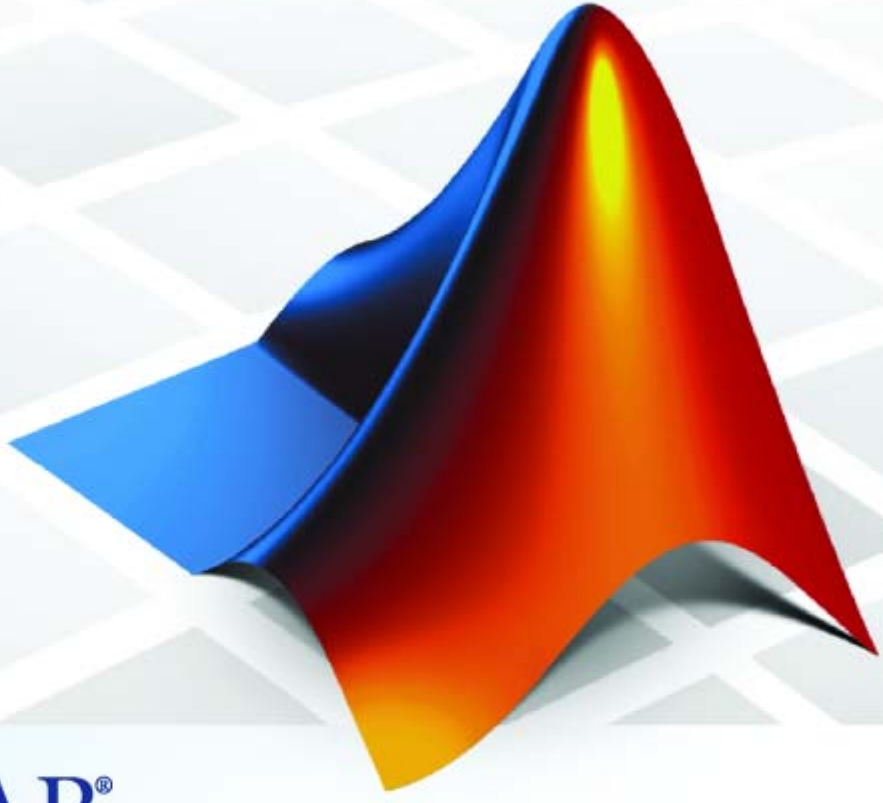
Datafeed Toolbox 3
User's Guide



MATLAB®

The MathWorks™
Accelerating the pace of engineering and science

How to Contact The MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`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

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)

Contents

1 Getting Started

2 Communicating with Financial Data Servers

3 Retrieving Data

4 The Datafeed Toolbox Graphical User Interface

5 Functions — By Category

Examples

A

Index

1

Getting Started

What Is Datafeed Toolbox?

Datafeed Toolbox for MATLAB® effectively turns your MATLAB workstation into a financial data acquisition terminal. Datafeed 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

Datafeed Toolbox supports connections to financial data servers that are provided by the following corporations:

- 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)
- Hyperfeed Technologies, Inc.® (http://www.hyperfeed.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 Corporation ®(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.

- The following data service providers require you to install proprietary software on your PC. For more information about how to obtain this software, contact your data server sales representative.
  - Bloomberg
  - Interactive Data Pricing and Reference Data

- Haver Analytics
- Hyperfeed
- Kx Systems
- Reuters

If the required client software is not properly licensed for your machine, you receive the following error message when you attempt to connect to a Bloomberg, Interactive Data Pricing and Reference Data, or Haver Analytics data server:

```
Invalid MEX-file
```

- The following data service providers may require you to specify a proxy host and proxy port:
  - 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.
- Thomson Datastream requires the following:
  - A license for Dataworks from Thomson.
  - To connect to the Thomson Datastream API via 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.
- FactSet requires you to obtain a license to use FactSet's FAST technology. For more information, see the FactSet Web site at http://www.factset.com.
- Reuters requires you to use the Reuters RFA Configuration Editor to configure your Reuters connections. To do this:

  a. Set your classpath from a DOS prompt as follows:

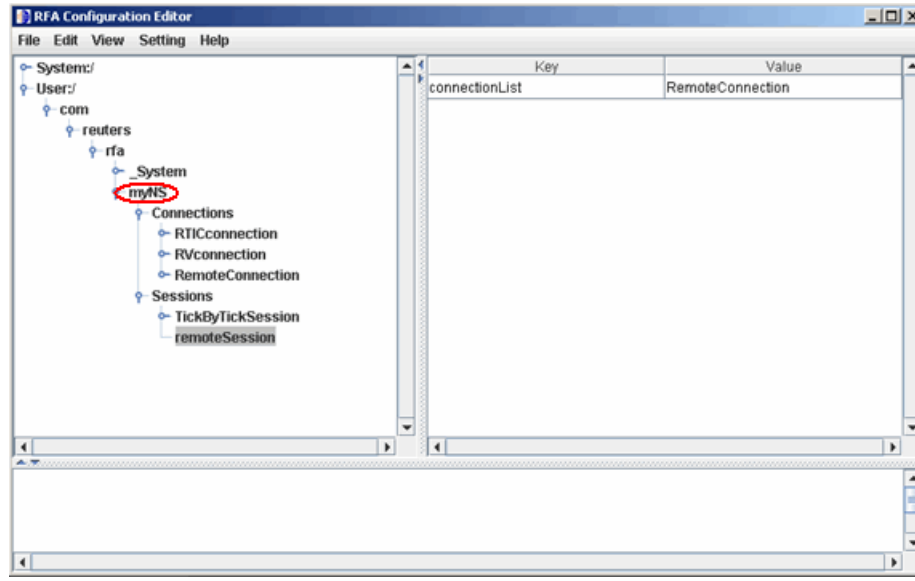
```
set CLASSPATH=%CLASSPATH%; ...
$MATLAB/toolbox/datafeed/datafeed/config_editor.jar
```

b Run the Configuration Editor:

```
java com.reuters.rfa.tools.config.editor.ConfigEditor
```

c Define a namespace, a connection, and a session associated with the connection.

In the following example, the session `remoteSession` with the namespace `MyNS` is added to the connection list for the connection `remoteConnection`.



d Disable DACS, the Reuters data entitlements system. To do so, add the following to your connection configuration:

```
dacs_CbeEnabled=false
Dacs_SbePubEnabled=false
Dacs_SbeSubEnabled=false
```

If you are running an SSL connection, also add to your connection configuration:

```
dacs_GenerateLocks=false
```

2

Communicating with Financial Data Servers

Communication Management

In this section...

Communicating with Data Servers

This section uses the Bloomberg financial data server as an example of how to use Datafeed Toolbox to establish communication with a financial data server and retrieve data. Communication with other supported data servers is accomplished with a virtually identical set of toolbox functions.

Core Functions

Datafeed Toolbox uses the following set of core functions to manage communication with each supported financial data server.

- To establish a connection to the appropriate data server, use:
  - `bloomberg`
  - `datastream`
  - `factset`
  - `fred`
  - `haver`
  - `hyperfeed`
  - `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 demonstrates how to use the `bloomberg` function to connect to the Bloomberg data server.

The syntax for the `bloomberg` function is:

```
Connect = bloomberg(PortNumber, 'IPAddress')
```

Note The `PortNumber` and `IPAddress` arguments are optional.

The IP address is entered as a MATLAB string. For example, the expression:

```
c = bloomberg(8194, '123.456.54.123')
```

Returns a Bloomberg connection object:

```
c =

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

The `connection` field within the object `c` contains the Bloomberg connection handle that is used to process future data requests.

If you want to accept the default port number and IP address provided when your Bloomberg software was installed, enter the following command in the MATLAB Command Window:

```
c = bloomberg
```

Verifying Connections

To verify that a data server connection is valid and open, use the `isconnection` function. For a connection object `c` previously created with one of the above connection functions, the 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

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 Properties

For the Bloomberg connection:

```
c = bloomberg(8194, '123.456.54.123')
```

The command:

```
p = get(c)
```

Returns the list of all valid connection properties and their values associated with the connection object `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 Bloomberg version for the connection object `c`, use the following command:

```
p = get(c,{'Port';'Version'})
```

Which returns:

```
p =
        port: 8194
     version: 1.8000
```

When returning a single property, for example, the connection handle, the function:

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

Returns:

```
p =
    84554360
```

For a single returned property the output is not 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.

3

Retrieving Data

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

The following example illustrates the use of the `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 (Bloomberg Default) Data

A header (default) data request to Bloomberg 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 from the Bloomberg connection, use `fetch` 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 data is requested.

Note Security names are case sensitive for Bloomberg `fetch`.

- The `'HEADER'` argument is entered literally.
- `'Flag'` denotes dates for which data can be retrieved. `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
```

Not all fields are applicable to `IBM US Equity`, the security on which we inquired.

