

User Guide

LIM Web Service Data Loader

Version 2



Table of Contents

Web Service Data Loader Overview.....	3
Architecture.....	3
Parsers.....	3
Default Parser Data Format.....	4
Submitting Data.....	5
Checking Job Status.....	6
Retrieving a Job Report.....	7
Job Status codes.....	8
Audit File Replication.....	8
Overview.....	8
Architecture.....	8
Audit File Format.....	9
Section 1 [config].....	10
Section 2 [metadata-rel].....	10
Section 3 [metadata-col].....	11
Section 4 [metadata-relcol].....	11
Section 5 [messages].....	11
Section 6 [data].....	12
Sample audit file:.....	12
Replication of data via BMIM.....	13

Web Service Data Loader Overview

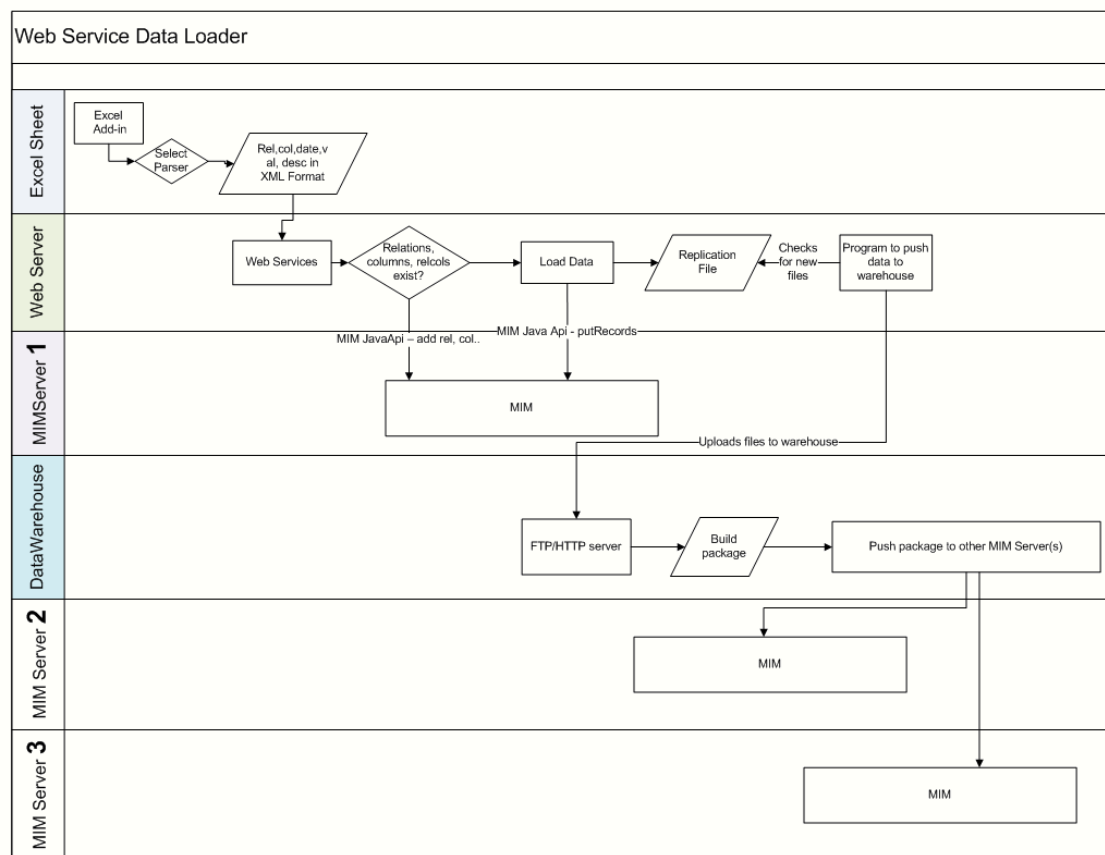
The web service data loader is intended to meet the needs of our customers to load data into the MIM with minimal interaction and understanding of the low level API calls needed to achieve loading data. The software will be a RESTfull based web service running inside apache tomcat. The user interface will be within Excel and load data that is in a formatted excel sheet using a selected parser.

- ▶ The Data loader is only compatible with Web Services Version 2.0.3 to Version 2.0.10.



Architecture

The diagram below provides the details of the data loading process from the Excel sheet level to the data in the MIM Server.



Parsers

The Upload services uses "parsers" to process data submitted to the service. A parser is responsible for converting an incoming data stream into a series of (relation, column, transaction date, value) records that can then be loaded into the MIM.

