eLibera Messaging Server 1.0

Written by Matthias Meisenberger

http://www.elibera.com / office@elibera.com

Version: 17.07.06, 07:45:59

Index

eLibera Messaging Server 1.0....1
- Overview:....2
- Installing and Starting the eLibera Messaging-Server 1.0:....3
  - Requirements:....3
  - Installation and basic configuration:....3
  - Linux execution of the server:....3
  - Logging-Options....4
- Configuration and Tuning of the eLibera Messaging-Server 1.0:....5
  - General configuration options:....5
    - PORT....5
    - MSG_ENCODING....5
    - MIN_WAIT_TIME_FOR_SENDING_ALIVE_BIT....5
    - TEST....5
  - Maximum number of clients and Clustering....5
    - MAX_CLIENTS_ONLINE....5
    - ALTERNATIVE_SERVERS....6
    - How to cluster multiple servers:....6
  - Tuning the performance of the server....6
    - MAX_PROCESSING_THREADS....6
    - MAX_SIZE_FOR_SOCKET_QUEUE....6
    - MAX_SIZE_FOR_CLIENT_OBJECT_POOL....7
    - STARTING_SIZE_FOR_CLIENT_OBJECT_POOL....7
    - MIN_CLIENT_OBJECTS_IN_POOL....7
- Problems:....8
  - java.net.BindException: Address already in use....8
  - java.net.SocketException: Too many open files....8
  - log4:WARN message at the server start-up....8
  - Error during start-up: Cannot open connection....8
  - Server doesn't start or throws strange error messages....8

Overview:

The eLibera Messaging Server is used for sending messages to the MLE-clients. It can be used as an instant messaging service or as a simple notification and information service.

Every message can contain multiple attachments which are deleted automatically if the user deletes the message on the client.

Installing and Starting the eLibera Messaging-Server 1.0:

Requirements:

The server only works with the Java 1.5 (or 5.0) VM form SUN Microsystems. If you run on a Linux system please check (by running „java -version“) if the correct Java Runtime Environment is installed. If not do it by installing the correct version either through „apt“ (only for Debian Systems, call „apt-get install sun-java5-jdk“) or by downloading and installing the Java VM from „**http://java.sun.com**“!

If you have installed multiple Java Version on your Linux-system please update your system to use the correct one by calling „update-alternatives java“ and selecting the Sun Java VM.

Installation and basic configuration:

To install the server you have to extract the contents of the ZIP-package you received in the directory where the server should reside.

Afterwards you have to edit the

```
hibernate_messaging.properties
```

file, which is located in the "conf"-directory of the server. This file defines the database configuration.

For a basic configuration for the Postgres-SQL database you have to change the following settings:

```
hibernate.connection.url jdbc:postgresql://localhost:5432/messagingserver
```

This line defines the connection URL to your Postgres-SQL database. If your database server doesn't run on localhost and the given port (5432, this is the standard Postgres-SQL port) you should change this value. The last part of this URL (in this case „messagingserver“) means the name of the database. This database must exist and should be empty. You can change this value to everything you want to.

```
hibernate.connection.username yourUsernameHere
```

This settings define your Postgres-SQL username, you want to access the database. Replace „yourUsernameHere“ with your actual database-account username.

```
hibernate.connection.password yourPasswordHere
```

This settings define your Postgres-SQL password according to your username, you want to access the database. Replace „yourPasswordHere“ with your actual database-account password.

If everything is correct the server will generate the database tables on it's on in the specified database on the server.

For the first start of the server you should check the logging information (by calling the start-up script with the option „log“) to see if the database generation was successful.

Please check out also the next chapter about the configuration and tuning of the server for further configuration options. But in most cases the server should work now with the standard configuration.

Linux execution of the server:

To start the server you have to call the „**eLiberaMessagingServer.sh**“ script:

```
./eLiberaMessagingServer.sh start
```

--> will start the server

```
./eLiberaMessagingServer .sh stop
```

--> will stop the server

```
./eLiberaMessagingServer .sh restart
```

--> will restart the server

```
./eLiberaMessagingServer .sh log
```

--> will print the console log of the server (system-out).

