AGNITAS
email & marketing automation

Install & Admin Guide

for

OpenEMM 25.11

AGNITAS AG

www.openemm.org

forum.openemm.org

Author:
Martin Aschoff

Table of Contents

1 Document History and Outlook

Version	Date	Change
1.0.0	October 21, 2025	Section 2.3: Added reference to OpenEMM connectors for integration platforms Make and N8N Section 3.1: Updated software stack and provided updated outlook Section 9.8 Added advice for upgrades to version 25.11
		Document fork for OpenEMM 25.11
1.0.3	October 28, 2025	Section 3.1: Added Node.js version to the list of requirements
1.0.2	August 22, 2025	Section 5.1: Simplified installation of MariaDB and Python driver Section 9.7: Added configuration hint for Tomcat 10.1.42 and later
1.0.1	May 28, 2025	Chapter 4 and section 5.1: various hints when using RHEL 9 Section 6.2: new command to make Postfix reboot-safe Chapter 8: added hint to enable cookies in the web browser to see statistic results Section 9.7: Added recommendation to upgrade *nodejs* and hint to check the status of the jobqueue
1.0.0	April 25, 2025	General: Removed references to CentOS 7, SLES 12, Java 11, Python 3.8 and MariaDB 10.6.8, because these versions are no longer supported by OpenEMM General: Renamed OpenEMM support tool OMT to OST Section 3.1: Updated software stack and provided updated outlook Section 5.2: Consolidated required configuration for MariaDB in one section Section 7.7.1: Removed section on *wkhtmltox*, which is no longer supported Section 9.7: Added advice for upgrades to version 25.04 Section 11.2: More information on JobQueue and JobWorkers Section 11.5: more tips how to tweak the configuration of MariaDB for best performance Section 11.6: new section on what to do if database is down
		Document fork for OpenEMM 25.04
1.0.2	October 16, 2024	Section 4.1 was removed, because OpenEMM 24.10 auto-installs all required Python modules Section 5.1: Added MariaDB version 10.11.7 as alternative for AlmaLinux 9
1.0.1	October 2, 2024	Section 2.1: Updated screenshots due to new GUI design Chapter 4: Added advice for incompatibility of Python 3.11 with MariaDB 10.5 and 10.6
1.0.0	October 1, 2024	Section 3.1: Updated software stack and provided updated outlook Section 3.2+3.3: new hint on legacy file *timezone* Chapter 4: Updated from Python 3.8 to Python 3.11 for future compatibility Section 9.9: Added advice for upgrades to version 24.10 Section 9.13: New section how to upgrade MariaDB
		Document fork for OpenEMM 24.10
1.0.1	March 25, 2024	Section 5.2: Changed way to set up MariaDB's root password for

		better compatibility with OMT
1.0.0	March 8, 2024	Section 3.1: Updated software stack and provided updated outlook Section 9.9: Special advice for upgrades to version 24.04 Removed all references to Apache Tomcat Native (deprecated)
		Document fork for OpenEMM 24.04
1.0.4	March 6, 2024	Section 5.1: Hint that MariaDB 10.6.8 does no longer work with RHEL/AlmaLinux 9 Section 7.2: Added parameter *name* in example of file *dbcfg* Section 7.7: Moved section 6.3.1+6.3.2 to 7.7, because complete installation is only possible at that time in the installation process
1.0.3	January 16, 2024	Section 10.6: new section how to configure a smart relay server
1.0.2	December 22, 2023	Chapter 4: Added hint to re-compile a self-compiled Python version in case of Linux upgrade Section 6.3.1: Changed file path for security reasons
1.0.1	October 31, 2023	Section 5.1: Replaced links for CentOS 7 RPMs with links for CentOS 8 RPMs Section 6.1: Added instructions on how to install and launch the firewall Section 6.2: Added info to make sure that Postfix starts at each server reboot Section 7.2: Corrected typo in statement "GRANT ALL PRIVILEGES" Section 9.3: Replaced sudo command so that configuration for sudoers is not needed
1.0.0	October 19, 2023	Section 3.1: Updated software stack and provided updated outlook Section 4 and 4.1: added new Python lib requirements (see also section 9.8) Section 6.2: new value for property *relay_domains* to remove requirement of root access for mailloop service (see also section 9.11) Section 6.3.1+6.3.2: Marked *wkhtmltox* as deprecated and provided alternative (based on *puppeteer*) Section 7.2.1: updated documentation of OMT's functionality Section 9.8: Special advice for upgrades to version 23.10
		Document fork for OpenEMM 23.10
1.0.1	July 28, 2023	Chapter 8: Added hint how to disable CSRF protection Section 8.1: Added usage of a virus scanner as reason for sending problems
1.0.0	July 20, 2023	General: Removed all MySQL references because it is no longer supported Section 3.1: Updated software stack and provided updated outlook Section 4.1: Replaced Python module pycrypto with cryptography Section 5.1: changed description to use RPMs instead of tarballs to simplify MariaDB installation Section 5.2: Relocated hint on character set and collation from section 10.3 because of its growing importance Section 7.2.1: Added info on replacing EMM logos for white-labeling with OMT Section 7.2.1: Added new option for configuration of file server.xml by OMT

		Section 9.7: Instructions for upgrading 22.10 to 23.04 Section 10.2: description how to set up and configure an HSTS header
		Document fork for OpenEMM 23.04
1.0.1	June 1, 2023	Section 4.1: Removed requirement for Python modules xlrd, xlwt & xutils Section 7.2.1: More details for system email addresses that have to be defined Section 8.1: Added even more things to check if email delivery fails
1.0.0	December 9, 2022	Section 3.1: Updated software stack and provided outlook Section 4.1: Added "msgpack" and "websockets" to list of Python packages Section 5.1: Changed MariaDB version from 10.5.8 to 10.6.8 and recommended an important bug fix Section 5.1+9.3: Added option to specify version of Python module "mariadb" Section 9.6: Instructions for upgrading 22.04 to 22.10 Section 11.2: New section on monitoring of OpenEMM's jobqueue Section 12.1.1: Replaced APR with NIO Connector, because APR has become deprecated

2 Introduction

If your OpenEMM instance has been already installed and you want to upgrade the current version, you may directly jump to chapter *OpenEMM Updates and Upgrades*.

2.1 Purpose of OpenEMM

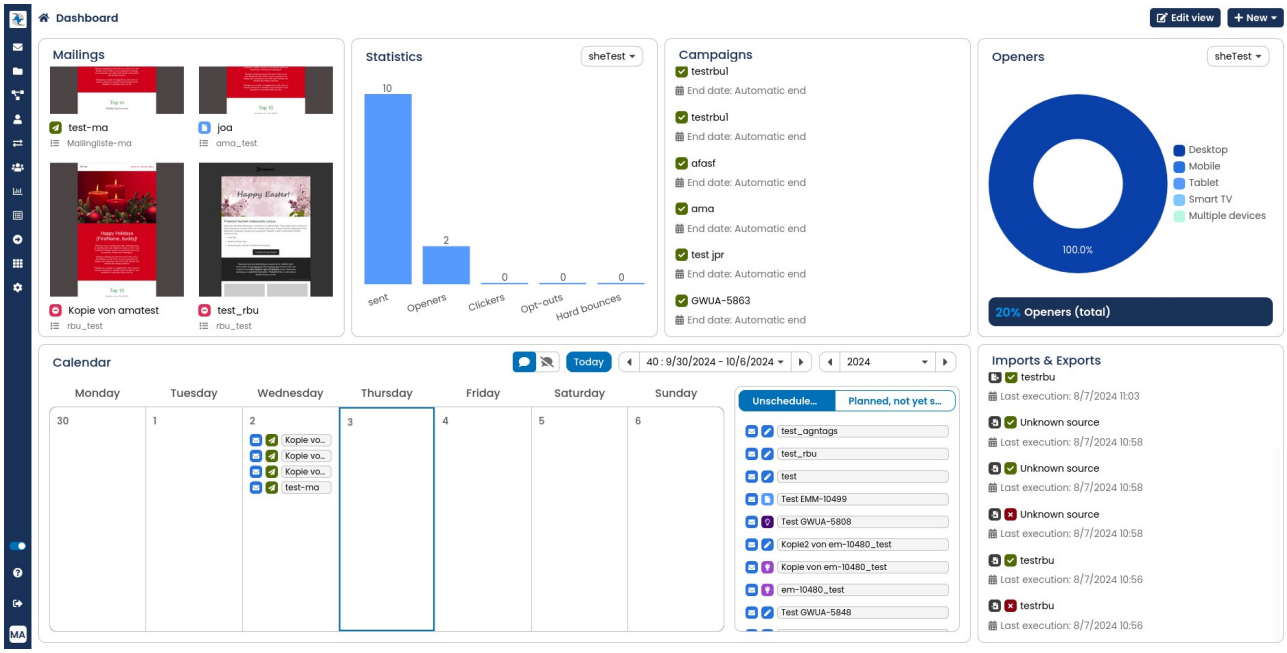
OpenEMM is a web-based enterprise application for email marketing, email newsletters, service emails (transaction emails and event or time triggered emails), marketing automation and lead management. It can be operated via a browser interface with great usability, or in headless mode via its SOAP or REST API. To summarize it, OpenEMM is a tool for customer relationship management by email.

OpenEMM offers tons of features for professional marketing users, among them:

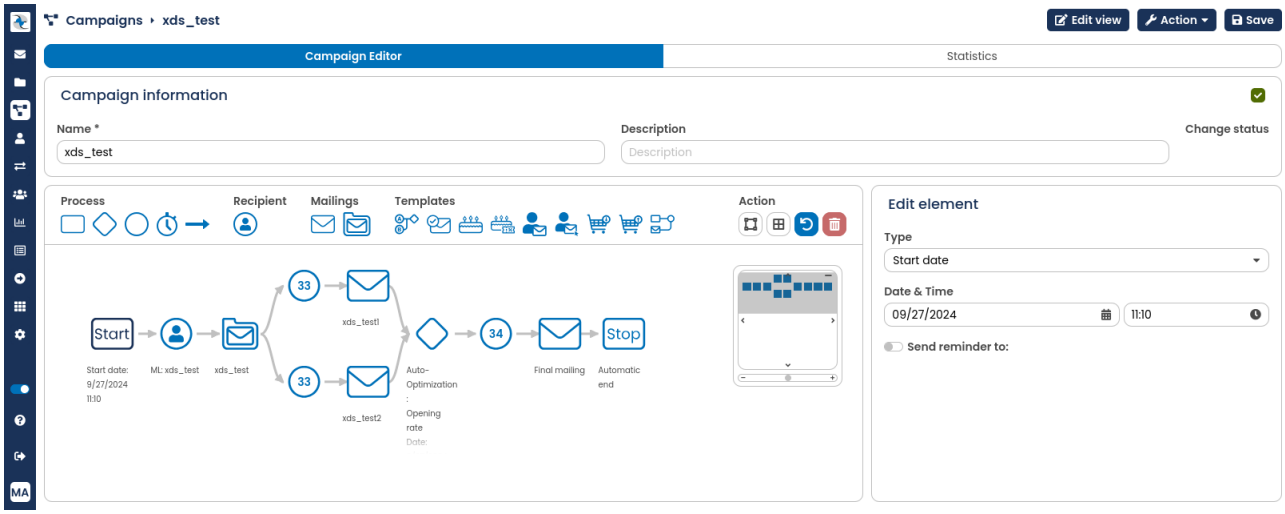
- a console based administration tool for checks, configuration, updates and backups (OST)
- a **responsive web user interface** with great usability and different languages
- a **mail template management** system
- a **visual web form builder**
- mailing, template and web forms import and export to load and exchange prepared mailing templates and web forms
- a **graphical workflow manager** to create complex campaigns with drag&drop
- individual and (GDPR compliant) anonymous tracking of mail openings, link clicks and deep tracking
- **automated bounce management**
- graphical **realtime statistics** with lots of KPIs and configurable reports (PDF and CSV)
- self-defined **target groups** based on recipient profiles, recpient's status and behaviour (created visually or with SQL-like syntax)
- a **scalable multiqueue mail backend** for maximum sending performance
- flexibly configurable data import and export with extensive reporting of results
- **predefined triggers** that can be accessed by HTTPS and can be registered as **webhooks** for 3rd party systems
- a **scripting** feature to enhance the functionality of OpenEMM with **customized triggers**
- sophisticated management of users, user roles and user rights
- an audit-proof searchable and exportable user activity log
- a system status menu with helpful info and configuration options for OpenEMM administrators
- an extensive set of **SOAP webservices** to manage OpenEMM from remote

- a feature-rich **RESTful API** to manage OpenEMM from remote
- a callback API to register **webhooks** for notifications of 3rd party systems about various mailing and recipient events
- **connectors for integration plaforms** Make and N8N

The GUI of OpenEMM works with web browsers Google Chrome, Mozilla Firefox and Microsoft Edge. To give you an impression of the web-based user interface of OpenEMM, the following two screenshots illustrate the configurable dashboard and how to build workflows and campaigns with OpenEMM:



OpenEMM dashboard with latest mailings, planning calendar, status of active campaigns,etc.