This document describes how to use the Upload service with its built-in "DefaultParser". This parser accepts data in XML format and processes it. This is the parser that interprets data sent from Excel, however it can be use to load data from any source so long as it is in the format described here.

Default Parser Data Format

The Upload service's DefaultParser accepts row and column data encoded in XML. Each row contains 4 columns that describe the MIM data to load: symbol (relation), column, transaction date, and the value. An optional fifth column may contain a text description of the symbol.

Column	Contents
1	The symbol (relation) name with a full path. Example: TopRelation:Category:SYMBOL_NAME
2	The column in which to store the data.
3	The transaction date in Excel numeric format (described below.)
4	The data to store.
5	(Optional) A text description of the symbol.

As an example, here are three rows with data in Excel:

	A	B	C	D	E
1	TopRelation:Category:NEW_RELATION_01	TopColumn:Price:Close	2010-01-01 00:00	100	Description of NEW_RELATION_01
2	TopRelation:Category:NEW_RELATION_02	TopColumn:Price:Close	2010-01-02 00:00	200	Description of NEW_RELATION_02
3	TopRelation:Category:NEW_RELATION_03	TopColumn:Price:Close	2010-01-03 00:00	300	

- ▶ The data from Excel converted to the DefaultParser's XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<ExcelData>
  <Rows>
    <Row num="1">
      <Cols>
        <Col num="1">TopRelation:Category:NEW_RELATION_01</Col>
        <Col num="2">TopColumn:Price:Close</Col>
        <Col num="3">40179</Col>
        <Col num="4">100</Col>
        <Col num="5">Description of NEW_RELATION_01</Col>
      </Cols>
    </Row>
    <Row num="2">
      <Cols>
        <Col num="1">TopRelation:Category:NEW_RELATION_02</Col>
        <Col num="2">TopColumn:Price:Close</Col>
        <Col num="3">40180</Col>
        <Col num="4">200</Col>
        <Col num="5">Description of NEW_RELATION_02</Col>
      </Cols>
    </Row>
    <Row num="3">
      <Cols>
        <Col num="1">TopRelation:Category:NEW_RELATION_03</Col>
        <Col num="2">TopColumn:Price:Close</Col>
        <Col num="3">40181</Col>
        <Col num="4">300</Col>
        <Col num="5" />
      </Cols>
  </Rows>
</ExcelData>
```

```

</Row>
</Rows>
</ExcelData>

```

Two things to notice:

- ▶ Rows 1 and 2 include the optional symbol description in Column number 5. Row number 3 does not.
- ▶ Column number 3 shows the dates in the spreadsheet converted to Excel's numeric date format. See [this page](#) for details on Excel's date format and how to convert a date to it.

Submitting Data

The Upload service accepts data via HTTP POST at this URL:

POST `http://webservice-host:port/rs/upload`

The service expects two parameters to be passed in the POST in addition to the data itself.

Parameter	Description
username	The name of the user uploading the data.
parsername	The name of the parser that will interpret the data. Example: DefaultParser

Both parameters are required.

- ▶ Example:

`http://webservice-host:port/rs/upload?username=<username>&parsername=DefaultParser`

When POSTing data the HTTP 'Content-Type' header must be set to "text/xml". If possible the 'Content-Length' header should be set to the number of bytes of XML being sent. If Content-Length is set the Upload service will report an error if it does not receive exactly that many bytes. This provides an additional check that the data was received successfully.

*For a basic overview of HTTP POST in a Java context, see [this page](#).

The Upload service processes uploaded data asynchronously. As soon as the service has received all the XML data the HTTP POST call will complete, and the server will return a response that indicates if the data was accepted successfully.

- ▶ The response format is:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <response status="Job submitted successfully." jobID="35" intStatus="100" />

```

Clients should check that the status code is 100. Any other status code should be considered an error.

The jobID represents the data submitted to the system. If the system is busy uploaded data may not be processed for some time. Clients can retrieve their job list (as described below) and use this jobID to check the status of their job(s).

Checking Job Status

The Upload service maintains lists of job submitted by all users and tracks the status of each job. Clients can retrieve their job list at any time to check if their uploaded data has been processed successfully.

Use HTTP GET to retrieve the job list:

GET <http://webservice-host:port/rs/upload/jobs>

The service expects a username parameter so it can retrieve the correct job list.

Parameter	Description
username	The service will return a list of this user's jobs.