Logging-Options

Besides the standard logging output there exists a detailed log-file, usually called:

```
log_full_eLiberaMessagingServer.log
```

To customize the logging output of the server you can edit the following configuration file:

```
logging_messaging.properties
```

The logging is defined after the rules of the JDK logging framework. Please see the JDK documentation for details.

Configuration and Tuning of the eLibera Messaging-Server 1.0:

The configuration of the messaging-server is done over the „settings_messaging.properties“ file located in the "conf"-directory of the messaging-server.

General configuration options:

PORT

```
PORT=6655
```

With the PORT option you could change the listening port of your server. It is not advised to change this option, because you would have to recompile the client software (MLE) for this new port.

MSG_ENCODING

```
MSG_ENCODING=UTF-16
```

This is a rather important option because it defines in which encoding the messages are stored in the database. If you want to use all the Unicode characters you should store all messages in Unicode in the server database and set this value to „UTF-16“.

MIN_WAIT_TIME_FOR_SENDING_ALIVE_BIT

```
MIN_WAIT_TIME_FOR_SENDING_ALIVE_BIT=5000
```

Defines the minimum interval the server checks, if the client is still alive (in milliseconds). Every interval one byte is send to the client. Don't use a too small value, otherwise the client gets too many useless data to process.

This value is also a good option to share the server resources on a fair basis to all connected clients, because every client has to wait at least this time, till he is processed again.

The actual interval can be higher, if the server is under heavy load.

TEST

```
TEST=0
```

This option is only used for testing the database. If you set this value to „1“ the server will generate test entries in the server database. If you set this value to „2“ the server will generate again test-values and then tries to delete them (with this option the database constrain is tested, if the MsgBinary-database records are not deleted, the constraint doesn't work, which means the message attachments won't be deleted!). So if you use the option 1 and 2 you should check the contents of your database to see what happens.

Leave this value to „0“ for the normal use of the server.

Maximum number of clients and Clustering

MAX_CLIENTS_ONLINE

```
MAX_CLIENTS_ONLINE=100000
```

Limits the number of online clients for this server.

The maximum number of this value is limited with your server license. If your license limits the number of clients to 5000, than increasing this value to a higher number has no effect.

ALTERNATIVE_SERVERS

```
ALTERNATIVE_SERVERS=socket\://128.1.1.1\:6656;socket\://128.1.1.2\:6657;socket\://128.1.1.3\:6658
```

Defines alternative socket-addresses to other running instances of a valid Messaging server.

To separate multiple socket-addresses use the ';' - character.

How to cluster multiple servers:

This option must be covered by your license, otherwise it won't work!

Clustering allows you to redirect clients to other physical servers instead of dealing with all clients on one single server.

The best approach for this option is to use one server with a very low „MAX_CLIENTS_ONLINE“-value as the entry-server (this is the server where all the MLE-clients try to connect to). If the server has a higher number of online clients than defined in the „MAX_CLIENTS_ONLINE“-value, it will start to redirect every new client to the defined servers in the „ALTERNATIVE_SERVERS“-option. These servers should now have a „MAX_CLIENTS_ONLINE“-value depending on their performance-capabilities and will deal now with the client requests.

After a successful redirect to a second server it is possible to redirect the client again to another server. Nevertheless, this is not advised, because otherwise the client might wait for a long time till it's actual request gets processed. The better option is, to use one entry server, which has a list of all additional servers and does all the redirecting work.

Furthermore the entry-server should use a very high „MAX_SIZE_FOR_SOCKET_QUEUE“-value and a very high „MAX_SIZE_FOR_CLIENT_OBJECT_POOL“-value to accept all the clients and then redirect them (these values are described in the next chapter). If the „MAX_SIZE_FOR_SOCKET_QUEUE“-value is too low it might occur that a new client gets discarded!