Building of workflows and campaigns with OpenEMM's graphical drag&drop editor

2.2 General Architecture

OpenEMM consists of several independent services for scalability. It runs on top of a well proven Open Source software stack without any further dependencies to any commercial software:

- Red Hat Enterprise Linux (**RHEL**) or Suse Linux Enterprise Server (**SLES**) or compatible
- Postfix
- MariaDB
- Java OpenJDK
- Apache Tomcat
- Python 3
- Node.js

2.3 Open Platform Design

A software like OpenEMM must not operate as an island, but it should be capable to be integrated with third party systems like a CRM or CMS software, an e-commerce shop, an ERP platform or a business intelligence software. Therefore, OpenEMM was designed to be a very open and flexible platform with lots of interfaces and extension capabilities. It can be used headless, too.

OpenEMM offers not only a highly customizable graphical interface (GUI) for its users, but also an easy to use URL API für URI tunneling and a rich SOAP webservice interface as well as a RESTful API to be used by third-party software. For outbound communication OpenEMM provides a callback API for webhooks to notify third-party software about various events.

You may also use integration platforms Make or N8N to connect and integrate OpenEMM with other applications. The OpenEMM Wiki (https://wiki.openemm.org) offers information on OpenEMM connectors for these platforms in chapter *6 Integrations*.

2.4 Architecture Components

The OpenEMM software is not a monolith, but it is divided into several independent backend and frontend **services** (software). Communication between the various OpenEMM services is done via the database, i.e. the database is the hub of the application.

To gain a better understanding of the purpose of the various backend and frontend services, this is a summary of their main jobs:

- **Backend:** The backend services read mail-related data from the database, generate the individual emails, create mail previews and send the mails out into the Internet. Backend services also register instant bounce messages and collects reponses (like autoresponder and mail replies by recipients) and delayed bounce data to update the database with results from the mail sendings. Backend services send out autoresponder mails defined in the OpenEMM GUI and forward all mail replies not filtered out to a predefined destination (feedback address).
- **GUI** (part of frontend): This service provides the browser-based user interface of OpenEMM.
- **Statistics** (part of frontend): This service generates statistics (tables with numerical values as well as visual charts and diagrams) for the OpenEMM GUI and creates reports in PDF and CSV format for download.
- **Webservices** (part of frontend): This service provides the webservice interface of OpenEMM.

3 General Requirements

This document will guide you through some necessary steps, which are needed to install and configure OpenEMM. It requires a little knowledge of Linux system administration, of MariaDB database administration and knowledge of Domain Name Services (DNS) to configure the domain names for various services.

For questions, comments, suggestions for improvement and other contributions to improve this guide, please feel free to use the OpenEMM support forum at forum.openemm.org

3.1 Software Stack

This is the software stack required by OpenEMM **24.04**:

- 64-bit version of **RHEL 8** or **9**, **AlmaLinux 8** or **9**, or **SLES 15**
- **Java Open JDK 17** or **21** (LTS version)
- Apache **Tomcat 10.1** or **11**
- **Python 3.11** or later (see chapter *Python 3 for OpenEMM* for details)
- DBMS: **MariaDB 10.6.10** or later (we use 10.11.7, see *MariaDB* for details)
- MTA: **Postfix** 2.6 or later
- **Node.js** 18 or later

It may be that OpenEMM also operates on later versions than the ones listed above, but this has not been tested in a production environment, yet.

To install OpenEMM and the required software stack you need a shell access (like *bash*) for your server as user *root*. All command-line examples are based on RHEL or SLES. Unless otherwise noted, you should run all commands as user *root* to make sure you own and you can grant the required permissions.

Instructions that are valid only for RHEL, are introduced with a header line **RHEL** and are ended with two blank lines. Instructions that are valid only for SLES, are introduced with a header line **SLES** and are ended with two blank lines, too.

3.1.1 Software Stack Roadmap / End of Life

Hera are some additional information on the software stack requirement of future versions of OpenEMM, so that you have enough lead time for your release plannings:

- **Linux:** Since EOL for Suse 12 was end of October 2024, OpenEMM does no longer support it.
- **Java:** OpenEMM 25.04 is compiled with Java 17 and needs a deployment of Java JDK 17 or 21 as runtime environment.
- **Tomcat:** Because OpenEMM is compiled with Java 17, it requires Jarkata EE libs. Jarkata EE is only supported by version 10.1 (or later) of web container Tomcat.

- **Python:** Starting with OpenEMM 25.04, the required minimum version for Python is 3.11, because security support for Python 3.8 has end in October 2024.
- **MariaDB:** Since OpenEMM does not support MySQL 8 or later (due to compatibility issues), we recommend to switch from MySQL to a current version of MariaDB now (see section *Special Advice for Upgrades to Version 22.10*), because Oracle's extended support of MySQL 5.7 has ended and OpenEMM 22.10 was the last version of OpenEMM officially supporting MySQL 5.7.
- **Postfix:** We have dropped support of Sendmail because while Postfix is actively developed further, progress of Sendmail stalls. Also, beginning with version 3.4, performance of Postfix is superior to Sendmail.

3.2 RHEL Operating System

Make sure that service *cron* is enabled on your server and that that *SELinux* (RHEL) is disabled. **Disable SELinux** by changing parameter *SELINUX* to value "permissive" or "disabled" in file *config* of directory */etc/selinux* and reboot the server.

If a legacy file *timezone* exists in directory */etc* of your server, please make sure that this file uses your local timezone, because EMM picks up this timezone. If file timezone does not exist, check if symlink *localtime* in directory */etc* points to your local timezone.

Update the operating system to its latest release. This will keep your system in the most stable state and harden it against various intrusion attempts.

Please be advised that if you have configured your operating systems to do automatic updates, these updates may cause problems. Example: The operating system updates the Java version EMM uses to a new version while OpenEMM is running. This will require a restart of OpenEMM to get rid of any inconsistencies caused by mixing the old and new Java version at runtime.

Install all required packages as noted below. Further dependencies will be resolved and installed automatically by the repository management software:

```
# dnf update
# dnf install gcc make
# dnf install xorg-x11-fonts-75dpi fontconfig freetype libX11 libXext
libXrender urw-fonts
```

For the backend of OpenEMM you have to compile and install Python 3.11 or later. But please follow the description in this manual and install the database driver module for Python only after the DBMS is installed. See section *Python 3 for OpenEMM* for details on compilation and deployment of Python 3.

The frontend of OpenEMM needs Java 17 or 21. You should use OS*T* to install it, in case it is missing on your server. See section *OST (OpenEMM Support Tool)* for details.

3.3 SLES Operating System

Make sure that service *cron* is enabled on all servers and that *SELinux* is disabled on all servers.

If a legacy file *timezone* exists in directory */etc* of your server, please make sure that this file uses your local timezone, because EMM picks up this timezone. If file timezone does not exist, check if symlink *localtime* in directory */etc* points to your local timezone.

Update the operating system to its latest release. This will keep your system in the most stable state and harden it against various intrusion attempts.

Please be advised that if you have configured your operating systems to do automatic updates, these updates may cause problems. Example: The operating system updates the Java version EMM uses to a new version while OpenEMM is running. This will require a restart of OpenEMM to get rid of any inconsistencies caused by mixing the old and new Java version at runtime.

Install all required packages as noted below. Further dependencies will be resolved and installed automatically by the repository management software:

```
# zypper install gcc
# zypper install zlib fontconfig libfreetype6 libX11-6 libXext6
libXrender1 xorg-x11-fonts ghostscript-fonts-std
# zypper install zip sudo wget
```

Get a list of all available repositories:

```
# SUSEConnect -list-extensions
```

For the backend of OpenEMM you have to compile and install Python 3.11 or later. But please follow the description in this manual and install the database driver module for Python only after the DBMS is installed. See section *Python 3 for OpenEMM* for details on compilation and deployment of Python 3.

The frontend of OpenEMM needs Java 17 or 21. You should use OST to install it, in case it is missing on your server. See section *OST (OpenEMM Support Tool)* for details.

Install and enable the required logging service:

```
# zypper install rsyslog
# systemctl start rsyslog
```

```
# systemctl enable rsyslog
```

Furthermore, directory */usr/sbin* has to be included in the PATH variable for all new users. Therefore, add the following line to file *.profile* in directory */etc/skel* before you create the user for OpenEMM:

```
export PATH=$PATH:/usr/sbin
```

4 Python 3 for OpenEMM

Quite often the Python version provided by your operating system is too old or incomplete. Therefore, you should install Python and all required modules by yourself.

Another hint: If you upgrade your Linux distribution to a later version (like migrating from AlmaLinux 8 to AlmaLinux 9), please check the Python version provided by the new distribution. If in doubt, de-install Python and re-install like described below.

Please be aware that the MariaDB database driver of Python 3.11 does not support MariaDB before version 10.6.10. Therefore, please upgrade MariaDB if you use an older version.

RHEL:

First, install some required development packages:

```
# dnf install gcc gcc-c++
# dnf install libgcrypt-devel libxml2-devel openssl-devel bzip2-devel glibc-langpack-en.x86_64
# dnf install gdbm-devel libffi-devel ncurses-devel
# dnf install readline-devel sqlite-devel zlib-devel xz-devel
```

Package *glibc-langpack-en.x86_64* is needed because of bug “LC_ALL: cannot change locale (en_US.UTF-8)”.

Package *gdbm-devel* is no longer included in the standard repository for RHEL 9. The *CRB repository* needs to be enabled for this. On AlmaLinux 9, this can be done with

```
dnf config-manager --set-enabled crb
dnf install gdbm-devel
```

Install Python 3.11 and the default version of Python 3 with package manager and development packages:

```
# dnf install python3.11 python3.11-devel python3.11-pip python3-devel python3-pip
```

SLES (valid for SLES 15 SP4 to SP6):

At first, install some required development packages:

```
# zypper install -y gawk gcc gcc-c++ gdbm-devel libbz2-devel libdb-4_8-devel libffi-devel libxml2-devel libnsl-devel libopenssl-devel libuuid-devel make ncurses-devel readline-devel sqlite3-devel tar wget xz-devel zlib-devel
```

List the available extensions with

```
# suseconnect -list-extensions
```

Search for a suitable extension with a name like *Python 3 Module 15 SP4 x86_64*. The entry also lists how to activate this extension, in this case:

```
# SUSEConnect -p sle-module-python3/15.4/x86_64
```

Now you can install all Python 3.11 packages that are required by EMM:

```
# zypper install python311 python311-pip python311-devel python311-dbm
```

5 DBMS for OpenEMM

5.1 MariaDB

If you want to use OpenEMM with a MariaDB database, you have to install the database software before you install the MariaDB database driver module for Python. If MySQL is preinstalled on any of your servers, you have to remove it first before installing MariaDB:

```
# systemctl stop mysql
# dnf remove mysql*
```

Even officially published versions of MariaDB may contain serious bugs, including severe data corruption, performance problems or security issues (see public bugtracker at https://jira.mariadb.org). Therefore, we can not recommend to blindly install the latest version available because of the potential risk of data loss. For example, we used MariaDB **10.6.8** for the EMM public cloud quite some time and for us this version worked fine (while the following version 10.6.9 contained a crash level bug and does not work at all with OpenEMM).

Please note that MariaDB 10.6.8 will no longer work with RHEL 9 or AlmaLinux 9, because some required dependencies are no longer available in the repositories of these Linux versions. Also, MariaDB 10.6.8 does not work with Python 3.11, which is required by OpenEMM. For AlmaLinux 9 and Python 3.11, we use MariaDB **10.11.7** in our production environment and this version works fine for us.

We recommend to install the MariaDB RPM files from the official MariaDB repository. If you want to use the same version we use, download the version mentioned above.

