The MIM and Real-Time Data

12301 Research Blvd.
Building IV, Suite 410
Austin, TX 78759

U.S. Help Desk Phone: +1-800-546-9646 (or direct +1-512-697-3000), select ext. 3400
U.K. Help Desk Free Phone: 0800 032 6063
Europe Help Desk Phone: +44 20 7190 2947
Help Desk Email: support@lim.com
+1-512-697-3001 (Fax)

Part Number: 083_97
Date: March 6, 2008

Logical Information Machines, Inc.
120 North LaSalle Street
Suite 2150
Chicago, IL 60602

(312) 456-3000

Table of Contents

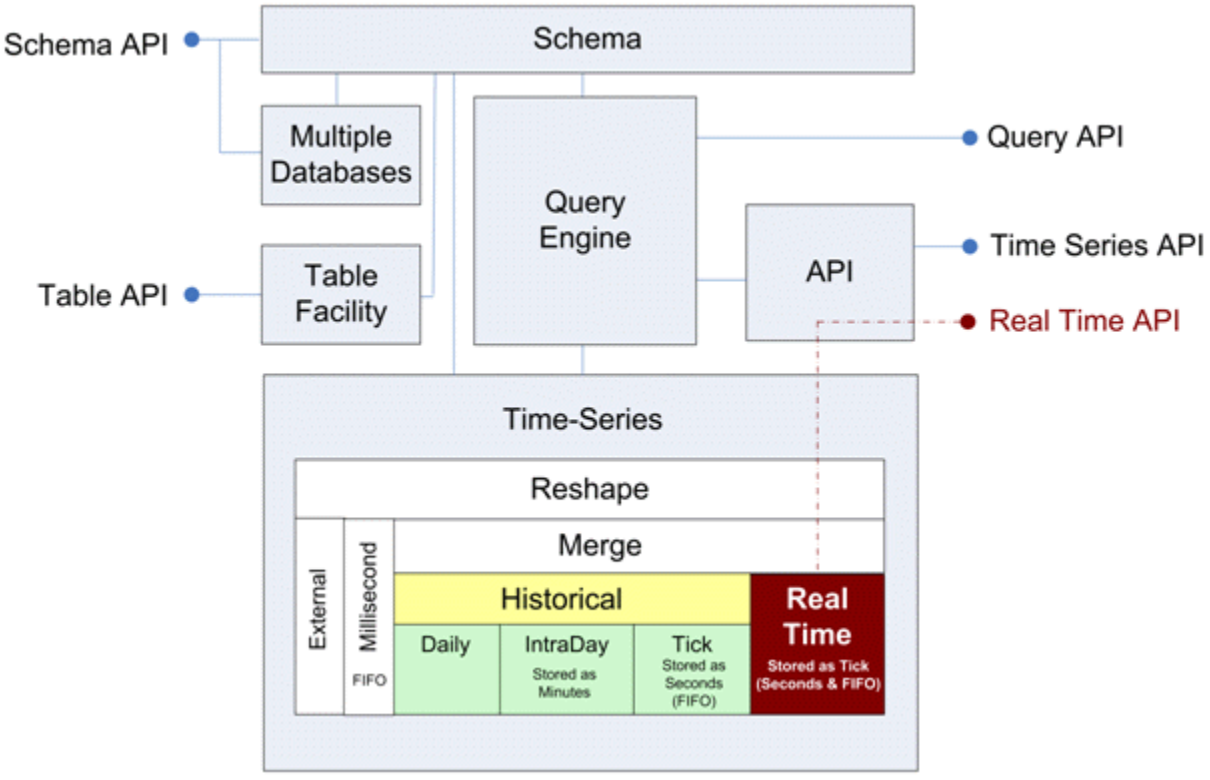
The MIM & Real-Time Data

How to Purchase

All MIM server installations can have real-time functionality enabled by purchasing a MIM Prime license. The ability to handle real-time data is built-in into every MIM system. To purchase a MIM Prime license contact a LIM Sales Representative at sales@lim.com.

