SAP®

Jürgen Aurisch
Markus Stausberg
Wolfgang Auer

SAP AG

Synchronization Manager

MaxDB

April, 20th, 2005

Version 7.6

Important Note

The information in this document describes configuration and deployment of the Synchronization Manager suite programs when run from the command line using java. This approach should only be taken in special, complex scenarios by advanced users.

The recommended way to configure and run the Synchronization Manager programs is to use the wrapper scripts which are provided in `'<independent program path>/bin'` **.**

For documentation, refer to the official MaxDB documentation, available at http://www.mysql.com/products/maxdb/ **.**

For a simple walkthrough example of Synchronization Manager, see

`'<independent program path>/app/syncman/example'` **.**

1 Introduction to MaxDB Synchronization

1.1 Motivation

Traditionally, database synchronization tools directly transfer data between two database instances using a network.

A connection has to be established, and at each end of it synchronization services must be started.

The participants have to be online at the same time to exchange the data. This requires coordinated action according to a fixed schedule at sites which could be at different locations and in different time zones.

If there is no permanent network connection, as in the case of databases in mobile applications, complex administrative issues arise, involving massive human intervention.

In addition, many simple replication concepts only support unidirectional transfer, from a master database instance to one or more client instances.

Changes made to the data in the client instances are silently overwritten with the master data.

Bidirectional synchronization requires a conflict resolving mechanism, which is difficult to implement and administer in a point-to-point scenario where each synchronization participant is interconnected with each other.

In asynchronous synchronization, these problems are avoided by setting up a transport system which implements abstract, permanent destinations to which synchronization data is sent and from where it can be retrieved by the recipients.

The data is stored at the destination until a data consumer connects and requests it.

MaxDB synchronization employs such a transport mechanism compliant to the 'Java Message Service' (JMS) specification, which is an industry standard.

1.2 Features

Code base 100% Java (administration GUI : SWT)

- Replication of DML operations (insert, update, delete)
- Database backed, recoverable operation
- Asynchronous communication, no permanent network connection required

- Bidirectional and unidirectional operation modes
- Different columns of a table can be handled independently
- GUI based definition and administration
- Conflict resolution in bidirectional mode
- Filtering, column type and name translation
- Auditing and collision logging
- No changes of the application table design necessary.
- Based on reliable messaging system.
- Unidirectional Synchronization to MySQL and MinDB clients

1.3 Components of MaxDB Synchronization and Data Flow

1.3.1 Defining the Replication Units

Before a database instance can participate in MaxDB synchronization, a synchronization administrator has to group the database tables into logical replication units. Furthermore, the administrator must define master – client relationships between replication units of different database instances and assign special attributes to them, if required. The definitions are made with a GUI tool which also creates the necessary meta data in the databases and the messaging system.

The definition of replication units and their attributes is described in more detail in a separate section.

When table data has changed, the change propagates through several stages in the synchronization system on its way to the recipient database instances:

1.3.2 Replication Trigger and Shadow Tables

To a table which is to be synchronized, internal insert, update and delete triggers are attached. When a change occurs to a dataset, the according trigger is fired and records the new state of the dataset to a shadow table which is invisible to the user (one can think of it as a kind of database log).

Also some versioning information is maintained to resolve potential conflicts.

1.3.3 Capture of Logged Replication Data

At arbitrary times or permanently, the synchronization service can be run, and the capture component of it converts the shadow table data into replication messages, which are then sent to their destination in the messaging system.

Each such message represents the effect of a database operation (DML, i.e. insert, update, or delete) on an individual dataset in a table, identified by its primary key.

1.3.4 The Messaging System

The messaging system consists of a message server and an open client API (JMS). The server allows to set up abstract destinations to which the clients can send messages and subscribe to .

A message to a destination is stored in the message server until it is consumed by a suitable client subscribed to the destination.

Message delivery is reliable in both the case of a server crash and loss of data in a network connection.

1.3.5 Consumption of Replication Messages

Once the message is delivered to the messaging client, the statement it represents is executed in the recipient database instance unless its version is considered 'out of date' by the consumption component of the synchronization service.

In such a case, the sender of the message is notified about the failure and the current dataset is retransmitted to the client.

1.3.6 Synchronization and Data Consistency

Normally, execution of DML statements is subjected to consistency checking (constraints and foreign keys) and may have side effects implemented by user defined triggers and foreign key delete rules.

In a synchronization scenario, it is assumed that consistency checking has already taken place at the participant which generates replication messages. Therefore such checking is turned off at the consuming participants by using a special type of database session. Similarly, such a session does not execute triggers and foreign key delete rules.

Care must be taken, if side effects affect database objects which are excluded from synchronization and will therefore be missing at the consuming participants.