For OpenEMM you need both the server and the client component of MariaDB. At first, enable the MariaDB repository so that the required dependencies are installed automatically:

```
# wget https://r.mariadb.com/downloads/mariadb_repo_setup
# bash mariadb_repo_setup --mariadb-server-version="mariadb-10.11"
```

Install the MariaDB-server package (including dependencies) first. For RHEL 9 use

```
# dnf install https://archive.mariadb.org/mariadb-10.11.7/yum/almalinux9-amd64/rpms/MariaDB-server-10.11.7-1.el9.x86_64.rpm
```

and for RHEL 8

```
# dnf install https://archive.mariadb.org/mariadb-10.11.7/yum/almalinux8-amd64/rpms/MariaDB-server-10.11.7-1.el8.x86_64.rpm
```

Afterwards, install the remaining package *MariaDB-devel* (needed for Python/PIP) with:

```
# dnf install MariaDB-devel
```

To protect your MariaDB installation from automatic updates by *dnf update*, lock your version with

```
# dnf install yum-plugin-versionlock
# dnf versionlock add MariaDB-server
```

Edit the master configuration file *my.cnf* in directory */etc*:

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
symbolic-links=0
log-error=/var/log/mariadb/mariadb.log
innodb_stats_persistent=0
!includedir /etc/my.cnf.d
```

To start MariaDB and to make sure it launches every time the server is rebooted:

```
# systemctl enable mariadb
# systemctl start mariadb
```

Finally, add the MariaDB database driver module for Python which allows Python to access the MariaDB API after you have installed Python 3.11 or later:

```
# python3 -m pip install mariadb
```

Please note that with RHEL 8 you must first set Python version 3.11 as the default, since the system's preinstalled Python version is not sufficient. Use the following command to configure the default version:

```
# alternatives --config python3
```

Enter the number of the desired option (for example, 1) to select Python 3.11 as default version.

It is not necessary to create a separate database user for OpenEMM, because the OST (OpenEMM Support Tool) will take care of it (see section *OST (OpenEMM Support Tool)* below).

5.2 MariaDB Configuration

Set the MariaDB root password right after installation with

```
# mysql -u root
MariaDB [(none)]> ALTER USER 'root'@'localhost' IDENTIFIED by
'<password>';
```

and save it in your local password store or on your server in a secret file like *.mysqlpw* in directory */root* with read and write permission only for user *root*:

```
# cd /root
# vi .mysqlpw
```

```
# chmod 600 .mysqlpw
```

To avoid problems operating OpenEMM with MariaDB, you **must not** use its default configuration, but you have to set or change the following properties in section *[mysqld]* of MariaDB's configuration file *my.cnf* (usually found in directory */etc*):

```
sql-mode = "STRICT_ALL_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION"
```

→ MariaDBs offers several modes of operation. Please make sure these three modes (and only these three modes) are set:

- If the *strict* mode is not selected, if OpenEMM tries to import a string into a database field which is too short to hold the complete string, MariaDB would simply cut off the string at the end to make it fit - without any warning! This would harm the integrity of your data.
- Preventing *auto-creation* of new database users is a security setting.
- The ban of *substitutions* makes sure that MariaDB uses OpenEMM's choice of database engine InnoDB in any case.

```
lower_case_table_names = 1
```

→ This enforces usage of lower case for table names on Linux systems (where the case of a letter makes a difference).

```
wait_timeout = 86400
```

→ This property is set to 28800 by default. This means that MariaDB automatically cuts the connection to OpenEMM after 8 hours of inactivity. This leads to an initial connection error when OpenEMM attempts to contact the MariaDB database next time. If your OpenEMM installation does not access its MariaDB database all the time, you should increase this value to at least one day (86400) or even to a whole week (604800).

```
innodb_lock_wait_timeout = 1800
```

→ This prevents MariaDB from canceling a database operation after a mere 50 seconds (the default value of MariaDB), which could lead to inconsistent data in the OpenEMM database.

```
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci
```

→ This avoids possible conflicts between OpenEMM code and MariaDB caused by different settings for character set and collations in the database.

```
max_allowed_packet = 10M
```

→ Please be aware that the default value of this parameter may only be 1 or 2 MByte. In this case, you can not load a single data packet (a file for example) bigger than 1 or 2 MByte into the database. This affects the upload of files with the OpenEMM upload feature or the upload of attachments for emails. Since the transfer of data to the database has some overhead, the value for *max_allowed_packet* should be a little bit higher than the value for *attachment.maxSize* in file *emm.properties*. (You can check the current value inMariaDB with “`SELECT @@max_allowed_packet;`”). This value also limits the maximum

size of SQL statements. In OpenEMM this is important for those SQL statements that retrieve statistical data for display in the workflow manager and dashboard calendar. In (the very unlikely) case that the GUI does not show these numbers and you find a corresponding error message in Tomcat's log *catalina.out*, double the value for parameter *max_allowed_packet* until it works.

```
autocommit = 1
```

→ If you want to set property *autocommit*, then set it to active. However, this is not strictly necessary, because MariaDB enables *autocommit* by default.

Please do not forget to restart MariaDB after changing the configuration file. For more advice on how to configure the MariaDB database, check out section *MariaDB Database Tuning*.

5.3 DNS Requirements

When setting up the DNS entries for your OpenEMM server, please make sure that your server holds a valid A record and a PTR record which points back to the hostname of your server (see /etc/hosts) for reverse lookups. This is important because most external mailservers that receive emails from your OpenEMM installation will do a reverse DNS lookup in order to check if the FQDN of your server and the PTR record of your server's IP address match. If not, this is an indication of a spambot network and quite often your emails will be rejected.

If you plan to use an SPF entry for the domain which is used for the sending address of your mass mails, make sure to add your server to this SPF record.

6 Server Preparations

Before you are able to install OpenEMM on the server you have to prepare the server first.

Create a group and a user openemm:

```
# groupadd openemm
# useradd -m -g openemm -d /home/openemm -s /bin/bash openemm
# passwd openemm
# su - openemm
```

6.1 Firewall Configuration

If necessary, install the firewall software, anable and start it:

```
# dnf install firewalld
# systemctl enable firewalld
# systemctl start firewalld
```

Open port 25 and port 8080 in your firewall and add a port forwarding from port 80 to 8080, so you do not have to enter the URL of your OpenEMM server with ":8080" at the end:

```
# firewall-cmd --get-active-zones
```

If your zone is "public" (if not, use the zone name you got with the aforementioned statement):

```
# firewall-cmd --zone=public --add-port=8080/tcp --permanent
# firewall-cmd --zone=public --add-port=25/tcp --permanent
# firewall-cmd --zone=public --add-forward-
port=port=80:proto=tcp:toport=8080 --permanent
# firewall-cmd --reload
```

If you want to use the HTTPS protocol instead of HTTP (which we strongly recommend for production environments), you need a forwarding from port 443 to 8443:

```
# firewall-cmd --zone=public --add-forward-
port=port=443:proto=tcp:toport=8443 --permanent
```

If you use SUSE, we recommend to use *iptables* which can be installed with

```
# systemctl mask SuSEfirewall2
# systemctl stop SuSEfirewall2
# zypper install iptables
```

6.2 Postfix Deployment

Since you want to use Postfix as SMTP server (MTA), you have to stop and remove Sendmail first (in case it is installed), and you have to install the required packages for Postfix (in case it is not already installed).

RHEL:

```
# systemctl stop sendmail
# dnf remove sendmail
# dnf install postfix sendmail-milter procmail
```

SLES:

```
# systemctl stop sendmail
# zypper remove sendmail
# zypper install postfix procmail libmilter1_0
```

RHEL + SLES:

Further dependencies will be resolved and installed automatically by the repository management software.

Switch the default SMTP server to Postfix with

```
# alternatives --set mta /usr/sbin/sendmail.postfix
```

and create a symlink so that OpenEMM can find the Postfix mail log file:

```
# ln -s /var/log/mail /var/log/maillog
```

After installation of Postfix, you have to change its configuration to unleash all features. To do this, change to the Postfix main configuration directory:

```
# cd /etc/postfix
```

Add some configuration parameters to Postfix' main configuration file *main.cf*:

```
inet_interfaces = all
inet_protocols = all
mailbox_command = /usr/bin/procmail
mailbox_size_limit = 0
message_size_limit = 0
maximal_queue_lifetime = 1d
bounce_queue_lifetime = 1d
smtp_tls_security_level = may
smtp_tls_protocols = !SSLv2, !SSLv3, !TLSv1, !TLSv1.1
smtp_tls_ciphers = high
smtp_tls_mandatory_ciphers = $smtp_tls_ciphers
hash_queue_depth = 2
enable_long_queue_ids = yes
```

```
relay_domains = hash:/home/openemm/var/run/relay.domains
transport_maps = hash:/home/openemm/var/run/transport.maps
smtpd_milters = unix:/home/openemm/var/run/bav.sock
```

If lines with parameters of the same name already exist in file *main.cf* (like *inet_interfaces, inet_protocols* or *smtpd_tls_security_level*), comment them with character # at the beginning to avoid any warning messages or overwrite them with the new values in case you do not want to keep the original values as backup.

The two files *relay.domains* and *transport.maps* as well as Linux socket *bav.sock* are automatically created at first startup time of the mailloop service. File *relay.domains* specifies your mailloop service domain name, so that responses like auto-replies and bounces sent to an email address with this domain name are accepted by Postfix for relaying. File *transport.maps* defines for the mailloop service domain name the service used for processing.

Additionally, you have to set parameter *myhostname* in file *main.cf* to the FQDN of your OpenEMM server like *openemm.domain.com*. Otherwise, mails would be sent with sender domain *localhost.loacldomain* instead.

If you want to be able to receive autoresponder, bounce and feedback mails encrypted with the TLS protocol, add

```
smtpd_use_tls = yes
smtpd_tls_loglevel = 2
smtpd_tls_security_level = may
smtpd_tls_received_header = yes
smtpd_tls_session_cache_timeout = 3600s
smtpd_tls_eecdh_grade = strong
smtpd_tls_cert_file = <path_to_CERT_file>
smtpd_tls_key_file = <path_to_KEY_file>
smtpd_tls_CAfile = <path_to_CERT_chain>
smtpd_tls_CApath = <path_to_CERT_directory>
smtpd_tls_protocols = !SSLv2, !SSLv3
smtpd_tls_ciphers = high
```

to file *main.cf*. Take care to replace the four placeholders with the real directory paths to the specified files to make sure that Postfix is able to receive TLS encrypted mails. Certificate *mail.crt* may be a self-signed certificate.

Finally, the configuration parameters for service "mailloop" are defined in configuration file *master.cf*. Add these two lines:

```
mailloop unix - n n - - pipe
  flags=RX user=openemm argv=/usr/bin/procmail /home/openemm/lib/bav.rc
```

Please do **not** omit the two space characters before keyword "flags" to indicate the parser that the line is continued!

Last not least, activate the TLS manager in file *master.cf* by uncommenting (i.e. removing the leading #) line

```
tlsmgr unix - - n 1000? 1 tlsmgr
```

To activate your manual changes to the configuration of Postfix, restart Postfix with

```
# systemctl restart postfix
```

To make sure that Postfix is started at each server reboot, add the service to run level 3 and 5:

```
# systemctl enable postfix
```

You might want to test this settings with a server reboot to be on the safe side.

In case you start Postfix separately from OpenEMM and Postfix complains that file *relay.domains* is missing, you can ignore this warning because OpenEMM take care to create this file at startup time in case it is missing.

6.3 Tomcat Deployment

OpenEMM needs web application engine Tomcat for the frontend services (GUI, statistics and webservices). Tomcat can be installed with OpenEMM Support Tool *OST*. For details please read section *OST (OpenEMM Support Tool)* below.

If you want to operate OpenEMM with the HTTPS protocol, the server key files (**.key*, **.pem*) and server certificate files (**.crt*, *cacerts*) for the TLS configuration (to allow HTTPS connections) must be provided from your side as these files are server and client specific. See section *Apache NIO Connector* for details.

6.4 Configuration of Operating System Logging Parameters

Current versions of RHEL and SLES provide a "feature" that drops messages from being logged if the server has a high workload. If you use the default configuration values of RHEL or SLES (i.e. 1,000 entries max. within 30 seconds), and if the OpenEMM server has a high workload due to a high mail output, this can lead to missing entries in the maillog. However, missing entries in the maillog mean that OpenEMM does not know whether mails were delivered successfully or not and may lead to an incomplete bounce management and incomplete statistics!

To make sure that even under high workload all messages are logged to the maillog, we recommend to change (or add) the following values of file *journald.conf* in directory */etc/systemd*:

```
RateLimitIntervalSec=10s
RateLimitBurst=20000
```

Afterwards, restart the journal daemon with

```
systemctl restart systemd-journald
```

to activate your changes.

SLES:

Additionally, add (or change) the following values in file *rsyslog.conf* of directory */etc* after the line with parameter *$IMJournalStateFile imjournal.state*:

```
$imjournalRatelimitInterval 10
$imjournalRatelimitBurst 20000
```

RHEL 8:

Change in file *rsyslog.conf* in directory */etc* lines

```
module(load="imjournal"             # provides access to the systemd journal
    StateFile="imjournal.state")    # File to store the position in the journal
```

to the following one-liner:

```
module(load="imjournal" StateFile="imjournal.state" ratelimit.interval="10"
ratelimit.burst="20000")
```

RHEL+ SLES:

Afterwards, restart the rsyslog service to activate your changes in file *rsyslog.conf*:

```
systemctl restart rsyslog
```

6.5 Miscellaneous

OpenEMM needs a minimum value of 16384 for kernel parameter *nofile*, which defines the maximum number of open files per process. *OST (OpenEMM Support Tool)* will check and change if necessary.

OpenEMM requires read access to the mail log file at */var/log/maillog* and *logrotate* has to be aware of this fact, too. Open file *syslog* in directory */etc/logrotate.d* and add the following line after the line *sharedscripts*:

```
create 0644
```

and run

```
# chmod 644 /var/log/maillog
```

to set the permissions of the current maillog.

7 OpenEMM Deployment

7.1 Download

We recommend to download the deployment & runtime package of OpenEMM from https://www.agnitas.de/en/download/openemm-binaries/. This package provides an installer tool for the backend and frontend code and eases installation, updates, administration and maintenance of OpenEMM significantly.

Of course, you can also download the source code of OpenEMM from GitHub at https://github.com/agnitas-org/openemm and compile and deploy the software manually yourself. See the build instructions at the end of the OpenEMM wiki page at https://wiki.openemm.org.

7.2 OST (OpenEMM Support Tool)

Download the OpenEMM runtime tarball (file name: *openemm-runtime-<release>.tar.gz*) to */home/openemm* to create the deployment and runtime environment for OpenEMM. Change to user *openemm*, unpack the tarball and start it with

```
# su - openemm
$ tar -xvzpf openemm-runtime-<release>.tar.gz
$ OST.sh
```

At the first start *OST* checks for certain packages and settings and offers to change them.

While OpenEMM needs Python 3.11 or higher to work, *OST* itself is fine with older versions 3 of, i.e. the python3 package of the operating system would be sufficient.

OST is an interactive command line tool, which should be your preferred way to check, configure and update various components of OpenEMM.