The list includes all of the jobs the user has submitted and the status of each (not yet processed, processing, failed, successful, etc.)

An example of the job list XML:

```
<?xml version="1.0" encoding="utf-8"?>
<UploadedJobs>
  <jobs>
    <job>
      <endtime>2010-03-04T14:25:20.294-06:00</endtime>
      <errorCount>1</errorCount>
      <id>42</id>
      <infoCount>0</infoCount>
      <starttime>2010-03-04T14:25:19.913-06:00</starttime>
      <jobstatus>
        <code>302</code>
        <message>Job failed.</message>
      </jobstatus>
      <warnCount>0</warnCount>
    </job>
    <job>
      <endtime>2010-03-04T14:26:03.102-06:00</endtime>
      <errorCount>1</errorCount>
      <id>43</id>
      <infoCount>0</infoCount>
      <starttime>2010-03-04T14:26:02.839-06:00</starttime>
      <jobstatus>
        <code>300</code>
        <message>Job completed successfully.</message>
      </jobstatus>
      <warnCount>0</warnCount>
    </job>
  </jobs>
</UploadedJobs>
```

```
</jobs>
</UploadedJobs>
```

Clients can parse this data to determine if their submitted jobs are done and whether they succeeded.

Retrieving a Job Report

The Upload service also maintains a detailed job report for each job. The report contains status messages generated while loading the data. The parser that does the work determines which messages are actually generated, so the messages can vary per parser. In general they should be treated as "informational" and the exact text of the messages could change in the future. Note that these messages are the same that appear in the Excel add-in GUI when the user double-clicks on a job.

There are 3 types of messages: INFO, WARN, and ERROR. The parser determines how to categorize each message it generates, however the categories are meant only as guidelines and have no bearing on whether a job succeeded or failed. In other words, a job that generates ERROR messages may still succeed overall. Use the job status (as described above) to determine the success or failure of a job.

Use HTTP GET to retrieve a job report:

GET http://webservice-host:port/rs/upload/jobreport/<job_id>

Where <job_id> is the id of the job you are interested in. No query parameters are required to retrieve a job report.

