Web Services for MATLAB

User Guide

MORNINGSTAR®

Table of Contents

Web Services for MATLAB

Overview

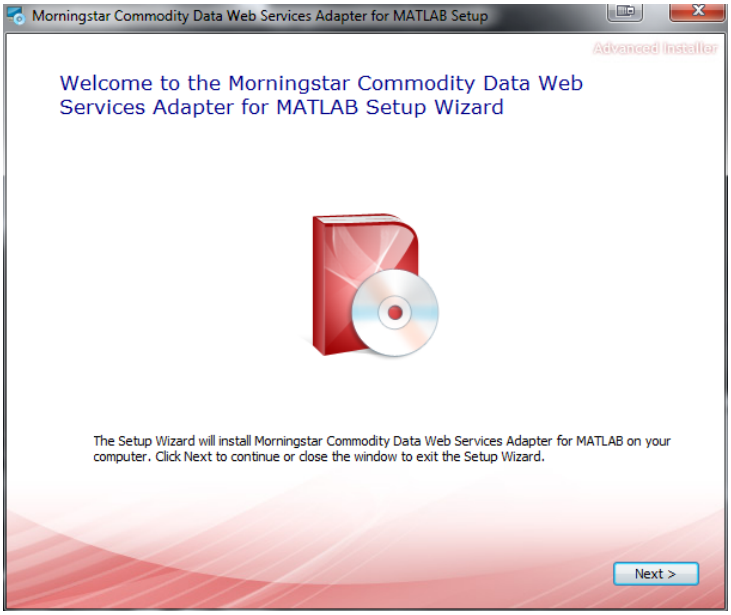
The Morningstar Commodity Data Web Services Adapter for MATLAB provides the ability to retrieve market data from a Morningstar Commodity Data Web Services Server within the MATLAB console. Users can employ the MATLAB functions provided to request data by either specifying a set of symbols and fields directly or employing the Morningstar Commodity Data Server Query Language queries. Data request results are formatted as MATLAB objects that contain data points, date and time information and other pertinent information. Users can get more information about each function provided by using MATLAB's help facility.

Installation

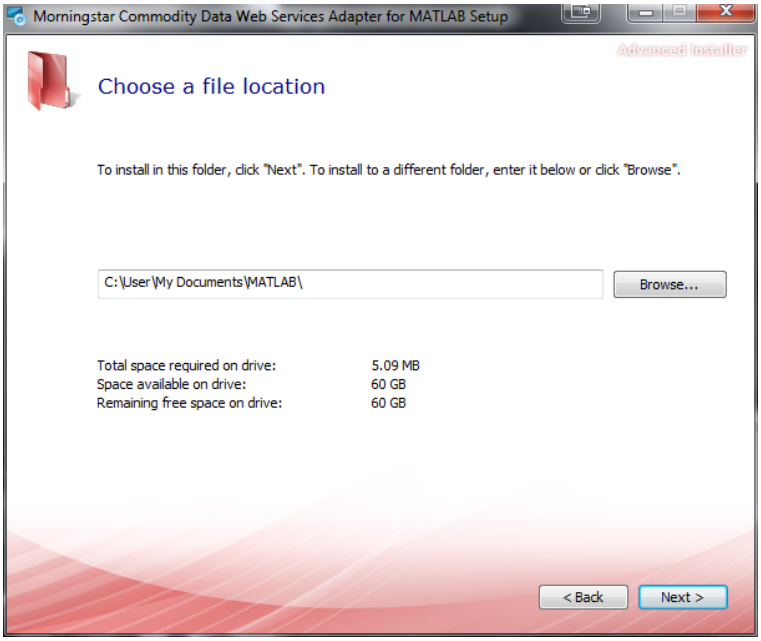
Installing Morningstar Commodity Data Web Services Adapter for MATLAB requires running the installer and directing the installation to your MATLAB working directory.

The installer can be sent via request by contacting commoditydata-support@morningstar.com.