```
Current menu: Main

Please choose (Blank => Quit):
  1. Show OpenEMM status
  2. Configuration
  3. Database Maintenance
  4. Security
  5. Install or update package from AGNITAS Website
  6. Install or update package from local file
  7. Install or update package from AGNITAS Cloud
  8. Show update history
  9. Switch OpenEMM version
 10. Restart OpenEMM
```

```
11. Send configuration and log files in email
 0. Quit
>
```

At launch time *OST* checks if kernel parameters in */etc/security/limits.conf* are big enough and, if not, offers to change them. To be able to change this file, you need to start *OST* as user *root*. After the change of the kernel parameters, you have to reboot the server.

At launch time *OST* also checks your software stack and offers to install required packages (including Java and Tomcat) and to set environment variables

- JAVA_HOME for Java (default path: */usr/lib/jvm/java*)
- CATALINA_HOME for Tomcat (default path: */home/openemm/opt/tomcat/*)
- PROXY to communicate with the Internet (optional, default: left empty)

These parameters will be written to file *setenv.sh* in directory */home/openemm/bin/*.

If one of these tools is not available on your server yet, *OST* offers to install it for you.

OST helps you to set up and configure a database connection (file *dbcfg*) including database user and password. In menu *Configuration → Change database configuration* you can enter the required properties like database type ("mariadb"), database name ("openemm"), database user (suggestion: "openemm") and your database password. Do not forget to confirm your changes with “save”!

If the database configuration file *dbcfg* does not exist in directory */home/openemm/etc/*, it is also created from scratch. A valid *dbcfg* file looks like this:

```
openemm: dbms=mariadb, host=localhost, name=openemm, user=openemm,
password=<password>, jdbc-connect=jdbc:mariadb://localhost/openemm?
zeroDateTimeBehavior=convertToNull&useUnicode=true&characterEncoding=UTF-
8, jdbc-driver=org.mariadb.jdbc.Driver
```

Please make sure, that every comma separating a key-value pair is followed by a blank. Only this format makes sure that you can use a comma within a password or other values.

If the OpenEMM database and its user does not exist, *OST* offers to set it up for you after you have entered the properties mentioned above. The empty OpenEMM database will be filled later during installation of the OpenEMM code package.

If for whatever reason the creation of the database user for OpenEMM does not work, you can create the user manually with

```
# mysql -u root -p
MariaDB> CREATE USER 'openemm'@'localhost' identified by '<password>';
MariaDB> GRANT ALL PRIVILEGES ON openemm.* TO 'openemm'@'localhost' ;
MariaDB> GRANT SELECT ON mysql.proc TO 'openemm'@'localhost';
MariaDB> GRANT RELOAD ON *.* TO 'openemm'@'localhost';
MariaDB> GRANT SUPER ON *.* TO 'openemm'@'localhost';
MariaDB> FLUSH PRIVILEGES;
```

Please replace placeholder *<password>* with a strong password.

You need the global permissions *SELECT ON mysql.proc* to read Stored Procedures, *RELOAD* to be able to flush the privileges and *SUPER* to be able to create database triggers. If you are not able to get permission *SUPER*, as a workaround you can set

```
log-bin-trust-function-creators = 1
```

in section *[mysqld]* of your database configuration file *my.cnf*.

7.2.1 Menus of OST

Menu "Show EMM status" starts a brief health check of the OpenEMM configuration on the current server and checks the integrity of the files via their checksums.

Menus "Configuration" help you to check, change or create certain settings of your OpenEMM configuration without having to deal directly with the OpenEMM database or configuration files:

```
Current menu: Configuration

Please choose (Blank => Back to Main):
  1. Configure basic environment (Java, Tomcat, Proxy)
  2. Change configuration of database connection
  3. Change basic configuration
  4. Change layout images
  5. Change client/account configuration
  6. Change jobqueue configuration
  0. Back to Main
 >
```

Sub-menu Change basic configuration offers you to define domain names for various EMM services:

Domain Name	Name of Parameter
OpenEMM GUI	system.url
Mailloop Service	system.defaultMailloopDomain
Redirect Service	system.defaultRdirDomain
Statistics Service	birt.url
Statistics Service	birt.url.intern
Webservices	webservices.url

When you call this sub-menu the first time and no value is set yet, you are asked to enter the FQDN of your OpenEMM server, including the access protocol (HTTP or HTTPS). Based on your input the values of the parameters mentioned in the table above are prefilled. You can change those values later at any time.

Property *birt.url.intern* should be set to value http://localhost:8080/birt in any case.

You should also configure various email addresses for mails from OpenEMM and notifications:

Name of Parameter	Description
mailaddress.sender	sender address for all frontend emails (like reports), also used for 2FA and error mails
mailaddress.replyto	reply-to address for all frontend emails (like reports), also used for 2FA and error mails
mailaddress.bounce	bounce address for all frontend emails (like reports), also used for 2FA and error mails (optional)
mailaddress.info.cleaner	email address that gets notifications if a nightly DB cleaning process did not finish in time
mailaddress.error	email address for EMM error notifications like DB connection error
mailaddress.frontend	email address that gets notifications for user form errors and for action script errors
mailaddress.upload.database	email address that gets notifications that a new file was uploaded in the upload menu
mailaddress.upload.support	email address that gets any instructions included in a file upload
mailaddress.report_archive	email address that gets report mails to be archived
mailaddress.support	email adress that gets messages sent via button "report a problem"
mailaddress.feature_support	email adress that gets messages sent via button "suggest a feature"

Sub-menu Change layout images offers you the possibility to change the logos of EMM for white-labeling. Just enter the file name of the logo you want to change and enter path and file name of the replacement:

```
favicon.ico          (the icon often displayed in the browsers tab,
                      besides the title)
logo.svg             (small logo in the webapplication above the
                      menu and login mask)
logo.png             (an alternative to the SVG logo)
edition_logo.png     (logo in login mask)
report_logo.png      (logo used inside pdf reports)

Please choose layout image to replace (Blank => Back):
 > logo.svg
Please enter new image filepath for replacement of 'logo.svg':
 > /home/user/image.png
```

Sub-menu Change client/account configuration offers you to change the domain names for the mailloop service and redirect service separately. Also, you should define an email address for property *contact_tech*, because OpenEMM will send certain warnings to this email address.

Menu Database Maintenance offers a backup and restore of the database present on the server.

With menu "Security" you can set a new password for admin user *emm-master* if you select option "Create new initial 'emm-master' password". You may use this option also in case you have forgotten the password for user *emm-master*.

Also, you can use option Configure TLS certificate (https) to customize Tomcat's configuration file *server.xml* with your TLS certificate files so that OpenEMM can be accessed by the HTTPS protocol. If you want to know more about the details, have a look at chapter *Apache NIO Connector*.

Menu "Install or update package from ..." supports you in downloading packages of a new OpenEMM release from various sources. Please be aware that you have to start *OST* as user *root* to be able to install or update a backend service. If you want to load a file from an external source (AGNITAS), make sure that the server *OST* is running on, is able to access this source. Otherwise download the file first with a different device, copy it to your server and install it as local file.

If you need to know which services and versions of OpenEMM were active at what time in the past, menu Show update history provides a list of all available services, their versions and the exact startup times.

Menu "Switch OpenEMM version" lets you roll back to a former version of an OpenEMM service or roll forward to a later version. However, if you switch back not to an older patch level (last 3 digits of the version number), but to an older minor or major version (like going back from 20.10 to 20.04), please be advised that the database schema is not rolled back and, therefore, some feature may not work properly because it wants to use a database field that does not exist any longer in the new database schema.

After a change of version or after an OpenEMM service update, use menu "Restart OpenEMM" to restart the OpenEMM installation on the current server (frontend and backend services, if available).

Menu "Send configuration and log files in email" helps you to create an email which sends all important OpenEMM configuration information to an email address of your choice for diagnostic purposes or as a backup. For that reason, the tool collects certain configuration and log files from the current server as well as the content of database configuration tables of your OpenEMM installation and sends them out as zipped and password-protected attachment. Since this attachment can become quite big, make sure that the receiving email address is able to handle this payload.

If *OST* reports an error during operation, re-start *OST* with option *-debug* and repeat the operation. In this case *OST* will show a stacktrace for the error, which will give you a hint what went wrong.

7.3 OpenEMM Deployment

For the deployment of the openemm code tarball you have to start *OST (OpenEMM Support Tool)* as user *root*, because some files of the tarball have to be deployed with root permissions:

```
$ sudo su -
# /home/openemm/bin/OST.sh
```

Use menu *Install or update package from AGNITAS website* to install the openemm code package. As first step, the menu offers to install an update of the runtime package, if a later version is available. Opposed to upgrading to a new major version of OpenEMM, an update to a new runtime version is always recommended.

Answer “no” to all downloads offers until the openemm code package is offered (if you started *OST* as user *root*). If an upgrade of a major version of OpenEMM is available (like 21.04) this upgrade opportunity is offered first. However, you should not blindly accept it if you are working on a production system (see section *Test and Production System* below).

During Deployment of the openemm code tarball several new directories and symlinks will be created in directory */home/openemm/*. Also, the OpenEMM database is automatically populated with its initial content.

If you do not want to install or update the code of OpenEMM, you can start *OST* as user *openemm*:

```
# su - openemm
$ OST.sh
```

7.4 Configuration

After deployment of the OpenEMM code but before launching it with the restart menu, use menu *Configuration → Change basic configuration* to set up and change the configuration of OpenEMM. Note: It is possible to launch OpenEMM without adapting the configuration, but in this case not all features of the software will work properly.

With OpenEMM 20.04 all configuration properties have been migrated from files *emm.properties, emm-ws.properties* and *Mailgun.ini* to the OpenEMM database. Therefore, you do not change these files any longer but the corresponding database entries listed in the sub-menu mentioned above. The advantage of the migration from files to database is, that further updates of OpenEMM do not overwrite any individual settings you might have made.

Sub-menu *Change basic configuration* asks you to enter the full URL of your server including protocol (http or https). From this value it deviates the settings for various essential properties.

Sub-menu Change client/account configuration offers you to change the redirect domain and the mailloop domain. These value are pre-filled by the basic configuration and you should change those values only if you know what you are doing:

- Set *rdir_domain* to the protocol and FQDN of your server, for example *https://openemm.example.com*. This domain will be used in all measurable links to redirect them through OpenEMM. This property **must** include the appropriate protocol.
- Set *mailloop_domain* to the domain of your sender address. The domain for the mailloop service must be different from the name of your OpenEMM server. It usually is a domain whose MX record points to your OpenEMM server. For example the FQDN of your sender address is *mailing.example.com* and its MX record points to *openemm.example.com* which is the address of your OpenEMM server. In this case, use domain *mailing.example.com* as domain for the sender address of your mailings. (There are other ways for configuration, described in section *Mailloop Configuration*.) If you do not configure the mailloop service, OpenEMM can only process instant bounces, i.e. you will not be able to catch all bounces. This property **must not** include any protocol.

Finally, create an initial password for admin user "emm-master" in menu *Security* → *Create new initial 'emm-master' password*. Write down the generated password and use user "emm-master" and this initial password later for your first login in the GUI of OpenEMM. At first login, you will be prompted to change your password to a new one of your choice. In case you forget your password for the admin user at a later time, you can always use this menu of *OST (OpenEMM Support Tool)* to set a new password.

7.5 Startup

Launch OpenEMM with menu *Restart OpenEMM* of *OST*. Experts may have a look at the Tomcat log during startup with