How the MIM Stores Data

LIM specializes in database management of real-time data for the commodity, financial and energy markets. LIM uses the Market Information Machine (MIM), a server and database solution to store real-time data alongside historical data. The MIM database historical store is split into 2 physical databases; the primary historical database and the real-time (RT) database. The primary historical database is designed to maximize fast retrieval time, whereas the real time is optimized for fast storage time.



Folding Data

It is required that at some point the real-time (tick) database is either folded or cleared to maintain performance. Folding will convert the tick storage into historical storage. Clearing will remove the points with the expectation of replacement with another dataset. Folding can either be performed as a single operation or folding can be done incrementally (on a daily basis, series-by-series). Once the data is folded into the historical database it can be retrieved at a much faster rate.

Reading Historical and Tick Data

How the MIM Aggregates Data

The MIM stores real-time data in the manner it was transmitted. At the same time, the MIM provides derived analytics.

There are 2 physical databases for data but only a single logical database for data is presented via the API's to the client. This allows real-time data to be used by clients without additional coding when such data is present in the server deployment on site. Clients can control the use of the real-time database with API flags.

At run time, the server will check the real-time database for data on a daily basis. For example, if historical data is present to 11/10/2004, then that data will be used. When real-time data is stored for 11/11/2004, the end data for the time series is immediately extended to 11/11/2004. The client will see the data in the daily 11/11/2004 values as well as intraday data.

Historical Database

For every relation and column the MIM can produce:

- Daily Values: Daily values are stored and used to aggregate weekly, monthly, quarterly and yearly values.
- Intraday Values: Intraday values are stored as minute values and may be used to aggregate hourly values.
- Tick Values: Tick values are stored as second values and may be used to aggregate minute and hourly values.
- Millisecond Values: Millisecond values are only stored in the Historical Database and cannot be stored in the Real-Time Database.

Real-Time Database

For every relation and column the MIM can produce:

- Real-Time Values: Real-time data is stored as tick values (to the second) and may be used to aggregate daily, intraday or tick by tick values.

Real-time data cannot be stored as millisecond values.

Aggregation Rules

The *aggregation_rule* specifies how data of one granularity is to be converted to other granularities; for example, how daily data is combined to form weekly data. The possible aggregation rules are as follows:

- open (first)
- high (maximum)
- low (minimum)
- close (last)
- sum
- average

These rules are specified in the schema, as a property of a relation-column. The aggregation rule in the following case is the same as the column.

The default *aggregation_rule* is "close", which specifies that the last value for the period should be used. Similarly, "open" specifies to use the first value for the period, "high" the highest value, and "low" the lowest value. For example, "close" would be used for the Close column of a relation such that the weekly close would be the last value for the week (i.e., the close for the last day of the week). When "sum" or "average" is specified, the sum or average of the values for the period is used. For example, "sum" would be used for the Volume column of a relation such that the weekly volume would be the sum of the daily volumes for that week. XMIM will always aggregate on whatever data is available for the specified period. For example, if the aggregation period is weekly and on a given week, Friday is a holiday, the aggregation will take place over the data for Monday through Thursday so that the weekly close will be Thursday's close.

By default, for the open, high, low and close columns, the aggregation rules are the same as the column.

Aggregation Example:

The following aggregation example is going from minutes to hours for open, high, low, close. The data points we need to make a minute bar at 12 p.m. are those in the range: 11:01 a.m. to 12:00 p.m. This is the 1 hour bar derived from the data below.