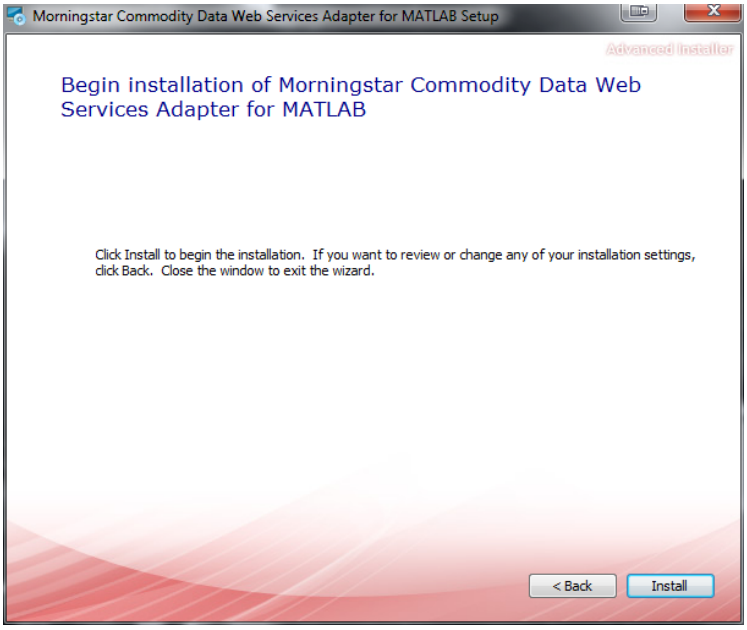
Once downloaded, run the installer.



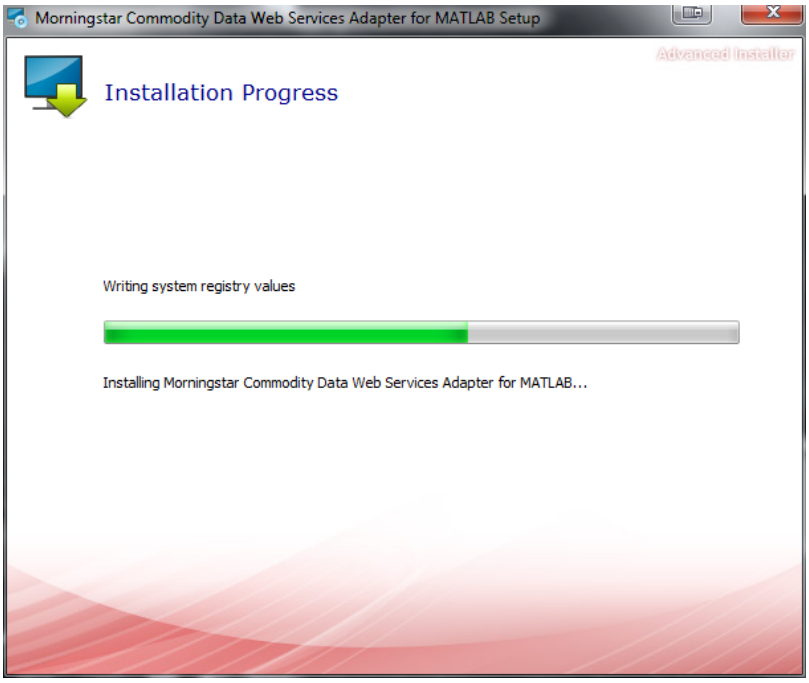
The default directory for installation is "%HOMEPATH\My Documents\MATLAB" which is the default working directory for MATLAB. If you use a different directory, navigate the installer to that directory.



