eLibera Gateway Server 1.0

Written by Matthias Meisenberger

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

Version: 17.07.06, 07:45:59

Index

Overview:

The eLibera Gateway Server is the central connection point for every MLE-client. Every request is send over this server. Furthermore this server provides a lot of converting options, like:

- Converting HTML pages to MLE-ML code (the XML language of the MLE-client)
- Converting RSS feeds to MLE-ML code
- Resizing images

Additionally this server provides you with a very powerful caching mechanism, which reduces processing power, memory usage, network load and the waiting time for the client.

Other features are a powerful option to block certain users and devices from accessing this server and the possibility to even block certain HTTP-URLs.

Installing and Starting the eLibera Gateway-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_gateway.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/mliberaloggin
```

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 „mliberaloggin“) 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 „**eLiberaGatewayServer.sh**“ script:

```
./eLiberaGatewayServer.sh start
```

--> will start the server

```
./eLiberaGatewayServer.sh stop
```

--> will stop the server

```
./eLiberaGatewayServer.sh restart
```

--> will restart the server

```
./eLiberaGatewayServer.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_eLiberaGatewayServer.log
```

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

```
logging_gateway.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 Gateway-Server 1.0:

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

GATEWAY_SERVER_NAME

```
GATEWAY_SERVER_NAME=eLibera Gateway Server
```

This option describes the the name of the server. This option is sent to every HTTP-Server over the header variable „MLIBERA_GATEWAY“. Otherwise this option has no effect.

GATEWAY_SERVER_VERSION

```
GATEWAY_SERVER_VERSION=1
```

This option describes the the version of the server. This option is sent to every HTTP-Server over the header variable „MLIBERA_GATEWAY_VERSION“. Otherwise this option has no effect.

HTTP_CONNECTION_TIME_OUT

```
HTTP_CONNECTION_TIME_OUT=20
```

Defines how many seconds the server waits for the response from an HTTP server. Be aware that the MLE-client itself has a timeout, so don't set this timeout too high.

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 Gateway 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.

Caching Options

Caching is very important option to increase the performance of your server. The only limitation is the memory.

MAX_CACHE_SIZE

```
MAX_CACHE_SIZE=5000
```

This number defines how many HTTP-responses should be held in the memory. Only HTTP-responses which set the proper HTTP-Caching headers (and therefore allow the caching of the response) are add to this cache.

If your memory capacity allows it, you should use a very high value for this option, because it can increase the performance of the server significantly. In the best case it is not necessary to do an actual HTTP-request to the desired HTTP-server if there is an HTTP-response in this case.

Therefore the caching mechanism also reduces the amount of data that must be transfered over the network.

Be aware that this number says nothing about how big each element in this cache is. In this cache might be images, video files or just text documents (will be the majority).

MAX_CACHE_ENTRIES_FOR_CONVERTED_HTML_IN_MEMORY

```
MAX_CACHE_ENTRIES_FOR_CONVERTED_HTML_IN_MEMORY=10000
```

This is a special value for HTML documents. The gateway server can convert HTML documents to MLE-ML documents (readable by the MLE). This converting-process is very resource intensive therefore the amount of converting-processes should be reduced as much as possible.

If an HTTP-response is cached on the server, this converting process is not necessary. Nevertheless a lot of HTTP-servers doesn't allow caching at all, because the content is generated dynamically (for example most newspaper web-pages forbid the caching even if it would be possible).

Therefore this second cache mechanism is used to prevent the converting-process, if caching was

not possible.

If your memory capacity allows it, you should increase this value to a very high number. This value can even be higher than the „MAX_CACHE_SIZE“ because you can be sure that only small text-documents are stored in this cache.

Database options: Logging, Advertising, Blocking, URLs

USE_DB

```
USE_DB=true
```

If you want to use any of the database options, you have to activate this value.

Setting this value to „false“ will turn off all the database options.

USE_DB_LOGGING

```
USE_DB_LOGGING=true
```

Turns the logging function on.

Every client session and every single client request will be logged in the database.

USE_DB_ADVERTISING

```
USE_DB_ADVERTISING=false
```

Advertising, which is described in the database, will be inserted in every converted HTML document.

USE_CHECK_FOR_BLOCKED_USERS

```
USE_CHECK_FOR_BLOCKED_USERS=true
```

It is possible to block certain clients to use this server. You can define the blocked users in a database table.

Every database row of the blocked users is hold in memory to decrease the client processing time.

BLOCKING_USER_CALCULATIONS_PER_DAY

```
BLOCKING_USER_CALCULATIONS_PER_DAY=1
```

All the data of the blocked users is hold in memory. With this value you can define how often the data in the memory is synchronized with the database.

A value of one means very day at mid-night the synchronization is done. A value of 24 means that the synchronization is done every hour. If you hardly every change the blocked user data, set this value to 1.