9/14/2006, 12:00pm, 5.0300, 5.1100, 4.9700, 5.0400

Date,	Time,	Open,	High,	Low,	Close
09/14/2006,	10:54,	5.0900,	5.0900,	5.0800,	5.0800
09/14/2006,	10:55,	5.0500,	5.0500,	5.0300,	5.0500
09/14/2006,	10:56,	5.0500,	5.0600,	5.0300,	5.0300
09/14/2006,	10:57,	5.0200,	5.0400,	5.0200,	5.0300
09/14/2006,	10:58,	5.0500,	5.0600,	5.0300,	5.0400
09/14/2006,	10:59,	5.0300,	5.0500,	5.0300,	5.0500
09/14/2006,	11:00,	5.0400,	5.0400,	5.0200,	5.0200
09/14/2006,	**11:01,**	**5.0300,**	**5.0600,**	**5.0300,**	**5.0400**
09/14/2006,	**11:02,**	**5.0500,**	**5.0500,**	**5.0400,**	**5.0450**
09/14/2006,	**11:03,**	**5.0500,**	**5.0500,**	**5.0400,**	**5.0400**
09/14/2006,	**11:05,**	**5.0600,**	**5.0700,**	**5.0600,**	**5.0600**
09/14/2006,	**11:06,**	**5.0200,**	**5.0200,**	**5.0000,**	**5.0100**
09/14/2006,	**11:07,**	**5.0000,**	**5.0200,**	**4.9900,**	**5.0000**
09/14/2006,	**11:08,**	**5.0000,**	**5.0050,**	**4.9700,**	**4.9700**
09/14/2006,	**11:09,**	**5.0000,**	**5.0400,**	**5.0000,**	**5.0200**
09/14/2006,	**11:37,**	**5.0900,**	**5.1100,**	**5.0900,**	**5.1000**
09/14/2006,	**11:38,**	**5.0900,**	**5.1000,**	**5.0900,**	**5.1000**
09/14/2006,	**11:39,**	**5.1100,**	**5.1100,**	**5.1100,**	**5.1100**
09/14/2006,	**11:40,**	**5.0900,**	**5.1100,**	**5.0900,**	**5.1100**
09/14/2006,	**11:41,**	**5.1100,**	**5.1100,**	**5.1000,**	**5.1000**
09/14/2006,	**11:42,**	**5.1050,**	**5.1100,**	**5.1000,**	**5.1100**
09/14/2006,	**11:43,**	**5.1000,**	**5.1050,**	**5.1000,**	**5.1050**
09/14/2006,	**11:44,**	**5.1000,**	**5.1100,**	**5.1000,**	**5.1100**
09/14/2006,	**11:47,**	**5.0700,**	**5.1000,**	**5.0700,**	**5.0900**
09/14/2006,	**11:59,**	**5.0500,**	**5.0500,**	**5.0500,**	**5.0500**
09/14/2006,	**12:00,**	**5.0400,**	**5.0400,**	**5.0400,**	**5.0400**
09/14/2006,	12:01,	5.0600,	5.0600,	5.0600,	5.0600
09/14/2006,	12:02,	5.0500,	5.0500,	5.0400,	5.0400
09/14/2006,	12:03,	5.0600,	5.0600,	5.0600,	5.0600

The red numbers are the numbers where the aggregated value comes from.

The lines in bold are the lines that we use to calculate the 12 p.m. bar.

Likewise, the 1 p.m. hourly bar would be computed from the minute bars from 12:01 p.m. to 1:00 p.m.

The 2 p.m. hourly bar would be computed from the minute bars from 1:01 p.m. to 2:00 p.m.

Another example:

09/14/2006,10:57,	5.0200,	5.0400,	5.0200,	5.0300
09/14/2006,10:58,	5.0500,	5.0600,	5.0300,	5.0400
09/14/2006,10:59,	5.0300,	5.0500,	5.0300,	5.0500
09/14/2006,11:00,	5.0400,	5.0400,	5.0200,	5.0200
09/14/2006,11:03,	**5.0500,**	**5.0500,**	**5.0400,**	**5.0400**
09/14/2006,11:05,	**5.0600,**	**5.0700,**	**5.0600,**	**5.0600**
09/14/2006,11:06,	**5.0200,**	**5.0200,**	**5.0000,**	**5.0100**
09/14/2006,11:07,	**5.0000,**	**5.0200,**	**4.9900,**	**5.0000**
09/14/2006,11:08,	**5.0000,**	**5.0050,**	**4.9700,**	**4.9700**
09/14/2006,11:09,	**5.0000,**	**5.0400,**	**5.0000,**	**5.0200**
09/14/2006,11:37,	**5.0900,**	**5.1100,**	**5.0900,**	**5.1000**
09/14/2006,11:38,	**5.0900,**	**5.1000,**	**5.0900,**	**5.1000**
09/14/2006,11:39,	**5.1100,**	**5.1100,**	**5.1100,**	**5.1100**
09/14/2006,11:40,	**5.0900,**	**5.1100,**	**5.0900,**	**5.1100**
09/14/2006,11:41,	**5.1100,**	**5.1100,**	**5.1000,**	**5.1000**
09/14/2006,11:42,	**5.1050,**	**5.1100,**	**5.1000,**	**5.1100**
09/14/2006,11:43,	**5.1000,**	**5.1050,**	**5.1000,**	**5.1050**
09/14/2006,11:44,	**5.1000,**	**5.1100,**	**5.1000,**	**5.1100**
09/14/2006,11:47,	**5.0700,**	**5.1000,**	**5.0700,**	**5.0900**
09/14/2006,12:01,	5.0600,	5.0600,	5.0600,	5.0600
09/14/2006,12:02,	5.0500,	5.0500,	5.0400,	5.0400
09/14/2006,12:03,	5.0600,	5.0600,	5.0600,	5.0600

In this case there isn't a minute bar exactly 11:01 or at exactly 12:00. We must look at all the available data points in the range from 11:01 a.m. to 12:00 p.m.

The open will be the first one within the range. The close will be the last one within the range.

Data Extraction

There are two logical paths for extracting data from the server: `XmimGetRecords` and `XmimQueryExecute`. The general form of all API readers is: `XmimConnect`, `XmimGetRecords`, and `XmimDisconnect`. For best performance, connections should be pooled and reused whenever possible.

Using XmimGetRecords

`XmimGetRecords` is the most efficient API and reads directly from the stores. `XmimGetRecords` can extract daily, 5 minute bars, 1 minute bars, 5 second bars, etc. `XmimGetRecords` can be used to reshape the data into any data frequency. When the units are set to 1 second, `XmimGetRecords` will extract data from the real-time database in FIFO (e.g., more than 1 tick per second). The program xmim_get is provided in binary and source form. This program is a good example of using the `XmimGetRecords` interface.

Using XmimQueryExecute

`XmimGetRecords` cannot provide derived functions such as an average of values. To access derived values, the query engine is accessed via `XmimQueryExecute` (user will need MIM server version 4.6 or greater). The query execute can produce derived studies at a frequency lower than 1 minute. For example, one can generate the 5 second average with the query engine.

Setting the Current Tick Usage Flag

The MIM server represents data as a single database to readers. The API's express interest in the real-time tick data by using flags to the API calls. By adding the flag `XMIM_APPEND_TO_ALL` (`XMIM_APPEND_TO_ALL` is the default) to data reading APIs any real-time data collected into the real-time database will be returned combined with historical data. In the MIM database, historical data is always returned before real tick data.

`XMIM_CURRENT_TICK_USAGE`, is used to indicate whether historic data only should be retrieved or whether data from the high-frequency real-time database should also be retrieved. The corresponding data type for this argument is:

```
typedef enum {
            XMIM_APPEND_TO_ALL,       /* Append current tick to historic          */
            XMIM_APPEND_TO_NONE,      /* Do not use current tick                  */
            XMIM_APPEND_TO_DAILY,     /* Append current tick to daily data        */
            XMIM_APPEND_TO_TICK       /* Append to intraday and real tick         */
} XmimCurrentTickUsage;
```