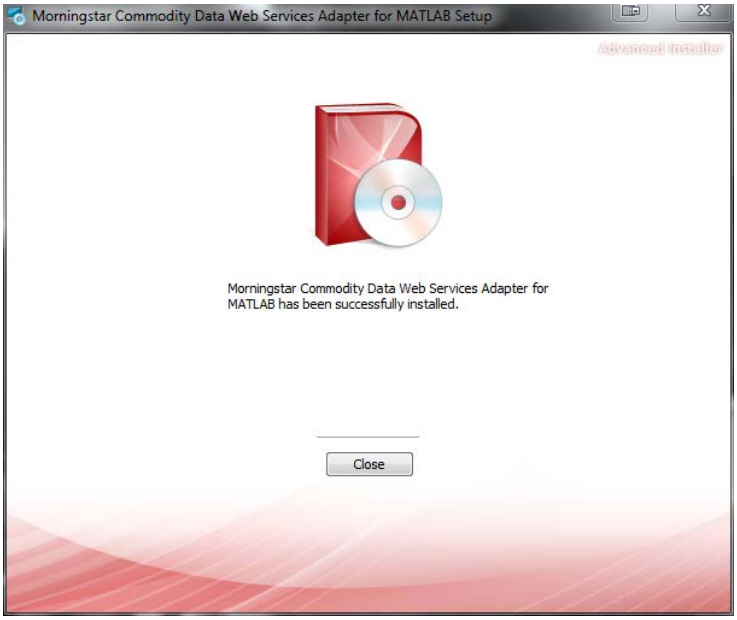
Once you have selected your MATLAB working folder, the installer is ready to run.



The installer will place all of the required files where needed.



After the installation has finished you are ready to start using the Web Services Adapter for MATLAB. Consult the 'Getting Started' section for more information.



Getting Started

Once the installer has finished, start MATLAB and run the following command to being using MATLAB API:

```
setURL('https://yourwebservicesurlhere.com')
```

This command will need to be run every time you restart MATLAB to ensure that the connection is valid, and all security features are working correctly. This command will retrieve the certificate from the provided URL enabling encrypted traffic.

The `setURL` command will also dynamically load the required java files into MATLAB for future use.

Web Services Connection Functions

Setting the Web Services URL

All Web Service Adapter methods that interact with a web services require a URL to be set. This URL instructs MATLAB API where to send its requests for data and authorization. This MATLAB function will also add the required java filed to MATLAB's classpath. All other function calls will fail until setURL is used making it the first function used for every session.

This method will fail if the connection to the URL is down or the URL was invalid.

Usage:

```
setURL(String webServiceURL)
```

Argument(s)	
webServiceURL	target web services URL.

Return Value:

If the URL is valid and accessible, a message telling you it has completed successfully. Otherwise it will return a message about what has failed.

Example(s):

The following example demonstrates a successful setURL:

```
>> setURL('https://ws.morningstarcommodity.com')

ans =

Commodity Web Service URL successfully set to
https://ws.morningstarcommodity.com
```

The following example demonstrates an unsuccessful setURL:

```
>> setURL('https://please-fail.com')

 ans =

 ERROR: Could not reach https://please-fail.com check your network
connection
```

Create an Authentication Token

All Web Service Adapter data methods (other than setURL) require an authentication token to run. An authentication token is an encoded String that holds user information as well as certain settings such as how to format results, date-time formats, etc.

A valid authentication token can be obtained only after the web services URL has been set (see setURL) and if the user credentials provided to getAuthToken can be used to successfully authenticate with the web services.

If you do not have credentials, or are having trouble obtaining a valid token, contact your IT support or Morningstar support for assistance.

Usage:

```
token = getAuthToken(String username, String password)
```

Argument(s)	
username	Login used to access Morningstar web services. Consult your IT personnel or Morningstar support if you not have a username.
password	Password for the login specified for the username parameter.

Return Value:

If the username and password can be authenticated successfully, getAuthToken will return an encoded String. If there are any issues with authentication, getAuthToken will return a String indicating the error encountered during authentication.

Example(s):

The following example demonstrates how to obtain an authentication token:

```
token = getAuthToken('matlabuser@mycompany.com','mypassword')
```

Retrieving Data

Executing DataServer Queries

Description:

Use the executeQuery method to run queries.

Usage

```
String res = executeQuery(String authToken, String mimQuery)
```

Argument(s)	
authToken	used to authenticate with web services. See help for authToken method to find out more about obtaining an authentication token.
mimQuery	character String containing the query to be executed.