2 Defining Synchronization

2.1 Prerequisites

2.1.1 Replication User

Create a replication user at each database instance which participates in synchronization. The replication user can only be created by a user who has the SYSDBA privilege. The replication user needs to be a DBA user.

You can create the user with the Database Manager GUI or by executing a SQL Command with e.g. SQL Studio. The key word for the replication user is “replication”.

Example:

```
create user <user> password <password> dba not exclusive replication
```

2.1.2 Message Server

Synchronization of MaxDB is based on reliable and asynchronous messaging. Several synchronization participants exchange their data through the message server. To install and run the message server, refer to the message server documentation.

2.2 Synchronization Manager GUI

2.2.1 Libraries

- **Eclipse SWT/JFace**

  The Synchronization Manager GUI is based on Java and the Eclipse libraries of SWT/JFace.

  The necessary Java libraries for the SWT/JFace are:

  ```
  osgi.jar, jface.jar, jfacetext.jar, swt.jar, text.jar, runtime.jar
  ```

  Beside these jar files the swt shared library suitable for your platform is needed. For convenience the win32 versions of `swt.jar` and the shared library are included in the SyncMan distribution.

For other platforms refer to http://www.eclipse.org .

- **sapdbc.jar**
  To connect to MaxDB with JDBC, `sapdbc.jar` (with a version that matches the MaxDB database software version) is needed. It is available at http://www.mysql.com/products/maxdb/ .
- **j2ee.jar**
  The Synchronization Manager GUI also acts as a JMS Client, which requires the `j2ee.jar` (from the Java 2 Enterprise SDK, available at http://java.sun.com/j2ee/).
- **msgserver.jar**
  The client methods of the message server are contained in `msgserver.jar`.
- **syncservice.jar**
  This jar file `syncservice.jar` is used to perform the initial synchronization of database tables.
- **syncmangui.jar**
  The Synchronization Manager GUI itself is contained in `syncmangui.jar`.

2.2.2 Main Class

The main class of the Synchronization Manager GUI is:

```
com.sap.sdb.syncMan.gui.SyncManGUI
```

2.2.3 Example Command Line

Copy the `j2ee.jar` and the `sapdbc.jar` into the `syncman/lib` directory.

Here '`lib`' is the directory where the SWT shared libraries are located.

To start the SyncManGUI in the syncman directory, use:

```
java -Djava.library.path=lib classpath .\lib;.\lib\runtime.jar;.\lib\os
gi.jar;.\lib\jface.jar;.\lib\jfacetext.jar;.\lib\swt.jar;.\lib\text.jar
;.\lib\sapdbc.jar;.\lib\msgserver.jar;.\lib\syncservice.jar;.\lib\syncm
angui.jar;.\lib\j2ee.jar com.sap.sdb.syncMan.gui.SyncManGUI
```

Under Linux the elements of the class path are separated by a colon.

Alternatively a batch file with the predefined class path can be used:

```
syncmangui
```

2.3 Defining the Synchronization Schema

The definition of a synchronization scenario is performed on the master database instance. When the Synchronization Manager GUI connects to the master database instance, it creates the meta data belonging to the replication user.

2.4 Replication Tables

Only tables which were granted to the replication user with the select, insert, update and delete privilege can be replicated.

After connecting to the master database instance (Session->Connect Database), the tables which are to be replicated can be chosen. Select the tab folder “Replication Tables” and add the tables (Replication Tables → Add Table).

2.4.1 Column Groups

A replication table may consist of column groups which are either versioned or not versioned. Column groups are used for collision detection when the same dataset is updated by different clients or by the master and a client. There can be only one versioned column group. If a table contains blob columns, a versioned column group is automatically generated.

The data changes of a versioned column group collide, if the version of the client column group does not match the version of the master column group at the affected key in the table. In not-versioned column groups the data changes collide, if at least one column of the column group’s “before image” does not match the corresponding column of the master column group at the affected key in the table.

2.5 Replication Unit

A replication unit is a collection of replication tables. For these tables a vertical filter can be defined by selecting the columns or column groups which should be used for synchronization.

A replication unit is either a master or a client unit and has one of the properties *out***,** *in* or *in/out* according to whether the replication mode is unidirectional or bidirectional. (e.g. a master out unit can replicate data from a master to a client, but cannot receive replication data.)

To define a master replication unit select the tab folder “Replication Units” and choose the menu Replication Units->Add Master Unit. Now replication tables can be added to this unit. To a master unit one can add replication client units. The type of a client unit is determined by the type of its master unit. The tables of a client unit may differ in owner, name and the names of its column to the tables of a master unit.

master unit out	→	client unit in
master unit in/out	→	client unit in client unit in/out

A client name, a database server, a database name and the replication user describe a client unit.

2.5.1 Activating a Unit

Before use in synchronization, a unit has to be activated. During activation all necessary triggers, shadow tables and version tables are created in the involved database instances.

