   |
| <b>Arguments</b>     | <table><tr><td><code>Connect</code></td><td>FRED connection object created with the <code>fred</code> function.</td></tr></table>   | <code>Connect</code> | FRED connection object created with the <code>fred</code> function. |
| <code>Connect</code> | FRED connection object created with the <code>fred</code> function.   |                      |   |
| <b>Description</b>   | <code>close(Connect)</code> closes the connection to the FRED data server.  |                      |   |
| <b>Examples</b>      | <p>Make a connection <code>c</code> to a FRED data server:</p> <pre>c = fred('http://research.stlouisfed.org/fred2/')</pre> <p>Close this connection:</p> <pre>close(c)</pre> |                      |   |
| <b>See Also</b>      | <code>fred</code>   |                      |   |



**Purpose** Request data from FRED data servers

**Syntax**

```
data = fetch(Connect, 'Security')
data = fetch(Connect, 'Security', 'D1')
data = fetch(Connect, 'Security', 'D1', 'D2')
```

### Arguments

|            |  |
|------------|--|
| Connect    | FRED connection object created with the fred function.                                       |
| 'Security' | MATLAB string containing the name of a security in a format recognizable by the FRED server. |
| 'D1'       | MATLAB string or date number indicating the date from which to retrieve data.                |
| 'D2'       | MATLAB string or date number indicating the date range from which to retrieve data.          |

### Description

For a given security, `fetch` returns historical data using the connection to the FRED data server.

`data = fetch(Connect, 'Security')` returns data for the security `Security`, using the connection object `Connect`.

`data = fetch(Connect, 'Security', 'D1')` returns data for the security `Security`, using the connection object `Connect`, for the date `D1`. If you specify today's date for `D1`, `fetch` returns data from yesterday.

`data = fetch(Connect, 'Security', 'D1', 'D2')` returns all data for the security `Security`, using the connection object `Connect`, for the date range `'D1'` to `'D2'`.

---

**Note** You can specify dates in any of the formats supported by `datestr` and `datenum` that show a year, month, and day.

---

## Examples

Fetch all available daily U.S. to Euro foreign exchange rates:

```
d = fetch(f, 'DEXUSEU')
d =
    Title: 'U.S. / Euro Foreign Exchange Rate'
    SeriesID: 'DEXUSEU'
    Source:
    'Board of Governors of the Federal Reserve System'
    Release: 'H.10 Foreign Exchange Rates'
    SeasonalAdjustment: 'Not Applicable'
    Frequency: 'Daily'
    Units: 'U.S. Dollars to One Euro'
    DateRange: '1999-01-04 to 2006-06-19'
    LastUpdated: '2006-06-20 9:39 AM CT'
    Notes: 'Noon buying rates in New York City for
    cable transfers payable in foreign currencies.'
    Data: [1877x2 double]
```

Fetch data for 01/01/2007 through 06/01/2007:

```
d = fetch(f, 'DEXUSEU', '01/01/2007', '06/01/2007')
d =
    Title: ' U.S. / Euro Foreign Exchange Rate'
    SeriesID: ' DEXUSEU'
    Source:
    ' Board of Governors of the Federal Reserve System'
    Release: ' H.10 Foreign Exchange Rates'
    SeasonalAdjustment: ' Not Applicable'
    Frequency: ' Daily'
    Units: ' U.S. Dollars to One Euro'
    DateRange: ' 1999-01-04 to 2006-06-19'
    LastUpdated: ' 2006-06-20 9:39 AM CT'
    Notes: ' Noon buying rates in New York City for
    cable transfers payable in foreign currencies.'
    Data: [105x2 double]
```

## See Also

`fred.close`, `fred.get`, `fred.isconnection`

**Purpose** Retrieve properties of FRED connection objects

**Syntax**  
`value = get(Connect, 'PropertyName')`  
`value = get(Connect)`

## Arguments

`Connect` FRED connection object created with the `fred` function.

`'PropertyName'` A MATLAB string or cell array of strings containing property names. Property names are:

- `'url'`
- `'ip'`
- `'port'`

## Description

`value = get(Connect, 'PropertyName')` returns a MATLAB structure containing the value of the specified properties for the FRED connection object.

`value = get(Connect)` returns the value for all properties.

## Examples

Establish a connection, `c`, to a FRED data server:

```
c = fred('http://research.stlouisfed.org/fred2/')
```

Retrieve the port and IP address for the connection:

```
p = get(c, {'port', 'ip'})  
p =  
    port: 8194  
    ip: 111.222.33.444
```

## See Also

`fred.close`, `fred.fetch`, `fred.isconnection`

# fred.isconnection

---

**Purpose** Verify whether connections to FRED data servers are valid

**Syntax** `x = isconnection(Connect)`

**Arguments**

Connect            FRED connection object created with the fred function.

**Description** `x = isconnection(Connect)` returns `x = 1` if a connection to the FRED data server is valid, and `x = 0` otherwise.

**Examples**

Establish a connection, `c`, to a FRED data server:

```
c = fred('http://research.stlouisfed.org/fred2/')
```

Verify that `c` is a valid connection:

```
x = isconnection(c)
x = 1
```

**See Also**

`fred.close`, `fred.fetch`, `fred.get`

|                    |   |
|--------------------|---|
| <b>Purpose</b>     | Connect to local Haver Analytics database   |
| <b>Syntax</b>      | <code>H = haver(Databasename)</code>  |
| <b>Arguments</b>   | <code>Databasename</code> Local path to the Haver Analytics database.   |
| <b>Description</b> | <code>H = haver(Databasename)</code> establishes a connection to a Haver Analytics database.  |
| <b>Examples</b>    | Create a connection to the Haver Analytics database at the path <code>d:\work\haver\data\haverd.dat</code> :<br><br><code>H = haver('d:\work\haver\data\haverd.dat')</code> |
| <b>See Also</b>    | <code>haver.close</code> , <code>haver.fetch</code> , <code>haver.get</code> , <code>haver.isconnection</code>  |

# haver.aggregation

---

**Purpose** Set Haver Analytics aggregation mode

**Syntax** X = aggregation (C)  
X = aggregation (C,V)

**Description** X = aggregation (C) returns the current aggregation mode.  
X = aggregation (C,V) sets the current aggregation mode to V. The following table lists possible values for V.

| Value of V | Aggregation mode | Behavior of aggregation function   |
|------------|------------------|--|
| 0          | strict           | aggregation does not fill in values for missing data.                              |
| 1          | relaxed          | aggregation fills in missing data based on data available in the requested period. |
| 2          | forced           | aggregation fills in missing data based on some past value.                        |
| -1         | Not recognized   | aggregation resets V to its last valid setting.                                    |

**See Also** haver, haver.close, haver.fetch, haver.get, haver.infohaver.isconnection, haver.nextinfo

**Purpose** Close Haver Analytics database

**Syntax** `close(H)`

**Arguments**

|   |   |
|---|---|
| H | Haver Analytics connection object created with the <code>haver</code> function. |
|---|---|

**Description** `close(H)` closes the connection to the Haver Analytics database.

**Examples** Establish a connection H to a Haver Analytics database:

```
H = haver('d:\work\haver\data\haverd.dat')
```

Close the connection:

```
close(H)
```

**See Also** `haver`

# haver.fetch

---

**Purpose** Request data from Haver Analytics database

**Syntax**  
D = fetch(H,S)  
D = fetch(H,S,Startdate,Enddate)  
D = fetch(H,S,Startdate,Enddate,P)

## Arguments

|           |   |
|-----------|---|
| H         | Haver Analytics connection object created with the <code>haver</code> function.   |
| S         | Haver Analytics variable.   |
| Startdate | MATLAB string or date number indicating the <code>startdate</code> from which to retrieve data.   |
| Enddate   | MATLAB string or date number indicating the <code>enddate</code> of the date range.   |
| P         | A specified period. You can enter the period as: <ul style="list-style-type: none"><li>• D for daily values</li><li>• W for weekly values</li><li>• M for monthly values</li><li>• Q for quarterly values</li><li>• A for annual values</li></ul> |

**Description** `fetch` returns historical data via a Haver Analytics connection object.

D = `fetch(H,S)` returns data for the Haver Analytics variable S, using the connection object H.

D = `fetch(H,S,Startdate,Enddate)` returns data for the Haver Analytics variable S, using the connection object H, between the dates `Startdate` and `Enddate`.



`D = fetch(H,S,Startdate,Enddate,P)` returns data for the Haver Analytics variable `S`, using the connection object `H`, between the dates `Startdate` and `Enddate`, in time periods specified by `P`.

## Examples

### Establish a Connection to a Haver Analytics Database

Connect to the Haver Analytics daily demonstration database `haverd.dat`:

```
H = haver('d:\work\haver\data\haverd.dat')
```

### Retrieving Variable Data

Return data for the variable `FFED`:

```
D = fetch(H, 'FFED')
```

### Retrieving Variable Data for a Specified Date Range

Return data for `FFED` from 01/01/1997 to 09/01/2007:

```
D = fetch(H, 'FFED', '01/01/1997', '09/01/2007')
```

### Retrieving Monthly Variable Data for a Specified Date Range

Return data for `FFED`, converted to monthly values, from 01/01/1997 to 09/01/2007:

```
D = fetch(H, 'FFED', '01/01/1997', '09/01/2007', 'M')
```

## See Also

`haver.close`, `haver.get`, `haver.isconnection`, `haver`, `haver.info`, `haver.nextinfo`

# haver.get

---

**Purpose** Retrieve properties from Haver Analytics connection objects

**Syntax**  
`V = get(H, 'PropertyName')`  
`V = get(H)`

## Arguments

**H** Haver Analytics connection object created with the `haver` function.

**'PropertyName'** A MATLAB string or cell array of strings containing property names. The property name is `Databasename`.

## Description

`V = get(H, 'PropertyName')` returns a MATLAB structure containing the value of the specified properties for the Haver Analytics connection object.

`V = get(H)` returns a MATLAB structure, where each field name is the name of a property of `H`. Each field contains the value of the property.

## Examples

Establish a Haver Analytics connection, `HDAILY`:

```
HDAILY = haver('d:\work\haver\data\haverd.dat')
```

Retrieve the name of the Haver database:

```
V = get(HDAILY, {'databasename'})  
V =  
databasename: d:\work\haver\data\haverd.dat
```

## See Also

`haver.close`, `haver.fetch`, `haver.isconnection`, `haver`

**Purpose** Retrieve information about Haver Analytics variables

**Syntax** `D = info(H,S)`

**Arguments**

- H Haver Analytics connection object created with the `haver` function.
- S Haver Analytics variable.