USE_CHECK_FOR_DB_SERVER_ENTRIES

```
USE_CHECK_FOR_DB_SERVER_ENTRIES=true
```

It is possible to block certain HTTP-servers or provide the gateway-server with additional information about the given server. This information is stored in a database table.

Every database row of a server entry is hold in memory to decrease the client processing time.

SERVER_ENTRY_CALCULATIONS_PER_DAY

```
SERVER_ENTRY_CALCULATIONS_PER_DAY=1
```

All the data of the server entries are hold in memory. With this value you can define how often the data in the memory is synchronized with the database.

A value of one means very day at mid-night the synchronization is done. A value of 24 means that the synchronization is done every hour. If you hardly every change the server entry data, set this value to 1.

Additional HTML converting options

SKWEEZER_URL

```
SKWEEZER_URL=
```

There are online services which optimize HTML pages for mobile devices. To convert these optimized HTML pages leads to better results than to convert the actual, not-optimized HTML-pages (but it still works!).

Nevertheless the performance will decrease because instead of one single request two HTTP-requests have to be done (if no caching instance works!).

Known skweezer services are:

- http://www.skweezer.net/skweeze.aspx?i=1&q=
- http://www.google.com/gwt/n?u=
- http://www.iyhy.com/?a=

Leave this value blank to turn of the use of this feature.

SKWEEZER_URL_REG_PATTERN

```
SKWEEZER_URL_REG_PATTERN=
```

This is a regular expression pattern to define if a HTTP-request already goes to the skweezer service (than it would be stupid to do 2 requests) or not (than we have to do a second request to the skweezer service).

Valid patterns would be:

- [^.]*\\.google.[^/]*/gwt/n\\?*
- [^.]*\\.skweezer.net/skweeze.aspx\\?*
- [^.]*\\.iyhy.com/\\?*

Leave this value blank to turn of the use of this feature.

Blocking Users

Sometimes it is necessary to block certain users to access your server. To do this you have to create a new database record in the „**blockedusers**“-database table and you have to turn on this feature in the configuration file.

For every record you can define a message that is transfered to the user why he is blocked. You can define this text-message in the „blockmsg“ field of every blocking-record.

To get valid values for your blocking-records you should take a log at the logging-tables, there you can find all the required information. These two tables are also the best source to find the users you want to block.

If you want to block all clients with a certain application name you simply have to create a blocking record where the field „**appname**“ ist set to application name you want to block, the block message is set and the rest of the fields should be empty.

If you want to block a client with a certain device instance ID (this value should be unique for every client) you simply have to create a blocking record where the field „**deviceinstanceid**“ ist set to device instance id you want to block, the block message is set and the rest of the fields should be empty.

If you want to block one client with a certain application name and a certain username, you simply have to create a blocking record where the field „**appname**“ ist set to application name you want to block, the field „**username**“ is set to the username you want to block, the block message is set and the rest of the fields should be empty.

Keep in mind that these records have to be synchronized with the memory of the server. This ocures at a certain interval you have defined in the configuration file. So usually a change in the database has not an immediate effect on the server!

Blocking servers and adding additional information to special HTTP servers:

Sometimes it is necessary to block certain HTTP-servers so your users can not access them. To do this you have to create a new record in the „**serverentry**“-database table.

A new entry must define the „**url**“-field (which must be unique in the table). You should remove the „http://“ or „https://“ prefix of the server URL, otherwise this feature won't work.

If you want to block the server, you simply have to set the „**locked**“-field to „true“ and no user can access this server.

If you want to turn of the caching mechanism for a server, you simply have to set the „**callowcache**“-field to „false“ (otherwise it must be „true“!!!).

If you want to mark a server as a platform server, you simply have to set the „**plattform**“-field to „true“ (otherwise it must be „false“!!!).

If you want to turn of the skweezer-feature for a server , you simply have to set the „**noskweeze**“-field to „true“ (otherwise it must be „false“!!!).

If you want to mark a server as a HTTPS-server, you simply have to set the „**https**“-field to „true“ (otherwise it must be „false“!!!).

To use this feature you have turn it on in the configuration file. Keep in mind that these records have to be synchronized with the memory of the server. This ocures at a certain interval you have defined in the configuration file. So usually a change in the database has not an immediate effect on the server!

Adding advertising:

The gateway-server allows you to include advertising to every converted HTML document. To do this you have to create one or more records in the „**werbung**“-database table.

Every record must consist out of a the „**xml**“-field which contains the actual advertising in MLE-ML code and the „**client**“-field, which defines the application name of the client. Afterwards every client with this application-name will get this advertising included in every HTML document. A new advertising record will only get used for new connected-clients.

To use this feature you have turn it on in the configuration file.

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_gateway.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.