OTRS Update and Installation Guide

Release 8.0

OTRS AG

May 07, 2020

Contents

This work is copyrighted by OTRS AG (https://otrs.com), Zimmersmühlenweg 11, 61440 Oberursel, Germany.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be found on the GNU website.

Introduction

OTRS (Open Technology – Real Service) is an open source ticket request system with many features to manage customer telephone calls and emails. It is distributed under the GNU General Public License (GPL) and tested on various Linux platforms. Do you receive many e-mails and want to answer them with a team of agents? You're going to love OTRS!

1.1 About This Manual

This manual is intended for use by system administrators. The chapters describe the installation and updating of the OTRS software.

There is no graphical user interface for installation and updating. System administrators have to follow the steps described in the following chapters.

All console commands look like `username> command-to-execute`. Username indicates the user account of the operating system, which need to use to execute the command. If a command starts with `root>`, you have to execute the command as a user who has root permissions. If a command starts with `otrs>`, you have to execute the command as the user created for OTRS.

> **Warning:** Don't select `username>` when you copy the command and paste it to the shell. Otherwise you will get an error.

We supposed that OTRS will be installed to `/opt/otrs`. If you want to install OTRS to a different directory, then you have to change the path in the commands or create a symbolic link to this directory.

```
root> ln -s /path/to/otrs /opt/otrs
```

Hardware and Software Requirements

OTRS can be installed on Linux and on other Unix derivates (e.g. OpenBSD or FreeBSD). To run OTRS on Microsoft Windows is not possible.

To run OTRS, you' ll also need to use a web server as reverse proxy and a database server. Apart from that, you should install Perl and/or install some additional Perl modules on the OTRS machine.

Perl must be installed on the same machine as OTRS. The database back end and the web server may be installed locally or on another host.

For Perl, you will need some additional modules which can be installed either with the Perl shell and CPAN, or via the package manager of your operating system (rpm, yast, apt-get).

OTRS has a console command to check the environment and the missing modules.

```
otrs> /opt/otrs/bin/otrs.CheckEnvironment.pl
```

If some packages are missing, you can get an install command for your operating system, if you run the script with `--list` option.

```
otrs> /opt/otrs/bin/otrs.CheckEnvironment.pl --list
```

If all needed packages are installed, the output of the environment check script shows the installed packages and the version numbers.

```
Checking for Perl Modules:
  o Archive::Tar.....................ok (v2.24)
  o Archive::Zip.....................ok (v1.63)
  o Crypt::Eksblowfish::Bcrypt.......ok (v0.009)
  o Crypt::SSLeay....................ok (v0.73_06)
  o CryptX...........................ok (v0.061)
  o Date::Format.....................ok (v2.24)
  o DateTime.........................ok (v1.50)
  o DBI..............................ok (v1.641)
  o DBD::mysql.......................ok (v4.046)
```

(continues on next page)

```
  o DBD::ODBC........................Not installed!  Use: 'apt-get install -y␣
→libdbd-odbc-perl' (optional - Required to connect to a MS-SQL database.)
  o DBD::Oracle......................Not installed!  Use: 'cpan DBD::Oracle'␣
→(optional - Required to connect to a Oracle database.)
  o DBD::Pg.........................Not installed!  Use: 'apt-get install -y␣
→libdbd-pg-perl' (optional - Required to connect to a PostgreSQL database.)
  o Digest::SHA.......................ok (v5.96)
  o Encode::HanExtra..................ok (v0.23)
  o EV................................ok (v4.22)
  o IO::Socket::SSL...................ok (v2.060)
  o JSON::XS..........................ok (v3.04)
  o List::Util::XS....................ok (v1.46_02)
  o LWP::UserAgent....................ok (v6.35)
  o Mail::IMAPClient..................ok (v3.39)
    o Authen::SASL....................ok (v2.16)
    o Authen::NTLM....................ok (v1.09)
  o Moose.............................ok (v2.2011)
  o Net::DNS..........................ok (v1.17)
  o Net::LDAP.........................ok (v0.65)
  o Search::Elasticsearch.............ok (v6.00)
  o Specio............................ok (v0.42)
  o Specio::Subs......................ok (v0.42)
  o Template..........................ok (v2.27)
  o Template::Stash::XS...............ok (undef)
  o Text::CSV_XS......................ok (v1.36)
  o Time::HiRes.......................ok (v1.9741)
  o XML::LibXML.......................ok (v2.0132)
  o XML::LibXSLT......................ok (v1.96)
  o XML::Parser.......................ok (v2.44)
  o YAML::XS..........................ok (v0.74)

Checking for External Programs:
  o GnuPG.............................ok (v2.2.8)
  o npm...............................ok (v5.8.0)
    o Node.js.........................ok (v8.11.4)
  o OpenSSL...........................ok (v1.1.1/OpenSSL)
```