Return Value:
A MATLAB struct consisting of values, headings, and dateList.

If a LET statement with a list of symbols was used in the query, the result set will have multiple objects, each having a headings, dateList, and values components.

The values data structure is a matrix that holds the values for each rel-col pair requested. Description of each column of the values matrix is provided in headings. The dateList is a cell array which holds the date and time information for each row in the values matrix.

Queries will ignore the Data Precision setting and return the raw numbers WebServices retrieves from the server itself. Matlab may format the numbers to fit the workspace settings.

Example(s):
The following is an example of executing a simple DataServer query:

```
>> res = executeQuery(token, 'SHOW 1: Close of NG when date is within 1 week')

res =

dateList: {5x1 cell}

headings: {'1'}

values: [5x1 double]
```

The result set returned as a result of executing a query is similar to that of getRecords with the exception that the headings cell array holds the name of the attribute specified in the SHOW statement. For example, if the attribute is specified as CloseOfNG as in the following query:

```
>> res = executeQuery(token, 'SHOW CloseOfNG: Close of NG when date is within
1 week');

res =

dateList: {5x1 cell}
```

```
headings: {'CloseOfNG'}

values: [5x1 double]
```

The executeQuery command also supports multiple reports as seen in the example below. Note that res(1) corresponds to IBM, res(2) corresponds to MSFT and so on:

```
>> res = executeQuery(token, 'LET @MyList = IBM, MSFT, DELL, CSCO SHOW 1: Bar
of @MyList when date is within 1 month');

res =

1x4 struct array with fields:

dateList

headings

values

>> res(1)

ans =

dateList: {22x1 cell}

headings: {'1'  '1'  '1'  '1'}

values: [22x4 double]

>> res(2)

ans =

dateList: {22x1 cell}

headings: {'1'  '1'  '1'  '1'}

values: [22x4 double]
```

Execution options can be specified for queries as long as there is a newline character after the last execution option:

```
>> res = executeQuery(token, ['%exec.units: 1 hour' char(10) ' SHOW 1: Bar of
IBM when date is after 2009']);
```

```
res =

dateList: {602x1 cell}

headings: {'1'  '1'  '1'  '1'}

values: [602x4 double]
```

Retrieving Specific Data from the DataServer

Description:

Use the getRecords call to obtain time-series data for a set of relations and columns.

Usage:

```
String xmlResult = getRecords(String authToken, String relList, String
colList, String units, String fromDateTime, String toDateTime)
```

Argument(s)	
authToken	Used to authenticate with web services - See help for authToken method to find out more about obtaining an authentication token.
relList	comma-separated list of relation names
colList	comma-separated list of column names
units	Unit of data to return. Valid units are:MILLSECONDS, SECONDS, MINUTES, HOURS, DAYS, WEEKS, MONTHS, QUARTERS, YEARS
fromDateTime	Specifies the starting date in for the date range to pull data for. If a time value is also specified, that time value is used as the starting time for the time range in addition to the date range. User needs to set the date and time format a time range can be specified. See the help section for setDateTimeFormat command for more information.
toDateTime	Specifies the ending date in for the date range to pull data for. If a time value is also specified, that time value is used as the ending time for the time range in addition to the date range. User needs to set the date and time format a time range can be specified. See the help section for setDateTimeFormat

	command for more information.

Return Value:
A MATLAB struct consisting of values, headings, and dateList.

The values data structure is a matrix that holds the values for each rel-col pair requested. Description of each column of the values matrix is provided in headings. The dateList is a cell array which holds the date and time information for each row in the values matrix.

GetRecords requests will honor the Data Precision setting but the numbers are subject to formatting by the Matlab workspace settings. Matlab will not lose accuracy but may append zeros or express the number in scientific notation.