**Description** `D = info(H,S)` returns information about the Haver Analytics variable, S.

**Examples**

Establish a Haver Analytics connection H:

```
H = haver('d:\work\haver\data\haverd.dat')
```

Request information for the variable after FFED:

```
D = info(H, 'FFED2')
VarName: 'FFED2'
    StartDate: '01-Jan-1991'
    EndDate: '31-Dec-1998'
    NumberObs: 2088
    Frequency: 'D'
    DateTimeMod: 1.1335e+009
    Magnitude: 0
    DecPrecision: 2
    DifType: 1
    AggType: 'AVG'
    DataType: '%'
    Group: 'Z05'
    Source: 'FRB'
    Descriptor:
'Federal Funds [Effective] Rate (% p.a.)'
    ShortSource: 'History'
```

LongSource: 'Historical Series'

## **See Also**

`haver.close`, `haver.get`, `haver.isconnection`, `haver`,  
`haver.nextinfo`

|                    |   |   |   |
|--------------------|---|---|---|
| <b>Purpose</b>     | Verify whether connections to Haver Analytics data servers are valid  |   |   |
| <b>Syntax</b>      | <code>X = isconnection(H)</code>  |   |   |
| <b>Arguments</b>   | <table><tr><td>H</td><td>Haver connection object created with the <code>haver</code> function.</td></tr></table>  | H | Haver connection object created with the <code>haver</code> function. |
| H                  | Haver connection object created with the <code>haver</code> function.   |   |   |
| <b>Description</b> | <code>X = isconnection(H)</code> returns <code>X = 1</code> if the connection is a valid Haver Analytics connection, and <code>X = 0</code> otherwise.  |   |   |
| <b>Examples</b>    | <p>Establish a Haver connection H:</p> <pre>H = HAVER('d:\work\haver\data\haverd.dat')</pre> <p>Verify that H is a valid Haver Analytics connection:</p> <pre>X = isconnection(H) X = 1</pre> |   |   |
| <b>See Also</b>    | <code>haver.close</code> , <code>haver.fetch</code> , <code>haver.get</code> , <code>haver</code>   |   |   |

# haver.nextinfo

---

**Purpose** Retrieve information about next Haver Analytics variable

**Syntax** `D = nextinfo(H,S)`

## Arguments

**H** Haver Analytics connection object created with the `haver` function.

**S** Haver Analytics variable.

**Description** `D = nextinfo(H,S)` returns information for the next Haver Analytics variable after the variable, `S`.

## Examples

Establish a Haver Analytics connection `H`:

```
H = haver('d:\work\haver\data\haverd.dat')
```

Request information for the variable following `FFED`:

```
D = nextinfo(H, 'FFED')
VarName: 'FFED2'
      StartDate: '01-Jan-1991'
      EndDate:   '31-Dec-1998'
      NumberObs: 2088
      Frequency: 'D'
      DateTimeMod: 1.1335e+009
      Magnitude: 0
      DecPrecision: 2
      DifType: 1
      AggType: 'AVG'
      DataType: '%'
      Group: 'Z05'
      Source: 'FRB'
      Descriptor:
'Federal Funds [Effective] Rate (% p.a.)'
      ShortSource: 'History'
```

LongSource: 'Historical Series'

**See Also**

`haver.close`, `haver.get`, `haver`, `haver.info`, `haver.isconnection`

# havertool

**Purpose** Run Haver Analytics graphical user interface (GUI)

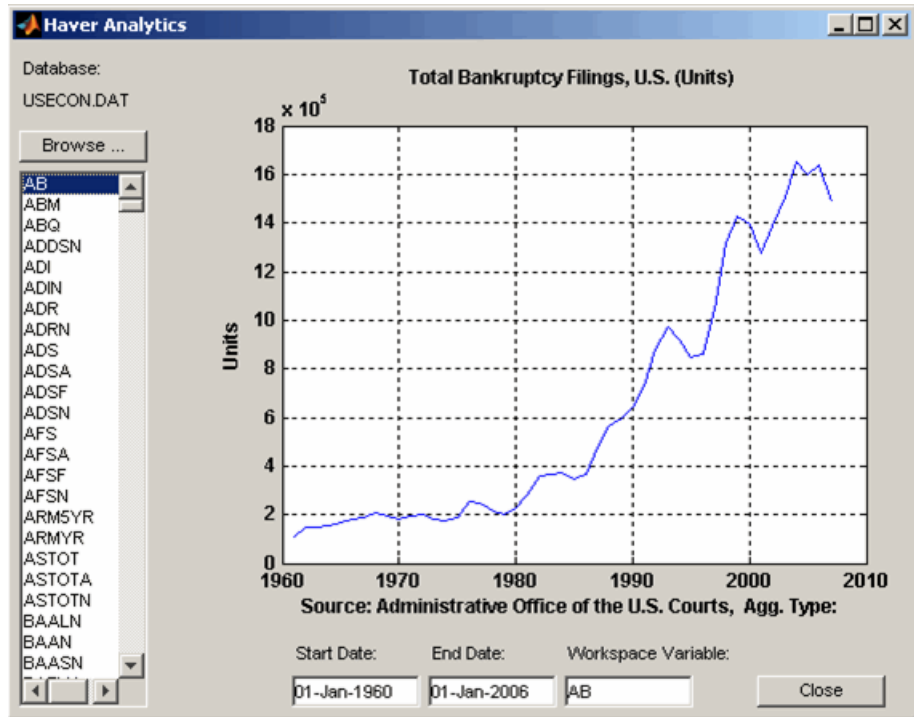
**Syntax** havertool(H)

**Arguments**

H Haver Analytics connection object created with the haver function.

**Description**

havertool(H) runs the Haver Analytics graphical user interface (GUI). The GUI appears in the following figure.





The GUI fields and buttons are:

- **Database:** The currently selected Haver Analytics database.
- **Browse:** Allows you to browse for Haver Analytics databases, and populates the variable list with the variables in the database you specify.
- **Start Date:** The data start date of the selected variable.
- **End Date:** The data end date of the selected variable.
- **Workspace Variable:** The MATLAB variable to which havertool writes data for the currently selected Haver variable.
- **Close:** Closes all current connections and the Haver Analytics GUI.

## Examples

Establish a Haver Analytics connection H:

```
H = haver('d:\work\haver\data\haverd.dat')
```

Open the graphical user interface (GUI) demonstration:

```
havertool(H)
```

## See Also

haver

# idc

---

|                    |   |
|--------------------|---|
| <b>Purpose</b>     | Connect to Interactive Data Pricing and Reference Data's RemotePlus data servers  |
| <b>Syntax</b>      | <code>Connect = idc</code>  |
| <b>Description</b> | <code>Connect = idc</code> connects to the Interactive Data Pricing and Reference Data's RemotePlus server. <code>Connect</code> is a connection handle used by other functions to obtain data. |
| <b>Examples</b>    | Connect to a Interactive Data Pricing and Reference Data's RemotePlus server:<br><br><code>c = idc</code>   |
| <b>See Also</b>    | <code>idc.close</code> , <code>idc.fetch</code> , <code>idc.get</code> , <code>idc.isconnection</code>  |

|                    |   |         |  |
|--------------------|---|---------|--|
| <b>Purpose</b>     | Close connections to Interactive Data Pricing and Reference Data's RemotePlus data servers  |         |  |
| <b>Syntax</b>      | <code>close(Connect)</code>   |         |  |
| <b>Arguments</b>   | <table><tr><td>Connect</td><td>Interactive Data Pricing and Reference Data's RemotePlus connection object created with the <code>idc</code> function.</td></tr></table>       | Connect | Interactive Data Pricing and Reference Data's RemotePlus connection object created with the <code>idc</code> function. |
| Connect            | Interactive Data Pricing and Reference Data's RemotePlus connection object created with the <code>idc</code> function.  |         |  |
| <b>Description</b> | <code>close(Connect)</code> closes the connection to the Interactive Data Pricing and Reference Data's RemotePlus server.   |         |  |
| <b>Examples</b>    | <p>Establish an Interactive Data Pricing and Reference Data's RemotePlus connection, <code>c</code>:</p> <pre>c = idc</pre> <p>Close this connection:</p> <pre>close(c)</pre> |         |  |
| <b>See Also</b>    | <code>idc</code>  |         |  |

# idc.fetch

---

**Purpose** Request data from Interactive Data Pricing and Reference Data's RemotePlus data servers

**Syntax**

```
data = fetch(Connect, 'Security', 'Fields')
data = fetch(Connect, 'Security', 'Fields', 'FromDate',
            'ToDate')
data = fetch(Connect, 'Security', 'Fields', 'FromDate',
            'ToDate', 'Period')
data = fetch(Connect, '', 'GUILookup', 'GUICategory')
```

**Arguments**

|  |   |
|--|---|
| Connect  | Interactive Data Pricing and Reference Data's RemotePlus connection object created with the <code>idc</code> function.  |
| 'Security'   | A MATLAB string containing the name of a security in a format recognizable by the Interactive Data Pricing and Reference Data's RemotePlus server.  |
| 'Fields'   | A MATLAB string or cell array of strings indicating specific fields for which to provide data. Valid field names are in the file <code>@idc/idcfields.mat</code> . The variable <code>bbfieldnames</code> contains the list of field names. |
| 'FromDate'   | Beginning date for historical data.   |
| <hr/> <b>Note</b> You can specify dates in any of the formats supported by <code>datestr</code> and <code>datenum</code> that show a year, month, and day. <hr/> |   |
| 'ToDate'   | End date for historical data.   |

- 'Period'            Period within date range.
- 'GUICategory'      GUI category. Possible values are:
- 'F' (All valid field categories)
  - 'S' (All valid security categories)

## Description

`data = fetch(Connect, 'Security', 'Fields')` returns data for the indicated fields of the designated securities. Load the file `idc/idcfields` to see the list of supported fields.

`data = fetch(Connect, 'Security', 'Fields', 'FromDate', 'ToDate')` returns historical data for the indicated fields of the designated securities.

`data = fetch(Connect, 'Security', 'Fields', 'FromDate', 'ToDate', 'Period')` returns historical data for the indicated fields of the designated securities with the designated dates and period. Consult the Remote Plus documentation for a list of valid 'Period' values.

`data = fetch(Connect, '', 'GUILookup', 'GUICategory')` opens the Interactive Data Pricing and Reference Data's RemotePlus dialog box for selecting fields or securities.

## Examples

Open the dialog box to look up securities:

```
D = fetch(Connect, '', 'GUILookup', 'S')
```

Open the dialog box to select fields:

```
D = fetch(Connect, '', 'GUILookup', 'F')
```

## See Also

`idc.close`, `idc.get`, `idc`, `idc.isconnection`

# idc.get

---

**Purpose** Retrieve properties of Interactive Data Pricing and Reference Data's RemotePlus connection objects

**Syntax**  
`value = get(Connect, 'PropertyName')`  
`value = get(Connect)`

**Arguments**

|              |  |
|--------------|--|
| Connect      | Interactive Data Pricing and Reference Data's RemotePlus connection object created with the <code>idc</code> function.   |
| PropertyName | (Optional) A MATLAB string or cell array of strings containing property names. Property names are: <ul style="list-style-type: none"><li>• 'Connected'</li><li>• 'Connection'</li><li>• 'Queued'</li></ul> |

**Description**

`value = get(Connect, 'PropertyName')` returns the value of the specified properties for the Interactive Data Pricing and Reference Data's RemotePlus connection object. `PropertyName` is a string or cell array of strings containing property names.

`value = get(Connect)` returns a MATLAB structure. Each field name is the name of a property of `Connect`, and each field contains the value of that property.

**See Also** `idc.close`, `idc.get`, `idc`, `idc.isconnection`

|                    |  |         |  |
|--------------------|--|---------|--|
| <b>Purpose</b>     | Verify whether connections to Interactive Data Pricing and Reference Data's RemotePlus data servers are valid  |         |  |
| <b>Syntax</b>      | <code>x = isconnection(Connect)</code>   |         |  |
| <b>Arguments</b>   | <table><tr><td>Connect</td><td>Interactive Data Pricing and Reference Data's RemotePlus connection object created with the <code>idc</code> function.</td></tr></table>  | Connect | Interactive Data Pricing and Reference Data's RemotePlus connection object created with the <code>idc</code> function. |
| Connect            | Interactive Data Pricing and Reference Data's RemotePlus connection object created with the <code>idc</code> function.   |         |  |
| <b>Description</b> | <code>x = isconnection(Connect)</code> returns <code>x = 1</code> if the connection is a valid Interactive Data Pricing and Reference Data's RemotePlus connection, and <code>x = 0</code> otherwise.                    |         |  |
| <b>Examples</b>    | <p>Establish an Interactive Data Pricing and Reference Data's RemotePlus connection <code>c</code>:</p> <pre>c = idc</pre> <p>Verify that <code>c</code> is a valid connection:</p> <pre>x = isconnection(c) x = 1</pre> |         |  |
| <b>See Also</b>    | <code>idc.close</code> , <code>idc.fetch</code> , <code>idc.get</code> , <code>idc</code>  |         |  |

**Purpose** Connect to Kx Systems, Inc. kdb+ databases

**Syntax**  
K = kx(IP,P)  
K = kx(IP,P,ID)

**Arguments**

|    |  |
|----|--|
| IP | IP address for the connection to the Kx Systems, Inc. kdb+ database.                   |
| P  | Port for the Kx Systems, Inc. kdb+ database connection.                                |
| ID | The <i>username:password</i> string for the Kx Systems, Inc. kdb+ database connection. |

**Description** K = kx(IP,P) connects to the Kx Systems, Inc. kdb+ database given the IP address IP and port number P.

K = kx(IP,P,ID) connects to the Kx Systems, Inc. kdb+ database given the IP address IP, port number P, and *username:password* string ID.