2.1 Hardware Requirements

Hardware requirements highly depend on the usage of OTRS. OTRS can be used to process a few tickets per month or to process hundreds of tickets per day. The storage requirement also depends on the number of tickets and size of attachments.

We recommend using a machine with:

- AMD Ryzen 7 3700X Octa core or comparable CPU
- 64 GB RAM
- 2 × 1 TB NVMe SSD (Software-RAID 1)
- Gigabit LAN

2.2 Software requirements

Perl

- Perl 5.16.0 or higher
- Perl packages listed by `/opt/otrs/bin/otrs.CheckEnvironment.pl` console command

Web Servers

- Apache2
- nginx
- Any other web server that can be used as a reverse proxy

Databases

- MySQL 5.7 or higher
- MariaDB 10.2 or higher
- PostgreSQL 9.2 or higher
- Oracle 10g or higher

Other dependencies

- Elasticsearch from version 6.5.x to 6.8.x (higher versions are not supported)
- Elasticsearch modules `analysis-icu` and `ingest-attachment`
- Node.js 8.9 or higher

Web browsers

- Apple Safari version 7 or higher
- Google Chrome
- Microsoft Internet Explorer 11
- Microsoft Edge
- Mozilla Firefox version 32 or higher
- Any other modern web browser with JavaScript support

Note: OTRS 9 will not support Internet Explorer anymore.

Installation

This chapter describes the installation and basic configuration of the central OTRS framework.

Follow the detailed steps in this chapter to install OTRS on your server. You can then use its web interface to login and administer the system.

3.1 Preparation: Disable SELinux

Note: If your system uses SELinux, you should disable it, otherwise OTRS will not work correctly.

Here's how to disable SELinux for RHEL/CentOS/Fedora.

1. Configure `SELINUX=disabled` in the `/etc/selinux/config` file:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

2. Reboot your system. After reboot, confirm that the `getenforce` command returns *Disabled*:

```
root> getenforce
Disabled
```

3.2 Step 1: Unpack and Install the Application

You can obtain either `otrs-x.y.z.tar.gz` or `otrs-x.y.z.tar.bz2`. Unpack the source archive (for example, using `tar`) into the directory `/opt`, and create a symbolic link `/opt/otrs` that points to `/opt/otrs-x.y.z`. **Do not forget** to replace the version numbers!

Note: Package `bzip2` is not installed in some systems by default. Make sure, that `bzip2` is installed before unpacking `otrs-x.y.z.tar.bz2`.

Unpack command for `otrs-x.y.z.tar.gz`:

```
root> tar -xzf otrs-x.y.z.tar.gz -C /opt
```

Unpack command for `otrs-x.y.z.tar.bz2`:

```
root> tar -xjf otrs-x.y.z.tar.bz2 -C /opt
```

It is recommended to create a symbolic link named `/opt/otrs` that always points to the latest OTRS version. Using symbolic link makes easy to manage the OTRS updates, because you can leave untouched the directory of the previous version, only the symbolic link needs to change. If you need to revert the update, you can change the target of the symbolic link back.

