User Guide

LIM Web Services Adapter for MATLAB

Version 1.0

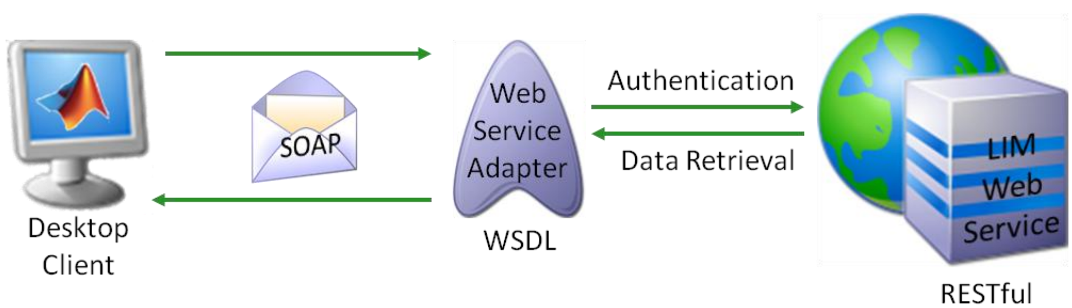
LIM Web Services Adapter for MATLAB

Overview

The LIM Web Services Adapter provides SOAP-based web services that allow access to LIM data. Applications that can act as Web service clients, such as MATLAB, can retrieve the list of available functions, initiate data retrieval requests and receive datasets using the LIM Web Services Adapter interface.

Result sets that are responses to data requests can be formatted in either an XML format that can easily be converted to MATLAB objects or using the default LIM XML format. The XML schemas for both result sets are provided in the appendix.

The LIM Web Services Adapter provides a web services interface access LIM data for users who prefer using SOAP-based web services and would like a WSDL to specify the service contract for functionality provided by the web service. It uses the LIM RESTful web services as a back-end to access LIM data through MIM queries and data requests. The following diagram shows the flow between the components for clients requesting data through the LIM Web Services Adapter.



Flow

- The desktop client, (MATLAB), retrieves the list of functions from WSDL. This occurs once per web service adapter update to get any newly added functionality
- Authentication token is retrieved. This occurs once or more based on changes to the user’s password.
- Use authentication token to make data requests.

The Adapter serves as a SOAP interface to the existing LIM web services. Actual authentication as well as all data requests are done by the LIM web service. The adapter is the broker between the client and the backend.

The main differences between the web services provided by the adapter and RESTful services are summarized in the table below:

	LIM RESTful WS	LIM WS Adapter (SOAP)
Are function calls synchronous/asynchronous?	Asynchronous	Synchronous
Features included	Complete Set of Functions LIM provides through web Services.	Data Retrieval Functions Only
Provides WSDL	No	Yes
Best For When …	Application Development, Scalability	Client Applications that easily consume WSDL's (such as MATLAB)

Installation

The installation process involves deploying the server component and obtaining the XML Toolbox for MATLAB to easily convert data request results into MATLAB objects. See the Known Issues section of this document for resolutions to any issues that may prevent operations.

Installing Web Service

Prerequisites:

- Existing LIM Web Services Deployment
- Tomcat Server (version 6.0)
- Java 1.6 or later

- Setup Configuration Parameters for the Web Service.
  The adapter needs the URL of the LIM web services it is going to use as its backend. This is done by adding "Dlim.wsurl=$LIMWSURL" to the $JAVA_OPTS environment variable where $LIMWSURL denotes the URL for accessing LIM Web Services. For example, if the URL where web services are, is https://ws.lim.com, the option would be set by adding "-Dlim.wsurl=https://ws.lim.com" to JAVA_OPTS.

  If the application is being deployed into an existing LIM environment, you can add this configuration to the existing $JAVA_OPTS variable defined in .limrc file.

- Install Tomcat Server Components.
  The prerequisite jar files are located under the tomcat/lib directory. Copy the files in tomcat/lib into the lib directory under $CATALINA_HOME. For most installations, this will be $LIMUSER_HOME/appserver/lib.

- Restart Tomcat.
  Shutdown and startup the Tomcat process for the changes to take effect.