Example(s):
The following example demonstrates retrieving data for the High and Low columns for relations IBM and MSFT:

```
>> res1 = getRecords(token,'IBM,MSFT','High,Low','DAYS','2009-04-01', '2010-04-01');

res1 =

dateList: {262x1 cell}

headings: {'High of IBM'  'Low of IBM'  'High of MSFT'  [1x11 char]}

values: [262x4 double]
```

Uploading Data

Data can be uploaded back to the Commodity Data server from Matlab. The API provides the ability to

1) Create an upload job. This will be the container for data points that will later be uploaded to the server. You must specify the date format that will be used for the timestamp. The user

will also need to specify the parser type. For more information on parser types, refer to the data upload section of the commodity data excel add-in users guide.

2) Add data points to the upload job. You can add one or many data points. Note that the timestamp must conform to the format specified in the job.
3) Submit the upload via the 'uploadData' command. The command requires a valid authentication token and an upload job. Once submitted, the upload job will format the data into XML which the server can consume and transfer the data to the server using a web service call.

Creating an Upload Job

Create a container object that will be used to construct the data upload request.

Data points that are to be uploaded to the server are added to this object via the 'addToUpload' function.

Once all data points have been added, the contents of the object can be submitted for upload via the 'uploadData' method. Upon submission, this object will format the data points into an XML format that can be processed on the server.

Note that the date-time information for each data point in the container must conform to the date-time format given in the format parameter.

Usage

`job =` createUploadJob(String dateTimeFormat, String parserType)

Argument	Description
dateTimeFormat	String indicating the date-time format Example: yyyy-mm-dd
parserType	The parser to be used for interpreting the data uploaded data on the server. Please see the Commodity Add-In documentation Section: Upload for to see the parser list.

Time Format acceptable fields

Letter	Date or Time Component	Example
Y	Year	1996; 96
M	Month in year	July
w	Week in year	27
W	Week in month	2
D	Day in year	189
d	Day in month	10
F	Day of week in month	2
E	Day in week	Tuesday; Tue
a	AM/PM marker	PM
H	Hour in day (0-23)	3
k	Hour in day (1-24)	23
K	Hour in am/pm (0-11)	0
h	Hour in am/pm (1-12)	12
m	Minute in hour	30
s	Second in minute	55
S	Millisecond	978

Return Value:
An empty container object used to construct the data upload request.

Adding Data to an Upload Job

Function name: addToUpload - adds a datapoint to upload container object.

Multiple calls to this method can be made with the same uploadJob container object to add a set of data points.

Usage

dataObj = addToUpload(UploadJob payload,
String rel,
String col,
String dateTime,
String value)

Argument(s)	
payload	the container object obtained from a call to createUploadJob.
rel	The fully-qualified relation name (start with "TopRelation"). The relation name must be upper-case (the part after the last ':' in the name). For example, TopRelation:TestCategory:TestRelation -> INVALID, relation name is not uppercase TopRelation:TestCategory:TESTREL -> VALID For more information about valid relation names, please see the Upload section in the Commodity Data Add-In Users Guide. If the relation does not already exist, it will be created.
col	The column name. The column must already be defined on the server. Example: Open, High, Low, Close, Volume, LmpVal
dateTime	The date data is going to be added to. the date must be conformed to the format of the uploadJob container object
Value	data corresponding to the relation

Return Value:

The container object which holds the specified datapoint.

Submit Upload Job to Server

Function name: UploadData - Submits a data upload request contained in the uploadJob container to the server.

Usage: data <- uploadData(String authToken, UploadJob uploadJob)

Argument(s)	
authToken	Authenticate with Commodity Web Services. See help for getAuth method to find out more about obtaining an authentication token.
uploadjob	The container object that holds the data to be uploaded.