Execute this command to create a symbolic link:

```
root> ln -fns /opt/otrs-x.y.z /opt/otrs
```

3.3 Step 2: Install Additional Programs and Perl Modules

Use the following script to get an overview of all installed and required CPAN modules and other external dependencies.

```
root> perl /opt/otrs/bin/otrs.CheckEnvironment.pl
Checking for Perl Modules:
  o Archive::Tar.....................ok (v1.90)
  o Archive::Zip.....................ok (v1.37)
  o Crypt::Eksblowfish::Bcrypt.......ok (v0.009)
...
```

Note: Please note that OTRS requires a working Perl installation with all *core* modules such as the module `version`. These modules are not explicitly checked by the script. You may need to install a `perl-core` package on some systems like RHEL that do not install the Perl core packages by default.

To install the required and optional packages, you can use either CPAN or the package manager of your Linux distribution.

Execute this command to get an install command to install the missing dependencies:

```
root> /opt/otrs/bin/otrs.CheckEnvironment.pl --list
```

OTRS requires a supported stable version of Node.js to be installed. Please refer to the Node.js installation instructions.

3.4 Step 3: Create the OTRS User

Create a dedicated user for OTRS within its own group:

```
root> useradd -r -U -d /opt/otrs -c 'OTRS user' otrs -s /bin/bash
```

3.5 Step 4: Activate the Default Configuration File

There is an OTRS configuration file bundled in `$OTRS_HOME/Kernel/Config.pm.dist`. You must activate it by copying it without the `.dist` filename extension.

```
root> cp /opt/otrs/Kernel/Config.pm.dist /opt/otrs/Kernel/Config.pm
```

3.6 Step 5: Configure the Apache Web Server

OTRS comes with an own built-in web server that is used behind Apache as a reverse proxy (or any other reverse proxy server). A few Apache modules are needed for correct operation:

- `mod_headers`
- `mod_proxy`
- `mod_proxy_http`
- `mod_proxy_wstunnel`

On some systems like Debian and SuSE, these modules need to be specifically enabled:

```
root> a2enmod headers
root> a2enmod proxy
root> a2enmod proxy_http
root> a2enmod proxy_wstunnel
```

Most Apache installations have a `conf.d` directory included. On Linux systems you can usually find this directory under `/etc/apache` or `/etc/apache2`. Log in as root, change to the `conf.d` directory and link the appropriate template in `/opt/otrs/scripts/apache2-httpd.include.conf` to a file called `zzz_otrs.conf` in the Apache configuration directory (to make sure it is loaded after the other configurations).

```
# Debian/Ubuntu:
root> ln -s /opt/otrs/scripts/apache2-httpd.include.conf /etc/apache2/sites-
↪enabled/zzz_otrs.conf
```

Now you can restart your web server to load the new configuration settings. On most systems you can do that with the command:

```
root> systemctl restart apache2.service
```

3.7 Step 6: Set File Permissions

Please execute the following command to set the file and directory permissions for OTRS. It will try to detect the correct user and group settings needed for your setup.

```
root> /opt/otrs/bin/otrs.SetPermissions.pl
```

3.8 Step 7: Setup the Database

The following steps need to be taken to setup the database for OTRS properly:

- Create a dedicated database user and database.
- Create the database structure.
- Insert the initial data.
- Configure the database connection in `Kernel/Config.pm`.

Note: Please note that OTRS requires `utf8mb4` as database storage encoding.