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 numerous 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:

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

For example, use the `bloomberg` function to establish a connection `c1` to a Bloomberg data server:

```
c1 = bloomberg(8234, '123.457.78.999')
```

Then run the command:

```
d = fetch(c1,'IBM US Equity','GETDATA', {'Open';'Last_Price'})
```

MATLAB returns:

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

Retrieving Time Series Data

The `fetch` function 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` may 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')
```

The result looks as follows:

```
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, and 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 the command:

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

Data for the field is returned 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 uncertain what the symbol might be. 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, and `Market` indicates the type of security (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:

```
data = fetch(c1, 'New', 'LOOKUP', 'Govt')
```

Returns a list of possible values:

```
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'
```

4

The Datafeed Toolbox Graphical User Interface

Introduction

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

The Datafeed Toolbox 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 Corporation
  - 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. Use this dialog box with Bloomberg and Interactive Data Pricing and Reference Data connections to find the ticker symbol for a specific security when you know at least part of the name of the security.

  For more information on how to use this dialog box, see “Using the Securities Lookup Dialog Box” on page 4-9.

Using the Datafeed Dialog Box

In this section...

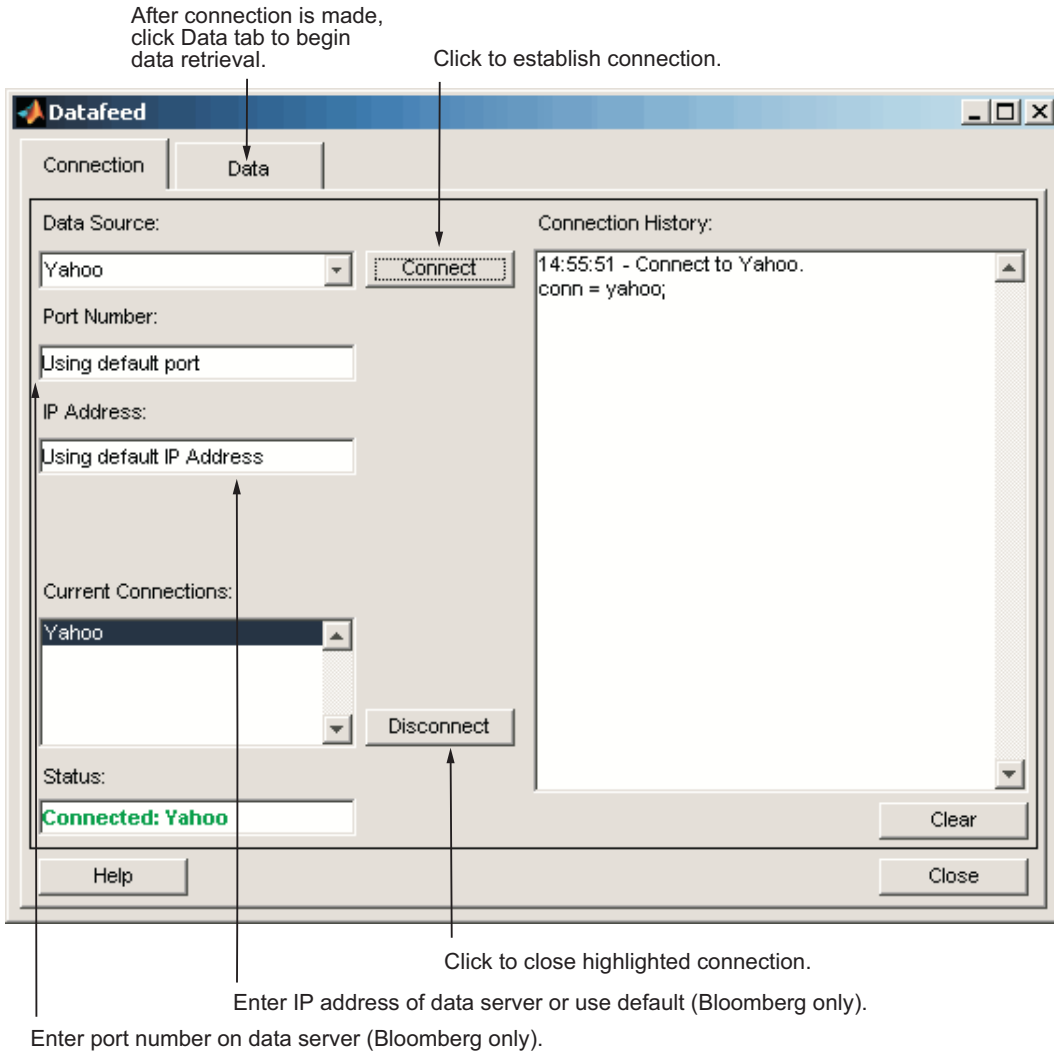
About the Datafeed Dialog Box

The Datafeed dialog box establishes the connection with the data server and manages data retrieval. Enter the command `dftool` to display the Datafeed dialog box on your screen. The Datafeed dialog box consists of two tabbed dialog boxes:

- The **Connection** tab establishes communication with a data server. For more information, see "Connecting to Data Servers" on page 4-3.
- 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 that you obtain. For more information, see "Setting Overrides" on page 4-7.

Connecting to Data Servers

The **Connection** tab establishes a connection to one or more data servers. For FactSet, Federal Reserve Economic Data, Yahoo!, and Interactive Data Pricing and Reference Data connections, choose the data server from the **Data Source** selection list and click the **Connect** button. For a Bloomberg connection, you can specify a specific IP address and port number on the Bloomberg server, or alternatively, just click the **Connect** button and accept the default values.



1 (Bloomberg only) Enter the port number on the data server in the **Port Number** box (or use the default).

2 (Bloomberg only) Enter the IP address of the data server in the **IP Address** box (or use the default).

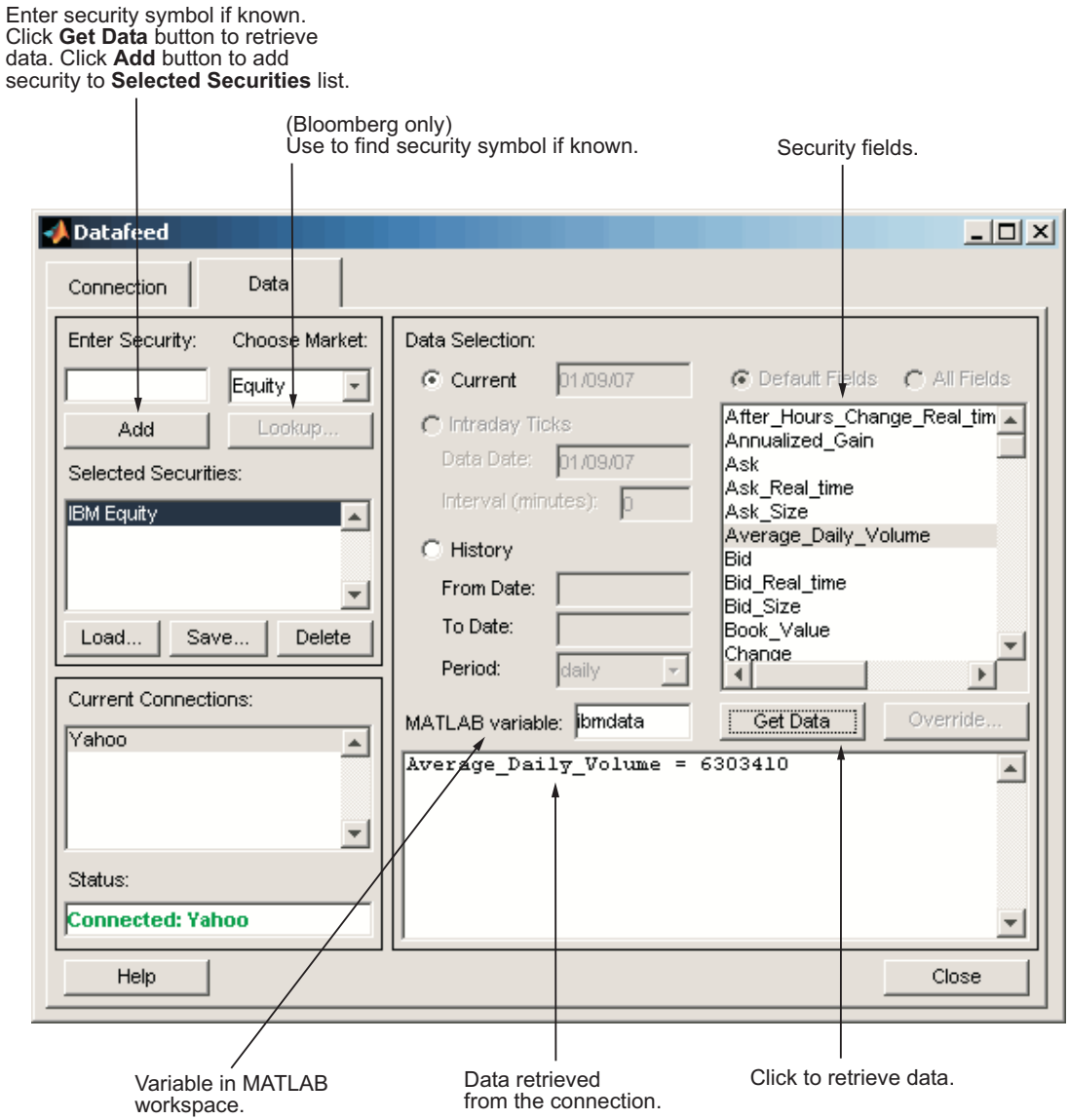
3 Click the **Connect** button to establish the connection.