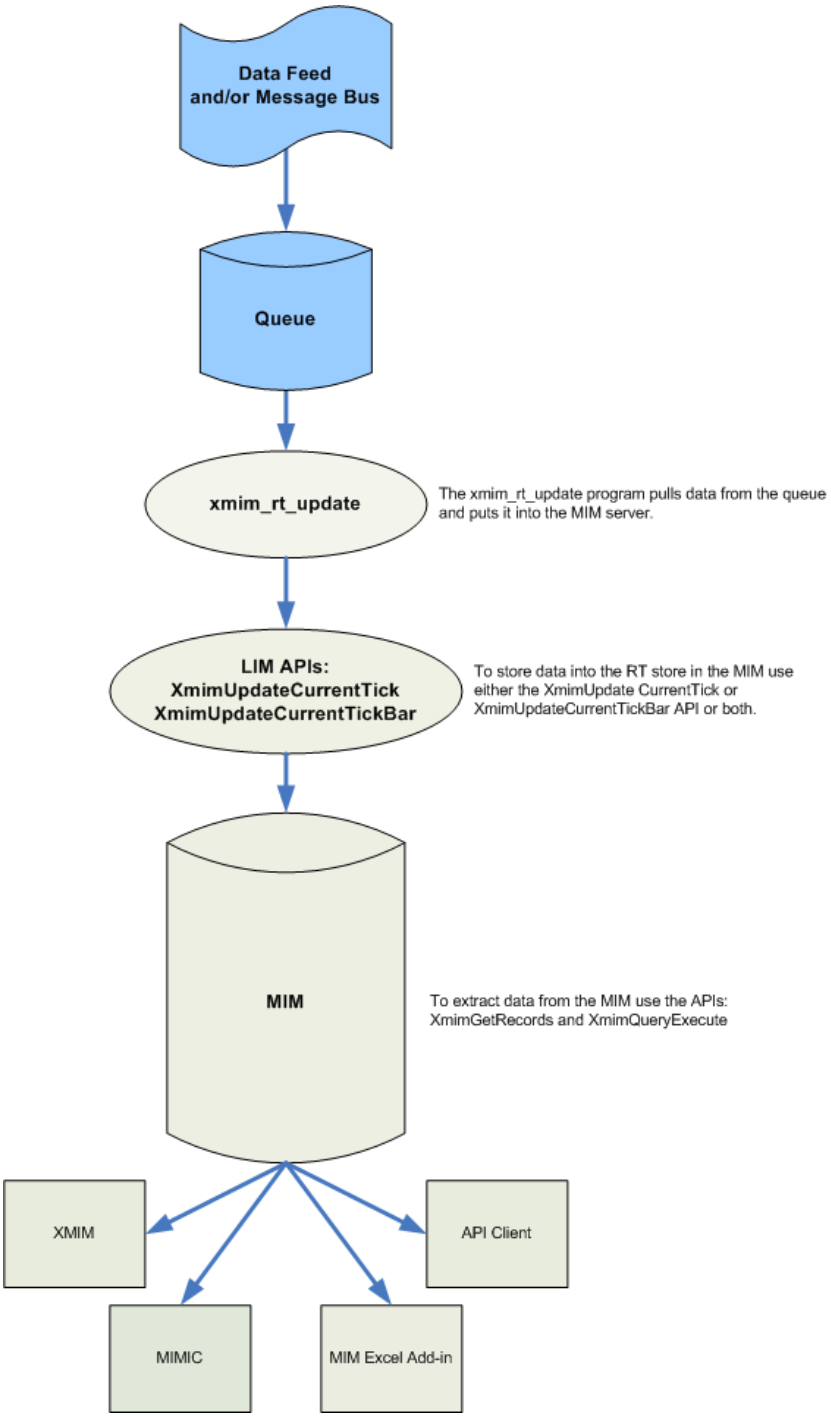
The default is `XMIM_APPEND_TO_NONE` such that data from the real-time database will not be utilized. If `XMIM_APPEND_TO_ALL` is specified and the ending date/time range specified extends beyond what is available in the historic database, then data from the real-time database (if available) will be aggregated into the appropriate units and appended to the historic data. `XMIM_APPEND_TO_DAILY` and `XMIM_APPEND_TO_TICK` can be used to specify that current tick data should be appended only to daily data retrieval or only to tick (both intraday and real tick) data retrieval, respectively. Thus, if historic data is available ending yesterday, the current tick facility is being updated via a tick feed, daily bars are retrieved with the default `toDate` and `XMIM_APPEND_TO_ALL` or `XMIM_APPEND_TO_DAILY` is specified, the result will include historic daily data with an additional daily bar constructed from today's data stored so far in the current tick facility. Likewise, for intraday or real tick data retrieval, the real-time database will be accessed (if so specified) and the data stored so far will be aggregated and appended to the historic data. Note that since the high-frequency real-time database is optimized for fast data storage at the expense of slower data retrieval, if appending from real-time database is indicated, the data retrieval will be much slower than if only historical data is retrieved (historical data is optimized for quick retrieval).