- Deploy the LIM Web Services Adapter WAR File.
  If you have your Tomcat server set up to automatically deploy applications, copy the LimWsAdapter.war file into the $CATALINA_HOME/webapps directory where $CATALINA_HOME is the base path of the Tomcat installation. You can also use the

Tomcat Web Application Manager to deploy the WAR file. See Tomcat documentation for more information.

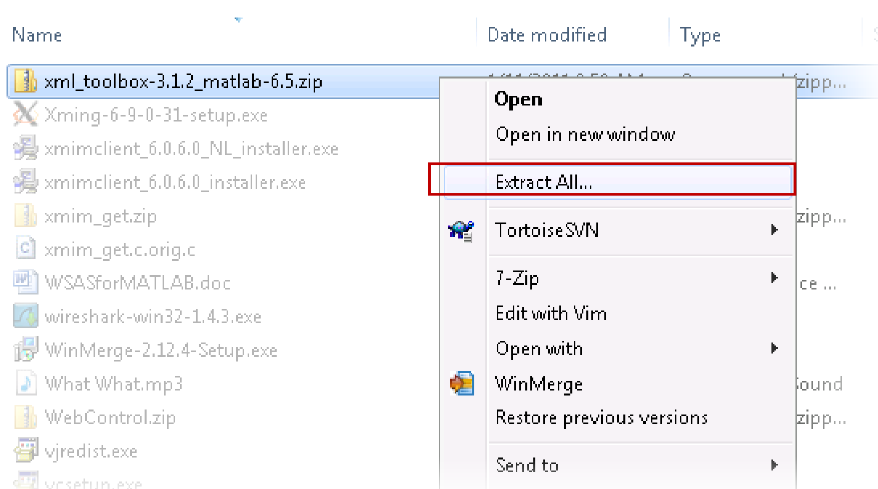
- Test Installation.
  To check whether deployment succeeded, access the following link using a web browser: $BASE_URL/LimWsAdapter/LimWs where $BASE_URL is the base address for the Tomcat server. For example, if the adapter was deployed Tomcat server with the base URL http://tomcat.mycompany.com:9090, the address to the test page is http://tomcat.mycompany.com:9090/LimWsAdapter/LimWs

Installing XML Toolbox for MATLAB

The XML Toolbox is a MATLAB add-on which provides the ability to easily convert XML to MATLAB data structures. It can be obtained from MATLAB Central File Exchange as a free download.

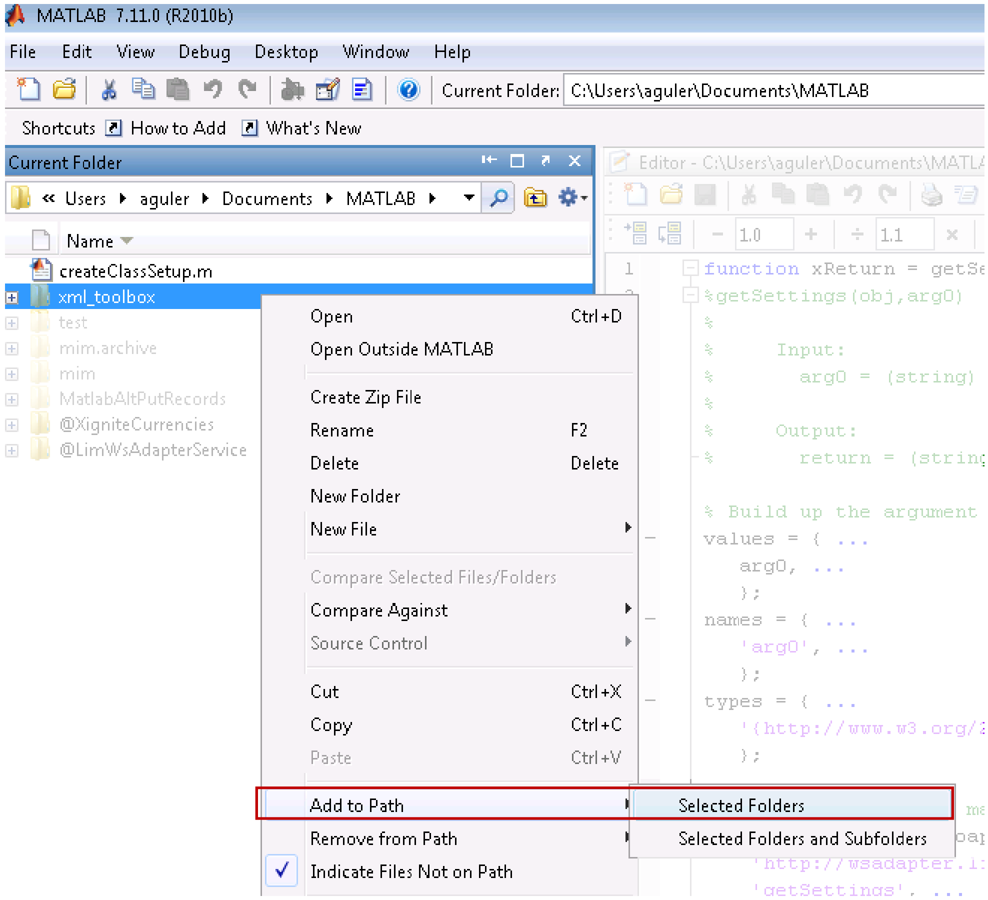
- Download the package from MATLAB Central File Exchange