Sample job report XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <JobResultImpl>
    <Messages>
      <msgs type="INFO">
        <text>Loaded 4 values for 4 relcols.</text>
      </msgs>
    </Messages>
  </JobResultImpl>
```

Job Status codes

STATUS	STATUS ID	STATUS MESSAGE
<i>SUBMITTED_NEW</i>	100	Job submitted successfully
<i>SUBMITTED_WAITING</i>	102	Job is awaiting execution
<i>SUBMITTED_TOO_BUSY</i>	103	Server was busy to run the job. Will try to run later
<i>RUNNING_INIT</i>	202	Job is initializing
<i>RUNNING_INIT_OK</i>	203	Job initialized successfully
<i>RUNNING_PARSING</i>	200	Parsing data.
<i>RUNNING_LOADING</i>	201	Loading data.
<i>FINISHED_OK</i>	300	Job completed successfully
<i>FINISHED_WARNING</i>	301	Job completed with warning(s)
<i>FINISHED_FAILURE</i>	302	Job failed
<i>FINISHED_CANCELLED</i>	303	The job was cancelled
<i>FINISHED_NO_START</i>	304	A server error prevented the job from running
<i>UNKNOWN</i>	-1	Job status could not be determined

Audit File Replication

Overview

The objective of this project is to facilitate replication of custom data loaded by the users of the LIM web service data loader. The web service process produces an audit file to track the changes users are making to the MIM via the loader. This file contains the needed information to reproduce those same changes in another MIM server. This will allow our customers to make changes to one server and have those same changes applied to their other MIM servers.

Architecture

An audit file is produced every time the web service loads data into the MIM. These audit files will be periodically transmitted to the data warehouse. The data warehouse processes the audit files and converts them to update packages. These packages are then sent to the customer's other MIM servers through the normal warehouse distribution system.

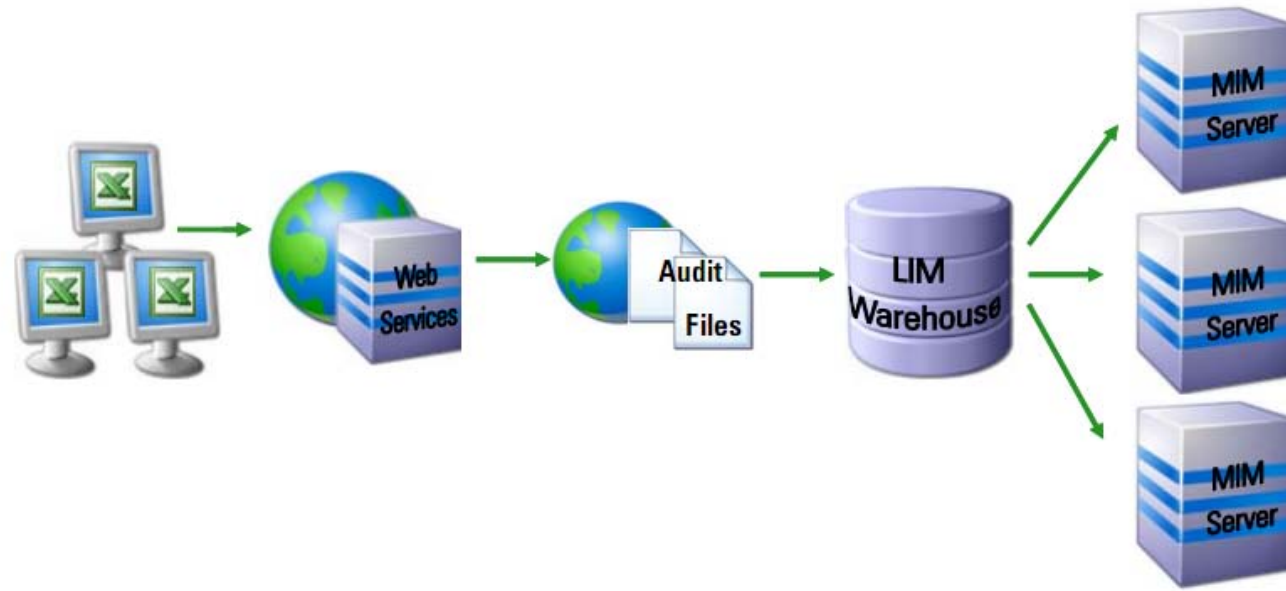


fig 1

Audit File Format

The loader produces an audit file containing the information the user is requesting to load into the MIM and other information pertaining to the load. This file is placed into a spool area designated in the upload service configuration.

The file contains the following.

- ▶ Configuration information : user name, database name, transaction time, etc.
- ▶ Metadata for relations, columns and relcols
- ▶ The data payload
- ▶ Any status messages output during the load

The complete details are as follows.

The file contains six sections denoted with [], with one of the following names: config, metadata-rel, metadata-col, metadata-relcol, messages, and data. Comment rows start with a ";" and can be located anywhere in the file.

Each row begins with a unique sequence number that indicates the actual order in which the operations were performed. For most purposes it can be ignored but it is included in case it is necessary to know the order each action was executed. The exception is the [data] section, which does not include sequence numbers. Data is always loaded after all metadata and is loaded in the order as shown in the file.

Section 1 [config]

This section contains key value pairs containing config information and other processing information.

The following are the defined keys for the [config] section.

Element	Description
server_id	The unique identifier of the server that generated the audit file.
uuid	A globally unique identifier for this audit file
username	Name of the user that loaded the data
database	Name of the MIM database the data was loaded into. It does not include path information. Example: "data.cust"
host	Hostname of the MIM the data was loaded into
port	The MIM server port
submitdt	Time and date the data was sent to the upload service. The format is: yyyy-MM-ddTHH:mm:ss[+,-]HHmm
loaddt	Time and date the submitted data was loaded into the MIM. The format is the same as 'submitdt'.
tzname	The "long" name of the time zone. Example "America/Los Angeles".
parser	The name of the parser used when submitting the data.
datadf	(Optional) The date format used in the [data] section. The default is "yyyyMMdd".
datatf	(Optional) The time format used in the [data] section. The default is "HHmmss".

Section 2 [metadata-rel]

Each line consists of comma-separated values that describe a single relation.

The line format is: seq,action,relation name,relation path,type,description,trading days,start time,end time,[roll date], [roll policy]

Element	Description
seq	Unique integer that indicates the order actions were taken
action	C = create, U = update, D = delete
relation name	The name of the relation
relation path	The full path of the relation
type	category, normal, futures, or futures_contract
description	Text description of the relation
trading days	Any combination of M,T,W,H,F,S,D i.e., 5 day = MTWHF
start time	Start trading time in HH:mm format
end time	End trading time in HH:mm format
roll date	("futures" relations only) Future roll date
roll policy	("futures" relations only) Future roll policy

Example:

5,U,TRACK_WHEAT,TopRelation:LIM:Wheat,futures,"",MTWHF,00:00,23:59,"Last Data Day","Actual Prices",

Section 3 [metadata-col]

Each line consists of comma-separated values that describe a single column. The line format is:

seq,action,column name,column path,type,description

Element	Description
seq	Unique integer that indicates the order actions were taken
action	C = create, U = update, D = delete
column name	The name of the column
column path	The full path of the column
type	category or normal
description	Text description of the column

Example:

1,U,BidFlat,TopColumn:Price,normal,""

Section 4 [metadata-relcol]

Each line consists of comma-separated values that describe a single relcol.

The line format is: seq,action,relation,column,aggregation,type,object type

Element	Description
seq	Unique integer that indicates the order actions were taken
action	C = create, U = update, D = delete
aggregation	open, high, low, close, sum, average
type	base, sparse
object type	integer, float, double

Example:

7,C,TopRelation:LIM:Wheat:TRACK_WHEAT:TRACK_WHEAT_1900F,TopColumn:Price:BidFlat,
close,base,double

Section 5 [messages]

This section contains any error, warning or informational messages. These messages are generated when the data is initially parsed and loaded. They are not relevant to the data warehouse or the replication process, so they may be ignored by the warehouse parsing process.

Example:

2,warn,Failed to parse row # 7 : Parent future must be created before adding contract
[MARK_FUTURE_1900F]

Section 6 [data]

Each line consists of comma-separated values that describe a single data point.

relation,column,date,[time],value

Element	Description
relation	The relation with its full path
column	The column with its full path
date	The transaction date
time	(Optional) The transaction time (for intraday data)
value	The value

Example:

TopRelation:LIM:Wheat:FOB_WHEAT:FOB_WHEAT_2009N,TopColumn:Price:BidBasis,20100205,1.000000

Sample audit file:

[config]