Tuning the performance of the server

Besides these options, also take a look on the Caching-options. These options are also very significant for increasing the performance of the server.

MAX_PROCESSING_THREADS

```
MAX_PROCESSING_THREADS=50
```

This is the most important option according to the performance of the server. It defines how many concurrent running threads (processes) are used to process all the client requests, which are connect to this server (also the redirecting of clients is done by these threads).
If this value is too low, and lots of clients are connected to the server, the client might have to wait a very long time, till it's request is processed. On the other side, if this value is too high, your server might run out of resources and get very slow.

MAX_SIZE_FOR_SOCKET_QUEUE

```
MAX_SIZE_FOR_SOCKET_QUEUE=100000
```

Determines the size of the queue for all new accepted sockets (clients). So if a client gets connected to the server, it first gets pushed to this queue before it is processed. If this queue is already full, the client gets discarded! To avoid this the queue should be big enough to hold the maximum number of clients that would connect at the same time to your server. Additionally you should add 25% to this value, to be able to cover this amount of clients even if your server is currently on heavy load.

If you expect that no more than 10000 clients will connect at the same time to your server, set this value to 12500 or higher.

A client that was processed (authenticated) is removed from this queue immediately.

MAX_SIZE_FOR_CLIENT_OBJECT_POOL

```
MAX_SIZE_FOR_CLIENT_OBJECT_POOL=100000
```

To increase performance the server reuses client objects, instead of creating for every client a new object. These client-objects are stored in this pool. The maximum amount of objects in this pool can be defined here.

It is a good advise to set this number to the maximum amount of clients you expect to be online at the same time plus 10 to 20%.

STARTING_SIZE_FOR_CLIENT_OBJECT_POOL

```
STARTING_SIZE_FOR_CLIENT_OBJECT_POOL=90000
```

This value describes how many objects should be created at server-start-up in the client object pool. If you have the memory capacity, you can set this value to the same one as defined in „MAX_SIZE_FOR_CLIENT_OBJECT_POOL“.

MIN_CLIENT_OBJECTS_IN_POOL

```
MIN_CLIENT_OBJECTS_IN_POOL=100
```

If the pool runs below this number, it will start to create new client objects. Otherwise the server waits till a client disconnects and it's client objects gets back to the pool.

Problems:

java.net.BindException: Address already in use

This means another server is already listening to the given port. Please choose another PORT-value in the settings_messaging.properties file or check that no other instance of the server is already running.

java.net.SocketException: Too many open files

For Linux Systems:
Get the current limit of your system by typing:

```
ulimit -n
```

or

```
cat /proc/sys/fs/file-nr
```

To increase the limit, simply edit the file „/etc/security/limits.conf“ and add the following line:

```
username hard nofile 50000
```

where username is the name of user in which context the server is running and 50000 is the new limit you want to set.

log4:WARN message at the server start-up

During the start-up of the server the following message is shown in the log:

```
log4j:WARN No appenders could be found for logger (org.hibernate.cfg.annotations.Version).
log4j:WARN Please initialize the log4j system properly.
```

This warning message can be ignored, it has no effects on the server!

Error during start-up: Cannot open connection

During start-up of the server you get the following error message:

```
org.hibernate.exception.GenericJDBCException: Cannot open connection
    at org.hibernate.exception.SQLStateConverter.handledNonSpecificException(SQLStateConverter.java:103)
    at org.hibernate.exception.SQLStateConverter.convert(SQLStateConverter.java:91)
    at org.hibernate.exception.JDBCExceptionHelper.convert(JDBCExceptionHelper.java:43)
```

This means that the information you provided to connect to the Database is not correct.
Check if the database name really exists on your database server.
Check if the username and password are valid and are allowed to connect to this database on the given database server. Furthermore, this user need to have the permission to create new tables and sequences.

Server doesn't start or throws strange error messages

Please check that the Java VM 1.5 (or 5.0, it is the same) from SUN is installed on your system and is the default Java VM. Please go through the requirement section of installation-chapter.