- Right click on the xml_toolbox file and select “Extract All” to extract the contents of the package



After extracting the download contents, you should have a folder named *xml_toolbox* which includes the commands provided with the package and documentation.

Add the Folder to MATLAB Path



Refer to MATLAB website for further information about deploying the XML Toolbox.

Accessing LIM WS Adapter from MATLAB

You can access the web services provided by the LIM Adapter by either running createClassFromWsdl and using the MATLAB class and methods it generates or directly calling functions by using MATLAB SOAP functions. The examples provided in this document will be based on using createClassFromWsdl function. For using MATLAB SOAP functions to access LIM Adapter functionality, please refer to MATLAB documentation.

As of the release of this document, there is a known issue with making SOAP requests from within MATLAB that prevents the functions from working. Please see the Known Issues for the workaround.

Use the createClassFromWsdl function to obtain the MATLAB object and methods that correspond to the LIM Ws Adapter:

```
>> createClassFromWsdl('http://mycompany.com:8080/LimWsAdapter/LimWs?wsdl')

Retrieving document at 'http://mycompany.com:8080/LimWsAdapter/LimWs?wsdl'

ans =

LimWsAdapterService
```

The function requires the user to specify the location of the web service's WSDL file. Note that the URL for the WSDL file location is the URL that is used to test the web service plus the string "?wsdl". When the function executes successfully, it will return the name of the object it created (LimWsAdapterService). It creates a folder with the same class name as the object (@LimWsAdapterService) which includes the methods available from the web service.

Once the class is successfully created, an object of that class needs to be created to be used in all methods of that class:

```
>> obj = LimWsAdapterService

    endpoint: 'http://mycompany.com:8080/LimWsAdapter/LimWs'

        wsdl: 'http://mycompany.com:8080/LimWsAdapter/LimWs?wsdl'
```

Each time a method of @LimWsAdapter is called, the variable *obj* will be used as the first parameter provided. In all examples in this document, whenever *obj* is specified a function parameter, it is in reference to this object.

See MATLAB help documentation for more information on createClassFromWsdl and creating an object of the newly generated class corresponding to the web service.

Functions

Create an Authentication Token

All LIM Web Service Adapter methods except for limHelp and getVersion 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 if the user credentials provided to getAuthToken can be used to successfully authenticate with the LIM web services.

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

Usage:
string token = getAuthToken(LimWsAdapterService obj, string username, string password)

Argument(s)	
obj	MATLAB object of the class LimWsAdapterService. Required for all method calls to the LIM Web Services Adapter.
username	Login used to access LIM web services. Consult your IT personnel or LIM 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 area 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(obj, 'limuser@mycompany.com','mypassword')
```

Retrieving Data

Executing MIM Queries

Description:

Use the executeQuery method to run MIM queries.

Usage

String xmlRes = executeQuery(LimWsAdapterService obj, String authToken, String mimQuery)

Argument(s)	
obj	MATLAB object of the class LimWsAdapterService. Required for all method calls to the LIM Web Services Adapter.
authToken	used to authenticate with LIM web services. See help for authToken method to find out more about obtaining an authentication token.
mimQuery	character string containing the MIM query to be executed.

Return Value:

An XML string which can be parsed with xml_parse command (XML Toolbox) to create an object 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.

Example(s):

The following is an example of executing a simple MIM query:

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

xmlRslt =

<?xml version="1.0" encoding="UTF-8" standalone="yes"?><root idx="1" ...

>> res = xml_parse(xmlRslt)


res =

headings: {'1'}

values: [5x1 double]

dateList: {5x1 cell}
```