Return Value:
A string indicating upload status

Examples

Upload two data points (daily data):

```
>> uploadJob = createUploadJob('yyyy-MM-dd', 'DefaultParser-7day')

uploadJob =

UploadJob{data=[],
 dateTimeFormat='yyyy-MM-dd',
 parserType='DefaultParser-7day'}

>> addToUpload(uploadJob, 'TopRelation:Test:TESTREL1', 'Close', '2015-02-06',
56.43);
>> addToUpload(uploadJob, 'TopRelation:Test:TESTREL1', 'Close', '2015-02-05',
26.43);
>> uploadData(token, uploadJob)
```

```
ans =

Job 1471: Job completed successfully.

>> getRecords(token, 'TESTREL1', 'Close', 'Days', '2015-02-04','2015-02-06')

ans =

    dateList: {2x1 cell}
    headings: {'Close of TESTREL1'}
      values: [2x1 double]
```

Example #2 Upload Minutely/Hourly data:

```
uploadJob = createUploadJob('yyyy-MM-dd hh:mm', 'DefaultParser-7day')
>> addToUpload(uploadJob, 'TopRelation:Test:TESTREL101', 'Close', '2015-02-05
13:30', 36.31);
>> addToUpload(uploadJob, 'TopRelation:Test:TESTREL101', 'Close', '2015-02-05
13:45', 36.31);
>> uploadData(token, uploadJob)
>> executeQuery(token, ['%exec.units: 15 minutes' char(10) 'SHOW 1: Close of
TESTREL101'])

       ans =

           dateList: {96x1 cell}
           headings: {'1'}
             values: [96x1 double]

>> ans.values
 …
             NaN
         36.3100
          36.3100
        NaN
…
```

Utility Functions

Custom Formatting for GetRecords Date/Time Parameters

Description:

The date-time format is set to yyyy-MM-dd by default. To restrict the result set by specifying a time range in addition to a daterange, the user needs to set the date time format for the GetRecords parameter to include a time value. The user can specify the date and time format to match the from/to date/times they provide in their getRecords calls.

Usage

```
String token = setDateTimeFormat(String token, String format)
```

Argument(s)	
authToken	used to authenticate with web services - See help for authToken method to find out more about obtaining an authentication token.
format	String indicating the date-time format describing the date/time values specified in getRecords calls

Return Value:

The authentication token with updated date-time formatting information. Use this value as your authentication token for the new date-time format to take effect.

Example(s):

The following example demonstrates how to set the date time format to include hours and minutes using the 24-hour format:

```
>> token = setDateTimeFormat(token, 'yyyy-MM-dd HH:mm');

>> dispSettings(token)

User: matlabuser@mycompany.com

Date-Time Format: yyyy-MM-dd HH:mm

Missing Data Option: FILL_NAN

Skip NaN Option: SKIP_NONE

>> token = setDateTimeFormat(token, 'yyyy-MM-dd HH:mm');
```

```
>> res = getRecords(token,'IBM','Low','HOURS','2010-04-09 12:00', '2010-04-09
15:00');

res =

dateList: {4x1 cell}

headings: {'Low of IBM'}

values: [4x1 double]

>> res.dateList

ans =

'2010-04-09 12:00:00.000'

'2010-04-09 13:00:00.000'

'2010-04-09 14:00:00.000'

'2010-04-09 15:00:00.000'

>> res.values

ans =

127.4650

127.4600

127.5500

127.6100
```

Handling Missing Data in GetRecords Results

Description:

Provides control over how missing values are handled:

FILL_NAN: Fill with NaN's (default)

FILL_FORWARD: Fill with previous known value

FILL_BACKWARD: Fill with next known value

FILL_INTERP_LINEAR: Fill with linear interpolated value between last known value and next known value. See Commodity DataServer Data and Development Guide for more information.

FILL_INTERP_GEOMETRIC: Fill with geometric interpolated values between last known value and next known value. See Commodity DataServer Data and Development Guide for more information.

FILL_INTERP_LOGARITHMIC: Fill with logarithmic interpolated values between last known value and next known value. See Commodity DataServer Data and Development Guide for more information.

Usage

```
String token = setMissingDataFillOption(String authToken, String fillOption)
```

Argument(s)	
authToken	used to authenticate with web services - See help for authToken method to find out more about obtaining an authentication token.
fillOption	String indicating which option to use for handling missing data - See description above for valid parameters.

Return Value:

The authentication token with an updated missing data fill option. Use this new value as your authentication token for changes to take effect.