```
$ tail -f /home/openemm/logs/catalina.out
```

to check for any warnings or error messages. Be patient, because the startup process will take a minute or two.

After OpenEMM has been launched successfully, point your browser to your OpenEMM server and log in with user “emm-master” and the initial password you just created. OpenEMM will ask you to change the initial password to a new one of your choice.

If you want to install the context-sensitive online help feature of OpenEMM (including the user manual with about 500 pages), visit

- https://www.agnitas.de/en/download/openemm-manual/ (English)
- https://www.agnitas.de/download/openemm-handbuch/ (German)

to get the download link for the manual package. You can use *OST’s* menu *Install or update package from AGNITAS Cloud* to install the documentation.

7.6 Test and Production System

If OpenEMM is an important software for you (and we hope it is!) we recommend to set up a separate test system alongside your production system. This should be no probem for you, because the complete software stack for OpenEMM is open source and you do not have to pay a dime for the additional instance.

You can create a copy of your OpenEMM production database with menu *Database Maintenance → Backup MariaDB database* of *OST*. On your test system you should only create the empty OpenEMM database with menu *Configuration → Change configuration of database connection*. Now you can set up the database copy of your production system with menu *Database Maintenance → Restore MariaDB database*. But make sure that both systems use the same DBMS version to avoid any trouble.

After that, you only have to adjust the properties containing the server URL in menu *Configuration → Change basic configuration*, because they have been copied from your production database as well. Now you are ready to use your test system to try out different configuration settings, to test database backup and restore or to test an upgrade to a new major OpenEMM version – all that before applying any change to your precious production system.

7.7 Rendering of Thumbnail Images and PDF Documents

If you want to generate good-looking thumbnails and PDF files in the OpenEMM GUI, the OpenEMM server needs the installation of some more packages:

RHEL:

```
# dnf install gcc-c++ mesa-libgbm atk at-spi2-atk libXcomposite
libXdamage libXrandr libXrender libxkbcommon pango
# dnf module reset nodejs
# dnf module enable nodejs:20
# dnf install nodejs
```

SLES:

```
# zypper install gcc-c++ libgbm1 libgbm-devel
# SUSEConnect -p sle-module-web-scripting/15.4/x86_64
# zypper install nodejs18
```

Finally, **switch to user *openemm***, change to directory */home/openemm/webapps/emm/WEB-INF* and install the NPM package *puppeteer* via port 443 (HTTPS) from the NPM server (operated by GitHub/Microsoft):

```
$ cd /home/openemm/webapps/emm/WEB-INF
$ npm install
```

This means, that the OpenEMM server needs access to the Internet during deployment!

Furthermore, *puppeteer* must be able to access the browser frontend of OpenEMM from *localhost*. Therefore, make sure that your firewall rules do not block access of port 80 (HTTP) or 443 (HTTPS) from *localhost*.

8 OpenEMM Testing

For testing (and to initialize the system) you should first make sure that MariaDB contains the OpenEMM database and the DBMS is already up and running.

If you are not sure about OpenEMM's status use the first menu *Show OpenEMM status* of *OST* to check it and use menu *Restart OpenEMM* to restart the software if it does not look right.

To test the correct operation of each service we recommend

- creating a new mailing (testing the GUI of the frontend)
- sending the mailing to an existing email address (testing the backend)
- and sending it to an non-existing email address (testing the bounce management)*
- opening the mail and clicking the links (testing the redirect service of the frontend)
- checking in mailing statistics if openings and clicks were recorded (testing the redirect service and statistics service of the frontend)

*To test the bounce management you have to send the mailing not to admin or test recipients, but to normal recipients, because due to performance reasons mailings to admin and test recipients are sent out instantly without the overhead of a regular mailing. For that reason the bounce management process can not be applied to mailings for admin and test recipients.

Please note: Most statistics of OpenEMM are generated by a separate statistics service. For communication between the GUI of OpenEMM (emm service) and the statistics service, OpenEMM users **must** enable cookies in the web browser which is used to access OpenEMM. Otherwise, the GUI can not display the results provided by the statistics service.

If you hide OpenEMM behind firewalls and proxies and have problems to access the OpenEMM GUI, check Tomcat log *catalina.out* for any errors. If you see errors regarding CSRF protection, you may disable it with

```
UPDATE config_tbl SET value='false' WHERE class='security' AND
name='csrfProtection.enabled';
```

at the cost of missing CSRF protection.

8.1 Troubleshooting: OpenEMM Does Not Send Emails

The most "popular" problem of OpenEMM is, that no emails are sent out. Therefore, we have compiled a checklist with the most common reasons why OpenEMM does not send emails:

1. The mailing or the choosen target group has no recipients
2. OpenEMM has not been started. Please restart with *Restart* menu of *OST*.

3. Postfix' mail log file in */var/log/maillog* shows errors.
4. Packages *postfix* and *sendmail-milter* are not installed.
5. Postfix has not been started (check with command "systemctl status postfix" and start with "systemctl start postfix").
6. Postfix is no longer running (check with "ps aux | grep postfix/master").
7. Postfix' mailqueue at */var/spool/postfix* does not exist (solution: re-install Postfix).
8. The wrong MTA is active (Sendmail, qmail, Exim, etc.).
9. Port 25 has not been opened in the firewall.
10. Service *cron* is disabled (check with "systemctl status crond").
11. SELinux is enabled.
12. The OpenEMM server uses a virus scanner that prevents the start of the XML RPC backend process (see file *backend.log* in directory */home/openemm/log/* for an entry like *"Error: Could not find or load main class org.agnitas.backend.MailoutServerXMLRPC"*)
13. For reverse lookups, no PTR record does exist for the IP address of the OpenEMM server (so that mails are blocked as spam by mailbox providers on the receiving end).

9 OpenEMM Updates and Upgrades

With OpenEMM we make a difference between **updates** (changing to a new minor release, like from 20.10.000.050 to 20.10.000.100) and **upgrades** (changing to a new major release, like from 20.04 to 20.10).

If you plan to do an **upgrade**, we strongly recommend to **update** to the lastet minor release of your OpenEMM version, first. Only after that, execute the upgrade to the new major version. This makes sure, that the delta between the two major versions is as little as possible and the old version is well prepared for the upgrade.

If you do not operate a test environment for OpenEMM, but you apply updates and upgrades right to the production environment, **make sure to create a backup with OST → Database Maintenance → Backup MariaDB database before** (even more so, if you upgrade to a beta version of OpenEMM), because you might need the database backup in case of a roll back (see section *Rollbacks for OpenEMM* below for details).

9.1 Updating and Upgrading OpenEMM

Before you upgrade OpenEMM to a new major version, check the required software stack in section *Software Stack* first. If you have to update a component of the software stack like Java, Python or MariaDB, make sure to stop OpenEMM before by executing

```
# /home/openemm/bin/openemm.sh stop
```

to avoid any disruption or misbehaviour of OpenEMM after the update of the software component.

The OpenEMM Wiki at https://wiki.openemm.org shows the latest available versions of the openemm code package at the top.

You can download the latest versions of the packages with menu *Install or update package from AGNITAS Website* of *OST*. Please answer “no” to all downloads offers until the openemm code package is offered.

An update of the openemm code package will also update the schema of the OpenEMM database, if necessary. Please keep in mind that you have to start *OST* as user *root*, if you want to update the openemm code package.

Do not forget to restart OpenEMM with menu *Restart OpenEMM* after you have installed a new openemm code package to activate it. But do the restart at an convenient time: Do not restart OpenEMM during a dispatch of a mailing, or right after the dispatch (due to the brief downtime of the redirect service which would cause missed openings and click redirects).

When the upgrade is finished and OpenEMM has been restarted, log into the GUI of OpenEMM and check the status of the automated background processes in tab *Job list* of sub-menu *System status* in menu *Administration*.

9.2 Do not Skip Major Releases when Upgrading

If you **upgrade** an OpenEMM version **prior to 24.10**, **do not skip an upgrade of a major version**, i.e. if you use OpenEMM 23.10 and you plan to upgrade to OpenEMM 24.10, do not skip the upgrade to version 24.04! You have to install and start the intermediate version at least once. This ensures that possible migration tasks included in the intermediate OpenEMM version are executed, so that the database and configuration of OpenEMM remains in a consistent state.

To make sure that you are able to upgrade to each major release of OpenEMM, update to the runtime version of the next major release and after that download and update the openemm code package. **Do not** update the runtime package to a new major release **without** updating and executing the openemm code package, too!

If you have skipped updating openemm code packages while updating the runtime package, you have to use the switch menu to switch back to an older runtime version (see section *Rollbacks for OpenEMM* below for details).

Even if you do not want to use an intermediate version, **you have to start this version once** (like 24.04 in the above-mentioned case). This makes sure that the startup process in the OpenEMM code is executed and initiates all pending migration tasks contained in the skipped OpenEMM version.

For example, the startup process of OpenEMM 20.04 migrates all configuration properties from configuration file *emm.properties* into the database (among other things). This migration is important, because OpenEMM 20.10 reads its configuration properties from the database.

9.3 Special Advice for Upgrades to Version 22.10

OpenEMM 22.10 needs two more Python packages on all servers with a backend service. If you installed Python 3.8 from the repository (see section *Python 3 for OpenEMM*), use

```
# pip3.8 install msgpack
# pip3.8 install websockets
```

If you compiled Python yourself, use

```
# python3 -m pip install msgpack
# python3 -m pip install websockets
```

Please note that this is the last version supporting MySQL. If you want to migrate from MySQL to MariaDB, we recommend this procedure:

1. stop OpenEMM and MySQL
2. export the database content to have a backup just in case, for instance using command *mysqldump:*
   ```
   # mysqldump -aCceQx --hex-blob --routines --triggers -u root -p -r openemm.sql openemm
   ```
3. change your DBMS from MySQL 5.7 to MariaDB 10.2 (because these versions offer the maximum of compatibility: https://mariadb.com/kb/en/upgrading-from-mysql-to-mariadb)

4. run *mariadb-upgrade* to migrate the DB structure from MySQL to MariaDB
5. change file *dbcfg* in directory */opt/agnitas.com/etc* for MariaDB
6. start MariaDB
7. start OpenEMM

Make sure that the database user used by OpenEMM has the required rights. If the user name is *openemm*, we recommend to create the user this way:

```
MariaDB> CREATE USER 'openemm'@'localhost' IDENTIFIED BY '<password>';
MariaDB> GRANT ALL PRIVILEGES ON openemm.* TO openemm;
MariaDB> GRANT SUPER ON *.* TO openemm;
MariaDB> FLUSH PRIVILEGES;
```

If the OpenEMM database does not work correctly, import it from your backup instead (for instance using command *mysql*).

Now, that you are running OpenEMM with MariaDB 10.2, which is no longer supported, we recommend to update to a later version of MariaDB. See section *MariaDB* for the version we use ourselves for the OpenEMM public cloud, and see section *Upgrading MariaDB* for advice on how to upgrade a MariaDB version.

9.4 Special Advice for Upgrades to Version 23.04

Please note that OpenEMM 23.04 does no longer support MySQL 5.7. See the section before for the process to migrate from MySQL to MariaDB.

The EMM database schema for workflows has changed several years ago. With OpenEMM 23.10 we will remove the code to support the legacy format. To check if you are affected, beginning with OpenEMM 23.04, *OST* will check your workflows at startup time. If *OST* detects an inactive legacy workflow, you will see message

“Your EMM database contains inactive legacy workflows [list of workflow names]. Please delete those workflows since they will no longer work with EMM 23.10 and later.”

If *OST* detects an active legacy workflow, you will see message

“Your EMM database contains active legacy workflows [list of workflow names] which can not be migrated. Please rebuild these workflows from scratch and delete the legacy workflows since they will no longer work with EMM 23.10 and later. If you need help, please contact support.”

Please re-build those workflows before you upgrade OpenEMM to version 23.10 or later.

If you want to check manually, if you have legacy workflows in your database, check in table *workflow_reaction_tbl* if it contains entries with field *is_legacy_mode* set to value 1.

Also, the report icon will no longer be supported in workflows beginning with OpenEMM 23.10. Therefore, *OST* will check all workflows for this icon, too. If a workflow is found, you should remove the report icon before you update to OpenEMM 23.10.

9.5 Special Advice for Upgrades to Version 23.10

OpenEMM 23.10 needs an update of Python lib *paramiko* and three new Python libs. Execute

```
# python3 -m pip install -U 'paramiko>=3.2.0'
# python3 -m pip install aiosmtplib
# python3 -m pip install asyncinotify
# python3 -m pip install asyncssh
```

to update and install the required libs.

Once you have finished the upgrade to OpenEMM 23.10, you can remove the requirement of root access for the mailloop service. To do this, stop Postfix on the server running the mailloop service as user *root* with

```
# systemctl stop postfix
```

and stop the mailloop service as user mailloop with

```
$ mailloop.sh stop
```

Now, change line

```
relay_domains = /home/mailloop/var/run/relay.domains
```

in configuration file *main.cf* in directory */etc/postfix* to

```
relay_domains = hash:/home/mailloop/var/run/relay.domains
```

and restart the mailloop service as user *mailloop* with