Before you connect to the database, add The Kx Systems, Inc. file `jdbc.jar` to the MATLAB `javaclasspath` using the `javaaddpath` command. The following example adds `jdbc.jar` to the MATLAB `javaclasspath` `c:\q\java`:

```
javaaddpath c:\q\java\jdbc.jar
```

---

**Note** In earlier versions of the Kx Systems, Inc. kdb+ database, this jar file was named `kx.jar`. If you are running an earlier version of the database, substitute `kx.jar` for `jdbc.jar` in these instructions to add this file to the MATLAB `javaclasspath`.

---

**Examples** Run the following command from a DOS prompt to specify the port number 5001:

```
q tradedata.q -p 5001
```



Connect to a Kx Systems, Inc. server using IP address LOCALHOST and port number 5001:

```
K = kx('LOCALHOST',5001)
handle: [1x1 c]
        ipaddress: 'localhost'
        port: 5001
```

**See Also**

`kx.close`, `kx.exec`, `kx.get`, `kx.fetch`, `kx.tables`

# kx.close

---

|                    |   |   |  |
|--------------------|---|---|--|
| <b>Purpose</b>     | Close connections to Kx Systems, Inc. kdb+ databases  |   |  |
| <b>Syntax</b>      | <code>close(K)</code>   |   |  |
| <b>Arguments</b>   | <table><tr><td>K</td><td>Kx Systems, Inc. kdb+ connection object created with the <code>kx</code> function.</td></tr></table> | K | Kx Systems, Inc. kdb+ connection object created with the <code>kx</code> function. |
| K                  | Kx Systems, Inc. kdb+ connection object created with the <code>kx</code> function.  |   |  |
| <b>Description</b> | <code>close(K)</code> closes the connection to the Kx Systems, Inc. kdb+ database.  |   |  |
| <b>Examples</b>    | Close the connection, K, to the Kx Systems, Inc. kdb+ database:<br><pre>close(K)</pre>  |   |  |
| <b>See Also</b>    | <code>kx</code>   |   |  |

**Purpose**

Request data from Kx Systems, Inc. kdb+ databases

**Syntax**

```
D = fetch(K, KSQL)
D = fetch(K, KSQL, P1, P2, P3)
```

**Arguments**

|            |  |
|------------|--|
| K          | Kx Systems, Inc. kdb+ connection object created with the <code>kx</code> function. |
| KSQL       | The Kx Systems, Inc. kdb+ command.   |
| P1, P2, P3 | Input parameters for the KSQL command.   |

**Description**

`D = fetch(K, KSQL)` returns data from a Kx Systems, Inc. kdb+ database in a MATLAB structure where `K` is the Kx Systems, Inc. kdb+ object and `KSQL` is the Kx kdb+ command. `KSQL` can be any valid kdb+ command. The output of this method is any data resulting from the command specified in `KSQL`.

`D = fetch(K, KSQL, P1, P2, P3)` executes the command specified in `KSQL` with one or more input parameters, and returns the data from this command.

**Examples**

Run the following command from a DOS prompt to specify the port number 5001:

```
q tradedata.q -p 5001
```

Connect to a Kx Systems, Inc. server using IP address `LOCALHOST` and port number 5001:

```
K = kx('localhost', 5001);
```

Retrieve data using the command `select from trade`:

```
D = fetch(K, 'select from trade');
D =
```

# kx.fetch

---

```
sec: {5000x1 cell}
  price: [5000x1 double]
  volume: [5000x1 int32]
  exchange: [5000x1 double]
  date: [5000x1 double]
```

Retrieve data, passing an input parameter 'ACME' to the command  
select from trade:

```
D = fetch(K, 'totalvolume', 'ACME');
D =
  volume: [1253x1 int32]
```

This is the total trading volume for the security ACME in the table trade.  
The function `totalvolume` is defined in the sample Kx Systems, Inc.  
kdb+ file, `tradedata.q`.

## See Also

`kx.exec`, `kx.insert`, `kx`

**Purpose** Retrieve Kx Systems, Inc. kdb+ connection object properties

**Syntax**  
`V = get(K, 'PropertyName')`  
`V = get(K)`

## Arguments

**K** Kx Systems, Inc. kdb+ connection object created with the `kx` function.

**'PropertyName'** A string or cell array of strings containing property names. The property names are:

- 'handle'
- 'ipaddress'
- 'port'

## Description

`V = get(K, 'PropertyName')` returns a MATLAB structure containing the value of the specified properties for the Kx Systems, Inc. kdb+ connection object.

`V = get(K)` returns a MATLAB structure where each field name is the name of a property of `K` and the associated value of the property.

## Examples

Get the properties of the connection to the Kx Systems, Inc. kdb+ database, `K`:

```
V = get(K)
V =
    handle: [1x1 c]
    ipaddress: 'localhost'
    port: '5001'
```

## See Also

`kx.close`, `kx.exec`, `kx.fetch`, `kx.insert`, `kx`

# kx.exec

---

**Purpose** Run Kx Systems, Inc. kdb+ commands

**Syntax** `exec(K,Command)`  
`exec(K,Command,P1,P2,P3)`

## Arguments

|          |   |
|----------|---|
| K        | Kx Systems, Inc. kdb+ connection object created with the <code>kx</code> function.  |
| Command  | Kx Systems, Inc. kdb+ command issued using the Kx Systems, Inc. kdb+ connection object created with the <code>kx</code> function. |
| P1,P2,P3 | Input parameters for Command.   |

## Description

`exec(K,Command)` executes the specified Command in Kx Systems, Inc. kdb+ without waiting for a response.

`exec(K,Command,P1,P2,P3)` executes the specified Command with one or more input parameters without waiting for a response.

## Examples

- Retrieve the data in the table `trade` using the connection to the Kx Systems, Inc. kdb+ database, K:

```
K = kx('localhost',5001);
```

- Use the `exec` command to sort the data in the table `trade` in ascending order.

```
exec(K,``date xasc`trade`);
```

Subsequent data requests also sort returned data in ascending order.

## See Also

`kx.fetch`, `kx.insert`, `kx`

**Purpose** Write data to Kx Systems, Inc. kdb+ databases

**Syntax** `insert(K,Tablename,Data)`

## Arguments

|                        |  |
|------------------------|--|
| <code>K</code>         | The Kx Systems, Inc. kdb+ connection object created with the <code>kx</code> function.         |
| <code>Tablename</code> | The name of the Kx Systems, Inc. kdb+ <code>Tablename</code> .                                 |
| <code>Data</code>      | The data that <code>insert</code> writes to the Kx Systems, Inc. kdb+ <code>Tablename</code> . |

**Description** `insert(K,Tablename,Data)` writes the data, `DATA`, to the Kx Systems, Inc. kdb+ table, `Tablename`.

**Examples** For the connection to the Kx Systems, Inc. kdb+ database,`K`, write data from `ACME` to the specified table:

```
insert(K, 'trade', {'`ACME', 133.51, 250, 6.4, '2006.10.24'})
```

**See Also** `kx.close`, `kx.fetch`, `kx.get`, `kx.tables`

# kx.isconnection

---

**Purpose** Verify whether connections to Kx Systems, Inc. kdb+ databases are valid

**Syntax** `X = isconnection(K)`

**Arguments**

|   |  |
|---|--|
| K | Kx Systems, Inc. kdb+ connection object created with the <code>kx</code> function. |
|---|--|

**Description** `X = isconnection(K)` returns `X = 1` if the connection to the Kx Systems, Inc. kdb+ database is valid, and `x = 0` otherwise.

**Examples** Establish a connection to a Kx Systems, Inc. kdb+ database, K:

```
K = kx('localhost',5001);
```

Verify that K is a valid connection:

```
X = isconnection(K)
X = 1
```

**See Also** `kx.close`, `kx.fetch`, `kx.get`, `kx`



**Purpose** Retrieve table names from Kx Systems, Inc. kdb+ databases

**Syntax** `T = tables(K)`

**Arguments**

**K** The Kx Systems, Inc. kdb+ connection object created with the `kx` function.

**Description** `T = tables(K)` returns the list of tables for the Kx Systems, Inc. kdb+ connection.

**Examples** Retrieve table information for the Kx Systems, Inc. kdb+ database using the connection `K`:

```
T = tables(k)
T =
    'intraday'
    'seclist'
    'trade'
```

**See Also** `kx.exec`, `kx.fetch`, `kx.insert`, `kx`

# rdthloader

---

**Purpose** Retrieve data from Reuters Datascope Tick History file

**Syntax**

```
x = rdthloader(file)
x = rdthloader(file,'date',{DATE1})
x = rdthloader(file,'date',{DATE1, DATE2})
x = rdthloader(file,'security',{SECNAME})
x = rdthloader(file,'start',STARTREC)
x = rdthloader(file,'records', NUMRECORDS)
```

**Arguments** Specify the following arguments as name-value pairs. You can specify any combination of name-value pairs in a single call to `rdthloader`.

|                         |   |
|-------------------------|---|
| <code>file</code>       | Reuters Datascope Tick History file from which to retrieve data.  |
| <code>'date'</code>     | Use this argument with <code>{DATE1, DATE2}</code> to retrieve data between and including the specified dates. Specify the dates as numbers or strings.   |
| <code>'security'</code> | Use this argument to retrieve data for <code>SECNAME</code> , where <code>SECNAME</code> is a cell array containing a list of security identifiers for which to retrieve data.  |
| <code>'start'</code>    | Use this argument to retrieve data beginning with the record <code>STARTREC</code> , where <code>STARTREC</code> is the record at which <code>rdthloader</code> begins to retrieve data. Specify <code>STARTREC</code> as a number. |
| <code>'records'</code>  | Use this argument to retrieve <code>NUMRECORDS</code> number of records.  |

**Description**

- `x = rdthloader(file)` retrieves tick data from the Reuters Datascope Tick History file `file` and stores it in the structure `x`.
- `x = rdthloader(file,'date',{DATE1})` retrieves tick data from `file` with date stamps of value `DATE1`.

- `x = rdthloader(file, 'date', {DATE1, DATE2})` retrieves tick data from `file` with date stamps between `DATE1` and `DATE2`.
- `x = rdthloader(file, 'security', {SECNAME})` retrieves tick data from `file` for the securities specified by `SECNAME`.
- `x = rdthloader(file, 'start', STARTREC)` retrieves tick data from `file` beginning with the record specified by `STARTREC`.
- `x = rdthloader(file, 'records', NUMRECORDS)` retrieves `NUMRECORDS` number of records from `file`.