4 When the `Connected` message appears in the **Status** box, click the **Data** tab to begin the process of retrieving data from the data server. For more information on the **Data** tab, see “Retrieving Data” on page 4-5.

5 Click the **Disconnect** button to terminate the session highlighted in the **Current Connections** box.

Retrieving Data

The **Data** tab manages the retrieval of data from the data server. It also allows you to set overrides on the data.



1 Enter the security symbol in the **Enter Security** box.

2 Indicate the type of data you are seeking in the **Data Selection** pane.

3 Indicate whether you want the default or full set of data in the **Fields** pane.

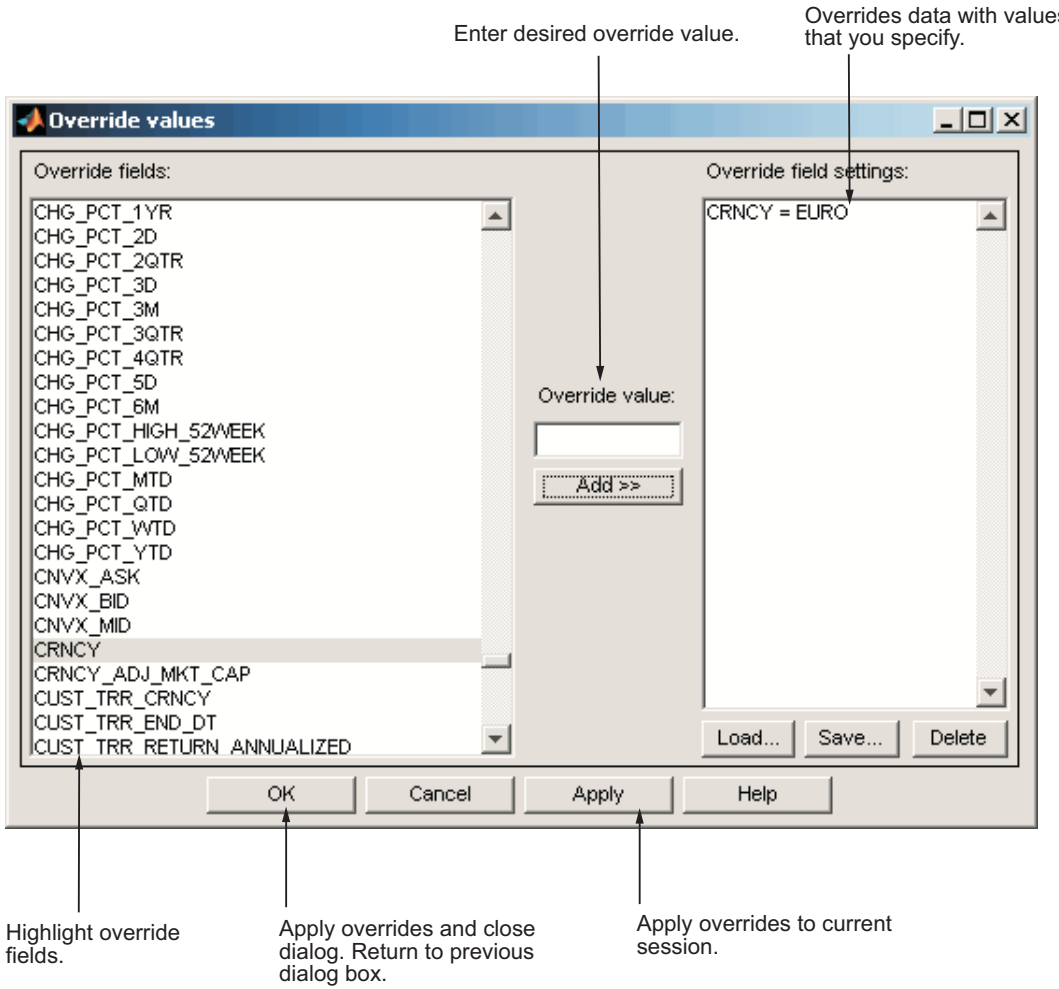
4 Click the **Get Data** button to retrieve data from the data server.

5 Click the **Override** button if you want to set overrides on the data you request from the data server.

Note If you do not know the symbol for a security, you can use the **Lookup** button to find it. For more information, see "Using the Securities Lookup Dialog Box" on page 4-9.

Setting Overrides

To set overrides on retrieved data, click the **Override** button. The Override values dialog box opens.

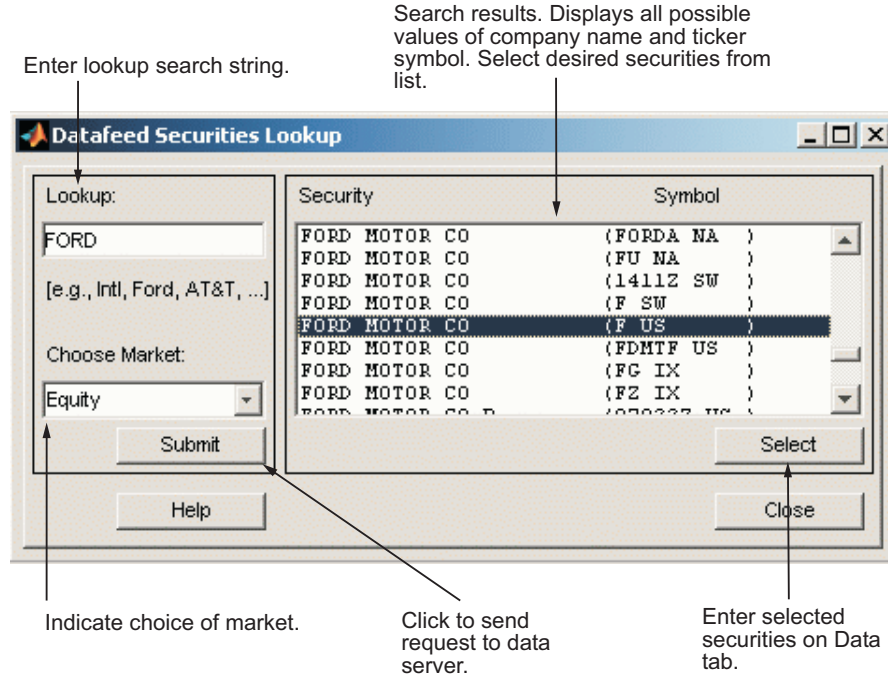

Enter desired override value.
Overrides data with values that you specify.
Override values
Override fields:
CHG_PCT_1YR
CHG_PCT_2D
CHG_PCT_2QTR
CHG_PCT_3D
CHG_PCT_3M
CHG_PCT_3QTR
CHG_PCT_4QTR
CHG_PCT_5D
CHG_PCT_6M
CHG_PCT_HIGH_52WEEK
CHG_PCT_LOW_52WEEK
CHG_PCT_MTD
CHG_PCT_QTD
CHG_PCT_WTD
CHG_PCT_YTD
CNVX_ASK
CNVX_BID
CNVX_MID
CRNCY
CRNCY_ADJ_MKT_CAP
CUST_TRR_CRNCY
CUST_TRR_END_DT
CUST_TRR_RETURN_ANNUALIZED
Override value:
Add >>
Override field settings:
CRNCY = EURO
Load...
Save...
Delete
OK
Cancel
Apply
Help
Highlight override fields.
Apply overrides and close dialog. Return to previous dialog box.
Apply overrides to current session.