Example(s):

The following example demonstrates how to fill missing data using the FILL_FORWARD option:

```
>> res = getRecords(newToken, 'IBM', 'Close', 'Days', '2011-01-14', '2011-01-
18');

>> res.values,res.dateLis

ans =

  150.0000
```

```
       NaN

  150.6500

ans =

    '2011-01-14 00:00:00.000'

    '2011-01-17 00:00:00.000'

    '2011-01-18 00:00:00.000'

>> newToken = setMissingDataFillOption(newToken, 'FILL_FORWARD');

>> res = getRecords(newToken, 'IBM', 'Close', 'Days', '2011-01-14', '2011-01-
18');

>> res.values,res.dateList

ans =

  150.0000

  150.0000

  150.6500

ans =

    '2011-01-14 00:00:00.000'

    '2011-01-17 00:00:00.000'

    '2011-01-18 00:00:00.000'
```

Handling NaN Values in GetRecords Result Set

Description:
Allows user to specify whether NaN values should be included in the results returned from getRecords calls. Setting the option to SKIP_ALL_NAN will cause NaN values to be omitted from the result set. Setting the option to SKIP_NONE will cause NaN values to be included in results.

The option is set to SKIP_NONE by default.

Usage:

```
String token = setSkipNaN(String authToken, String option)
```

Argument(s)	
authToken	used to authenticate with web services - See help for authToken method to find out more about obtaining an authentication token.
option	String indicating whether NaN values should be included or omitted from the results returned by getRecords calls

Return Value:

The authentication token with an updated 'skip NaN' setting.Use this new value as your authentication token for changesto take effect.

Example(s):

The following example demonstrates how to include and exclude NaN values from getRecords results:

```
>> newToken = setSkipNaN(token, 'SKIP_ALL_NAN');

>> res = getRecords( newToken, 'IBM', 'Close', 'Days', '2011-01-14', '2011-01-18');

res =

dateList: {2x1 cell}

headings: {'Close of IBM'}

values: [2x1 double]

>> res.values,res.dateList

ans =

150.0000

150.6500

ans =

'2011-01-14 00:00:00.000'
```

```
'2011-01-18 00:00:00.000

>> newToken = setSkipNaN(token, 'SKIP_NONE');

>> res = getRecords(newToken, 'IBM', 'Close', 'Days', '2011-01-14', '2011-01-
18');

res =

dateList: {3x1 cell}

headings: {'Close of IBM'}

values: [3x1 double]

>> res.values,res.dateList

ans =

150.0000

NaN

150.6500

ans =

'2011-01-14 00:00:00.000'

'2011-01-17 00:00:00.000'

'2011-01-18 00:00:00.000'
```

Setting the Timeout for Web Data Requests

Description:

Allows user to specify the amount of time a data requests will attempt to retrieve data before returning a timed out message.

The option is set to five minutes by default.

Usage:

```
String token = setTimeout(String authToken, String time)
```

Argument(s)	
authToken	used to authenticate with web services - See help for authToken method to find out more about obtaining an authentication token.
time	string indicating the amount of time to wait in milliseconds.

Return Value:

Authentication token with an updated timeout setting.Use this new value as your authentication token for changes to take effect.

Example(s):

The following example demonstrates how to check and change the timeout:

```
>> dispSettings(token)

User: matlabuser@mycompany.com

Date-Time Format: yyyy-MM-dd

Missing Data Option: FILL_NAN

Skip NaN Option: SKIP_NONE

Timeout: 300000 ms

Data Precision: 3

>> newToken = setTimeout(token, 600000);

>> dispSettings(newToken)

User: matlabuser@mycompany.com

Date-Time Format: yyyy-MM-dd

Missing Data Option: FILL_NAN

Skip NaN Option: SKIP_NONE

Timeout: 600000 ms

Data Precision: 3
```

Setting Data Precision during GetRecords Requests

Description:
Allows user to specify the data precision WebServices will use during GetRecords requests. This setting directly reflects the xml DataRequest attribute of "scale", for more information how this attribute directly influences data retrieved see the WebServices documentation.