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:

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

>> res = xml_parse(xmlRslt)


res =

headings: {'CloseOfNG'}

values: [5x1 double]

dateList: {5x1 cell}
```

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:

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

>> res = xml_parse(xmlRslt)

res =

1x4 struct array with fields:

headings

values

dateList


>> res(1)

ans =

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

values: [22x4 double]

dateList: {22x1 cell}


>> res(2)

ans =

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

values: [22x4 double]

dateList: {22x1 cell}
```

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

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

>> res = xml_parse(xmlRslt)

res =

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

values: [602x4 double]

dateList: {602x1 cell}
```

getRecords Function

Description:

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

Usage:

string xmlResult = getRecords(LimWsAdapter obj, string authToken, string relList, string colList, string units, string fromDateTime, string toDateTime)

Argument(s)	
obj	MATLAB object of the class LimWsAdapterService - Required for all calls - See MATLAB documentation on using web services for more information.
authToken	used to authenticate with LIM 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	comma-separated list of data units. 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:

An XML string that contains the results of the data request. The resultscan be parsed with the xml_parse command (XML Toolbox) to create an object 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.

Example(s):

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

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

>> res1 = xml_parse(xmlRslt)

res1 =

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

values: [262x4 double]

dateList: {262x1 cell}
```

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(LimWsAdapterService obj, string token, string format)

Argument(s)	
obj	MATLAB object of the class LimWsAdapterService - Required for all calls - See MATLAB documentation on using web services for more information.
authToken	used to authenticate with LIM 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(obj, token, 'yyyy-MM-dd HH:mm');

>> format = getSettings(obj, token)


format =

User: limuser@mycompany.com Date-Time format: yyyy-MM-dd HH:mm Result Format:
MATLAB_FORMAT

>> token = setDateTimeFormat(obj, token, 'yyyy-MM-dd HH:mm');>> xmlRslt =
getRecords(obj,token,'IBM','Low','HOURS','2010-04-09 12:00', '2010-04-09 15:00');>> res =
xml_parse(xmlRslt)

res =

headings: {'Low of IBM'}

values: [4x1 double]

dateList: {4x1 cell}
```

```
>> 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
```

Options for 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 MIM Data and Development Guide for more information.

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

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

Usage
string token = setMissingDataFillOption(LimWsAdapter obj, string authToken, string fillOption)

Argument(s)	
obj	MATLAB object of the class LimWsAdapterService - Required for all calls - See MATLAB documentation on using web services for more information.
authToken	used to authenticate with LIM 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:

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

>> res = xml_parse(xmlRes);

>> 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'


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

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

>> res = xml_parse(xmlRes);

>> 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(LimWsAdapter obj, string authToken, string option)

Argument(s)	
obj	MATLAB object of the class LimWsAdapterService - Required for all calls - See MATLAB documentation on using web services for more information.
authToken	used to authenticate with LIM 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(obj, token, 'SKIP_ALL_NAN');

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

>> res = xml_parse(xmlRes)

res =

headings: {'Close of IBM'}

values: [2x1 double]

dateList: {2x1 cell}


>> 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(obj, token, 'SKIP_NONE');

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

>> res = xml_parse(xmlRes)

res =

headings: {'Close of IBM'}

values: [3x1 double]

dateList: {3x1 cell}


>> 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'
```

Result Set Formatting

Description:
The result set returned for data requests (GetRecords and ExecuteQuery) are formatted to be consumed by the xml_parse MATLAB method. If you prefer viewing the results in the basic LIM format, use the setDefaultFormat method.

Usage:
string token = setDefaultFormat(LimWsAdapterService obj, string token)
string token = setMatlabFormat(LimWsAdapterService obj, string token)

Argument(s)	
obj	MATLAB object of the class LimWsAdapterService - Required for all calls - See MATLAB documentation on using web services for more information.
authToken	used to authenticate with LIM 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:
character string which is the authentication token with the updated result format setting.