Using the Securities Lookup Dialog Box

Click the **Lookup** button of the Datafeed dialog box **Data** tab to display the Securities Lookup dialog box. For more information about the **Data** tab, see “Retrieving Data” on page 4-5.

The Securities Lookup dialog box provides a means to obtain the ticker symbol for a particular security when you know part of the name. Enter the ticker symbol into the **Enter Security** field on the **Data** tab. It is essential to enter the ticker symbol as specified; otherwise, the data server may provide no data or provide data for some other security.

Alternatively, you can highlight one or more securities in the list and click **Select**. The selected securities are added to the **Selected Securities** list on the **Data** tab.



5

Functions — By Category

Bloomberg (p. 5-2)	Get Bloomberg financial data
Thomson Datastream (p. 5-22)	Get Thomson Datastream financial data
FactSet (p. 5-30)	Get FactSet financial data
Haver Analytics (p. 5-39)	Get Haver Analytics financial data
Hyperfeed (p. 5-52)	Get Hyperfeed financial data
Interactive Data Pricing and Reference Data (p. 5-61)	Get Interactive Data Pricing and Reference Data financial data
Federal Reserve Economic Data (p. 5-68)	Get Federal Reserve Economic Data financial data
Kx Systems (p. 5-76)	Get Kx kdb+ financial data
Reuters (p. 5-88)	Get Reuters current market and real-time data
Yahoo! (p. 5-97)	Get Yahoo! financial data

Bloomberg

bloomberg	Connect to Bloomberg
close	Close Bloomberg connection
fetch	Request data from Bloomberg
get	Bloomberg connection properties
isconnection	True if valid Bloomberg connection
pricevol	Price and volume (demonstration)
showtrades	Recent trade data (demonstration)
stockticker	Trades with volumes (demonstration)

Purpose

Connect to Bloomberg

Syntax

```
Connect = bloomberg(PortNumber, 'IPAddress')
Connect = bloomberg
```

Arguments

`PortNumber`	Port on machine to which you are connecting.
`IPAddress`	A MATLAB string containing the Internet address of the machine to which you are connecting.

Description

`Connect = bloomberg(PortNumber, 'IPAddress')` establishes a connection to a Bloomberg data server using the port number, `PortNumber`, and the Internet address, `'IPAddress'`.

`Connect = bloomberg` establishes a connection to a Bloomberg data server using port number `8194` and the default Internet address provided when the Bloomberg software was installed on your machine.

Examples

Connect to the Bloomberg server on port `8194` of the machine with Internet address `111.222.33.444`:

```
c = bloomberg(8194,'111.222.33.444')
```

See Also

`close`, `fetch`, `get`, `isconnection` (Bloomberg functions)

close

Purpose

Close Bloomberg connection

Syntax

```
close(Connect)
```

Arguments

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

Description

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

Examples

Establish a Bloomberg connection `c`:

```
c = bloomberg(8194,'111.222.33.444')
```

Close this connection:

```
close(c)
```

See Also

`bloomberg` (Bloomberg functions)

Purpose

Request data from Bloomberg

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 in a format recognizable by the Bloomberg server. You can substitute a `CUSIP number` for a security name as needed.

Note The `Security` argument is case sensitive.

Note The `Security` argument may be a cell array of strings containing a list of securities.

`Flag`	A MATLAB string indicating the dates from which data is to be retrieved. Possible values are: • `DEFAULT`: Data from most recent bid, ask, or trade. If a `Flag` value is not specified, `'DEFAULT'` is assumed. • `TODAY`: Today's data only. • `ENHANCED`: Data from most recent date of each individual field.
`Currency`	(Optional) Currency in which historical returns are provided. Valid currencies are listed in the file `@bloomberg/bbfields.mat`. Default = `[]`.
`Ident`	(Optional) Security type identifier. Valid security type identifiers are listed in the file `@bloomberg/bbfields.mat`. Default = `[]`.
`Fields`	A MATLAB string or cell array of strings specifying specific fields for which data is requested. Valid field names are listed in the file `@bloomberg/bbfields.mat`. The variable `bbfieldnames` 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 or serial date number indicating date for the time series. Specify `now` for today's time-series data.
`Minutes`	(Optional) Tick interval in minutes.
`TickField`	(Optional) The field can be specified as a string or numeric value (e.g., `TickField = 'Trade'` or `TickField = 1` return data for ticks of type `Trade`. Use the command `dftool('ticktypes')` to return the list of intraday tick fields.
`FromDate`	Beginning date for historical data.

Note Dates can be specified 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
- `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 (e.g., `'n'`), the unspecified values are replaced by defaults.

Note If `Period` is not specified, default values are used.

`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 the Bloomberg connection.

`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. This is the equivalent of `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.

`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, an empty matrix is returned.

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

- Time
- Open
- High
- Low
- Value of last tick
- Volume total value of ticks
- Total value of ticks for the time range
- Number of ticks

Columns 7 and 8 are provided 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 further specify the date range by setting the time period with the optional `Period` argument. You can further specify the data to be returned by appending the `Currency` or `Ident` argument.

`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

Returning Header Data

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

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

Opening and Closing Prices

Retrieve the opening and closing prices:

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

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'})
```

Time Series

Retrieve today's time series:

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

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')
```

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')
```

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`, `close`, `get`, `isconnection` (Bloomberg functions)

Purpose

Bloomberg connection properties

Syntax

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

Arguments

`Connect`	Bloomberg connection object created with the `bloomberg` function.
`PropertyName`	(Optional) A MATLAB string or cell array of strings containing property names. Property names are: • `'Connection'` • `'IPAddress'` • `'Port'` • `'Socket'` • `'Version'`

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 Bloomberg connection, `c`:

```
c = bloomberg(8194,'111.222.33.444')
```

The command:

```
p = get(c, {'Port', 'IPAddress'})
```

Returns:

```
p =
     port: 8194
     ipaddress: 111.222.33.444
```

See Also

`bloomberg`, `close`, `fetch`, `isconnection` (Bloomberg functions)

Purpose

True if valid Bloomberg connection

Syntax

```
x = isconnection(Connect)
```

Arguments

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

Description

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

Examples

Establish a Bloomberg connection, `c`:

```
c = bloomberg(8194,'111.222.33.444')
```

Verify that `c` is a valid Bloomberg connection:

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

See Also

`bloomberg`, `close`, `fetch`, `get` (Bloomberg functions)

pricevol

Purpose

Price and volume (demonstration)

Syntax

```
pricevol(InputList)
```

Arguments

`InputList`	Fields for which real-time data is sought.

Description

`pricevol(InputList)` demonstrates the Bloomberg real-time data import functionality. `InputList` is an input list of the elements. These are described in the following table.

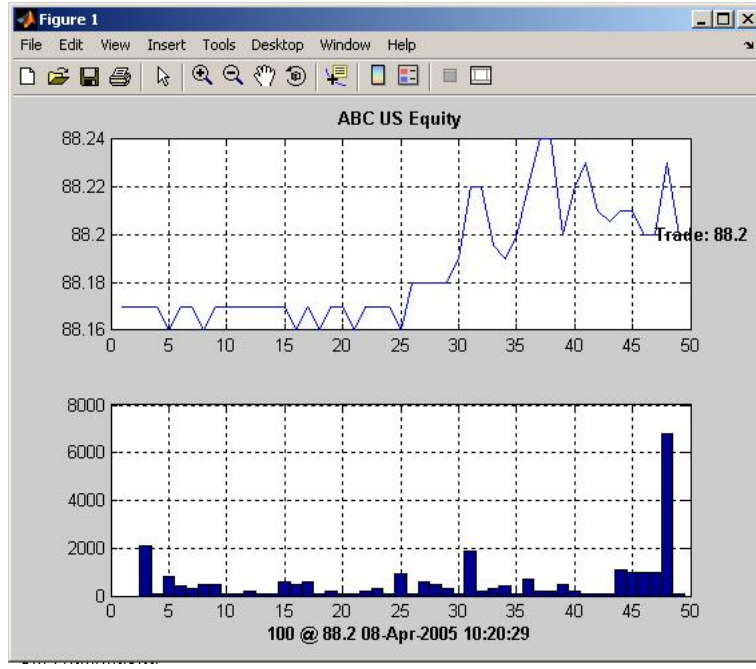
`InputList(1) = COM.Bloomberg.Data.1`	Bloomberg handle
`InputList(2) = 1`	Event ID
`InputList(3) = ('Security')`	Security string
`InputList(4) = 1`	Cookie
`InputList(5) = 2`	Field number ID
`InputList(6) = {[43.58]}`	Return data for the given tick
`InputList(7) = 0`	Status
`InputList(8)`	Structure containing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

`showtrades`, `stockticker` (Bloomberg functions)

showtrades

Purpose

Recent trade data (demonstration)

Syntax