3.8.1 MySQL or MariaDB

Log in to MySQL console as database admin user:

```
root> mysql -uroot -p
```

Create a database:

```
mysql> CREATE DATABASE otrs CHARACTER SET utf8mb4;
```

Special database user handling is needed for MySQL 8, as the default `caching_sha2_password` can only be used over secure connections. Create a database user in MySQL 8:

```
mysql> CREATE USER 'otrs'@'localhost' IDENTIFIED WITH mysql_native_password
→BY 'choose-your-password';
```

Create a database user in older MySQL versions:

```
mysql> CREATE USER 'otrs'@'localhost' IDENTIFIED BY 'choose-your-password';
```

Assign user privileges to the new database:

```
mysql> GRANT ALL PRIVILEGES ON otrs.* TO 'otrs'@'localhost';
mysql> FLUSH PRIVILEGES;
mysql> quit
```

Run the following commands on the shell to create schema and insert data:

```
root> mysql -uroot -p otrs < /opt/otrs/scripts/database/otrs-schema.mysql.sql
root> mysql -uroot -p otrs < /opt/otrs/scripts/database/otrs-initial_insert.
↪mysql.sql
root> mysql -uroot -p otrs < /opt/otrs/scripts/database/otrs-schema-post.
↪mysql.sql
```

Configure database settings in `Kernel/Config.pm`:

```
$Self->{DatabaseHost} = '127.0.0.1';
$Self->{Database}     = 'otrs';
$Self->{DatabaseUser} = 'otrs';
$Self->{DatabasePw}   = 'choose-your-password';
$Self->{DatabaseDSN}  = "DBI:mysql:database=$Self->{Database};host=$Self->
↪{DatabaseHost};";
```

Note: The following configuration settings are recommended for MySQL setups. Please add the following lines to `/etc/my.cnf` under the `[mysqld]` section:

```
max_allowed_packet  = 64M
query_cache_size    = 32M
innodb_log_file_size = 256M
```

3.8.2 PostgreSQL

Note: We assume, that OTRS and PostgreSQL server run on the same machine and PostgreSQL uses *Peer* authentication method. For more information see the Client Authentication section in the PostgreSQL manual.

Switch to `postgres` user:

```
root> su - postgres
```

Create a database user:

```
postgres> createuser otrs
```

Create a database:

```
postgres> createdb --encoding=UTF8 --owner=otrs otrs
```

Run the following commands on the shell to create schema and insert data:

```
otrs> psql < /opt/otrs/scripts/database/otrs-schema.postgresql.sql
otrs> psql < /opt/otrs/scripts/database/otrs-initial_insert.postgresql.sql
otrs> psql < /opt/otrs/scripts/database/otrs-schema-post.postgresql.sql
```

Configure database settings in `Kernel/Config.pm`:

```
$Self->{DatabaseHost} = '127.0.0.1';
$Self->{Database}     = 'otrs';
$Self->{DatabaseUser} = 'otrs';
$Self->{DatabasePw}   = 'choose-your-password';
$Self->{DatabaseDSN}  = "DBI:Pg:dbname=$Self->{Database};host=$Self->
↪{DatabaseHost};";
```

3.8.3 Finishing the Database Setup

To verify your database setup, run the following command:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Database::Check
Trying to connect to database 'DBI:Pg:dbname=otrs;host=localhost' with user
↪'otrs'...
Connection successful.
```

Once the database is configured correctly, please initialize the system configuration with the following command:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Config::Rebuild
Rebuilding the system configuration...
Done.
```

Note: For security reasons, please change the default password of the admin user `root@localhost` by generating a random password:

```
otrs> /opt/otrs/bin/otrs.Console.pl Admin::User::SetPassword root@localhost
Generated password 'rtB98S55kuc9'.
Successfully set password for user 'root@localhost'.
```

You can also choose to set your own password:

```
otrs> /opt/otrs/bin/otrs.Console.pl Admin::User::SetPassword root@localhost␣
↪your-own-password
Successfully set password for user 'root@localhost'
```

3.9 Step 8: Setup Elasticsearch Cluster

OTRS requires an active cluster of Elasticsearch (only versions between 6.5.x and 6.8.y are supported, newer versions are not supported). The easiest way is to setup Elasticsearch on the same host as OTRS and binding it to its default port. With that, no further configuration in OTRS is needed.