## Examples

- Retrieve all ticks from the file `file.csv` with date stamps of `02/02/2007`:

```
x = rdthloader('file.csv', 'date', {'02/02/2007'})
```

- Retrieve all ticks from `file.csv` between and including the dates `02/02/2007` and `02/03/2007`:

```
x = rdthloader('file.csv', 'date', {'02/02/2007', '02/03/2007'})
```

- Retrieve all ticks from `file.csv` for the security `XYZ.0`:

```
x = rdthloader('file.csv', 'security', {'XYZ.0'})
```

- Retrieve the first 10000 tick records from `file.csv`:

```
x = rdthloader('file.csv', 'records', 10000)
```

- Retrieve data from `file.csv`, starting at record 100000:

```
x = rdthloader('file.csv', 'start', 100000)
```

# rdthloader

---

- Retrieve up to 100000 tick records from `file.csv`, for the securities `ABC.N` and `XYZ.O`, with date stamps between and including the dates `02/02/2007` and `02/03/2007`:

```
x = rdthloader('file.csv', 'records', 100000, ...  
              'date', {'02/02/2007', '02/03/2007'}, ...  
              'security', {'ABC.N', 'XYZ.O'})
```

## See Also

`reuters`, `rnseloader`

**Purpose** Create Reuters sessions

**Syntax**

```
r = reuters (sessionName, serviceName)
r = reuters (sessionName, serviceName, user, position)
```

## Arguments

|                          |  |
|--------------------------|--|
| <code>r</code>           | Reuters session object created with the reuters function                     |
| <code>sessionName</code> | Name of the Reuters session, of the form <code>myNameSpace::mySession</code> |
| <code>serviceName</code> | Name of the service you use to connect to the data server.                   |
| <code>user</code>        | User ID you use to connect to the data server                                |
| <code>position</code>    | IP address of the data server to which you connect to retrieve data.         |

## Description

You must configure your environment before you use this function to connect to a Reuters data server. For more information, see “Reuters Data Service Requirements” on page 1-5.

`r = reuters (sessionName, serviceName)` starts a Reuters session where `sessionName` is of the form `myNameSpace::mySession` and `serviceName` specifies the name of the service you use to connect to the data server.

`r = reuters (sessionName, serviceName, user, position)` starts a Reuters session where `sessionName` is of the form `myNameSpace::mySession` and `serviceName` is the service to use, `user` is the user ID, and `position` is the IP address of the machine to which you connect to retrieve data. Use this form of the command if you require DACS authentication.

## Examples

### Connecting to Reuters Data Servers

Connect to a Reuters data server with session name 'myNS::remoteSession' and service name 'dIDN\_RDF':

```
r = reuters ('myNS::remoteSession', 'dIDN_RDF')
r =
  session: [1x1 com.reuters.rfa.internal.session.SessionImpl]
  user: []
  serviceName: 'dIDN_RDF'
  standardPI:
  [1x1 com.reuters.rfa.common.StandardPrincipalIdentity]
  eventQueue: [Error]
  marketDataSubscriber:
  [1x1 com.reuters.rfa.internal.session.
  MarketDataSubscriberImpl]
  marketDataSubscriberInterestSpec:
  [1x1 com.reuters.rfa.session.MarketDataSubscriber
  InterestSpec]
  client:
  [1x1 com.mathworks.toolbox.datafeed.MatlabReutersClient]
  mdsClientHandle:
  [1x1 com.reuters.rfa.internal.common.HandleImpl]
```

---

**Note** If you do not use the Reuters DACS authentication functionality, the following error message appears:

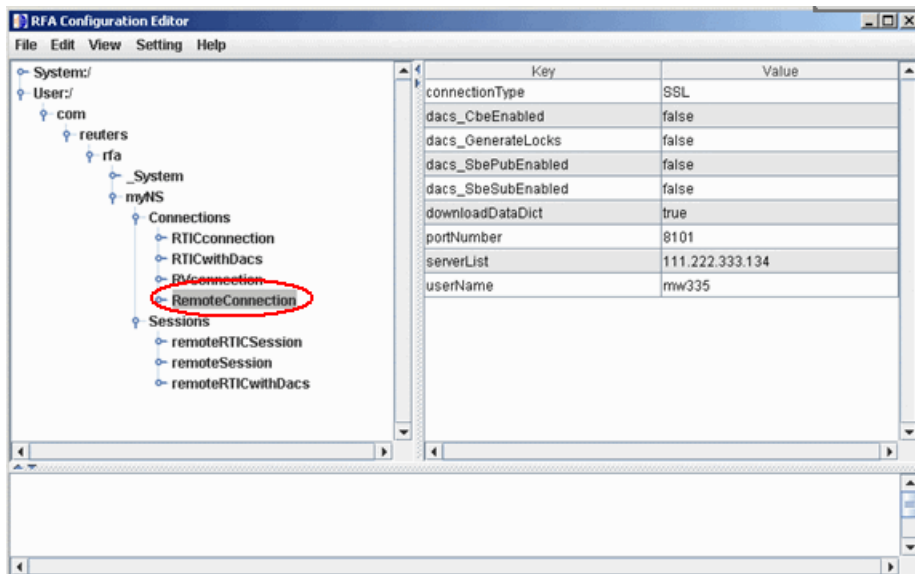
```
com.reuters.rfa.internal.connection.ConnectionImpl
initializeEntitlementsINFO:
com.reuters.rfa.connection.ssl.myNS.RemoteConnection
DACS disabled for connection myNS::RemoteConnection
```

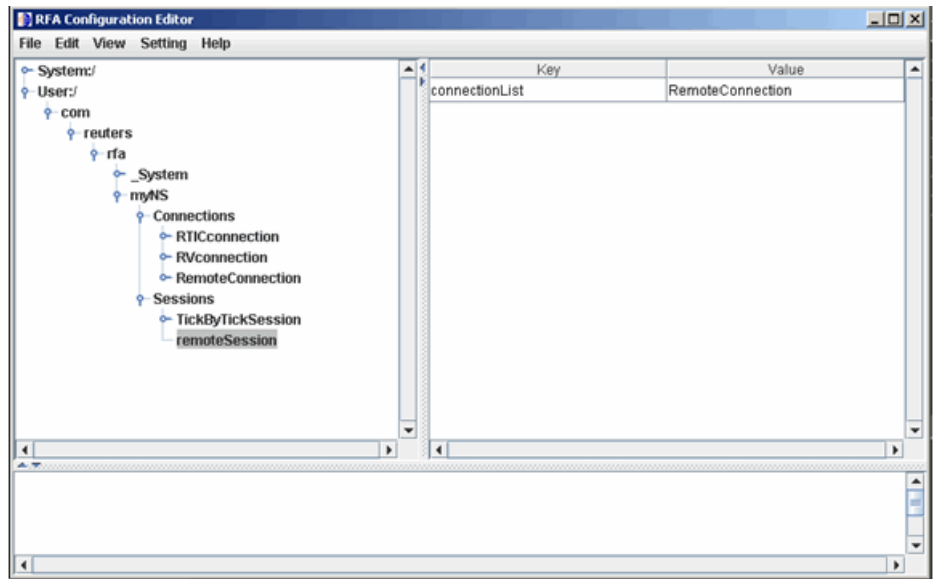
---

### Connecting to Reuters Data Servers Using DACS Authentication

- 1 Connect to a Reuters data server using DACS authentication, with session name 'myNS::remoteSession', service name 'dIDN\_RDF', user id 'ab123', and data server IP address '111.222.333.444/net':

```
r = reuters ('myNS::remoteSession', 'dIDN_RDF', ...  
'ab123', '111.222.333.444/net')
```





- 2 Add the following to your connection configuration:

```
dacs_CbeEnabled=false  
dacs_SbePubEnabled=false  
dacs_SbeSubEnabled=false
```

- 3 If you are running an SSL connection, add the following to your connection configuration:

```
dacs_GenerateLocks=false
```

## Connecting to Reuters Data Servers Without DACS Authentication

Connect to a Reuters data server with session name 'myNS::remoteSession' and service name 'dIDN\_RDF', without using DACS:

```
r = reuters ('myNS::remoteSession', 'dIDN_RDF')
```

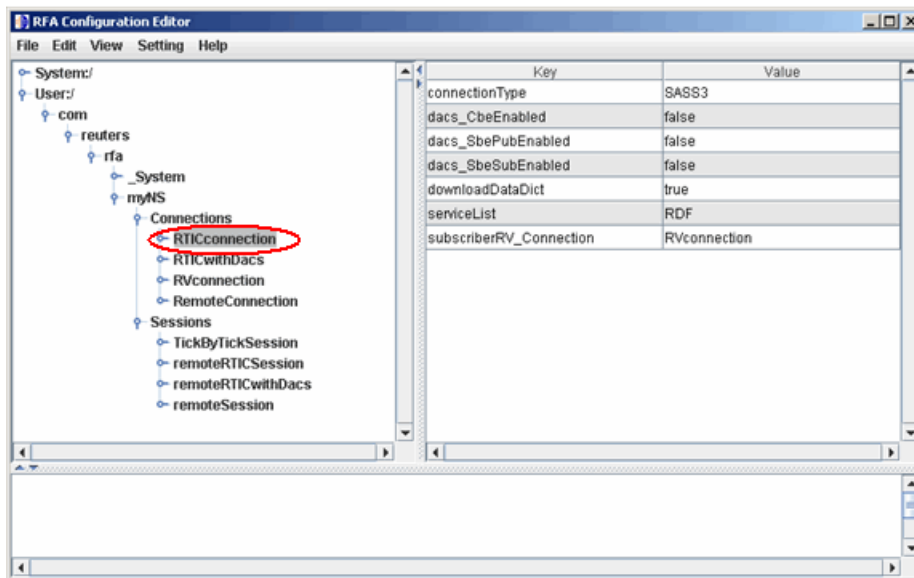


## Establishing an RTIC (TIC-RMDS Edition) Connection to Reuters Data Servers

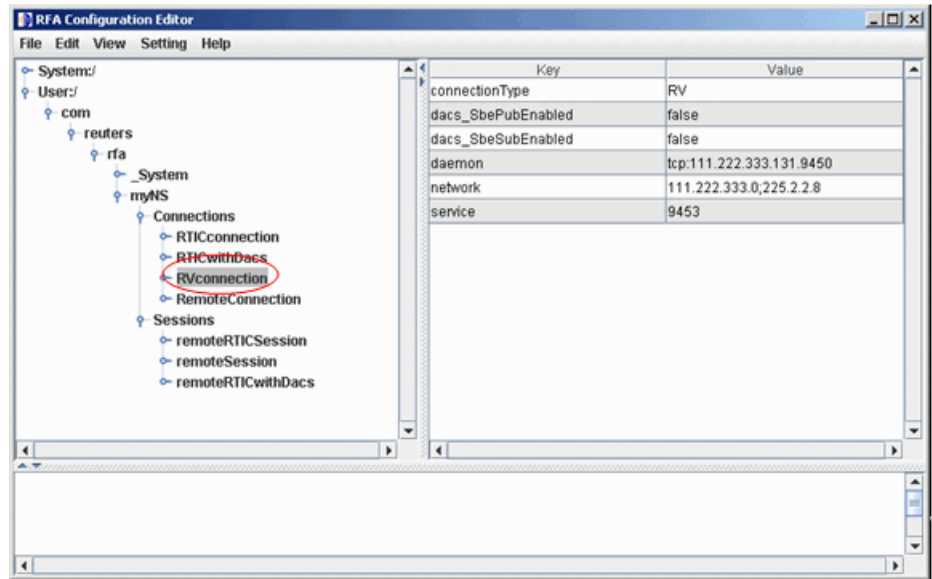
- Non-DACs-enabled

Make an RTIC (TIC-RMDS Edition) connection to a Reuters data server without DACS authentication, with session name 'myNS::remoteRTICSession', service name 'IDN\_RDF':

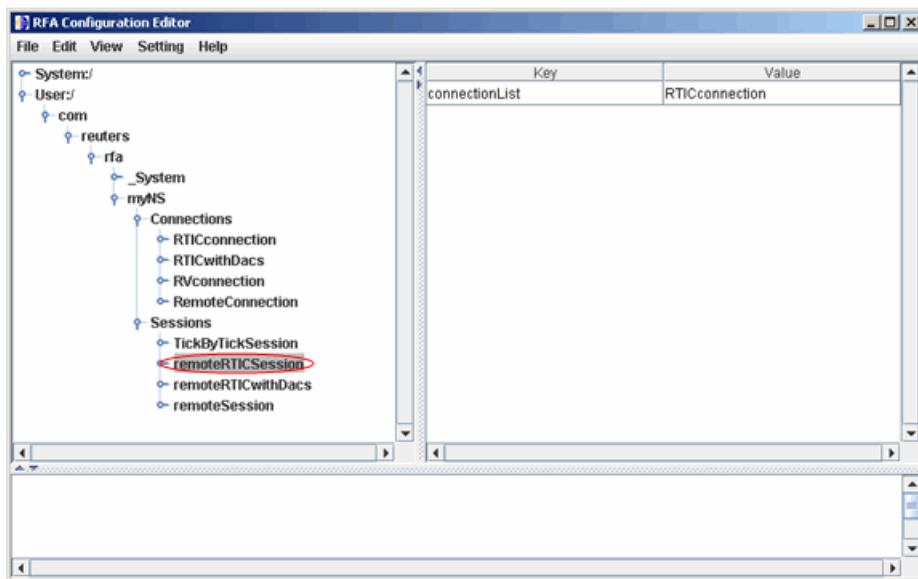
```
r = reuters ('myNS::remoteRTICSession','IDN_RDF')
```



This RTIC connection depends on the key subscriber RVConnection. Your RVConnection configuration should look as follows:



The RTICConnection configuration is referenced by the session remoteRTICSession, as shown in the following figure.