```
showtrades(InputList)
```

Arguments

`InputList`	Fields for which real-time data is sought.

Description

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

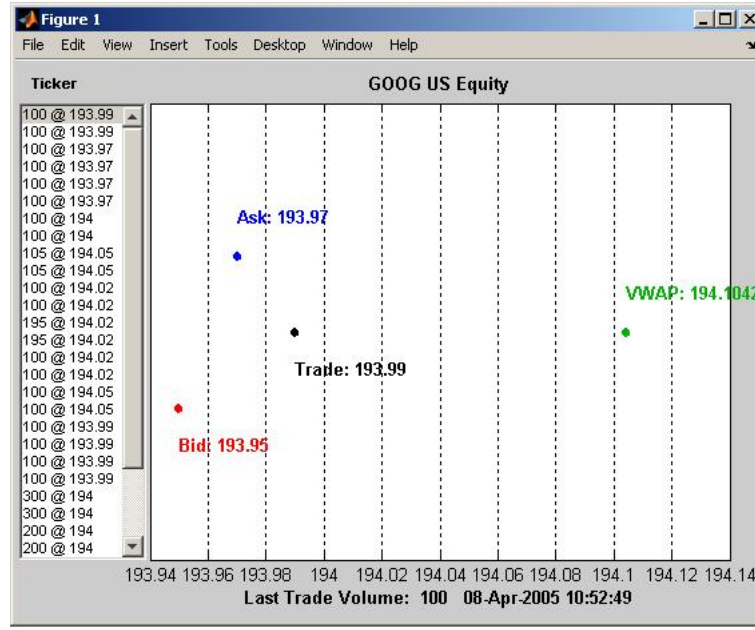
`InputList(1) = COM.Bloomberg.Data.1`	Bloomberg handle
`InputList(2) = 1`	Event ID
`InputList(3) = ('Security')`	Security string
`InputList(4) = 1`	Cookie
`InputList(5) = 2`	Field number ID
`InputList(6) = {[43.58]}`	Return data for the given tick
`InputList(7) = 0`	Status
`InputList(8)`	Structure containing 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`, `Bid`, `Ask`, and `VWAP` (volume-weighted adjusted price), and a list of the most recent trades with volumes:

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



See Also

`pricevol`, `stockticker` (Bloomberg functions)

stockticker

Purpose

Trades with volumes (demonstration)

Syntax

```
stockticker(InputList)
```

Arguments

`InputList`	Fields for which real-time data is sought.

Description

`stockticker(InputList)` demonstrates the Bloomberg real-time data import functionality. `InputList` is an input list of the elements. These are 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
```

```
** 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

`pricevol`, `showtrades` (Bloomberg functions)

Thomson Datastream

close	Close Thomson Datastream connection
datastream	Thomson Datastream API connection
fetch	Request data from Thomson Datastream
get	Thomson Datastream connection properties
isconnection	True if valid Thomson Datastream connection

Purpose

Close Thomson Datastream connection

Syntax

```
close(Connect)
```

Arguments

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

Description

`close(Connect)` closes the connection to the Thomson Datastream data server.

See Also

`datastream` (Thomson Datastream functions)

datastream

Purpose

Thomson Datastream API connection

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 you need 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 the Thomson Datastream content.

Examples

Establish a connection to the Thomson Datastream API, from which you can access Thomson Datastream content:

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

See Also

`close`, `fetch`, `get`, `isconnection` (Thomson Datastream functions)

Purpose

Request data from Thomson Datastream

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 `datastream` function.
`'Security'`	MATLAB string containing the name of a security or cell array of strings containing names of multiple securities, 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 data is to be retrieved.
`'Date'`	(Optional) MATLAB string indicating a specific calendar date for which data is requested.
`'FromDate'`	(Optional) Start date for historical data.

`'ToDate'`	(Optional) End date for historical data. If `'ToDate'` is provided, `'FromDate'` cannot be an empty value. **Note** Dates can be specified 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 the data is reported.

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')` additionally specifies the currency in which the data is reported.

Examples

The following examples use the Datastream `fetch` command to obtain data from Thomson Datastream.

Note The Datastream interface returns all data as strings. For example, when requesting `Price` data, it is returned in MATLAB as a cell array of strings within the structure. There is no way to determine the data type from the Datastream interface.

`data = fetch(Connect, 'ICI')` and `data = fetch(Connect, 'ICI', 'P')` both return the trailing one-year price time series for the specified instrument. (`'P'` is the default value for the `'Field'` argument.)

`data = fetch(Connect, 'ICI', {'P', 'PO'}, '08/01/2005')` returns the closing and opening prices for the specified instrument on the given date.

`data = fetch(Connect, {'ICI', 'IBM'}, {'P', 'PO'}, '8/01/2003', '8/01/2005, 'M')` returns the monthly closing and opening prices for the speciied securities during the specified date range.

See Also

`close`, `datastream`, `get`, `isconnection` (Thomson Datastream functions)

Purpose

Thomson Datastream connection properties

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`, and each field contains the value of that property.

See Also

`close`, `datastream`, `fetch`, `isconnection` (Thomson Datastream functions)

Purpose

True if valid Thomson Datastream connection

Syntax

```
x = isconnection(Connect)
```

Arguments

`Connect`	Thomson Datastream connection object created with the `datastream` 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 Thomson Datastream connection:

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

See Also

`close`, `datastream`, `fetch`, `get` (Thomson Datastream functions)

FactSet

close	Close FactSet connection
factset	Connect to FactSet
fetch	Request data from FactSet
get	FactSet connection properties
isconnection	True if valid FactSet connection

Purpose

Close FactSet connection

Syntax

```
close(Connect)
```

Arguments

`Connect`	FactSet connection object created with the `factset` function.

Description

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

See Also

`factset` (FactSet functions)

factset

Purpose

Connect to FactSet

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

`close`, `fetch`, `get`, `isconnection` (FactSet functions)

Purpose

Request data from FactSet

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 data is to be retrieved.
`Date`	Date string or serial date number indicating date for the requested data. If today's date is entered, yesterday's data is returned.
`FromDate`	Beginning date for date range.

Note Dates can be specified in any of the formats supported by `datestr` and `datenum` that display a year, month, and day.

`ToDate`	End date for date range.
`Period`	Period within date range. `Period` values are: • `'d'`: daily values • `'b'`: business day daily values • `'m'`: monthly values • `'mb'`: beginning monthly values • `'me'`: ending monthly values • `'q'`: quarterly values • `'qb'`: beginning quarterly values • `'qe'`: ending quarterly values • `'y'`: annual values • `'yb'`: beginning annual values • `'ye'`: ending annual values

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

1. Obtain the names of the available formula libraries:

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

2. Obtain the valid field names for the `FactSetSecurityCalcs` library:

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

3. Obtain closing price of a given security:

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

4. Obtain the closing price for the given dates for a given security using the default period of the data:

```
D = fetch(C,'ABC','price','8/01/99','8/10/99')
```

5. Obtain the monthly closing price for the given dates for a given security:

```
D = fetch(C,'ABC','price','8/01/99','8/10/99','m')
```

See Also

`close`, `factset`, `isconnection` (FactSet functions)

Purpose

FactSet connection properties

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: • `user` • `serial` • `password` • `cid`

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 FactSet:

```
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'

get(Connect, 'user')

ans =

Fast_User
```

See Also

`close`, `fetch`, `factset`, `isconnection` (FactSet functions)

isconnection

Purpose

True if valid FactSet connection

Syntax

```
x = isconnection(Connect)
```

Arguments

`Connect`	FactSet connection object created with the `factset` function.

Description

`x = isconnection(Connect)` returns `x = 1` if the connection is a valid FactSet connection, and `x = 0` if it is not.

Examples

Establish a FactSet connection:

```
c = factset
```

Verify that `c` is a valid FactSet connection:

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