MIM Scalability

The MIM server scales with symmetric multi-processors. The MIM is unique in that it can have multiple databases that can be mounted with one view. For more information, see the chapter on "Using Multiple Databases" in the *MIM Data and Development Guide*.

Bringing Real-Time Data into the MIM

The following shows the process for bringing in real-time data into the MIM.



For an example of how to use the real-time facility with a Reuters data feed, see the *Loading Triarch Data into the MIM Server* document.

Configuring a Historical Database for Real-Time Capture

Typical operating procedures for the MIM server have a historical database updated by a nightly or several times a day feed process. The real-time database is designed to be an "appendix" to the historical database. In order for the real-time API calls to function, the relations and columns need to exist in the schema.

Mapping Symbols

In general, the relation names of the real-time database can match the symbols of the feed. The MIM symbols can contain underscore, period, or the dollar sign. Other special characters should be avoided. For a complete list, see the chapter "Database and Data Guidelines" in the *MIM Data and Development Guide*. The structure of the database is not important to real-time capture. The symbols can be placed in any categories. Typical databases are many times derived from schemas LIM has developed. Other databases are purely mechanical breakdowns of the feed symbols in effect like alphabetical folders. If even a small set of historical data is available, the MIM data loading tools can be used to automate the creation of the database.

Futures Contract Symbols

Futures contract symbols can be placed in the database in advance of quotation. This will not harm the processing of the futures data in any way. Therefore, there are no requirements to configure futures contracts in real time.

Creating Symbols in Real Time

Symbols cannot be created in real time and keep up with the feed. However, it is allowed to create a small number of symbols while the feed is running. There will be a short delay in obtaining the lock and creating the symbol. Note: the program that creates the symbol does not have to be the real-time collector.

Aliasing

It does not matter how the feed capture names the symbols as the user can use aliasing to rename the symbols to a more user friendly naming convention. For more information on symbol naming conventions see the chapter on "Database and Data Guidelines" in the *Data and Development Guide*.

Capturing Real-Time Data into the Real-Time Database

There are four APIs used with updating the real-time database: `XmimUpdateCurrentTick`, `XmimUpdateCurrentTickBar`, `LockCurrentTickStore` and `UnlockCurrentTickStore`. `XmimUpdateCurrentTick` and `XmimUpdateCurrentTickBar` are used to make updates to the real-time database. `LockCurrentTickStore` and `UnlockCurrentTickStore` are used to lock and unlock the real-time database. The following details each API call:

XmimUpdateCurrentTick

To efficiently facilitate high-frequency updating, the time of insertion into the database needs to be minimized as a first priority. For this purpose, there is the `XmimUpdateCurrentTick` function, with the following synopsis:

```
XmimReturnCode XmimUpdateCurrentTick (XmimClientHandle       handle,
                                      XmimDate               date,
                                      int                    numRelations,
                                      XmimString             *relNames,
                                      int                    numColumns,
                                      XmimString             *colNames,
                                      XmimBoolean            realTick,
                                      int                    numRecords,
                                      XmimTime               *times,
                                      float                  *values);
XmimReturnCode XmimVaUpdateCurrentTick (XMIM_CLIENT_HANDLE,       handle,
                                        XMIM_DATE,                date,
                                        XMIM_RELATION_LIST,       relName1,
                                                                  relName2, ..., NULL,
                                        XMIM_COLUMN_LIST,         NULL,
                                        XMIM_REAL_TICK,           False,
                                        XMIM_NUM_RECORDS,         numRecords,
                                        XMIM_TIMES,               times,
                                        XMIM_VALUES,              values,
                                        XMIM_END_ARGS);
```