Example(s):
The following examples demonstrate modifying the XML format used to return results. To set the result format to the default LIM XML:

```
>> token = setDefaultFormat(obj, token);

>> format = getSettings(obj, token)


format =

User: limuser@mycompany.com Date-Time format: yyyy-MM-dd HH:mm Result Format: LIM_FORMAT
```

To set the result format to MATLAB, use the setMatlabFormat method call:

```
>> token = setMatlabFormat(obj, token);
```

Utility Functions

LIM Web Services Adapter Help

Description

Use the limHelp command to obtain information about the functionality provided by the LIM Web Services Adapter.

Usage:

limHelp(LimWsAdapterService obj,string cmdName)

Argument(s)	
obj	MATLAB object of the class LimWsAdapterService - Required for all calls - See MATLAB documentation on using web services for more information.
cmdName	name of the command to get help on. The name can either be the filename (e.g. getRecords.m) or just the name itself (getRecords)

Return Value:

Text containing information about the command which the user inquired about

Example(s):

The following example demonstrates obtaining help for the getVersion command:

```
>> limHelp(obj,'getVersion')

ans =

View Version of LIM WS Adapter
```

```
Description:
Use the getVersion command to obtain the version of this software:
etc……
```

View Current Settings

Description:
Use the getSettings call to view the current settings.

Usage:
getSettings(LimWsAdapterService obj, string authToken)

Argument(s)	
obj	MATLAB object of the class LimWsAdapterService - Required for all calls - See MATLAB documentation on using web services for more information.
authToken	used to authenticate with LIM 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:

```
>> getSettings(obj, token)

ans =

User: limuser@mycompany.com
Date-Time format: yyyy-MM-dd
```

```
Result Format: MATLAB_FORMAT

Missing Data Option: FILL_NAN

Skip NaN Option: SKIP_NONE
```

View Version of LIM WS Adapter

Description:

Use the getVersion command to obtain the version of this software:

Usage:

getVersion(LimWsAdapterService obj)

Argument(s)	
obj	MATLAB object of the class LimWsAdapterService - Required for all calls - See MATLAB documentation on using web services for more information.

Return Value:

Version of LIM Web Services Adapter being used.

Example(s):

The following example shows how to obtain version information:

```
>> getVersion(obj)

ans =

Version: 1.0 (build 1.0.3)
```

Known Issues

After the user runs creteClassFromWsdl and successfully creates the LimWsAdapterService MATLAB class, the functions generated will most likely not work. This is due to an issue with MATLAB and how it constructs SOAP messages to communicate. SOAP message that MATLAB constructs do not validate against the XML schema published by the web service. This results in the web service being called with empty parameters. The workaround is to edit each function and change the SOAP binding style from 'document' to 'rpc'. The following example is specific to getAuthToken but implementing the workaround should be similar for all other functions.

- Open the file getAuthToken.m using MATLAB or a text editor.
- Find the following line:

```
values,names,types,'document');
```

- Modify the line to:

```
values,names,types,'rpc');
```

- Save the file.

Appendix A MATLAB Result Format XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xs:element name="root">
<xs:complexType>
<xs:choice maxOccurs="unbounded">
<xs:element ref="headings" />
<xs:element ref="dateList" />
<xs:element ref="values" />
</xs:choice>
<xs:attribute name="idx" use="required" type="xs:integer" />
<xs:attribute name="size" use="required" />
<xs:attribute name="type" use="required" type="xs:NCName" />
<xs:attribute name="xml_tb_version" use="required" type="xs:decimal" />
</xs:complexType>
</xs:element>
<xs:element name="headings">
<xs:complexType>
<xs:sequence>
<xs:element maxOccurs="unbounded" ref="item" />
</xs:sequence>
<xs:attribute name="idx" use="required" type="xs:integer" />
<xs:attribute name="size" use="required" />
<xs:attribute name="type" use="required" type="xs:NCName" />
</xs:complexType>
</xs:element>
<xs:element name="dateList">
<xs:complexType>
<xs:sequence>
<xs:element maxOccurs="unbounded" ref="item" />
```