```
x = 1
```

See Also

`close`, `fetch`, `factset`, `get` (FactSet functions)

Haver Analytics

close	Close Haver Analytics database
fetch	Request data from Haver Analytics
get	Haver Analytics connection properties
haver	Connect to local Haver Analytics database
havertool	Haver Analytics graphical user interface demonstration
info	Information about Haver Analytics variable
isconnection	True if valid Haver Analytics connection
nextinfo	Information about Haver Analytics variable

haver

Purpose

Connect to local Haver Analytics database

Syntax

```
H = haver(Databasename)
```

Arguments

`Databasename`	Local path to the Haver Analytics database.

Description

`H = haver(Databasename)` establishes a connection to a Haver Analytics database.

Examples

To create a connection to the Haver Analytics database at a specified path, enter

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

See Also

`close`, `fetch`, `get`, `isconnection` (Haver Analytics functions)

Purpose

Close Haver Analytics database

Syntax

```
close(H)
```

Arguments

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

Description

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

Examples

Establish a Haver Analytics connection `H`:

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

Close the connection:

```
close(H)
```

See Also

`haver` (Haver Analytics functions)

fetch

Purpose

Request data from Haver Analytics

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 `haver` function.
`S`	Haver Analytics variable.
`Startdate`	MATLAB string or date number indicating the `startdate` from which data is to be retrieved.
`Enddate`	MATLAB string or date number indicating the `enddate` of the date range.
`P`	A specified period. The period can be entered as: • `D` for daily values • `W` for weekly values • `M` for monthly values • `Q` for quarterly values • `A` for annual values

Description

`fetch` returns historical data using the Haver Analytics connection.

Examples

Using the Haver Analytics daily demonstration database `haverd.dat`:

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

Returns all data for the variable `FFED`:

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

Returns the data for `FFED` for a specified date range:

```
D = fetch(H,'FFED','01/01/1971','07/01/1995')
```

Returns the data for `FFED` converted to monthly values for the specified date range:

```
D = fetch(H,'FFED','01/01/1971','07/01/1995','M')
```

See Also

`close`, `get`, `isconnection`, `haver`, `info`, `nextinfo` (Haver Analytics functions)

Purpose

Haver Analytics connection properties

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` and each field contains the value of that property.

Examples

Establish a Haver Analytics connection, `HDAILY`:

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

The command:

```
V = get(HDAILY,{'databasename'})
```

Returns:

```
V =
      databasename: d:\work\haver\data\haverd.dat
```

See Also

`close`, `fetch`, `isconnection`, `haver` (Haver Analytics functions)

isconnection

Purpose

True if valid Haver Analytics connection

Syntax

```
X = isconnection(H)
```

Arguments

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

Description

`X = isconnection(H)` returns `X = 1` if the connection is a valid Haver Analytics connection, and `X = 0` if it is not.

Examples

Establish a Haver connection `H`:

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

Verify that `H` is a valid Haver Analytics connection:

```
X = isconnection(H)
X = 1
```

See Also

`close`, `fetch`, `get`, `haver` (Haver Analytics functions)

Purpose

Information about 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 after `FFED`:

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

MATLAB returns the following structure:

```
            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

close, get, haver, info, isconnection (Haver Analytics functions)

Purpose

Information about Haver Analytics variable

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')
```

MATLAB returns the following structure:

```
    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

close, get, isconnection, haver, nextinfo (Haver Analytics functions)

Purpose

Haver Analytics graphical user interface demonstration

Syntax

```
havertool(H)
```

Arguments

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

Description

`havertool(H)` runs the Haver Analytics graphical user interface demonstration.

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` (Haver Analytics functions)

Hyperfeed

close	Close Hyperfeed connection
fetch	Request data from Hyperfeed
get	Hyperfeed connection properties
hyperfeed	Connect to Hyperfeed
isconnection	True if valid Hyperfeed connection

Purpose

Close Hyperfeed connection

Syntax

```
close(Connect)
```

Arguments

`Connect`	Hyperfeed connection object created with the `hyperfeed` function.

Description

`close(Connect)` closes the connection to the Hyperfeed data server.

See Also

`hyperfeed` (Hyperfeed functions)

fetch

Purpose

Request data from Hyperfeed

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`	Hyperfeed connection object created with the `hyperfeed` function.
`'Security'`	A MATLAB string or cell array of strings containing the names of a securities in a format recognizable by the Hyperfeed server.
`'Fields'`	A MATLAB string or cell array of strings indicating the data fields for which data is to be retrieved. Some possible values are: • `Symbol` • `Last` • `Date` • `Time` • `Change` • `Open` • `High` • `Low` • `Volume`

`'Date'`	Date string or serial date number indicating date for the requested data. If today's date is entered, yesterday's data is returned.
`'FromDate'`	Beginning date for historical data.

Note Dates can be specified 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: • `'d'`: daily • `'w'`: weekly • `'m'`: monthly • `'v'`: dividends

Description

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

`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

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

```
c = hyperfeed('History');

ClosePrice = fetch(c,'ko','Close','Apr 6 00')

ClosePrice =

     730582.00         45.75
```

See Also

`close`, `get`, `hyperfeed`, `isconnection` (Hyperfeed functions)

Purpose

Hyperfeed connection properties

Syntax

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

Arguments

`Connect`	Hyperfeed connection object created with the `hyperfeed` function.
`'PropertyName'`	(Optional) A MATLAB string or cell array of strings containing property names. Property names are: • `'Connection'` • `'IPAddress'` • `'Port'` • `'Socket'` • `'Version'`

Description

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

```
c = hyperfeed('Price')
```

Use the `get` function to retrieve the connection property value:

```
h = get(c, Connection)
```

```
h=
```

```
    connection: 3
    table: 'Price'
```

See Also

close, fetch, hyperfeed, isconnection (Hyperfeed functions)

Purpose

Connect to Hyperfeed

Syntax

```
Connect = hyperfeed(Table)
```

Arguments

`Table`	A MATLAB string indicating the Hyperfeed table (database) to access. Possible values are: • `'Price'` • `'Profile'` • `'History'`

Description

`Connect = hyperfeed(Table)` connects to the indicated Hyperfeed table.

Examples

Connect to the Hyperfeed Price table:

```
c = hyperfeed('Price')
```

See Also

`close`, `fetch`, `get`, `isconnection` (Hyperfeed functions)

isconnection

Purpose

True if valid Hyperfeed connection

Syntax

```
x = isconnection(Connect)
```

Arguments

`Connect`	Hyperfeed connection object created with the `hyperfeed` function.

Description

`x = isconnection(Connect)` returns `x = 1` if the connection is a valid Hyperfeed connection, and `x = 0` if it is not.

Examples

Establish a Hyperfeed connection, `c`, to the Price table:

```
c = hyperfeed
```

Verify that `c` is a valid Hyperfeed connection:

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

```
x = 1
```

See Also

`close`, `fetch`, `get`, `hyperfeed` (Hyperfeed functions)

Interactive Data Pricing and Reference Data

close	Close Interactive Data Pricing and Reference Data connection
fetch	Request data from Interactive Data Pricing and Reference Data
get	Interactive Data Pricing and Reference Data connection properties
idc	Connect to Interactive Data Pricing and Reference Data
isconnection	True if valid Interactive Data Pricing and Reference Data connection

close

Purpose

Close Interactive Data Pricing and Reference Data connection

Syntax

```
close(Connect)
```

Arguments

`Connect`	Interactive Data Pricing and Reference Data connection object created with the `idc` function.

Description

`close(Connect)` closes the connection to the Interactive Data Pricing and Reference Data server.

Examples

Establish an Interactive Data Pricing and Reference Data connection, `c`:

```
c = idc
```

Closes this connection:

```
close(c)
```

See Also

`idc` (Interactive Data Pricing and Reference Data functions)

Purpose

Request data from Interactive Data Pricing and Reference Data

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 connection object created with the `idc` function.
`'Security'`	A MATLAB string containing the name of a security in a format recognizable by the Interactive Data Pricing and Reference Data server.
`'Fields'`	A MATLAB string or cell array of strings indicating specific fields for which data is to be provided. Valid field names are in the file `@idc/idcfields.mat`. The variable `bbfieldnames` contains the list of field names.
`'FromDate'`	Beginning date for historical data.

Note Dates can be specified 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.
`'GUICategory'`	GUI category. Possible values are: • `'F'` (All valid field categories) • `'S'` (All valid security categories)

fetch

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 dialog box for selecting fields or securities.