Additionally, OTRS requires plugins to be installed into Elasticsearch:

```
root> /usr/share/elasticsearch/bin/elasticsearch-plugin install --batch␣
↪ingest-attachment
root> /usr/share/elasticsearch/bin/elasticsearch-plugin install --batch␣
↪analysis-icu
```

Note: Restart Elasticsearch afterwards, or indexes will not be built.

To verify the Elasticsearch installation, you can use the following command:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::DocumentSearch::Check
Trying to connect to cluster...
  Connection successful.
```

3.10 Step 9: Start the OTRS Daemon and Web Server

The new OTRS daemon is responsible for handling any asynchronous and recurring tasks in OTRS. The built-in OTRS web server process handles the web requests handed over from Apache. Both processes must be started from the `otrs` user.

```
otrs> /opt/otrs/bin/otrs.Daemon.pl start
otrs> /opt/otrs/bin/otrs.WebServer.pl
```

3.11 Step 10: First Login

Now you are ready to login to your system at http://localhost/otrs/index.pl as user `root@localhost` with the password that was generated (see above).

Use http://localhost to access the external interface.

3.12 Step 11: Setup Systemd Files

OTRS comes with example systemd configuration files that can be used to make sure that the OTRS daemon and web server are started automatically after the system starts.

```
root> cd /opt/otrs/scripts/systemd
root> for UNIT in *.service; do cp -vf $UNIT /usr/lib/systemd/system/;␣
→systemctl enable $UNIT; done
```

With this step, the basic system setup is finished.

3.13 Step 12: Setup Bash Auto-Completion (optional)

All regular OTRS command line operations happen via the OTRS console interface. This provides an auto completion for the bash shell which makes finding the right command and options much easier.

You can activate the bash auto-completion by installing the package `bash-completion`. It will automatically detect and load the file `/opt/otrs/.bash_completion` for the `otrs` user.

After restarting your shell, you can just type this command followed by TAB, and it will list all available commands:

```
otrs> /opt/otrs/bin/otrs.Console.pl
```

If you type a few characters of the command name, TAB will show all matching commands. After typing a complete command, all possible options and arguments will be shown by pressing TAB.

Note: If you have problems, you can add the following line to your `~/.bashrc` to execute the commands from the file.

```
source /opt/otrs/.bash_completion
```

3.14 Step 13: Further Information

We advise you to read the OTRS *Performance Tuning* chapter.

Updating

Note: It is highly recommended to perform a test update on a separate testing machine first.

Updating from an earlier version of OTRS 8 You can update directly from any previous to the latest available patch level release.

Updating from OTRS 7 You can update from any OTRS 7 patch level release to the latest available OTRS 8 patch level release.

Updating from OTRS 6 or earlier You cannot update from OTRS 6 or earlier directly to OTRS 8. Full updates to all available minor versions have to be made sequentially instead. For example, if you come from OTRS 5, you first have to perform a full update to OTRS 6, then to OTRS 7 and finally to OTRS 8.

See also:

See the admin manual of the previous versions of OTRS for the update instructions.

4.1 Step 1: Stop All Relevant Services and the OTRS Daemon

Please make sure there are no more running services or cron jobs that try to access OTRS. This will depend on your service configuration and OTRS version.