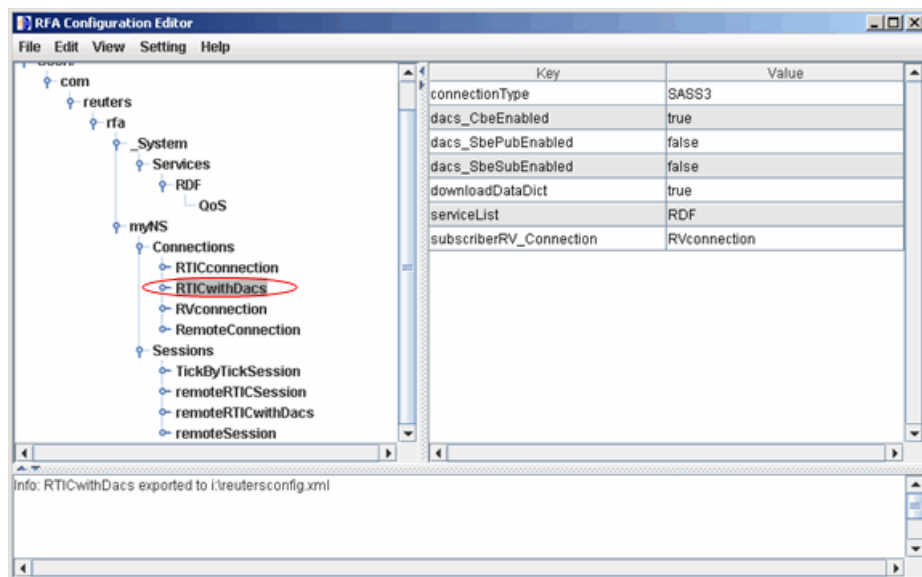
Messages like the following may appear in the MATLAB Command Window when you establish a non-DACs-enabled connection. These messages are informational and can safely be ignored.

```
Oct 5, 2007 2:28:31 PM
com.reuters.rfa.internal.connection.
ConnectionImpl initializeEntitlements
INFO: com.reuters.rfa.connection.ssl....
    myNS.RemoteConnection
    DACS disabled for connection myNS::RemoteConnection
```

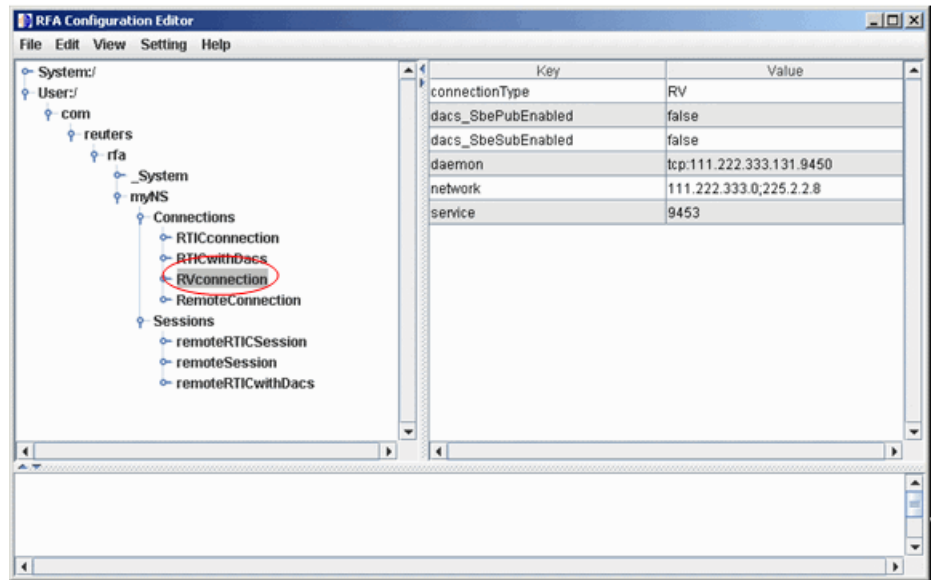
- **DACs-enabled**

Make an RTIC (TIC-RMDS Edition), DACS-enabled connection to a Reuters data server, with session name 'myNS::remoteRTICWithDACs', service name 'IDN\_RDF', user id 'ab123', and data server IP address '111.222.333.444/net':

```
r = reuters ('myNS::remoteRTICWithDACs', 'IDN_RDF', ...
'ab123', '111.222.333.444/net')
```



This RTIC connection depends on the key subscriber RVConnection. Your RVConnection configuration should look as follows:



Messages like the following may appear in the MATLAB Command Window when you establish a DACs-enabled connection. These messages are informational and can be ignored safely.

```
Oct 5, 2007 2:27:14 PM ...
com.reuters.rfa.internal.connection.
ConnectionImpl$ConnectionEstablishmentThread runImpl
INFO: com.reuters.rfa.connection.sass3.myNS.RTICwithDacs
Connection successful: ...
    componentName :myNS::RTICwithDacs,
subscriberRVConnection:
{service: 9453, network: 192.168.107.0;225.2.2.8,
daemon: tcp:192.168.107.131:9450}
Oct 5, 2007 2:27:14 PM
com.reuters.rfa.internal.connection.sass3....
    Sass3LoggerProxy log
INFO: com.reuters.rfa.connection.sass3.myNS.RTICwithDacs
SASS3JNI: Received advisory from RV session@
```

```
(9453,192.168.107.0;225.2.2.8,tcp:192.168.107.131:9450):
_RV.INFO.SYSTEM.RVD.CONNECTED
Oct 5, 2007 2:27:14 PM
com.reuters.rfa.internal.connection.ConnectionImpl
makeServiceInfo
WARNING: com.reuters.rfa.connection.sass3....
    myNS.RTICwithDacs
Service list configuration has no
    alias defined for network
serviceName IDN_RDF
```

If messages like the following appear in the MATLAB Command Window when you establish a DACs-enabled connection:

```
SEVERE: com.reuters.rfa.entitlements._Default.Global
DACs initialization failed:
com.reuters.rfa.dacs.AuthorizationException:
Cannot start the DACs Library thread due to -
Cannot locate JNI library - RFADacsLib
```

Then add an entry to the `$MATLAB/toolbox/local/librarypath.txt` file that points to the directory containing the following files:

- `FDacsLib.dll`
- `sass3j.dll`
- `sipc32.dll`

## See Also

`reuters.fetch`

# reuters.close

---

**Purpose** Release connections to Reuters data servers

**Syntax** `close(r)`

**Arguments** `r` Reuters session object created with the `reuters` function

**Description** `close(r)` releases the Reuters connection `r`.

**Examples** Release the connection `r` to the Reuters data server, and unsubscribe all requests associated with it:

```
close(r)
```

**See Also** `reuters`



**Purpose** Request data from Reuters data servers

**Syntax**  
`d = fetch (r,s)`  
`d = fetch (r,s, callback)`

**Arguments**

|                       |   |
|-----------------------|---|
| <code>r</code>        | Reuters session object created with the reuters function  |
| <code>s</code>        | Reuters security object                                   |
| <code>callback</code> | MATLAB function that runs for each data event that occurs |

**Description** `d = fetch (r,s)` returns the current data for the security `s`, given the Reuters session object `r`.

`d = fetch (r,s, callback)` uses the Reuters session object `r` to subscribe to the security `s`. MATLAB runs the `callback` function for each data event that occurs.

**Examples      Retrieving Current Securities Data**

Retrieve the current data for the security `GOOG.0` using the Reuters session object `r`:

```
d = fetch(r, 'GOOG.0')
```

Following is a partial listing of the returned security data:

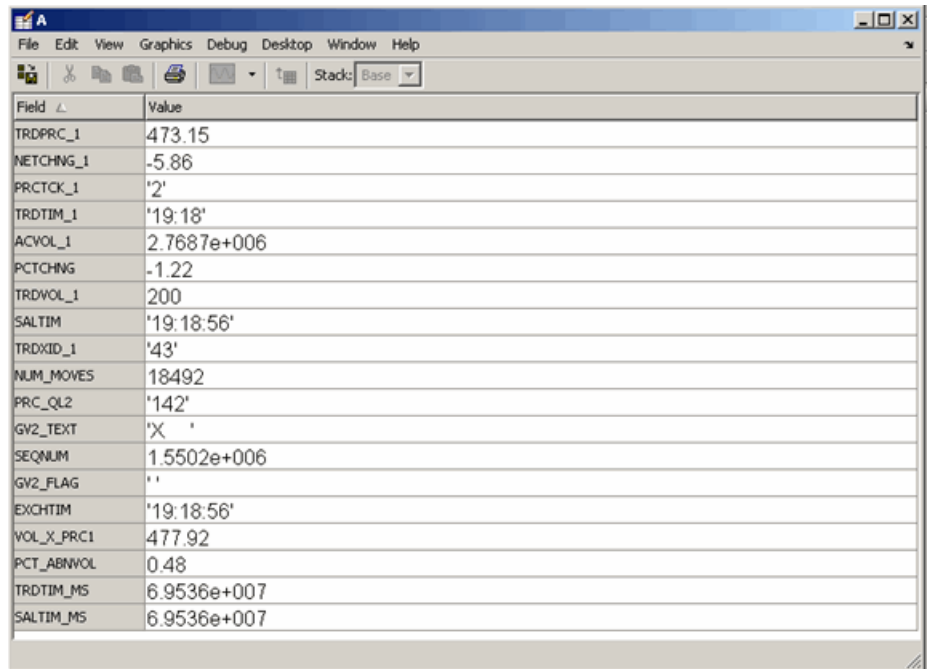
```
d =
  PROD_PERM: 74.00
  RDNDISPLAY: 66.00
  DSPLY_NAME: 'DELAYED-15GOOGLE'
  RDN_EXCHID: '0'
  TRDPRC_1: 474.28
  TRDPRC_2: 474.26
  TRDPRC_3: 474.25
  TRDPRC_4: 474.25
```

```
TRDPRC_5: 474.25
NETCHNG_1: -4.73
HIGH_1: 481.35
LOW_1: 472.78
PRCTCK_1: '1'
CURRENCY: '840'
TRADE_DATE: '30 APR 2007'
```

## Subscribing to a Security

To subscribe to a security and process the data in real time, specify a callback function. MATLAB runs this function each time it receives a real-time data event from Reuters. In this example, the callback function, `rtdemo`, returns the subscription handle associated with this request to the base MATLAB workspace, `A`. The `openvar` function is then called to display `A` in the Variable Editor. A partial list of the data included in `A` appears in the following figure.

```
d = fetch(r, 'GOOG.O', 'rtdemo')
d = com.reuters.rfa.internal.common.SubHandleImpl@75cea3
openvar('A')
```