Examples

Open the dialog box for looking up securities:

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

Open the dialog box for selecting fields:

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

See Also

`close`, `get`, `idc`, `isconnection` (Interactive Data Pricing and Reference Data functions)

Purpose

Interactive Data Pricing and Reference Data connection properties

Syntax

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

Arguments

`Connect`	Interactive Data Pricing and Reference Data connection object created with the `idc` function.
`PropertyName`	(Optional) A MATLAB string or cell array of strings containing property names. Property names are: • `'Connected'` • `'Connection'` • `'Queued'`

Description

`value = get(Connect, 'PropertyName')` returns the value of the specified properties for the Interactive Data Pricing and Reference Data 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

`close`, `get`, `idc`, `isconnection` (Interactive Data Pricing and Reference Data functions)

idc

Purpose

Connect to Interactive Data Pricing and Reference Data

Syntax

```
Connect = idc
```

Description

`Connect = idc` connects to the Interactive Data Pricing and Reference Data server. `Connect` is a connection handle used by other functions to obtain data.

Examples

Connect to the Interactive Data Pricing and Reference Data server:

```
c = idc
```

See Also

`close`, `fetch`, `get`, `isconnection` (Interactive Data Pricing and Reference Data functions)

Purpose

True if valid Interactive Data Pricing and Reference Data connection

Syntax

```
x = isconnection(Connect)
```

Arguments

`Connect`	Interactive Data Pricing and Reference Data connection object created with the `idc` function.

Description

`x = isconnection(Connect)` returns `x = 1` if the connection is a valid Interactive Data Pricing and Reference Data connection, and `x = 0` if it is not.

Examples

Establish an Interactive Data Pricing and Reference Data connection `c`:

```
c = idc
```

Verify that `c` is a valid Interactive Data Pricing and Reference Data connection:

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

See Also

`close`, `fetch`, `get`, `idc` (Interactive Data Pricing and Reference Data functions)

Federal Reserve Economic Data

close	Close FRED connection
fetch	Request data from FRED
fred	Connect to FRED
get	FRED connection properties
isconnection	True if valid FRED connection

Purpose

Connect to FRED

Syntax

```
Connect = fred(URL)
Connect = fred
```

Arguments

`URL`	Create a connection using a specified URL.

Description

`Connect = fred(URL)` establishes a connection to a FRED data server.

`Connect = fred` verifies that the URL `http://research.stlouisfed.org/fred2/` is accessible and creates a connection.

Examples

Connect to the FRED server at the specified URL:

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

See Also

`close`, `fetch`, `get`, `isconnection` (FRED functions)

close

Purpose

Close FRED connection

Syntax

```
close(Connect)
```

Arguments

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

Description

`close(Connect)` closes the connection to the FRED data server.

Examples

Establish a FRED connection, `c`:

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

Close this connection:

```
close(c)
```

See Also

`fred` (FRED functions)

Purpose

Request data from FRED

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 data is to be retrieved. Returns all data for the date `'D1'`; if `'D1'` is today's date, the data from yesterday is returned.
`'D2'`	MATLAB string or date number indicating the date range from which data is to be retrieved. Returns all data for the given security 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.

Description

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

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 a specified date range:

```
d = fetch(f,'DEXUSEU','01/01/2006','06/01/2006')
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

`close`, `get`, `isconnection` (FRED functions)

Purpose

FRED connection properties

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 FRED connection, `c`:

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

Run the following command:

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

MATLAB returns:

```
p =
     port: 8194
     ip: 111.222.33.444
```

See Also

`close`, `fetch`, `isconnection` (FRED functions)

Purpose

True if valid FRED connection

Syntax

```
x = isconnection(Connect)
```

Arguments

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

Description

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

Examples

Establish a FRED connection, `c`:

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

Verify that `c` is a valid FRED connection:

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

See Also

`close`, `fetch`, `get` (FRED functions)

Kx Systems

close	Close Kx kdb+ connection
exec	Execute Kx kdb+ command without waiting for response
fetch	Request data from Kx kdb+ database
get	Get Kx kdb+ connection properties
insert	Write data to Kx kdb+ database
isconnection	True if valid Kx kdb+ connection
kx	Connect to Kx kdb+ database
tables	Table names from Kx kdb+ database

Purpose

Connect to Kx kdb+ database

Syntax

```
K = kx(IP,P)
K = kx(IP,P,ID)
```

Arguments

`IP`	IP address for the connection to the Kx kdb+ database.
`P`	Port for the Kx kdb+ database connection.
`ID`	The *username:password* string for the Kx kdb+ database connection.

Description

`K = kx(IP,P)` makes a connection to the Kx kdb+ database given an IP address, `IP`, and port number, `P`. `K = kx(IP,P,ID)` makes a connection to the Kx kdb+ database given an IP address, `IP`, port number, `P`, and *username:password* string, `ID`.

Note The Kx file `kx.jar` must be added to the MATLAB javaclasspath using the `javaaddpath` command. In the following example, `kx.jar` is added to the MATLAB javaclasspath `c:\q\java`:

```
javaaddpath C:\q\java.kx.jar
```

Examples

Run the following command from a DOS prompt:

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

Run the command:

```
K = kx('LOCALHOST',5001)
```

MATLAB returns:

```
K =
        handle: [1x1 c]
```

```
ipaddress: 'localhost'
     port: 5001
```

See Also

`close`, `exec`, `get`, `fetch`, `tables` (Kx functions)

Purpose

Close Kx kdb+ connection

Syntax

```
close(K)
```

Arguments

`K`	Kx kdb+ connection object created with the `kx` function.

Description

`close(K)` closes the connection to the Kx kdb+ database.

Examples

Run the following command from a DOS prompt:

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

Close this connection:

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

See Also

`kx` (Kx functions)

fetch

Purpose

Request data from Kx kdb+ database

Syntax

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

Arguments

`K`	Kx kdb+ connection object created with the `kx` function.
`KSQL`	The Kx kdb+ command.
`P1,P2,P3`	Input parameters for the `KSQL` command.

Description

`D = fetch(K,KSQL)` returns data from a Kx kdb+ database in a MATLAB structure where `K` is the Kx 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:

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

Run the command:

```
K = kx('localhost',5001);
D = fetch(K,'select from trade');
```

MATLAB returns:

```
D =
       sec: {5000x1 cell}
```

```
          price: [5000x1 double]
         volume: [5000x1 int32]
       exchange: [5000x1 double]
           date: [5000x1 double]
```

The `'ACME'` input parameter returns:

```
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 kdb+ file, `tradedata.q`.

See Also

`exec`, `insert`, `kx` (Kx functions)

Purpose

Get Kx kdb+ connection properties

Syntax

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

Arguments

`K`	Kx 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 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

Run the following command from a DOS prompt:

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

Run the command:

```
K = kx('LOCALHOST',5001);
V = get(K)
```

MATLAB returns:

```
V =
```

```
    handle:  [1x1 c]
ipaddress: 'localhost'
     port: '5001'
```

See Also

close, exec, fetch, insert, kx (Kx functions)

Purpose

Execute Kx kdb+ command without waiting for response

Syntax

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

Arguments

`K`	Kx kdb+ connection object created with the `kx` function.
`Command`	Kx kdb+ command issued using the Kx kdb+ connection object created with the `kx` function.
`P1,P2,P3`	Input parameters for `Command`.

Description

`exec(K,Command)` executes the specified `Command` in Kx 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

Run the following command from a DOS prompt:

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

Retrieve the data in the table `trade`:

```
K = kx('localhost',5001);
exec(K,'`date xasc`trade');
```

The `exec` command sorts the data in the table `trade` in ascending order. Data subsequently fetched from the table is ordered in this manner.

See Also

`fetch`, `insert`, `kx` (Kx functions)

Purpose

Write data to Kx kdb+ database

Syntax

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

Arguments

`K`	The Kx kdb+ connection object created with the `kx` function.
`Tablename`	The name of the Kx kdb+ `Tablename`.
`Data`	The data to be written to the Kx kdb+ `Tablename`.

Description

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

Examples

Run the following command from a DOS prompt:

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

Run the commands:

```
K = kx('localhost',5001);
insert(K,'trade',{'`ACME',133.51,250,6.4,'2006.10.24'})
```

See Also

`close`, `fetch`, `get`, `tables` (Kx functions)

isconnection

Purpose

True if valid Kx kdb+ connection

Syntax

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

Arguments

`K`	Kx kdb+ connection object created with the `kx` function.

Description

`X = isconnection(K)` returns `X = 1` if the connection is a valid Kx kdb+ connection, and `x = 0` if it is not.