```
root> systemctl stop postfix
root> systemctl stop apache2
root> systemctl stop otrs-daemon
root> systemctl stop otrs-webserver
```

4.2 Step 2: Backup Files and Database

Create a backup of the following files and folders:

- `Kernel/Config.pm`
- `Kernel/Config/Files/User/*.pm`
- `Kernel/WebApp.conf`
- `var/*`
- as well as the database

Warning: Don't proceed without a complete backup of your system. Use the *Backup* script for this.

4.3 Step 3: Install the New Release

You can obtain either `otrs-x.y.z.tar.gz` or `otrs-x.y.z.tar.bz2`. Unpack the source archive (for example, using `tar`) into the directory `/opt`, and create a symbolic link `/opt/otrs` that points to `/opt/otrs-x.y.z`. **Do not forget** to replace the version numbers!

Note: Package `bzip2` is not installed in some systems by default. Make sure, that `bzip2` is installed before unpacking `otrs-x.y.z.tar.bz2`.

Unpack command for `otrs-x.y.z.tar.gz`:

```
root> tar -xzf otrs-x.y.z.tar.gz -C /opt
```

Unpack command for `otrs-x.y.z.tar.bz2`:

```
root> tar -xjf otrs-x.y.z.tar.bz2 -C /opt
```

It is recommended to create a symbolic link named `/opt/otrs` that always points to the latest OTRS version. Using symbolic link makes easy to manage the OTRS updates, because you can leave untouched the directory of the previous version, only the symbolic link needs to change. If you need to revert the update, you can change the target of the symbolic link back.

Execute this command to create a symbolic link:

```
root> ln -fns /opt/otrs-x.y.z /opt/otrs
```

4.3.1 Restore Old Configuration Files

- `Kernel/Config.pm`
- `Kernel/Config/Files/User/*.pm`
- `Kernel/WebApp.conf`

4.3.2 Restore Article Data

If you configured OTRS to store article data in the file system you have to restore the `article` folder to `/opt/otrs/var/` or the folder specified in the system configuration.

4.3.3 Restore Already Installed Default Statistics

If you have additional packages with default statistics you have to restore the stats XML files with the suffix `*.installed` to `/opt/otrs/var/stats`.

```
root> cd OTRS-BACKUP/var/stats
root> cp *.installed /opt/otrs/var/stats
```

4.3.4 Set File Permissions

Please execute the following command to set the file and directory permissions for OTRS. It will try to detect the correct user and group settings needed for your setup.

```
root> /opt/otrs/bin/otrs.SetPermissions.pl
```

4.3.5 Install Required Programs and Perl Modules

Please refer to the section *Step 2: Install Additional Programs and Perl Modules* in the installation guide that explains how to verify external dependencies such as Perl modules and Node.js.

In addition to that, OTRS 8 also requires an active cluster of Elasticsearch (only versions between 6.5.x and 6.8.y are supported, newer versions are not supported). Please refer to the *Step 8: Setup Elasticsearch Cluster* section in the installation guide.

4.4 Step 4: Run the Migration Script

The migration script will perform many checks on your system and give you advice on how to install missing Perl modules etc., if that is required. If all checks succeeded, the necessary migration steps will be performed. Please also run this script in case of patch level updates.

Run the migration script:

```
otrs> /opt/otrs/scripts/DBUpdate-to-8.pl
```

> **Warning:** Do not continue the upgrading process if this script did not work properly for you. Otherwise malfunction or data loss may occur.

The migration script also checks if ACLs and system configuration settings are correct. In case of an invalid system configuration setting, script will offer you an opportunity to fix it by choosing from a list of possible values. In case the script runs in a non-interactive mode, it will try to automatically fix invalid settings. If this fails, you will be asked to manually update the setting after the migration.

If there are outdated ACLs, the system will not be able to fix them automatically, and they need to be corrected by the administrator. Please see the last step for manual changes for details.

4.5 Step 5: Update Installed Packages

Note: Packages for OTRS 7 are not compatible with OTRS 8 and have to be updated.

You can use the command below to update all installed packages. This works for all packages that are available from online repositories. You can update other packages later via the package manager (this requires a running OTRS daemon).