The image shows a screenshot of a debugger window with a menu bar (File, Edit, View, Graphics, Debug, Desktop, Window, Help) and a toolbar. Below the toolbar is a table with two columns: 'Field' and 'Value'. The table contains the following data:

| Field      | Value       |
|------------|-------------|
| TRDPRC_1   | 473.15      |
| NETCHNG_1  | -5.86       |
| PRCTCK_1   | '2'         |
| TRDTIM_1   | '19:18'     |
| ACVOL_1    | 2.7687e+006 |
| PCTCHNG    | -1.22       |
| TRDVOL_1   | 200         |
| SALTIM     | '19:18:56'  |
| TRDXID_1   | '43'        |
| NUM_MOVES  | 18492       |
| PRC_QL2    | '142'       |
| GW2_TEXT   | 'X '        |
| SEQNUM     | 1.5502e+006 |
| GW2_FLAG   | ' '         |
| EXCHTIM    | '19:18:56'  |
| VOL_X_PRC1 | 477.92      |
| PCT_ABNWOL | 0.48        |
| TRDTIM_MS  | 6.9536e+007 |
| SALTIM_MS  | 6.9536e+007 |

**See Also** `reuters`, `reuters.close`, `reuters.stop`

# reuters.get

---

**Purpose** Retrieve properties of Reuters session objects

**Syntax**  
`e = get(r)`  
`e = get(r,f)`

**Arguments**

|                |   |
|----------------|---|
| <code>r</code> | Reuters session object created with the <code>reuters</code> function |
| <code>f</code> | Reuters session properties list                                       |

**Description**

`e = get(r)` returns Reuters session properties for the Reuters session object `r`.

`e = get(r,f)` returns Reuters session properties specified by the properties list `f` for the Reuters session object `r`.

**See Also** `reuters`

|                    |  |                |   |                |  |
|--------------------|--|----------------|---|----------------|--|
| <b>Purpose</b>     | Unsubscribe securities   |                |   |                |  |
| <b>Syntax</b>      | <code>stop(r)</code><br><code>stop(r,d)</code>   |                |   |                |  |
| <b>Arguments</b>   | <table><tr><td><code>r</code></td><td>Reuters session object created with the <code>reuters</code> function</td></tr><tr><td><code>d</code></td><td>Subscription handle returned by <code>reuters/fetch</code></td></tr></table>   | <code>r</code> | Reuters session object created with the <code>reuters</code> function | <code>d</code> | Subscription handle returned by <code>reuters/fetch</code> |
| <code>r</code>     | Reuters session object created with the <code>reuters</code> function  |                |   |                |  |
| <code>d</code>     | Subscription handle returned by <code>reuters/fetch</code>   |                |   |                |  |
| <b>Description</b> | <p><code>stop(r)</code> unsubscribes all securities associated with the Reuters session object <code>r</code>.</p> <p><code>stop(r,d)</code> unsubscribes the securities associated with the subscription handle <code>d</code>, where <code>d</code> is the subscription handle returned by <code>reuters/fetch</code>.</p> |                |   |                |  |
| <b>Examples</b>    | <p>Unsubscribe securities associated with a specific request <code>d</code> and a Reuters connection object <code>r</code>:</p> <pre>stop(r,d)</pre> <p>Unsubscribe all securities associated with the Reuters connection object <code>r</code>:</p> <pre>stop(r)</pre>  |                |   |                |  |
| <b>See Also</b>    | <code>reuters.fetch</code> , <code>reuters</code>  |                |   |                |  |

# rkd

---

**Purpose** Establish Reuters Knowledge Direct connection

**Syntax** `r = rkd (username, password)`

**Arguments**

|                       |   |
|-----------------------|---|
| <code>username</code> | User name required for a Reuters Knowledge Direct connection. |
| <code>password</code> | Password required for a Reuters Knowledge Direct connection.  |

**Description** `r = rkd (username, password)` establishes a Reuters Knowledge Direct connection.

**Examples** Establish a Reuters Knowledge Direct connection `r`:

```
r = rkd('username', 'password')
r =
    user: 'username'
    password: '*****'
```

**See Also** `rkd.fetch`

|                    |   |
|--------------------|---|
| <b>Purpose</b>     | Close Reuters Knowledge Direct connection   |
| <b>Syntax</b>      | <code>close(c)</code>   |
| <b>Arguments</b>   | <code>c</code> Reuters Knowledge Direct connection object.                              |
| <b>Description</b> | <code>close(c)</code> closes a Reuters Knowledge Direct connection.                     |
| <b>Examples</b>    | Close the Reuters Knowledge Direct connection <code>c</code> :<br><code>close(c)</code> |
| <b>See Also</b>    | <code>rkd</code>  |

# rkd.fetch

---

**Purpose** Request Reuters Knowledge Direct data

**Syntax**  
`y = fetch (r,s)`  
`y = fetch (r,s, 'name1', value1, 'name2', value2 ...)`

**Arguments**

|   |   |
|---|---|
| r | Reuters Knowledge Direct connection object. |
| s | Name of security for which to request data. |

**Optional Input Arguments**

|                 |  |
|-----------------|--|
| 'companyIdType' | Specify the type of security for which to request data. Valid values are: <ul style="list-style-type: none"><li>• 'TICKER' (default)</li><li>• 'RIC'</li><li>• 'VALOR'</li><li>• 'CUSIP'</li><li>• 'ISIN'</li><li>• 'SEDOL'</li><li>• 'ESTIMATEID'</li></ul> |
| 'countryCode'   | Enter a string value for 'countryCode', for example, 'US'. This field is empty by default.   |



'fperiod'  
(fiscal year)

Specify one or more fiscal years for which to return data. Enter multiple values as a cell array of strings. Valid values are:

- 'PRVS2'
- 'PRVS'
- 'CURR'
- 'NEXT'
- 'NEXT2'

'fperiod' is empty by default. This returns data for the previous 3 years.

'esttype'

Specify the type of estimate data to return. Enter multiple values as a cell array of strings. Valid values are:

- '\$PRIMARY'
- 'EPS'
- 'EPSEBG'
- 'EPSREP'
- 'FFO'
- 'REVENUE'
- 'EBITDA'
- 'CPS'
- 'DPS'
- 'NAV'
- 'NPROFIT'
- 'NPROFITREP'
- 'NPROFITEBG'
- 'OPROFIT'

- 'PPROFIT'
- 'PPROFITREP'
- 'PPROFITEBG'
- 'EBIT'
- 'LTGROWTH'
- 'TARGETPRICE'
- 'STOPINION'
- 'SUPOPINION'
- 'ROA'
- 'ROE'
- 'BVPS'

'esttype' is empty by default. This returns data for the previous 3 years for all esttypes.

'agg'  
(aggregation)

Enter multiple values for agg as a cell array of strings. Valid values are as follows:

- 'ATTR' (attributed)
- 'CONS' (consensus)
- 'ALL' (attributed and consensus)

'agg' is empty by default.

'earntype'

Get primary earnings data by assigning '\$PRIMARYEARN' to 'earntype'. This field is empty by default.

'hperiod'  
(historical  
period)

Specify a time period for which to obtain consensus estimates. This argument applies only to reports whose reportType value is ConsensusEstimates. Valid values are as follows:

- 'CURR' (Current period)
- '1WA' (1 week ago)
- '1MA' (1 month ago)
- '2MA'
- '3MA'
- '4MA'
- '5MA'
- '6MA'
- '7MA'
- '8MA'
- '9MA'
- '10MA'
- '11MA'
- '12MA'
- '13MA'
- '14MA'
- '15MA'
- '16MA'
- '17MA'
- '18MA'
- '3MHST' (3-month history)

- '12MHST'
- '18MHST'

'hperiod' is empty by default. This returns the current data.

'rTime'  
(real-time request)      Entering 1 for 'rTime' makes a real-time request for earnings estimates. This field is empty by default.

'reportType'      Specify what type of report to return. Valid values are:

- 'EarningsEstimates'
- 'ConsensusEstimates'
- 'EarningsEstimatesMetaInfo'

## Description

`y = fetch (r,s)` requests estimate data for all `esttype` fields of the security `s`, from the Reuters Knowledge Direct connection `r`.

`y = fetch (r,s, 'name1', value1, 'name2', value2 ...)` requests estimate data for all fields of the security `s`, from the Reuters Knowledge Direct connection `r` for an optional input list of parameter name/value pairs. For more information, see “Optional Input Arguments” on page 6-94.

## Examples

**1** Request EPS earnings estimates for company 'XYZ':

```
d = fetch(r,'XYZ','esttype','EPS')
d =
    ReutersResearchAPIResponse: [1x1 struct]
    GetEarningsEstimatesResponse: [1x1 struct]
        Status: [1x1 struct]
        Name: [1x1 struct]
        CoId: [1x1 struct]
    Security: [1x1 struct]
    Exchange: [1x1 struct]
```

```

Country: [1x1 struct]
  SecId: [1x1 struct]
MarketDataItem: [1x1 struct]
  Sector: [1x1 struct]
  Industry: [1x1 struct]
  Primary: [1x1 struct]
  Currency: [1x1 struct]
CurFiscalPeriod: [1x1 struct]
  Annual: [1x1 struct]
  Interim: [1x1 struct]
  FYActual: [1x1 struct]
  FYPeriod: [1x1 struct]
  ActValue: [1x1 struct]

```

## 2 Display information about the company:

```

Company = [{'XYZ'} d.Name.value;...
  {'ID'}          d.CoId.value(1);...
  {'Primary'}    d.Primary.value{1};...
  {'Primary'}    d.Primary.value{2};...
  {'Primary'}    d.Primary.value{3};...
  {'Primary'}    d.Primary.value{4};...
  {'Primary'}    d.Primary.value{5};...
  {' ' '};...
  {'Sector'}     d.Sector.value{1};...
  {'Code'}       d.Sector.code{1};...
  {'Set'}        d.Sector.set{1};...
  {' ' '};...
  {'Industry'}  d.Industry.value{1};...
  {'Code'}       d.Industry.code{1};...
  {'Set'}        d.Industry.set{1};...
  {' ' '};...
  {'Currency'}  d.Currency.value{1};...
  {'Code'}       d.Currency.code{1};...
  {'Set'}        d.Currency.set{1};...
  {'Type'}      d.Currency.type{1};...
  {' ' '};...