The date argument specifies the date for which tick data is to be updated in the high-frequency real-time database and the `relNames` and `colNames` give the relations and columns to be updated. As with the `XmimPutRecords` and `XmimReadFacts` routine, if no columns are specified, then Open, High, Low, and Close will be used as the default columns. The high frequency updating facility is available both for real tick data and for intraday tick data; the `realTick` argument indicates which type of tick data is involved (False indicates intraday and True indicates real tick). The number of records to update is given by `numRecords` and the times and values supplied in the times and values arrays. The number of records must correspond to the number of entries in the times array.

Example Usage

```
NG_2004Z, Close, true, 1 record, 10/5/2004, 13:25:00, 3.045
```

XmimUpdateCurrentTickBar

An additional routine, `XmimUpdateCurrentTickBar`, is available for use in cases where the tick data update consists of a single relation with a single price value (i.e., there is only a single update record) and it is desired to propagate the update to the entire bar such that when the data is retrieved, bar charts can be constructed. Utilizing this routine represents a significant optimization for the updating process such that it should always be used if the conditions apply. The synopsis is as follows:

```
XmimReturnCode XmimUpdateCurrentTickBar (XmimClientHandle        handle,
                                         XmimString              relName,
                                         XmimDate                date,
                                         XmimTime                time,
                                         float                   value);
XmimReturnCode XmimVaUpdateCurrentTickBar (XMIM_CLIENT_HANDLE,      handle,
                                           XMIM_RELATION,           relName,
                                           XMIM_DATE,               date,
                                           XMIM_TIME,               time,
                                           XMIM_VALUE,              value,
                                           XMIM_END_ARGS);
```

The `relName` argument gives the relation to be updated in the high-frequency real-time database. The date and time arguments specify the date and time of the update and the value supplies the price value. This routine can be used for updating either real tick or intraday tick data but, in the case of intraday data, the user must make sure to zero out the seconds field (or use `XMIM_INVALID_TIME` prior to setting the minute and hour fields). It is recommended that to obtain the best possible performance, the non-Va version of this routine be used[1].

Example using Real Tick Data

```
NB_2004Z, 10/5/2004, 13:25:00, 3.045
```

XmimLockCurrentTick and XmimUnlockCurrentTick

The MIM maintains the data added by `XmimUpdateCurrentTick` (or `XmimUpdateCurrentTickBar`) separately from the historical tick data. A database lock must be obtained prior to any updates; however, in the case of real-time data updates, a special lock that is specific to the real-time database is provided so that the historical data may remain unlocked. The routines for locking and unlocking the real-time database are as follows:

```
XmimReturnCode XmimLockCurrentTick (XmimClientHandle          handle);
XmimReturnCode XmimVaLockCurrentTick (XMIM_CLIENT_HANDLE,        handle,
                                      XMIM_END_ARGS);
XmimReturnCode XmimUnlockCurrentTick (XmimClientHandle        handle);
XmimReturnCode XmimVaUnlockCurrentTick (XMIM_CLIENT_HANDLE,      handle,
                                        XMIM_END_ARGS);
```

1 This is contrary to the general recommendation where the preferred usage is of the Va routines. In general, the Va routines allow for more flexibility and ease of change. There is, however, a slight overhead associated with the use of the Va routines and, in this particular case, where speed is so critical, the non-Va version provides the optimal usage.

If the MIM cannot acquire a write-lock for the real-time database, `XmimLockCurrentTick` will fail. In general, updates to the real-time database will be immediately visible to any clients reading from the database.