```
otrs> /opt/otrs/bin/otrs.Console.pl Admin::Package::UpgradeAll
```

4.6 Step 6: Start your Services

Now the services can be started. This will depend on your service configuration, here is an example:

```
root> systemctl start postfix
root> systemctl start apache2
```

Note: The OTRS daemon is required for correct operation of OTRS such as sending emails. Please activate it as described in the next step.

4.7 Step 7: Start the OTRS Daemon and Web Server

OTRS comes with example systemd configuration files that can be used to make sure that the OTRS daemon and web server are started automatically after the system starts.

```
root> cd /opt/otrs/scripts/systemd
root> for UNIT in *.service; do cp -vf $UNIT /usr/lib/systemd/system/;
→systemctl enable $UNIT; done
root> systemctl start otrs-daemon
root> systemctl start otrs-webserver
```

Now you can log into your system.

4.8 Step 8: Manual Migration Tasks and Changes

Warning: This step is required only for major update from OTRS 7.

With OTRS 8 a new default password policy for agents and customer users is in effect. The password policy rules can be changed in the system configuration (`PreferencesGroups###Password` and `Customer-PersonalPreference###Password`).

Password Policy Rule	Default
`PasswordMinSize`	8
`PasswordMin2Lower2UpperCharacters`	Yes
`PasswordNeedDigit`	Yes
`PasswordHistory`	10
`PasswordTTL`	30 days
`PasswordWarnBeforeExpiry`	5 days
`PasswordChangeAfterFirstLogin`	Yes

It is also possible to add some password policy exceptions for agents or customer users. Please check the following configuration options for details:

- `PasswordPolicy::ExceptAgentPasswordComplexity`

- `PasswordPolicy::ExceptAgentPasswordTTL`

- `PasswordPolicy::ExceptAgentPasswordHistory`

- `PasswordPolicy::ExceptAgentPasswordChangeAfterFirstLogin`

- `PasswordPolicy::ExceptCustomerPasswordComplexity`

- `PasswordPolicy::ExceptCustomerPasswordTTL`

- `PasswordPolicy::ExceptCustomerPasswordHistory`

- `PasswordPolicy::ExceptCustomerPasswordChangeAfterFirstLogin`

4.9 Recommendation: Convert Your MySQL Database

OTRS 7 and previous versions only supported the `utf8` MySQL database character set, which is not able to store all Unicode characters. Since OTRS 8 the `utf8mb4` character set is also supported, which removes this limitation. It is recommended to convert your database at your convenience with the built-in migration script.

Running the migration script without options will display only if something needs to be converted.

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Database::MySQL::UTF8MB4Migration
The following tasks need to be done:
  - 1 table(s) need to be converted to 'DYNAMIC' row format
  - the database needs to be converted to 'utf8mb4'
  - 411 column(s) need to be converted to 'utf8mb4'
  - 156 table(s) need to be converted to 'utf8mb4'
```

Use the `--force` option to perform the actual migration.

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Database::MySQL::UTF8MB4Migration -
↪-force
...
```

After the migration is done, you need to add a new switch to file `Kernel/Config.pm` to activate the `utf8mb4` support.

```
$Self->{'Database::mysql::utf8mb4'} = 1;
```

Note: OTRS 8 can be operated in the `utf8` character set, without this migration. But OTRS 9 will only support the `utf8mb4` character set. Therefore this migration needs to be done during the life cycle of OTRS 8 at some point.

CHAPTER 5

Backup and Restore

OTRS has built-in scripts for backup and restore. Execute the scripts with −h option for more information.

5.1 Backup

Note: To create a backup, write permission is needed for otrs user for the destination directory.

```
otrs> /opt/otrs/scripts/backup.pl -h
```

The output of the script:

```
Backup an OTRS system.

Usage:
 backup.pl -d /data_backup_dir [-c gzip|bzip2] [-r DAYS] [-t␣
↪fullbackup|nofullbackup|dbonly]

Options:
 -d                        - Directory where the backup files should place to.
 [-c]                      - Select the compression method (gzip|bzip2).␣
↪Default: gzip.
 [-r DAYS]                 - Remove backups which are more than DAYS days old.
 [-t]                      - Specify which data will be saved␣
↪(fullbackup|nofullbackup|dbonly). Default: fullbackup.
 [-h]                      - Display help for this command.