```

```
    {'Fiscal Year'} d.CurFiscalPeriod.fYear{1};...
    {'End Month'}   d.CurFiscalPeriod.fyem{1};...
    {'Period Type'} d.CurFiscalPeriod.periodType{1};...
  ]
Company =
  'XYZ'           'XYZ Inc.'
  'ID'            'US_33584'
  'Primary'       'PRX'
  'Primary'       'Mean'
  'Primary'       'EPS'
  'Primary'       'Q'
  'Primary'       [1x278 char]
                ''
  'Sector'        'Technology'
  'Code'          '57'
  'Set'           'RBSS2004'
                ''
  'Industry'      [1x28 char]
  'Code'          '57211'
  'Set'           'RBSS2004'
                ''
  'Currency'      'U.S. Dollars'
  'Code'          'USD'
  'Set'           'ISO'
  'Type'          'CONSENSUS'
                ''
  'Fiscal Year'   '2008'
  'End Month'     '12'
  'Period Type'   'Q'
```

### 3 Display security data (partial data appears here due to space constraints):

```
SecurityInfo = [...
  {'Code' 'Country' '' '' 'Sec IDs' '' ''}};...
  {' 'Set' 'Code' 'XYZ' 'Set' 'Type' 'ID'}};...
  d.Security.code d.Country.set d.Country.code d.Country.value ...
```

```
{' ' ' '};
cell(6,4) d.SecId.set d.SecId.type d.SecId.value;...
]
SecurityInfo =
  'Sec IDs'          ''          ''
  'Set'             'Type'       'ID'
  ''                ''          ''
  'LOCAL'          'RIC'         'XYZ.0'
  'LOCAL'          'Display RIC' 'XYZ.0Q'
  'LOCAL'          'TICKER'      'XYZ'
  'LOCAL'          'ISIN'         'US33584P5089'
  'LOCAL'          'SEDOL'       'A0208X3'
  'LOCAL'          'CUSIP'       '33584P508'
```

#### 4 Display market data:

```
MarketData = [{'Code' 'Type' 'Unit' 'Updated' 'Data'}];...
d.MarketDataItem.currCode d.MarketDataItem.type...
d.MarketDataItem.unit d.MarketDataItem.updated ...
d.MarketDataItem.value]
MarketData =
  'Code' 'Type'      'Unit' 'Updated'          'Data'
  'USD'  'CLPRICE'  'U'   '2008-01-18T00:00:00' '600.250000'
  'USD'  'SHARESOUT' 'U'   ''                  [] '312840485'
  'USD'  'MARKETCAP' 'M'   ''                  [] '187782.5011'
  'USD'  '52WKHIGH' 'U'   '2007-11-07T00:00:00' '747.2400'
  'USD'  '52WKLOW'  'U'   '2007-03-05T00:00:00' '437.0000'
```

#### 5 Display EPS data:

```
ActualsData = [{'Year' 'Type' 'End Year' 'End Month' ...
'Updated' 'Value'}]; ...
d.FYPeriod.fYear d.FYPeriod.periodType d.FYPeriod.endCalYear ...
d.FYPeriod.endMonth d.ActValue.updated d.ActValue.value]
ActualsData =
  'Year' 'Type' 'End Year' 'End Month' 'Updated'          'Value'
  '2007' 'Q'   '2007'     '9'         '2007-10-18T20:03:53' '3.9100'
```

|        |     |        |      |                       |           |
|--------|-----|--------|------|-----------------------|-----------|
| '2007' | 'Q' | '2007' | '6'  | '2007-07-19T20:07:43' | '3.5600'  |
| '2007' | 'Q' | '2007' | '3'  | '2007-04-19T20:06:47' | '3.6800'  |
| '2006' | 'A' | '2006' | '12' | '2007-01-31T21:06:56' | '10.5900' |
| '2006' | 'Q' | '2006' | '12' | '2007-01-31T21:05:45' | '3.1800'  |
| '2006' | 'Q' | '2006' | '9'  | '2006-10-19T20:04:12' | '2.6200'  |
| '2006' | 'Q' | '2006' | '6'  | '2006-07-20T20:05:12' | '2.4900'  |
| '2006' | 'Q' | '2006' | '3'  | '2006-04-20T20:06:09' | '2.2900'  |
| '2005' | 'A' | '2005' | '12' | '2006-02-01T19:09:12' | '5.7000'  |
| '2005' | 'Q' | '2005' | '12' | '2006-02-01T19:09:12' | '1.5400'  |
| '2005' | 'Q' | '2005' | '9'  | '2005-10-20T20:06:53' | '1.5100'  |
| '2005' | 'Q' | '2005' | '6'  | '2005-07-21T20:08:38' | '1.3600'  |
| '2005' | 'Q' | '2005' | '3'  | '2005-04-21T20:23:25' | '1.2900'  |
| '2004' | 'A' | '2004' | '12' | '2005-02-02T17:45:56' | '2.7500'  |
| '2004' | 'Q' | '2004' | '12' | '2005-02-02T17:45:56' | '0.9200'  |
| '2004' | 'Q' | '2004' | '9'  | '2004-10-22T12:13:38' | '0.7000'  |

## See Also

rkd



**Purpose** Get properties of Reuters Knowledge Data connection

**Syntax** `v = get(c, 'propertyname')`  
`v = get(c)`

**Arguments**

|                             |   |
|-----------------------------|---|
| <code>c</code>              | Reuters Knowledge Data connection object created with the <code>rkd</code> function.  |
| <code>'propertyname'</code> | A string or cell array of strings containing names of the connection object's properties. <code>user</code> is a valid property name. |

**Description**

`v = get(c, 'propertyname')` returns the values of the specified properties of the Reuters Knowledge Data connection object.

`v = get(c)` returns a structure where each field name is the name of a property of `c`, and each field contains the name of the property.

**See Also** `rkd`, `rkd.close`, `rkd.fetch`

# rkd.isconnection

---

|                    |   |
|--------------------|---|
| <b>Purpose</b>     | Verify whether Reuters Knowledge Data connections are valid   |
| <b>Syntax</b>      | <code>x = isconnection(c)</code>  |
| <b>Arguments</b>   | <code>c</code> Reuters Knowledge Data connection object created with the <code>rkd</code> function.   |
| <b>Description</b> | <code>x = isconnection(c)</code> returns <code>x = 1</code> if the connection is valid. Otherwise <code>x = 0</code> .  |
| <b>Examples</b>    | Establish a Reuters Knowledge Data connection:<br><pre>c = rkd</pre> Verify that the connection, <code>c</code> , is valid:<br><pre>x = isconnection(c) x = 1</pre> |
| <b>See Also</b>    | <code>rkd</code> , <code>rkd.close</code> , <code>rkd.fetch</code> , <code>rkd.get</code>   |

**Purpose** Retrieve data from Reuters Newscope sentiment archive file

**Syntax**

```
x = rnseloder(file)
x = rnseloder(file, 'date', {DATE1})
x = rnseloder(file, 'date', {DATE1, DATE2})
x = rnseloder(file, 'security', {SECNAME})
x = rnseloder(file, 'start', STARTREC)
x = rnseloder(file, 'records', NUMRECORDS)
```

**Arguments** Specify the following arguments as name-value pairs. You can specify any combination of name-value pairs in a single call to `rnseloder`.

|                         |  |
|-------------------------|--|
| <code>file</code>       | Reuters Newscope sentiment archive file from which to retrieve data.   |
| <code>'date'</code>     | Use this argument with <code>{DATE1, DATE2}</code> to retrieve data between and including the specified dates. Specify the dates as numbers or strings.  |
| <code>'security'</code> | Use this argument to retrieve data for <code>SECNAME</code> , where <code>SECNAME</code> is a cell array containing a list of security identifiers for which to retrieve data.   |
| <code>'start'</code>    | Use this argument to retrieve data beginning with the record <code>STARTREC</code> , where <code>STARTREC</code> is the record at which <code>rnseloder</code> begins to retrieve data. Specify <code>STARTREC</code> as a number. |
| <code>'records'</code>  | Use this argument to retrieve <code>NUMRECORDS</code> number of records.   |

**Description**

- `x = rnseloder(file)` retrieves data from the Reuters Newscope sentiment archive file `file`, and stores it in the structure `x`.
- `x = rnseloder(file, 'date', {DATE1})` retrieves data from `file` with date stamps of value `DATE1`.

# rnseloder

---

- `x = rnseloder(file, 'date', {DATE1, DATE2})` retrieves data from `file` with date stamps between `DATE1` and `DATE2`.
- `x = rnseloder(file, 'security', {SECNAME})` retrieves data from `file` for the securities specified by `SECNAME`.
- `x = rnseloder(file, 'start', STARTREC)` retrieves data from `file` beginning with the record specified by `STARTREC`.
- `x = rnseloder(file, 'records', NUMRECORDS)` retrieves `NUMRECORDS` number of records from `file`.

## Examples

- Retrieve data from the file `file.csv` with date stamps of `02/02/2007`:

```
x = rnseloder('file.csv', 'date', {'02/02/2007'})
```

- Retrieve data from `file.csv` between and including `02/02/2007` and `02/03/2007`:

```
x = rnseloder('file.csv', 'date', {'02/02/2007', '02/03/2007'})
```

- Retrieve data from `file.csv` for the security `XYZ.0`:

```
x = rnseloder('file.csv', 'security', {'XYZ.0'})
```

- Retrieve the first 10000 records from `file.csv`:

```
x = rnseloder('file.csv', 'records', 10000)
```

- Retrieve data from `file.csv`, starting at record 100000:

```
x = rnseloder('file.csv', 'start', 100000)
```

- Retrieve up to 100000 records from `file.csv`, for the securities `ABC.N` and `XYZ.O`, with date stamps between and including the dates `02/02/2007` and `02/03/2007`:

```
x = rnseloder('file.csv', 'records', 100000, ...  
             'date', {'02/02/2007', '02/03/2007'}, ...  
             'security', {'ABC.N', 'XYZ.O'})
```

**See Also**      `reuters`, `rdthloader`

**Purpose** Connect to Yahoo! data servers

**Syntax** `Connect = yahoo`  
`Connect = yahoo('URL', 'IPAddress', PortNumber)`

**Arguments**

|            |  |
|------------|--|
| URL        | Must be <code>http://quote.yahoo.com</code> .                          |
| IPAddress  | A MATLAB string containing the IP address of the proxy server machine. |
| PortNumber | Port number on proxy server.   |

**Description** `Connect = yahoo` verifies that the URL `http://quote.yahoo.com` is accessible and creates a connection handle.

`Connect = yahoo('URL', 'IPAddress', PortNumber)` connects to Yahoo! through a proxy server using the IP address and port number provided. This form of the `yahoo` function may be required when connecting to Yahoo! from behind an internal firewall.

**Examples** Connect to the Yahoo! data server:

```
Connect = yahoo
Connect =
    url: 'http://quote.yahoo.com'
```

Connect to the Yahoo! data server, providing an IP address and port number on a proxy server:

```
Connect = yahoo('http://quote.yahoo.com', '111.222.33.444', 5678)
Connect =
    url: 'http://quote.yahoo.com'
    ip: '111.222.33.444'
    port: 5678
```

**See Also** `yahoo.close`, `yahoo.fetch`, `yahoo.get`, `yahoo.isconnection`

|                    |   |         |  |
|--------------------|---|---------|--|
| <b>Purpose</b>     | Close connections to Yahoo! data servers  |         |  |
| <b>Syntax</b>      | <code>close(Connect)</code>   |         |  |
| <b>Arguments</b>   | <table><tr><td>Connect</td><td>Yahoo! connection object created with the <code>yahoo</code> function.</td></tr></table> | Connect | Yahoo! connection object created with the <code>yahoo</code> function. |
| Connect            | Yahoo! connection object created with the <code>yahoo</code> function.  |         |  |
| <b>Description</b> | <code>close(Connect)</code> closes the connection to the Yahoo! data server.  |         |  |
| <b>See Also</b>    | <code>yahoo</code>  |         |  |

# yahoo.fetch

---

**Purpose** Request data from Yahoo! data servers

**Syntax**

```
data = fetch(Connect, 'Security')
data = fetch(Connect, 'Security', 'Fields')
data = fetch(Connect, 'Security', 'Date')
data = fetch(Connect, 'Security', 'Fields', 'Date')
data = fetch(Connect, 'Security', 'FromDate', 'ToDate')
data = fetch(Connect, 'Security', 'Fields', 'FromDate',
            'ToDate')
data = fetch(Connect, 'Security', 'FromDate', 'ToDate',
            'Period')
```

**Arguments**

|          |   |
|----------|---|
| Connect  | Yahoo! connection object created with the yahoo function.   |
| Security | A MATLAB string or cell array of strings containing the name of a security in a format recognizable by the Yahoo! server. |

---

**Note** Retrieving historical data for multiple securities at one time is not supported for Yahoo. You can fetch historical data for only a single security at a time.