```
$ mailloop.sh start
```

Finally restart Postfix as user *root* with

```
# systemctl start postfix
```

Please note that OpenEMM 23.10 does no longer use deprecated *wkhtmltox*, but NPM package *puppeteer* to render thumbnail images and PDF documents. Therefore, the installation of nodejs and a one-time execution of *npm* is necessary. See section *Rendering of Thumbnail Images and PDF Documents* for details.

9.6 Special Advice for Upgrades to Version 24.04 and 24.10

Please make sure to switch to Java 17 as soon as possible, because Java 11 will be no longer supported by OpenEMM 25.04.

Also, we recommend to switch to Python 3.11 as soon as possible, because Python version 3.11 will be a minimum requirement of OpenEMM 25.04.

9.7 Special Advice for Upgrades to Version 25.04

OMT (OpenEMM Maintenance Tool) was renamed to *OST* (OpenEMM Support Tool).

OpenEMM 25.04 requires Java **17** and Python **3.11** or later. Please make sure to have both versions available on your EMM servers.

MariaDB has to be at least at version 10.6.10 to work with Python 3.11. We use MariaDB version **10.11.7** for our production environment.

OpenEMM 25.04 does not support deprecated *wkhtmltox* any longer, but uses NPM package *puppeteer* to render thumbnail images and PDF documents.

After the upgrade, you **must delete** file *derby-10.14.<patch>.jar* in directory */home/openemm/tomcat/lib (<patch>* is the placeholder for the exact version).

We recommend to switch from nodejs 18 to version 20:

RHEL:

```
# dnf module reset nodejs
# dnf module enable nodejs:20
# dnf install nodejs
```

SLES:

```
# zypper remove nodejs18
# zypper install nodejs20
```

If you use Tomcat version 10.1.42 or later, you may have problems to import multiple files and see error message "FileCountLimitExceededException" in log file *catalina.out* of Tomcat. To avoid this problem, set parameters *maxParameterCount* and *maxPartCount* of the *Connector* configuration in configuration file *server.xml* to value **1024** each.

When you are finished with the EMM upgrade, log into the GUI as user *emm-master* and check in menu *Administration*, sub-menu *System Status* in tab *Show Jobqueue* that all jobs are working.

9.8 Special Advice for Upgrades to Version 25.11

If you use Tomcat version 10.1.42 or later, you may have problems to import multiple files and see error message "FileCountLimitExceededException" in log file *catalina.out* of Tomcat. To avoid this problem, set parameters *maxParameterCount* and *maxPartCount* of the *Connector* configuration in configuration file *server.xml* to value **1024** each.

Please make sure to switch to Java **21** as soon as possible, because Java 17 will no longer be supported by future versions of OpenEMM.

When you are finished with the OpenEMM upgrade, log into the GUI as user *emm-master* and check in menu *Administration*, sub-menu *System Status* in tab *Show Jobqueue* that all jobs are working.

9.9 Rollbacks for OpenEMM

If you are not happy with a version you updated or upgraded to, you can roll back OpenEMM to a former version or roll forward to a later version with menu *Switch OpenEMM version*. This menu lists the active versions of the EMM services available on the current server so that you can switch separately

- the runtime environment *RUNTIME* (including OST itsself)
- the openemm frontend (consisting of GUI service *EMM*, statistics service *STATISTICS* and webservices service *WS*)
- the openemm backend (consisting of various services packaged into *BACKEND-OPENEMM*)
- the documentation package *MANUAL* (including all context sensitive help pages)

After you have selected a specific service you get a list of all versions available on the server so that you can chose the version you want to activate.

Warning: If you switch back the openemm frontend or backend not to an older minor version (last 3 digits of the version number), but to an older major version (like going back from 20.10 to 20.04), be advised that the database schema can not be rolled back and, therefore, some feature may not work properly in case it wants to use a database field that does no longer exist in the new database schema.

Therefore, if you want to switch back OpenEMM to an older major version, we recommend to restore the database backup of the corresponding version.

If you need to know which services and versions of OpenEMM were active at what time in the past, menu *Show update history* of *OST* provides a list of all available services, their versions and the exact startup times.

Do not forget to restart OpenEMM with menu *Restart OpenEMM* after you have switched the frontend or backend version of OpenEMM.

If you get offered updates only to the latest version of OpenEMM, you have to switch back the runtime version. For instance, a runtime version 20.10 only offers you updates to versions of OpenEMM 20.10. But as soon as you have upgraded to a runtime version 21.04, you get offered upgrades to OpenEMM 21.04 as well.

9.10 Templates and Web Forms

You do not have to start from scratch when producing mailings or creating web forms in OpenEMM. At https://www.agnitas.de/en/download-center/ you can download templates and web forms which you can import into OpenEMM. **Make sure to replace in web forms any placeholders for a company ID with value "1" if this is not done automatically during import.**

9.11 Preparations before Updating MariaDB

If you plan to update your version of MariaDB, due to a bug in older versions of MariaDB (MDEV-19292), you may get an error "rowsize too large" afterwards which prevents you from using your OpenEMM database after the update of MariaDB.

Statement from MariaDB regarding this problem:

"Prior to MariaDB 10.2.26, MariaDB 10.3.17, and MariaDB 10.4.7, MariaDB doesn't properly calculate the row sizes while executing DDL. In these versions, unsafe tables can be created, even if InnoDB strict mode is enabled."

To check if you are affected by this bug, please execute SQL statement

```
SQL> SELECT count(*) FROM information_schema.innodb_sys_tables WHERE name
LIKE 'openemm/%' AND ROW_FORMAT = 'Compact';
```

If the result of this SQL statement is greater than 0, your OpenEMM database uses tables using the format which may cause "unsafe" tables. **In this case, we strongly recommend to convert all database tables to format "dynamic" with statement**

```
ALTER TABLE <table name> ROW_FORMAT = DYNAMIC;
```

before upgrading your version of MariaDB (after you made a backup of your database, of course)!

To simplify this task for you, *OST* offers an automated conversion in menu *Database Maintenance*, sub-menu *Check MariaDB table format*. Alternatively, here is a little script that does the conversion of all tables (please execute as database user *root*):

```
DELIMITER $$
create procedure tmp_convert_row ()
Begin
   DECLARE done INT DEFAULT FALSE; DECLARE dbtable varchar(100);
   declare tab_curse cursor for select SUBSTRING(name, 9) from
information_schema.innodb_sys_tables where name like 'openemm/%' and
ROW_FORMAT = 'Compact';
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    open tab_curse;
     read_loop: LOOP
        fetch tab_curse into dbtable;
    IF done THEN LEAVE read_loop; END IF;
    SET @SQLText = CONCAT('alter table ', dbtable, '
ROW_FORMAT=DYNAMIC');
        PREPARE stmt FROM @SQLText; EXECUTE stmt; DEALLOCATE PREPARE
stmt;
   end loop;
   close tab_curse;
END$$
DELIMITER ;
call tmp_convert_row ();
```

9.12 Upgrading MariaDB

MariaDB recommends these steps to upgrade a MariaDB version:

1. Modify the repository configuration, so the system's package manager installs the required version.
2. Stop MariaDB.
3. Uninstall the old version of MariaDB.
4. Install the new version of MariaDB.
5. Make any desired changes to configuration options in option files, such as *my.cnf*. This includes removing any options that are no longer supported.
6. Start MariaDB.
7. Run *mariadb-upgrade*.

10 Advanced Configuration

This chapter describes some more advanced configuration options. Please be aware that you should backup first any files you modify in case the configuration is broken afterwards.

10.1 Mailloop Configuration

The mailloop service enables OpenEMM to process bounces (and autoresponder mails) which can be sent back to the sender address hours or even days after OpenEMM dispatched the mailing. We call these late bounces “asynchronous bounces”.

You need to define a dedicated mailloop service domain name which is different from the OpenEMM server hostname (set in */etc/sysconfig/network/*). While you have to set up the A record for the OpenEMM server hostname, you have to set an MX (Mail Exchange) record for the mailloop service domain name, which points to the OpenEMM server hostname for correct mail routing.

For each new bounce filter created in the OpenEMM GUI, OpenEMM creates a new filter address, by default based on the mailloop service domain name. The OpenEMM user should define a mail forwarding for the address(es) used as sender address in its mailings, to direct all incoming response to the filter address(es) for further processing by the mailloop service (see “Alternative A” below).

In our example below the subdomain of the OpenEMM server hostname is *openemm* and the mailloop service subdomain name will be *mailloop*. The (abbreviated) DNS entry for *domain.com* should look like this:

```
openemm                IN    A     0     <ip address>
mailloop.domain.com.   IN    MX    10    openemm.domain.com.
```

Replace expression <*ip address*> with the IP address of the OpenEMM server, which also runs the mailloop service.

The first line assigns the IP address to the regular hostname, and the second line defines the MX record for the mailloop service domain name, meaning that host *openemm* accepts emails sent to host *mailloop*.

Validate your setup by using a tool like *dig* or *host*, for example

```
# host -t A openemm.domain.com
# host -t MX mailloop.domain.com
```

10.1.1 Bounce Filtering Alternatives

When you send emails and want to take advantage of the bounce management for asynchronous bounces, there are three possibilities to set up your bounce management filtering. This chapter explains the different options you have.

The *sender address* mentioned in the options below is the email address used as the sender address in your mailing or newsletter, i.e. the "From:" address (and it is also used as the "envelope from" address, if not explicitly configured otherwise).

The *filter address* is the email address hosted by the mailloop service, and it is configured in the GUI of OpenEMM (feature "bounce filter"). The default filter address is created as "reply_<unique number>" in the local part and with the domain name of the mailloop service. In the example below the domain name is "mailloop.domain.com", i.e. the standard filter address created at first would be reply_1@mailloop.domain.com.

The popular option (alternative A):

Use whatever sender address you like, example: *news@agnitas.com*. Create a filter address by setting up a bounce filter in OpenEMM (see user manual). This filter will auto-generate a filter address like reply_1@mailloop.domain.com.

Implement a forward mechanism in the email account of the sender address to forward incoming mail, which was sent back to the sender address, to the filter address of OpenEMM. If you use news@agnitas.com as sender address, set the forwarding for domain agnitas.com with

```
# echo "news: reply_1@mailloop.domain.com" >> /etc/aliases
# newaliases
```

The flow of automated responses to your mailings (like autoresponder mails and bounce notifications) starts at the email address of the recipient, goes back from there to your sender address and is forwarded from the sender address to the filter address.

The advantage of this model is that you can choose any domain for the sender address you want, but you have to implement an external forward mechanism.

The advanced option (alternative B):

Use a sender address with the domain name of the mailloop service (for example news@mailloop.domain.com). Since no real email addresses exist for this sender domain name, normally it would not be possible to reply to an email with this sender address. To forward responses to a valid email address you have to define a bounce filter:

In the GUI configuration for the bounce filter set field *Filter address* manually to your individual sender address (in this example news@mailloop.domain.com) and this address is then bound to this bounce configuration. Due to performance reasons it may take a few minutes until a newly created entry is known by the system.

The flow of automated responses to your mailings (like autoresponder mails and bounce notifications) now goes directly from the email address of the recipient to your sender address, which is equal to the filter address of the bounce filter.

The advantage of this alternative is, that no external forward mechanism is needed, but your sender address has to use the domain name of your mailloop service. Also, you may not use policy "strict" for DMARC, since the FQDN of your mailloop service will differ from the FQDN of your mail server.

The sophisticated option (alternative C):

There is a third alternative for your filter configuration which allows you to set up a filter address of your own choice without having to implement an external forward mechanism. In this case you create a bounce filter with an individual filter address of your choice, like

reply@yourdomain.com. Make sure that in the DNS record of the domain used in your filter address, the MX record points to your OpenEMM server like

```
yourdomain.com. IN    MX    10    openemm.domain.com.
```

Now you can use this filter address in a similar way as in alternative B, but with a different domain name (in this example: yourdomain.com). If you use SPF, make sure that the SPF record of this domain used in your filter address includes the domain of your OpenEMM server.

In this case the flow of automated responses to your mailings (like autoresponder mails and bounce notifications) goes directly from the email address of the recipient to your sender address, which is equal to the filter address of the bounce filter.

The advantage of this alternative is, that no external forward mechanism is needed like in alternative A and that the domain of the sender address and the filter address can differ opposed to alternative B.

Caveat: Alternatives B or C will not accept some reserved local parts as part of the filter address. These reserved local parts are currently

- reply_<number>
- aml_<number>
- fbl

10.2 Configuration of Default Settings

Menu *Configuration*, sub-menu *Change configuration in DB* allows you to change certain configuration parameters of the OpenEMM frontend like file paths, server addresses and limiting values. You should have a look at the list of parameters to understand which parameters can be changed.

In the section of parameters with prefix *mailaddress* you should define email addresses for support requests by your users and email addresses for certain notification mails. By default, these email addresses are set to an invalid sender domain and, therefore, would never leave the OpenEMM server.

Some parameters of interest can be modified directly in database table *company_info_tbl*. Parameters with prefix *max* limit certain resources to avoid an overload of the database and parameters with prefix *expire* define, after how many days certain entries are deleted from the database to limit the required storage space.

Table c*onfig_tbl* holds additional parameters valid for the instance of OpenEMM, especially URLs and file paths to components used by OpenEMM as well as system email addresses.

If the OpenEMM database holds more than 10,000 recipients and you open the recipient list you will be greeted with message *The option you selected is too large to be displayed completely. Please limit your selection to reduce the amount of recipients.*

If you want more than 10,000 recipients to be processed for the recipient list (which will take longer to display), set field *max_recipients* in database table *company_tbl* to the value you want:

```
SQL> UPDATE company_tbl SET max_recipients = 100000;
```

To increase security, OpenEMM blocks logins when the same IP address generates a certain number of failed logins. The default value for the maximum number of failed logins is 10 and the default value for the lock out time is 60 seconds. You can change both values in the database with a new entry in table *config_tbl*. Examples for max. 3 retries and 5 minutes lock time:

```
SQL> INSERT INTO config_tbl (class, name, value, creation_date,
change_date) VALUES ('loginTracking', 'webui.maxFails', '3',
current_timestamp, current_timestamp);
SQL> INSERT INTO config_tbl (class, name, value, creation_date,
change_date) VALUES ('loginTracking', 'webui.ipBlockTimeSeconds', '300',
current_timestamp, current_timestamp);
```

We recommend that you set a HSTS header (HTTP Strict Transport Security) for the responses of OpenEMM. This ensures that the browser of the OpenEMM user always uses the secure HTTPS protocol to connect with the frontend of OpenEMM and no longer permit the use of the insecure HTTP protocol.

To activate this header, use this SQL code:

```
SQL> INSERT INTO http_response_headers_tbl (header_name, header_value,
overwrite) VALUES ('Strict-Transport-Security', 'max-age=15768000', 1);
```

the *max-age* parameter defines the validity period ot this header in seconds.

If you are really into it, have a look at the source code of class *ConfigValue.java* to see, what else you can configure in OpenEMM.

If any change to the database configuration of OpenEMM does not come into effect within 5 minutes, you have to restart OpenEMM.

10.3 Configuration of Webservices

The webservice interface runs as a separate web application in directory */home/openemm/webapps/webservices*.

After OpenEMM has been launched you may request the WSDL file for the webservices via URL

```
http://<domain>/2.0/emmservices.wsdl
```

To be able to access the webservices of OpenEMM you have to create a webservice user with a password first. See the user manual for details.

10.4 Configuration of DKIM Support

OpenEMM supports DKIM keys as an essential standard to improve the deliverability of emails. DKIM key configuration and administration is done by two shell scripts in directory */home/openemm/bin*:

- *dkim-creat* lets you create and implement new DKIM keys. Its syntax is `dkim-creat <domain> <selector> <length>`
- *dkim-mgr* is the core tool to maintain DKIM keys. It is used by *dkim-creat*.

To create a new DKIM key, launch *dkim-creat* as user *merger* in your working directory with 3 parameters:

- the domain for which a DKIM key should be created (<domain>)
- a selector like "emm" to identify the correct DKIM key (<selector>)
- the bit length of the key - we recommend 2.048 bits, maximum should be 4.096 bits (<length>)

Example:

```
$ dkim-creat agnitas.com emm 2048
```

dkim-creat generates three files in your current directory:

- *<selector>@<domain>.pub*, containing the public key
- *<selector>@<domain>.priv*, containing the private key
- *install-<selector>@<domain>.sh*, a little shell script to save your private DKIM key in the EMM database

Furthermore, *dkim-creat* shows the necessary configuration of the DKIM entry with the public DKIM key in the DNS record of your sender domain. Example:

```
Creating private/public key pair emm@agnitas.com.priv/emm@agnitas.com.pub
for agnitas.com (selector emm) with 2048 bit
Installation sample for your DNS:
emm._domainkey.agnitas.com. IN TXT "v=DKIM1;
p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAtWA2bIcq3n95h7hixhTDdSt2Bjc
y209N7OOAHm/81lSY/
PbkY7n5x5LELGvMuuOfg7QVChTM5dslDmJ2EHG8TSptZRuOPCw1fW2X3J8zkJK74wLSSVUoKW
TYwefp7+WYfoY+XxLzc40DWS1kxYNSvO9KaKxctMYYiKruAtdpnDBuK/
tEbTJZBBWH55vwTmXhYbX6L2ZsKIyKjueGfx7RTm3GrVAWRTpBo9krBbM0PL4dP8h3yySr9Hv
/WwXk19qeaC0oksVKChiXpc8CCt1gdLWQ5FeqoKnc6LkMkl4th0gPY/
voJB6EQA1BC5ZNbpYxgma1UEv1bG37iKV7hJ+LhQIDAQAB"
Creating install-emm@agnitas.com.sh .. add done.
```

Execute script *install-<selector>@<domain>.sh* after you have entered your new public DKIM key in the DKIM entry of the DNS record for your sender domain. The script then saves your private DKIM key into the OpenEMM database.

When you start *dkim-mgr* without parameters it shows you the OpenDKIM key entries in your EMM database in table *dkim_key_tbl*.

10.5 Configuration of a Smart Relay Server

To improve deliverability of your mailings or for security reasons it may make sense to use a smart relay to send out emails with OpenEMM.

To configure a smart relay for OpenEMM, define in Postfix configuration file *main.cf* in directory */etc/postfix/* line

```
relayhost = [192.168.0.1]
```

where *192.168.0.1* is an example for the IP address of your smart relay server. You may also use the FQDN of your smart relay server instead of its IP address.

Please make sure that port 25 is open on your smart relay server and be advised that a smart relay complicates bounce management, especially if the MX entry of the DNS record for your sender domain does not point to your smart relay.

To make sure that the smart relay reports bounces back to EMM, you should define a forwarding configuration for the Postfix service on the smart relay server so that incoming bounces are forwarded to EMM. If the smart relay uses Postfix, we recommend a file *mail-transport* in directory */etc/postfix/* with this content:

```
news@domain1.com          reply_1@mailloop.domain.com
news@domain2.com          reply_2@mailloop.domain.com
news@domain3.com          reply_3@mailloop.domain.com
```

In this example the left side shows the sender addresses you use for your emails and the right side shows the filter addresses created by OpenEMM (in GUI menu "Response Processing").

In Postfix configuration file *main.cf* in directory */etc/postfix/* add these two lines:

```
relay_domains = domain1.com, domain2.com, domain3.com
virtual_mailbox_maps = hash:/etc/postfix/mail-transport
```

to make sure that the map in file *mail-transport* is processed by Postfix.

Activate this new forwarding configuration with

```
# postmap /etc/postfix/mail-transport
# systemctl reload postfix
```

11 OpenEMM Administration

11.1 Automated Startup

If you want OpenEMM to automatically launch at server reboot, you can use a systemd unit file for that purpose. Create a new file *openemm.service* in directory */etc/systemd/system/* with the following content (please note that the *After* properties belong into one line):

```
[Unit]
Description=OpenEMM startup script
After=var-run.mount network.target local-fs.target time-sync.target
postfix.service

[Service]
User=openemm
Group=openemm
Type=oneshot
RemainAfterExit=true
LimitNOFILE=16384
ExecStart=/home/openemm/bin/openemm.sh start
ExecStop=/home/openemm/bin/openemm.sh stop
TimeoutSec=300
StandardOutput=journal+console

[Install]
WantedBy=multi-user.target
```

After deploying this file, reload the systemd-daemon and enable the openemm service with

```
# systemctl daemon-reload
# systemctl enable openemm
```

At next server reboot, OpenEMM will be started automatically.

11.2 Jobqueue Monitoring

The OpenEMM JobQueue executes several cron-job-like background processes of OpenEMM, called JobWorkers. You can see a subset of the JobQueue data in the GUI of OpenEMM: Log in and go to menu *Administration* → Sub menu *System Status* → tab *Show Jobqueue* for details. This view is helpful to check the status of those JobWorkers.

You should make sure that you have configured an email address for error messages (see DB info below), because some JobWorkers like AutoOptimization or WorkflowReactionHandler take care of automated processes and should always be active. For other JobWorkers it makes sense to check their status at least from time to time,

because some hanging jobs like cleanup processes will have no immediate effects but can drag down performance of EMM over time.

If you do not want to check the status of EMM jobs in the frontend, you can also access the EMM database directly. For MariaDB, the SQL statement is

```
SQL> SELECT id, description, lastresult, running, nextstart, laststart
FROM Job_queue_tbl
WHERE deleted = 0 AND
( (lastresult != 'Ok') -- has errors
  OR (nextstart < CURRENT_TIMESTAMP() - INTERVAL 10 MINUTE) -- overdue
  OR ( (running = 1) AND
    (laststart < CURRENT_TIMESTAMP() - INTERVAL 4 HOUR)) -- (too) long running
);
```

To automate the jobqueue check, you may run this statement from a shell script and analyze its ouput with a monitoring software like Icinga.

Table *job_queue_tbl* lists all JobWorkers periodically executed by OpenEMM's JobQueue. You can control the execution times of the cleanup jobs via their entries in *job_queue_tbl* because the mass deletion of information can place serious strain on your database resources. Here are some information on the data fields of the *job_queue_tbl*:

- ID: id of job
- CREATED: creation date of job entry
- LASTSTART: last time the job was started
- RUNNING: status of execution (>0 = currently running)
- LASTRESULT: result of last execution (OK or NULL if no errors happend, or error message)
- STARTAFTERERROR: flag for start after error (>0 = yes)
- LASTDURATION: run time of latest execution of the job in milli seconds
- NEXTSTART: next time the job will launch
- RUNCLASS: Java class triggered by the job (example: org.agnitas.util.quartz.DBCleanerJobWorker executes the DB cleanup job)
- EMAILONERROR: email address to notify in case of an error in a job
- DELETED: flag to indicate if this job was deleted (>0 = deleted)
- INTERVAL: schedule for job execution, some format and examples:
  - Format: ****M*, example: ***00* (daily execution on the hour)
  - Format: ****0;***M*, example: ****0;***5* (execution every 5 minutes)
  - Format: *HHMM*, example: *1200* (daily execution at noon)
  - Format: *DaDaDa:HHMM*, example: *Su:1200* (execution on Sundays at noon), *MoTuWeThFr:1800* (execution from Monday to Friday at 18:00)

11.3 Database Backup

For MariaDB there exist plenty of strategies for database backups and tons of books and Internet resources on that subject. However, if you run only a medium MariaDB database with a few GByte of data and if you can live with an interruption of services of very few minutes, you may simply use tool *mysqldump*:

```
# mysqldump -aCceQx --hex-blob --routines --triggers -u root -p -r
openemm.sql openemm
```

Executed at the command line, this statement copies a database dump in a very robust format into text file *openemm.sql*. The database dump can be imported back into an empty database *openemm* simply with

```
# mysql -u root -p openemm < openemm.sql
```

Menu *Database Maintenance* of *OST* offers a backup and restore of the OpenEMM database based on these commands.

If you run a bigger MariaDB database which should not be stopped during backup time, we recommend the use of the tool *Percona XtraBackup*.

11.4 Generic Database Tuning

80% of all application performance problems are really database performance problems. If you run a big OpenEMM installation and you are not satisfied with the application's performance, here are some database tuning tips you should try.

If certain tenants of your OpenEMM database hold a long list of recipients, you may speed up database operations like calculating statistics significantly with a combined index on several fields of table *customer_1_binding_tbl.*

We recommend the following two indices in case they do not exist yet:

```
SQL> CREATE INDEX custbind1$mlid_user_cuid$idx ON customer_1_binding_tbl
(mailinglist_id, user_status, customer_id);
SQL> CREATE INDEX custbind1$user_mlid_cuid$idx ON customer_1_binding_tbl
(user_status, user_type, mailinglist_id, customer_id);
```

If you use any other recipient profile field than *email* for duplicate checks in imports, you should put an index on this field in *customer_1_tbl* to significantly speed up file imports:

```
SQL> CREATE INDEX cust1$<fieldname>$idx ON customer_1_tbl (<fieldname>) ;
```

Replace placeholder <*fieldname*> with the name of the recipient profile field you use for duplicate checks.

11.5 MariaDB Database Tuning

InnoDB is the default storage engine of MariaDB. While InnoDB supports row locking and real transactions for better crash protection, the internal data structure is more complex compared to simpler storage engines like MyISAM. This leads to larger table sizes, slower

writes, slower full table scans and slower handling of BLOBs and CLOBs. Also, backup and recovery via *mysqldump/mysql* is slower.

To get the best performance from MariaDB and because the configuration parameters of storage engine InnoDB can make a big difference, you should define at least the following properties in section *[mysqld]* of MariaDB's configuration file *my.cnf* (usually found in directory */etc*):

- *innodb_buffer_pool_size*: default value is only 128 MByte, which is way too small for bigger databases with lots of InnoDB tables. **If your EMM database runs on a dedicated server**, *innodb_buffer_pool_size* should be set to **50%** of the RAM size of this DBMS server if this server has **less than 32 GByte RAM**. If the DBMS server has **32 GByte RAM or more**, set *innodb_buffer_pool_size* to **75%** of the server RAM size.
- *innodb_log_file_size*: default value is 96 MByte. Its value should be set to **25%** of the size of *innodb_buffer_pool_size*, but not higher than **256 MByte** to limit recovery time after a database crash.
- *innodb_file_per_table*: By default the InnoDB engine saves all table data into system tablespace file *ibdata1* in directory */var/lib/mysql*. Set *innodb_file_per_table* to value **1** to force InnoDB to create a separate file for each database table.
- *innodb_lock_wait_timeout*: Default value is 50 (seconds). For big databases this value is too small. We recommend to use **1800** (30 minutes) or even **3600** (60 minutes) for this value.
- *innodb_rollback_on_timeout*: When the timeout defined in *innodb_lock_wait_timeout* is reached, by default only the blocking statement is rolled back, **not** the entire transaction. This could lead to data inconsistencies. Therefore, we recommend to set *innodb_rollback_on_timeout* to value **1** to enforce rollbacks of the entire transactions.

Restart MariaDB to activate these changes.

You can check if the new values are active with