2.6 Synchronization Service Configuration

In addition to the synchronization meta data the synchronization services require some configuration data (e.g. JDBC driver and database names).This data may be provided as command line options, or in a configuration file in special cases, but by default it is stored in the participant database instances at activation time.

2.7 Message Server Destinations

The replication data is transported by the message server through ‘destinations’. These destinations are either topics (broadcasting) or queues (point-to-point communication).

To create these destinations, connect to the message server (Session → Connect Message Server). Select the tab “Message Server” and choose the menu “Synchronize Destinations”. The necessary destinations will be created on the message server automatically. The destinations can only be created when the master unit is activated.

3 Running the Synchronization Service

After successful definition and activation, data changes will be logged in the shadow tables of the participating databases. To transfer these data changes, the message server must be running (refer to the message server documentation), and the synchronization services must be started.

The jar files `j2ee.jar` and `sapdbc.jar` have to be copied into the `syncman/lib` directory.

3.1.1 Starting the Synchronization Service

To start the synchronization service in the syncman directory, use:

```
java -classpath .\lib;.\lib\syncservice.jar;.\lib\msgserver.jar;.\lib\sapd
bc.jar;.\lib\j2ee.jar com.sap.sdb.syncMan.SyncService
   -user <replication_user> -password <replication_user_password>
   -database_url "jdbc:sapdb://<server>/<database>?timeout=0&reconnect=false"
   -jdbc_driver com.sap.dbtech.jdbc.DriverSapDB [-admin_port <port_number>]
```

The default administration port number is 7223.

Alternatively the database credentials can be obtained with the option –A <access_key> from the SyncMan Access File.

Under Linux the elements of the class path are separated by a colon.

Alternatively a batch file with the predefined class path can be used:

```
syncservice -user <replication_user> -password <replication_user_password>
   -database_url "jdbc:sapdb://<server>/<database>?timeout=0&reconnect=false"
   [-admin_port <port_number>]
```

3.1.2 Terminating the Synchronization Service

The synchronization service is stopped by terminating the java process or by sending a shutdown command to the synchronization service.

```
java –classpath .\lib\syncservice.jar com.sap.sdb.syncMan.SyncServiceAdmin
   -host <server> -port <admin_port>
   -password <replication_user_password> shutdown
```

Alternatively the password can be obtained with the option –A <access_key> from the SyncMan Access File.

Alternatively a batch file with the predefined class path can be used:

```
syncservicestop -host <server> -port <admin_port>
   -password <replication_user_password> shutdown
```

The host is the machine where the synchronization service is running (default: localhost).

The port is determined when the synchronization service is started (default: 7223).

3.1.3 User Names and Passwords on the Command Line

The MaxDB database and all database tools adhere to a special policy concerning case sensitivity of user names and passwords which apply as well to all user names and passwords supplied to the messaging software. This is as follows:

If such a value is provided without being surrounded with double quote signs, it will implicitly be converted to upper case. To supply a value containing lowercase letters the value has to be surrounded with double quote signs. Note that in most shells it is required to escape the quote signs in order to pass them to the invoked program. This is usually done by using preceding backslash characters. Example :

```
java –classpath [...] –password \"secret\"
```

This also applies to respective values supplied in XML configuration files, where escaping is done using a CDATA section.

4 SyncMan Access Key

In a SyncMan Access Key database and Message Server credentials can be stored. The command line help provides all necessary information.

To display the command line help you can call:

```
java -classpath .\lib\syncmanacc.jar
                        com.sap.sdb.syncMan.acc.SyncManACC -?
```


5 Appendix: MySQL as Synchronization Participant

Currently, a MySQL database instance can only take part as a write-only client in a MaxDB Synchronization scenario. This means that replication units on a MySQL database can only exist as in-type client units connected to an out-type master unit.

Despite this restriction, definition of units and running the Synchronization Service in the MySQL case is fairly straightforward as described above.
However, care must be taken of the different user concept in MySQL and in providing the correct JDBC driver to the Synchronization Manager GUI and the Synchronization Service.

5.1 Replication User

In MySQL there is no special user role for replication. Hence a dedicated database user must be created which has read and write privileges on all user tables which are to be replicated. This user is then used in all places where in the MaxDB case the replication user would be used.

There must be one dedicated MySQL database instance in a MySQL server on an individual computer where the replication user stores its data and meta data.

This database instance is chosen in the first place by specifying it in the according connect dialog of the Synchronization Manager GUI. This database must then be used in all later Synchronization Service operations.

5.2 JDBC Driver

The appropriate .jar file containing the JDBC Driver implementation for the given MySQL software installation must be obtained from http://www.mysql.com/ or a mirror site.

This file must be used in all places in the above documentation instead of `sapdbc.jar`.

Also, the according driver class must be used instead of the MaxDB driver class in all of the above instructions.