The option is set to a precision of 3 decimal places by default; some data will have more than three decimal places and will be rounded upon retrieval. Matlab formats numbers according to workplace settings. By default Matlab will display numbers with four decimal places regardless of how precise the number actually is. Matlab preserves the accuracy of numbers returned so it will not alter the data.

Usage:

```
String token = setDataPrecision(String authToken, String precision)
```

Argument(s)	
authToken	used to authenticate with web services - See help for authToken method to find out more about obtaining an authentication token.
precision	number of decimal places to maintain upon data retrieval from WebServices.

Return Value:
The authentication token with an updated precision setting.Use this new value as your authentication token for changes to take effect.

Example(s):
The following example demonstrates how to check and change the timeout.

```
>> dispSettings(token)

User: matlabuser@mycompany.com

Date-Time Format: yyyy-MM-dd

Missing Data Option: FILL_NAN
```

```
Skip NaN Option: SKIP_NONE

Timeout: 300000 ms

Data Precision: 3

>> newToken = setDataPrecision(token, 5);

>> dispSettings(newToken)

User: matlabuser@mycompany.com

Date-Time Format: yyyy-MM-dd

Missing Data Option: FILL_NAN

Skip NaN Option: SKIP_NONE

Timeout: 600000 ms

Data Precision: 5
```

View Current Settings

Description:

Use the dispSettings call to view the current settings.

Usage:

```
dispSettings(String authToken)
```

Argument(s)	
authToken	used to authenticate with web services - See help for authToken method to find out more about obtaining an authentication token.

Return Value:

A list of key-value pairs corresponding to current settings values.

Example(s):

The following example demonstrates retrieving current settings values:

```
>> dispSettings(token)
```

```
User: matlabuser@mycompany.com

Date-Time Format: yyyy-MM-dd

Missing Data Option: FILL_NAN

Skip NaN Option: SKIP_NONE

Timeout: 300000 ms

Data Precision: 3
```


View the Contents of an UploadJob

Description:

Use the dispUploadJob command to view the current datapoints within an UploadJob

Usage:

```
dispUploadJob(UploadJob job)
```

Argument(s)	
job	the UploadJob created through the createUploadJob function and possibly modified through the addToUploadJob function as described above

Return Value:

None, the job's contents will be printed to the Matlab workspace. The Relation and Column printed will only be the Relation/Column name, the path will be left out.

Example(s):

The following example shows how to create and UploadJob, add data to that UploadJob, and then view the data added to it.

```
>> job = createUploadJob;

>> job = addToUploadJob(job,'TopRelation:TESTREL','TopColumn:TESTCOL','1998-
07-07',5778357);

>> dispUploadJob(job)

Number   Relation        Column          Date            Value

   01.   TESTREL         TESTCOL         1998-07-07      5778357.0
```

View Version of WS Adapter

Description:

Use the dispVersion command to obtain the version of this software and the web services located at the set URL:

Usage:

dispVersion(String authToken)

Argument(s)	
authToken	used to authenticate with web services - See help for getAuthToken method to find out more about obtaining an authentication token.

Return Value:

Version of Web Services and MATLAB API being used.

Example(s):

The following example shows how to obtain version information:

```
>> dispVersion(authToken)
```

```
Commodity MATLAB API version Version 2.0

Commodity Web Services version 2.3.2.46
```

Troubleshooting

If during installation you are unsure about the location of you working directory, start MATLAB and enter the command **pwd**. This will print the current working directory. Afterwards quit MATLAB and return to the installation process.

If the URL has not been set during the current MATLAB session, all methods will error and return:

```
Undefined variable "com" or class
"com.lim.wsadapter.MatlabMethods.getAuthToken".
```

To fix, run:

```
setURL('https://yourwebserviceurl.com')
```

If the error persists or if *setURL* is returning the error, uninstall the MATLAB API from you Windows Control Panel and reinstall the MATLAB API, double check that the directory you are installing to is the directory returned by the MATLAB **pwd** command.