```
sql> SHOW VARIABLES LIKE 'innodb_%';
```

within MariaDB.

If you want to know more about the InnoDB configuration parameters, we recommend to have a look at https://mariadb.com/kb/en/innodb-system-variables.

11.6 OpenEMM Recovery after Database Downtime

If the OpenEMM database was down or the database connection was lost (for whatever reason), please follow these steps to achieve the maximum data integrity:

1. Stop OpenEMM. If the database downtime was planned, stop OpenEMM before database shutdown, of course. You stop OpenEMM by executing

```
# /home/openemm/bin/openemm.sh stop
```

2. After the OpenEMM database is up and running, re-start OpenEMM with

```
# /home/openemm/bin/openemm.sh start
```

11.7 Stopping the Sending in Case of Emergency

A mail dispatach can be stopped in the GUI of OpenEMM. But if you want to dig deeper and stop the sending of a certain mailing at the command line level, it is important to understand the process of its generation, distribution and dispatch:

Meta mail packages are generated by backend. These files are located in directory

```
/home/openemm/var/spool/META
```

and one part of the file name is the ID of the mailing. If files with the ID of the mailing to be stopped, are removed, they are no longer distributed by the backend for dispatch to its mailer service. However, to avoid internal conflicts, you have to stop process *pickdist* before deleting the files with

```
# /home/openemm/bin/pickdist.sh stop
```

and you have to re-start *pickdist* afterwards with

```
# /home/openemm/bin/pickdist.sh start
```

If the backend has already distributed meta mail packages for processing by its mailer service, you will find the packages which have not been processed yet or which are in process right now in directory

```
/home/openemm/var/spool/mail/incoming
```

If a file is already in process, deleting it does not help. You may check if a file is in process with

```
# ps aux | grep xmlback
```

If you see process *xmlback* with a file name as argument, this file is already in process, i.e. process *xmlback* generates the final email files out of the meta mail package.

Otherwise you may delete the file(s). In this case you should stop process *pickdist* before deleting the files with

```
# /home/openemm/bin/pickdist.sh stop
```

and you have to re-start *pickdist* afterwards with

```
# /home/openemm/bin/pickdist.sh start
```

If a meta mail package is in process by the mailer service, there is no easy way to delete the email files generated by these packages. You may stop MTA Postfix, check the mail queues and manually delete the files of all emails belonging to the mailing to be stopped.

Please be aware that the statistics for a certain mailing in the GUI of OpenEMM may not be correct if you manually delete meta mail packages of this mailing or email files of this mailing from the MTA queues.

11.8 Out of Memory

If you work with big lists and experience an error message like

```
Java.lang.OutOfMemoryError: Java heap space
```

you have to allocate more memory to the Java Virtual machine (JVM). You can increase the minimum and maximum memory in file *emm.sh.additional.properties* (which overwrites settings of *emm.sh*) in directory */home/console/bin* or */home/rdir/bin* by increasing the values of parameters **-Xms** for minimum and **-Xmx** for maximum memory in variable *JAVA_OPTS_EXTERNAL*. If you have allocated all memory available and the error remains, you should increase your server RAM to at least 2 GByte (better: 4 GByte) and modify the parameters accordingly.

11.9 Log Rotation

To prevent the Tomcat log from filling up the hard disk of the OpenEMM server, you may install a log rotation to get rid of old log files. Create file *tomcat-openemm* in directory */etc/logrotate.d* with this suggested content:

```
/home/openemm/logs/catalina.out {
  copytruncate
  daily
  rotate 7
  compress
  dateext
  size 10k
  missingok
  sharedscripts
  postrotate
    ##########################################################################
    # zip files older than 180 min and delete access_logs older than 90 days
    ##########################################################################
    find /home/openemm/logs/access -name "*.log" -mmin +180 -exec gzip -9 {} \;
    find /home/openemm/logs/access -name "*.log.gz" -mtime +90 -exec rm {} \;
    find /home/openemm/logs -name "*.gz" -mtime +10 -exec rm {} \;
    find /home/openemm/logs -name "*manager*.log" -mmin +180 -exec rm {} \;
    find /home/openemm/logs -name "localhost*.log" -mmin +180 -exec rm {} \;

    ################################################
    # delete files older than 5 days in logs/webapps/
    ################################################
    find /home/openemm/logs/webapps -type f -mtime +5 -exec rm {} \;

    ##############################
    # Delete old ARCHIVES > 10 days
    ##############################
```

```
    find /home/openemm/var/spool/ARCHIVE -type d -mtime +10 2>/dev/null | xargs
-r rm -fr

    ###################################
    # Delete old logfiles from backend
    ###################################
    find /home/openemm/var/log -name "*.log" -mtime +30 -exec rm {} \;
  endscript
}
```

For redirect servers you should replace value *10k* in line 6 to *1M* and replace *console* with *rdir* in all paths. You also may shorten the various retention times to your individual needs.

11.10 Changing Security-Related Files

OpenEMM passwords are saved in the database not only encrypted but salted as well. The salt file with the file extension *salt* is located in directory */home/openemm/conf/keys*.

If you want to change the salt file, please do it before you start operating OpenEMM, because otherwise all saved passwords will not work any longer. For generating a new salt just save a string of letters, digits and other characters of the ASCII character set (decimal values 33 to 126) with a maximum length of 32 characters in a simple text file and name the new file like the old salt file. You may change the file name in configuration file *emm.properties* of the corresponding service.

To prevent access of the statistics service by a random server, the statistics service is accessible via HTTPS protocol only, uses a private key and grants access only to those servers providing the corresponding public key. This public key is provided by the GUI service. The private key in file *birt_private.pem* is located in directory */home/openemm/conf/keys/*. The corresponding public key in file *birt_public.pem* is located in the same directory, since in OpenEMM GUI and statistics service operate on the same server.

If you think that the keys are not safe (enough) for your purpose, you may replace them and restart OpenEMM.

12 Apache NIO Connector

While it is technically possible to access OpenEMM with the HTTP protocol, this is certainly not recommended for production environments. One could even argue that is illegal in EU countries where the GDPR is in force. To use OpenEMM with secure HTTPS connections, for Tomcat you should use the NIO Connector.

OST provides template file *server.xml.template* in directory */home/openemm/conf* as basis for your individual *server.xml* file. Configuration of the NIO Connector is done by changing the connector type and its properties for Tomcat in tag *Connector* of Tomcat's server configuration file *server.xml* like this:

```
<Connector
  port="8443"
  protocol="org.apache.coyote.http11.Http11NioProtocol"
  scheme="https"
  secure="true"
  SSLEnabled="true"
  disableUploadTimeout="true"
  acceptCount="100"
  connectionTimeout="20000"
  maxThreads="1000"
  enableLookups="false"
  useBodyEncodingForURI="true"
  server="<server_name>">
  <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
  <SSLHostConfig
    disableCompression="true"
    honorCipherOrder="true"
    ciphers="ECDHE-ECDSA-AES256-GCM-SHA384 ECDHE-RSA-AES256-GCM-SHA384
ECDHE-ECDSA-CHACHA20-POLY1305 ECDHE-RSA-CHACHA20-POLY1305 ECDHE-ECDSA-
AES128-GCM-SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-ECDSA-AES256-SHA384
ECDHE-RSA-AES256-SHA384 ECDHE-ECDSA-AES128-SHA256 ECDHE-RSA-AES128-
SHA256"
    protocols="all">
    <Certificate
      certificateChainFile="<tls-ca-keychain-bundle.file>"
      certificateFile="<tls-certificate.file>"
      certificateKeyFile="<tls-private-key.file>" />
  </SSLHostConfig>
</Connector>
```

Replace placeholder *<server_name>* with the FQDN of your OpenEMM server. Replace *<tls-ca-keychain-bundle.file>* with the path and name of your *cacert* file. Replace *<tls-certificate.file>* with the path and name of your *crt* file. And replace *<tls-private-key.file>* with the path and name of your *key* file.

The modified *server.xml* file belongs into Tomcat's configuration directory */home/openemm/conf*. Alternatively, you may use option *Configure TLS certificate (https)* of *OST's* Security menu to configure file server.xml (see section *Menus of OST* for details).

Do not forget to activate a port forwarding from port 443 to 8443 as described in section *Firewall Configuration*, because Tomcat uses port 8443 for HTTPS traffic by default.

13 Glossary

13.1 Bounce Management

OpeEMM's automated bounce management provides you with the capability to keep your mailing lists clean and up-to-date automatically. A bounce message is an error message, which is sent from a mail server on the recipient's side to the sender if an email is not deliverable. Bounce management administers emails which are undeliverable temporarily (soft bounce) or permanently (hard bounce). It also filters error messages and autoresponder mails.

OpenEMM does not treat all hardbounce messages reported by remote mail servers as hardbounce. In fact, some messages are only treated as softbounces, although their bounce codes starting with 5 would indicate a hardbounce.

The reason for this kind of ignorant behaviour ist intentional, because some mail servers are not properly configured regarding the generation of hardbounce messages and mistakenly report permanent delivery errors - some even by intention to pretend that certain email addresses do not exist. If OpenEMM would handle those fake hardbounce messages as real hardbounces email addresses of existing recipients would be disabled. As result, we only try to accept bounces as hardbounces which are really proved to be hardbounces.

If you want to modify the bounce management of OpenEMM, file *bav.rule* in directory */home/openemm/lib* is the right place. This file lists in section *[hard]* bounce messages which are recognized as hardbounces, and section *[soft]* lists bounce messages recognized as softbounces. The messages are formatted as regular expressions to allow, among others, the use of wildcards. You may add your own set of messages here.

By the way, if a hardbounce message is recognized as a softbounce even if it is a real hardbounce, this is not a problem. Because a real hardbounce is reported for each mailing again and is counted as a softbounce each time, it will be finally caught by the softbounce scoring of OpenEMM and converted to a hardbounce in the end.

13.2 DNS

DNS is the abbreviation for Domain Name System. This system forwards requests directed to a FQDN to a certain IP address. Each entry in the DNS maps the IP address of a server to a human readable address. Example: In place of the IP address 83.169.23.100, which points to the AGNITAS website, you may use the DNS address *www.agnitas.com*, which is much more convenient (for a human).

13.3 FQDN

A Fully Qualified Domain Name (FQDN) links to an IP address of a server. The FQDN may be composed of letters and numbers and by using this option nobody has to remember the difficult number sequence (IP). A FQDN is divided in three levels:

- The affix of the domain is the Top Level Domain (TLD). Example: *com*, *org* or *de*
- The domain name will be inserted in front of the TLD. Example: *agnitas*
- The FQDN starts with the *hostname*. For webpages this is very often *www*

Example: The FQDN *www.yourdomain.com* is composed of

- *www* = hostname
- *yourdomain* = domain name
- *com* = TLD

As you can see, the FQDN consists of the hostname, the domain name and the top level domain separated by dots. The combination of domain name and TLD is commonly referred as *domain*. The FQDN can be expanded by a *subdomain* (like *miami*). The subdomain will be inserted between the hostname and the domain. Example: www.miami.yourdomain.com .

13.4 Softbounce Scoring

If an email address generates lots of softbounces (temporary delivery problems) this is actually an indication that the email address is undeliverable permanently (hardbounce). OpenEMM provides softbounce scoring to identify those email addresses and to convert them to hardbounces.

The rules for converting a softbounce to a hardbounce work like this:

1. Select all email addresses in the softbounce table which generated more than 40 softbounces and where the time-lag between the first and last bounce is longer than 30 days.
2. If no mail opening or link click was registered within the last 30 days for an email address which matchs the before-mentioned conditions, this address is flagged as a hardbounce.
3. If at least one opening or click was registered within the last 30 days, this address is removed from the softbounce table, i.e. its bounce count is reset to zero.

Third Party Licenses

OpenEMM relies on several great open source frameworks, libraries and tools. In order to give due credit to those fine projects, you will find the individual licenses of the open source projects used by OpenEMM in subdirectory */home/openemm/webapps/emm/licences*.