Examples

Run the following command from a DOS prompt:

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

In MATLAB, establish a Kx kdb+ connection:

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

Verify that `K` is a valid Kx kdb+ connection:

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

See Also

`close`, `fetch`, `get`, `kx` (Kx functions)

Purpose

Table names from Kx kdb+ database

Syntax

```
T = tables(K)
```

Arguments

`K`	The Kx kdb+ connection object created with the `kx` function.

Description

`T = tables(K)` returns the list of tables for the Kx kdb+ connection.

Examples

Run the following command from a DOS prompt:

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

Retrieve table information for the Kx kdb+ database:

```
K = kx('localhost',5001);
T = tables(k)
```

MATLAB returns:

```
T =

      'intraday'
    'seclist'
      'trade'
```

See Also

`exec`, `fetch`, `insert`, `kx` (Kx functions)

Reuters

close	Release Reuters session
fetch	Request data from Reuters
get	Reuters session properties
reuters	Create Reuters session
stop	Unsubscribe securities

Purpose

Release Reuters session

Syntax

```
close(r)
```

Arguments

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

Description

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

Examples

Release the Reuters connection `r` and unsubscribe all requests associated with it:

```
close(r)
```

See Also

`reuters` (Reuters functions)

fetch

Purpose

Request data from Reuters

Syntax

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

Arguments

`r`	Reuters session object created with the `reuters` function
`s`	Reuters security object
`callback`	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`. `callback` is the MATLAB function that runs for each data event that occurs.

Examples

Example 1: Retrieve the current data for the security `GOOG.O` using the Reuters session object `r`:

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

Following is a partial listing of the security data that MATLAB returns:

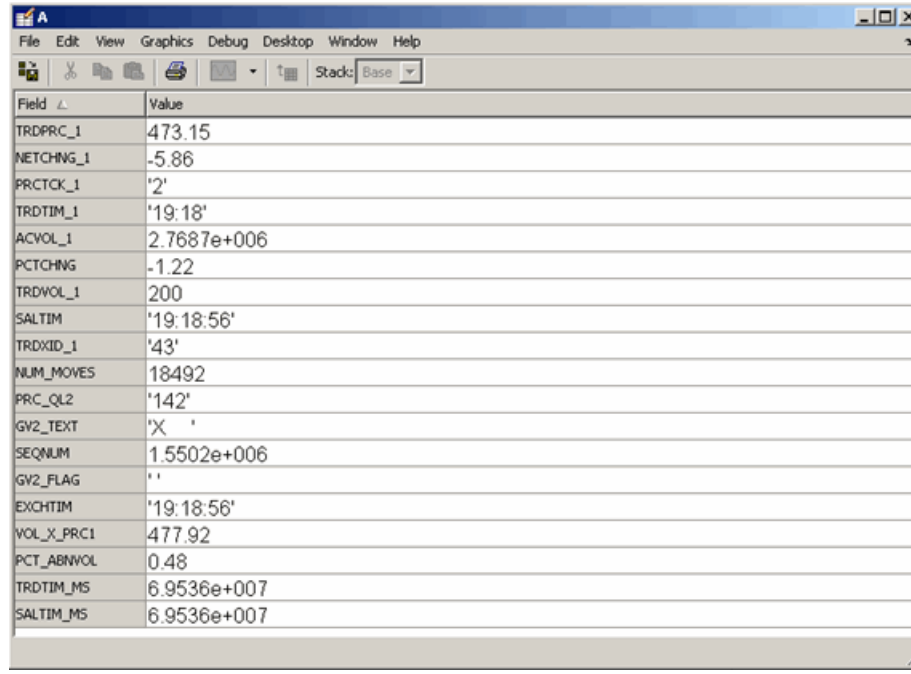
```
d =
PROD_PERM: 74.00
RDNDISPLAY: 66.00
DSPLY_NAME: 'DELAYED-15GOOGLE'
RDN_EXCHID: 'O'
TRDPRC_1: 474.28
TRDPRC_2: 474.26
TRDPRC_3: 474.25
TRDPRC_4: 474.25
TRDPRC_5: 474.25
```

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

Example 2: Subscribe to a security and process the data in real time.

To do this, you must specify a MATLAB callback function to be run each time a real-time data event is received 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 array 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')
```



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'
GV2_TEXT	'X '
SEQNUM	1.5502e+006
GV2_FLAG	''
EXCHTIM	'19:18:56'
VOL_X_PRC1	477.92
PCT_ABNVOL	0.48
TRDTIM_MS	6.9536e+007
SALTIM_MS	6.9536e+007

See Also

`reuters`, `close`, `stop` (Reuters functions)

Purpose

Reuters session properties

Syntax

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

Arguments

`r`	Reuters session object created with the `reuters` function
`f`	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` (Reuters functions)

Purpose

Create Reuters session

Syntax

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

Arguments

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

Description

`r = reuters (sessionName, serviceName)` initiates 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)` initiates 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. If you require DACS authentication, you must use this form of the command.

Examples

Example 1: Connect to Reuters with session name `'myNS::remoteSession'` and service name `'dIDN_RDF'`:

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

MATLAB returns the following:

```
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 The following error message appears if you do not use Reuters DACS authentication:

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

Example 2: Connect to Reuters 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')
```

See Also

`fetch` (Reuters functions)

stop

Purpose

Unsubscribe securities

Syntax

```
stop(r)
stop(r,d)
```

Arguments

`r`	Reuters session object created with the `reuters` function
`d`	Subscription handle returned by `reuters/fetch`

Description

`stop(r)` unsubscribes all securities associated with the Reuters session `r`.

`stop(r,d)` unsubscribes the securities associated with the subscription handle `d`, where `d` is the subscription handle returned by `reuters/fetch`.

Examples

Example 1: Unsubscribe securities associated with a specific request `d` and a Reuters connection object `r`:

```
stop(r,d)
```

Example 2: Unsubscribe all securities associated with the Reuters connection object `r`:

```
stop(r)
```

See Also

`fetch`, `reuters` (Reuters functions)

Yahoo!

close	Close Yahoo! connection
fetch	Request data from Yahoo!
get	Yahoo! connection properties
isconnection	True if valid Yahoo! connection
yahoo	Connect to Yahoo!

close

Purpose

Close Yahoo! connection

Syntax

```
close(Connect)
```

Arguments

`Connect`	Yahoo! connection object created with the `yahoo` function.

Description

`close(Connect)` closes the connection to the Yahoo! data server.

See Also

`yahoo` (Yahoo! functions)

Purpose

Request data from Yahoo!

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 data is to be retrieved. 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`.

`Date`	Date string or serial date number indicating date for the requested data. If today's date is entered, yesterday's data is returned.
`FromDate`	Beginning date for historical data.

Note Dates can be specified 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: • `'d'`: daily • `'w'`: weekly • `'m'`: monthly • `'v'`: dividends

Description

`data = fetch(Connect, 'Security')` returns data for all fields from Yahoo!'s Web site for the indicated security. Retrieving multiple securities at one time is not supported for Yahoo. 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

Example 1: 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
```

Example 2: Use 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
```

See Also

`close`, `get`, `isconnection`, `yahoo` (Yahoo! functions)

Purpose

Yahoo! connection properties

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`, and each field contains the value of that property.

Examples

Use the `yahoo` function to establish a connection to Yahoo!.

```
c = yahoo

c =

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

Now use the `get` function to retrieve the connection property value.

```
get(c, 'url')

ans =

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

See Also

`close`, `fetch`, `isconnection`, `yahoo` (Yahoo! functions)

isconnection

Purpose

True if valid Yahoo! connection

Syntax

```
x = isconnection(Connect)
```

Arguments

`Connect`	Yahoo! connection object created with the `yahoo` function.

Description

`x = isconnection(Connect)` returns `x = 1` if the connection is a valid Yahoo! connection, and `x = 0` if it is not.

Examples

The function

```
c = yahoo
```

establishes a Yahoo! connection, `c`.

Then

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

indicates that `c` is a valid Yahoo! connection.

See Also

`close`, `fetch`, `get`, `yahoo` (Yahoo! functions)

Purpose

Connect to Yahoo!

Syntax

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

Arguments

`URL`	Must be http://quote.yahoo.com.
`IPAddress`	A MATLAB string containing the Internet address of 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

Use the `yahoo` function to establish a connection to the Yahoo! data server.

```
Connect = yahoo

Connect =

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

Use the `yahoo` function to establish a connection 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

close, fetch, get, isconnection (Yahoo! functions)

A

Examples

Use this list to find examples in the documentation.

Communicating with Financial Data Servers

Retrieving Connection Properties

Retrieving Data

Index

H

I

K

L

M

N

P

R

S

T

V

Y