```
server_id=unique_server_id
uuid=29ac8211-caa7-4fe3-8ade-e4c531d79057
username=cingarfield
database=data.cust
host=aussm40-z47.lim.com
port=6400
submitdt=2010-03-01T14:21:59-0600
loaddt=2010-03-01T14:21:59-0600
tzname=America/Los_Angeles
parser=Default Parser
```

[metadata-rel]

```
2,U,Test,TopRelation,category,""
3,U,LIM,TopRelation:Test,category,""
4,U,Wheat,TopRelation:Test:LIM,category,""
5,U,TRACK_WHEAT,TopRelation:LIM:Wheat,futures,"",MTWHF,00:00,23:59,"Last Data Day","Actual Prices",
```

[metadata-col]

```
0,U,Price,TopColumn,category,""
1,U,BidFlat,TopColumn:Price,normal,""
```

[metadata-relcol]

```
7,C,TopRelation:LIM:Wheat:TRACK_WHEAT:TRACK_WHEAT_1900F,TopColumn:Price:BidFlat,close,base,double
9,C,TopRelation:LIM:Wheat:TRACK_WHEAT:TRACK_WHEAT_1900F,TopColumn:Price:AskFlat,close
```

,base,double

[messages]

9,info,An info message

10,warn,A warning message that, contains a comma

11,error,ERROR MESSAGE

[data]

;Here is all the data for this load

TopRelation:LIM:Wheat:TRACK_WHEAT:TRACK_WHEAT_1900F,TopColumn:Price:BidFlat,20100301
,133.000000

TopRelation:LIM:Wheat:TRACK_WHEAT:TRACK_WHEAT_1900F,TopColumn:Price:AskFlat,2010030
1,200.000000

Replication of data via BMIM

Assumptions for producing BMIM from the Audit file

- ▶ Action "C" equates to a BMIM add command
- ▶ Action "U" equates to a BMIM modify command
- ▶ Action "D" equates to a BMIM delete command
- ▶ Actions will be logged in correct sequential order
- ▶ Field "database" in the config section, is the simple name of the DB no path etc..
- ▶ If the symbol is not defined in section metadata-rel but used elsewhere, it is assumed that the symbol exists and no action to add is required.
- ▶ If the column is not defined in section metadata-col but used elsewhere, it is assumed that the column exists and no action to ad is required.
- ▶ All data lines may be written to a PRN file as is.
- ▶ No MIM db partitioning (definition files is required)
- ▶ The sequence number can be used to determine the order in which metadata was added, modified, etc., however this number has no meaning to BMIM.