---



**Fields**

A MATLAB string or cell array of strings indicating the data fields for which to retrieve data. A partial list of supported values for current market data are:

- 'Symbol'
- 'Last'
- 'Date'
- 'Time'
- 'Change'
- 'Open'
- 'High'
- 'Low'
- 'Volume'

A partial list of supported values for historical data are:

- 'Close'
- 'Date'
- 'High'
- 'Low'
- 'Open'
- 'Volume'
- 'Adj. Close\*'

For a complete list of supported values for market and historical data, see `yhfields.mat`.

# yahoo.fetch

---

|          |  |
|----------|--|
| Date     | Date string or serial date number indicating date for the requested data. If you enter today's date, fetch returns yesterday's data. |
| FromDate | Beginning date for historical data.  |

---

**Note** You can specify dates in any of the formats supported by `datestr` and `datenum` that show a year, month, and day.

---

|        |   |
|--------|---|
| ToDate | End date for historical data.   |
| Period | Period within date range. Period values are: <ul style="list-style-type: none"><li>• 'd': daily</li><li>• 'w': weekly</li><li>• 'm': monthly</li><li>• 'v': dividends</li></ul> |

## Description

`data = fetch(Connect, 'Security')` returns data for all fields from Yahoo's Web site for the indicated security.

---

**Note** This function does not support retrieving multiple securities at once. You must fetch a single security at a time.

---

`data = fetch(Connect, 'Security', 'Fields')` returns data for the specified fields.

`data = fetch(Connect, 'Security', 'Date')` returns all security data for the requested date.

`data = fetch(Connect, 'Security', 'Fields', 'Date')` returns security data for the specified fields on the requested date.

```
data = fetch(Connect, 'Security', 'FromDate', 'ToDate')
returns security data for the date range FromDate to ToDate.
```

```
data = fetch(Connect, 'Security', 'Fields', 'FromDate',
'ToDate') returns security data for the specified fields for the date
range FromDate to ToDate.
```

```
data = fetch(Connect, 'Security', 'FromDate', 'ToDate',
'Period') returns security data for the date range FromDate to ToDate
with the indicated period.
```

## Examples

### Retrieving Last Prices for a Set of Equities

Connect to the Yahoo! data server to obtain the last prices for a set of equities:

```
y = yahoo;
FastFood = fetch(y, {'ko', 'pep', 'mcd'}, 'Last')
FastFood =
    Last: [3x1 double]
FastFood.Last
ans =
    42.96
    45.71
    23.70
```

### Retrieving a Closing Price on a Specified Date

Obtain the closing price for Coca-Cola on April 6, 2000:

```
c = yahoo;
ClosePrice = fetch(c, 'ko', 'Close', 'Apr 6 00')
ClosePrice =
    730582.00    45.75
```

## See Also

yahoo.close, yahoo.get, yahoo.isconnection, yahoo

**Purpose** Retrieve properties of Yahoo! connection objects

**Syntax**  
`value = get(Connect, 'PropertyName')`  
`value = get(Connect)`

**Arguments**

|              |  |
|--------------|--|
| Connect      | Yahoo! connection object created with the yahoo function.  |
| PropertyName | (Optional) A MATLAB string or cell array of strings containing property names. Currently the only property name recognized is 'url'. |

**Description**

`value = get(Connect, 'PropertyName')` returns the value of the specified properties for the Yahoo! connection object.

`value = get(Connect)` returns a MATLAB structure where each field name is the name of a property of `Connect`. Each field contains the value of the property.

**Examples**

Connect to a Yahoo! data server:

```
c = yahoo
c =
    url: 'http://quote.yahoo.com'
```

Retrieve the URL of the connection:

```
get(c, 'url')
ans =
    url: 'http://quote.yahoo.com'
```

**See Also** `yahoo.close`, `yahoo.fetch`, `yahoo.isconnection`, `yahoo`

|                    |  |         |   |
|--------------------|--|---------|---|
| <b>Purpose</b>     | Verify whether connections to Yahoo! data servers are valid  |         |   |
| <b>Syntax</b>      | <code>x = isconnection(Connect)</code>   |         |   |
| <b>Arguments</b>   | <table><tr><td>Connect</td><td>Yahoo! connection object created with the yahoo function.</td></tr></table>   | Connect | Yahoo! connection object created with the yahoo function. |
| Connect            | Yahoo! connection object created with the yahoo function.  |         |   |
| <b>Description</b> | <code>x = isconnection(Connect)</code> returns <code>x = 1</code> if the connection is a valid Yahoo! connection, and <code>x = 0</code> otherwise.            |         |   |
| <b>Examples</b>    | <p>Connect to a Yahoo! data server:</p> <pre>c = yahoo</pre> <p>Verify that the connection, <code>c</code>, is valid:</p> <pre>x = isconnection(c) x = 1</pre> |         |   |
| <b>See Also</b>    | <code>yahoo.close</code> , <code>yahoo.fetch</code> , <code>yahoo.get</code> , <code>yahoo</code>  |         |   |



# Examples

---

Use this list to find examples in the documentation.

## **Communicating with Financial Data Servers**

“Connecting to the Bloomberg Data Server” on page 2-3

“Verifying Connections” on page 2-4

## **Retrieving Connection Properties**

“Retrieving Connection Properties” on page 2-5

“Example: Retrieving Bloomberg Connection Object Properties” on page 2-5

## **Retrieving Data**

“Retrieving Header Data” on page 3-3

“Retrieving Field Data” on page 3-6

“Retrieving Time Series Data” on page 3-7

“Retrieving Historical Data” on page 3-8

“Finding Ticker Symbols” on page 3-9



## B

- bloomberg 6-2
- Bloomberg®
  - connection handle 2-3
  - connection object 2-3
- bloomberg.pricevol 6-15
- bloomberg.showtrades 6-17
- bloomberg.stockticker 6-19

## C

- close
  - Bloomberg® 6-3
  - FactSet® 6-30
  - FRED® 6-38
  - Haver Analytics 6-45
  - Interactive Data Pricing and Reference
    - Data's RemotePlus™ 6-57
  - Kx Systems®, Inc. 6-64
  - Reuters® 6-86
  - rkd 6-93
  - Thomson® Datastream® 6-22
  - Yahoo!® 6-109
- connection handle 2-3
- connection object 2-3
- CUSIP number 6-4

## D

- data
  - default 3-3
  - field 3-6
  - header 3-3
  - historical 3-8
  - retrieving
    - using fetch function 3-2
  - time-series 3-7
- data servers
  - connecting to 2-2

- example using bloomberg function 2-3
  - using functions 2-2
- disconnecting from 2-7
- retrieving connection properties 2-5
- verifying connections 2-4

- data services
  - connection requirements 1-3
    - for FactSet® data server 1-4
    - for Reuters® data server 1-5
    - for Thomson® Datastream® data server 1-8
  - proxy information 1-4
  - software 1-3
  - data service providers 1-3
- Datafeed dialog box 4-3
  - Data tab 4-5
  - overriding data using 4-8
  - using Connection tab to connect to data servers 4-4
- Datafeed Dialog Box
  - starting 4-3
- Datafeed Toolbox™ software
  - definition 1-2
- datastream 6-21
- default data 3-3
- dftool 4-3
- disconnecting from data servers 2-7

## E

- exec
  - Kx Systems®, Inc. 6-68

## F

- factset 6-29
- Federal Reserve Economic Data (FRED®) 6-37

- fetch
  - Bloomberg® 6-4
  - FactSet® 6-31
  - FRED® 6-39
  - Haver Analytics 6-46
  - Interactive Data Pricing and Reference
    - Data's RemotePlus™ 6-58
  - Kx Systems®, Inc. 6-65
  - Reuters® 6-87
  - rkd 6-94
  - Thomson® Datastream® 6-23
  - Yahoo!® 6-110
- field data 3-6
- field names 3-6
- Flag values 3-4
- fred 6-37
- functions
  - Bloomberg®
    - bloomberg 6-2
    - close 6-3
    - fetch 6-4
    - get 6-12
    - isconnection 6-14
    - pricevol 6-15
    - showtrades 6-17
    - stockticker 6-19
  - FactSet®
    - close 6-30
    - factset 6-29
    - fetch 6-31
    - get 6-34
    - isconnection 6-36
  - FRED®
    - close 6-38
    - fetch 6-39
    - fred 6-37
    - get 6-41
    - isconnection 6-42
  - Haver Analytics
    - aggregation 6-44
    - close 6-45
    - fetch 6-46
    - get 6-48
    - haver 6-43
    - havertool 6-54
    - info 6-49
    - isconnection 6-51
    - nextinfo 6-52
  - Interactive Data Pricing and Reference
    - Data's RemotePlus™
      - close 6-57
      - fetch 6-58
      - get 6-60
      - idc 6-56
      - isconnection 6-61
  - Kx Systems®, Inc.
    - close 6-64
    - exec 6-68
    - fetch 6-65
    - get 6-67
    - insert 6-69
    - isconnection 6-70
    - kx 6-62
    - tables 6-71
  - Reuters®
    - close 6-86
    - fetch 6-87
    - get 6-90
    - reuters 6-75
    - stop 6-91
  - Reuters® Datascope Tick History
    - rdthloader 6-72
  - rkd
    - close 6-93
    - fetch 6-94
    - get 6-103
    - isconnection 6-104
    - rkd 6-92
  - rnseloder 6-105
  - Thomson® Datastream®

- close 6-22
  - datastream 6-21
  - fetch 6-23
  - get 6-27
  - isconnection 6-28
  - Yahoo!<sup>®</sup>
    - close 6-109
    - fetch 6-110
    - get 6-114
    - isconnection 6-115
    - yahoo 6-108
- G**
- get
    - Bloomberg<sup>®</sup> 6-12
    - FactSet<sup>®</sup> 6-34
    - FRED<sup>®</sup> 6-41
    - Haver Analytics 6-48
    - Interactive Data Pricing and Reference
      - Data's RemotePlus<sup>™</sup> 6-60
    - Kx Systems<sup>®</sup>, Inc. 6-67
    - Reuters<sup>®</sup> 6-90
    - rkd 6-103
    - Thomson<sup>®</sup> Datastream<sup>®</sup> 6-27
    - Yahoo!<sup>®</sup> 6-114
  - GETDATA argument 3-6
  - graphical user interface 4-1
- H**
- haver 6-43 to 6-44
  - Haver Analytics 6-43 to 6-44
  - havertool 6-54
    - Haver Analytics 6-54
  - HEADER argument 3-4
  - header data 3-3
  - header fields 3-3
  - historical data 3-8
  - HISTORY argument 3-8
- I**
- idc 6-56
  - info
    - Haver Analytics 6-49
  - insert
    - Kx Systems<sup>®</sup>, Inc. 6-69
  - isconnection
    - Bloomberg<sup>®</sup> 6-14
    - FactSet<sup>®</sup> 6-36
    - FRED<sup>®</sup> 6-42
    - Haver Analytics 6-51
    - Interactive Data Pricing and Reference
      - Data's RemotePlus<sup>™</sup> 6-61
    - Kx Systems<sup>®</sup>, Inc. 6-70
    - rkd 6-104
    - Thomson<sup>®</sup> Datastream<sup>®</sup> 6-28
    - Yahoo!<sup>®</sup> 6-115
- K**
- kx 6-62
- L**
- LOOKUP argument 3-9
- M**
- markets 3-9
- N**
- nextinfo
    - Haver Analytics 6-52
- R**
- rdthloader 6-72
  - retrieving connection properties 2-5
  - retrieving data 3-2
  - reuters 6-75

Reuters

Reuters® Configuration Editor 1-5

Reuters® Datascope Tick History file. *See*  
rdthloader

Reuters® Knowledge Direct. *See* rkd

Reuters® Newscope. *See* rnseloder

rkd 6-92

rnseloder 6-105

**S**

Securities Lookup dialog box 4-6

stop

Reuters® 6-91

**T**

tables

Kx Systems®, Inc. 6-71

ticker symbols 3-9

time-series data 3-7

TIMESERIES argument 3-7

**V**

verifying connections 2-4

**Y**

yahoo 6-108