```
</xs:sequence>
<xs:attribute name="idx" use="required" type="xs:integer" />
<xs:attribute name="size" use="required" />
<xs:attribute name="type" use="required" type="xs:NCName" />
</xs:complexType>
</xs:element>
<xs:element name="values">
<xs:complexType mixed="true">
<xs:attribute name="idx" use="required" type="xs:int" />
<xs:attribute name="size" use="required" />
<xs:attribute name="type" use="required" type="xs:NCName" />
</xs:complexType>
</xs:element>
<!-- The definition for values below is if you want to return value as cells ... -->
<!-- <xs:element name="values"> -->
<!-- <xs:complexType> -->
<!-- <xs:sequence> -->
<!-- <xs:element maxOccurs="unbounded" ref="item"/> -->
<!-- </xs:sequence> -->
<!-- <xs:attribute name="idx" use="required" type="xs:integer"/> -->
<!-- <xs:attribute name="size" use="required"/> -->
<!-- <xs:attribute name="type" use="required" type="xs:NCName"/> -->
<!-- </xs:complexType> -->
<!-- </xs:element> -->
<xs:element name="item">
<xs:complexType mixed="true">
<xs:attribute name="idx" use="required" type="xs:integer" />
<xs:attribute name="size" use="required" />
<xs:attribute name="type" use="required" type="xs:NCName" />
</xs:complexType>
</xs:element>
</xs:schema>
```

Appendix B LIM Result Set XML Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">


  <xs:element name="DataRequestResponse" type="dataRequestResponse"/>


  <xs:element name="Report" type="report"/>


  <xs:element name="ReportBlock" type="reportBlock"/>


  <xs:element name="ReportSummaryBlock" type="reportSummaryBlock"/>


  <xs:complexType name="dataRequestResponse">

    <xs:sequence>

      <xs:element name="Reports" type="report" minOccurs="0" maxOccurs="unbounded"/>

    </xs:sequence>

    <xs:attribute name="endTime" type="xs:dateTime"/>

    <xs:attribute name="id" type="xs:long" use="required"/>

    <xs:attribute name="startTime" type="xs:dateTime"/>

    <xs:attribute name="status" type="Status" use="required"/>

    <xs:attribute name="statusMsg" type="xs:string" use="required"/>

  </xs:complexType>


  <xs:complexType name="report">

    <xs:sequence>

      <xs:element name="ReportBlocks" type="reportBlock" minOccurs="0"
maxOccurs="unbounded"/>

    </xs:sequence>

    <xs:attribute name="title" type="xs:string"/>

  </xs:complexType>
```

```
<xs:complexType name="reportBlock">
  <xs:sequence>
    <xs:element name="RowDates" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="RowTimes" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="AttributeHeadings" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="ColumnHeadingIndices" type="xs:int" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="ColumnHeadings" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="ColumnPrecisions" type="xs:int" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="SummaryReport" type="reportSummaryBlock" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="Values" type="xs:double" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="dateFormat" type="xs:string"/>
  <xs:attribute name="timeFormat" type="xs:string"/>
  <xs:attribute name="numAttributes" type="xs:int" use="required"/>
  <xs:attribute name="numCols" type="xs:int" use="required"/>
  <xs:attribute name="numRows" type="xs:int" use="required"/>
</xs:complexType>


<xs:complexType name="reportSummaryBlock">
  <xs:sequence/>
  <xs:attribute name="avg" type="xs:double" use="required"/>
  <xs:attribute name="avgNegative" type="xs:double" use="required"/>
  <xs:attribute name="avgPositive" type="xs:double" use="required"/>
  <xs:attribute name="highest" type="xs:double" use="required"/>
  <xs:attribute name="lowest" type="xs:double" use="required"/>
  <xs:attribute name="pctNegative" type="xs:double" use="required"/>
  <xs:attribute name="pctPositive" type="xs:double" use="required"/>
  <xs:attribute name="stdDeviation" type="xs:double" use="required"/>
  <xs:attribute name="sum" type="xs:double" use="required"/>
  <xs:attribute name="variance" type="xs:double" use="required"/>
```

```
    <xs:attribute name="zStat" type="xs:double" use="required"/>
  </xs:complexType>

  <xs:simpleType name="Status">
    <xs:restriction base="xs:int">
      <xs:enumeration value="100"/>
      <xs:enumeration value="110"/>
      <xs:enumeration value="120"/>
      <xs:enumeration value="130"/>
      <xs:enumeration value="140"/>
      <xs:enumeration value="200"/>
      <xs:enumeration value="300"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```