Help:
Using -t fullbackup saves the database and the whole OTRS home directory␣
↪(except /var/tmp and cache directories).
```

```
Using -t nofullbackup saves only the database, /Kernel/Config* and /var␣
↪directories.
With -t dbonly only the database will be saved.

Output:
 Config.tar.gz           - Backup of /Kernel/Config* configuration files.
 Application.tar.gz      - Backup of application file system (in case of full␣
↪backup).
 VarDir.tar.gz           - Backup of /var directory (in case of no full␣
↪backup).
 DataDir.tar.gz          - Backup of article files.
 DatabaseBackup.sql.gz   - Database dump.
```

5.2 Restore

```
otrs> /opt/otrs/scripts/restore.pl -h
```

The output of the script:

```
Restore an OTRS system from backup.

Usage:
 restore.pl -b /data_backup/<TIME>/ -d /opt/otrs/

Options:
 -b                      - Directory of the backup files.
 -d                      - Target OTRS home directory.
 [-h]                    - Display help for this command.
```

Performance Tuning

There is a list of performance enhancing techniques for your OTRS installation, including configuration, coding, memory use, and more.

6.1 Ticket Search Index

OTRS uses a special search index to perform full-text searches across fields in articles from different communication channels.

To create an initial index, use this command:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Ticket::FulltextIndex --rebuild
```

Note: Actual article indexing happens via an OTRS daemon job in the background. While articles which were just added in the system are marked for indexing immediately, it could happen their index is available within a few minutes.

There are some options available for fine tuning the search index:

Ticket::SearchIndex::IndexArchivedTickets Defines if archived tickets will be included in the search index (not enabled by default). This is advisable to keep the index small on large systems with archived tickets. If this is enabled, archived tickets will be found by full-text searches.

Ticket::SearchIndex::Attribute Basic full-text index settings.

Note: Run the following command in order to generate a new index:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Ticket::FulltextIndexRebuild
```

Ticket::SearchIndex::Attribute

WordCountMax	☐	1000
WordLengthMax	☐	30
WordLengthMin	☐	3

Basic fulltext index settings. Execute "bin/otrs.Console.pl Maint::Ticket::FulltextIndexRebuild" in order to generate a new index.

Fig. 1: `Ticket::SearchIndex::Attribute` Setting

WordCountMax Defines the maximum number of words which will be processed to build up the index. For example only the first 1000 words of an article body are stored in the article search index.

WordLengthMin and **WordLengthMax** Used as word length boundaries. Only words with a length between these two values are stored in the article search index.

Ticket::SearchIndex::Filters Full-text index regular expression filters to remove parts of the text.

Ticket::SearchIndex::Filters

[,\&\<\>\?"\!*\|;\[\]\(\)\+\$\^=]

^['::]|['::]$

^[^\w]+$

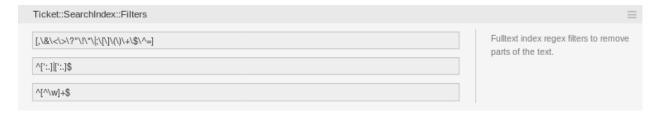Fulltext index regex filters to remove parts of the text.

Fig. 2: `Ticket::SearchIndex::Filters` Setting

There are three default filters defined:

- The first filter strips out special chars like: , & < > ? ” ! * | ; [ ] ( ) + $ ^ =
- The second filter strips out words which begin or ends with one of following chars: ': .
- The third filter strips out words which do not contain a word-character: a-z, A-Z, 0-9, _

Ticket::SearchIndex::StopWords English stop words for full-text index. These words will be removed from the search index.

Ticket::SearchIndex::StopWords###en

a

about

above

after

again

against

all

am

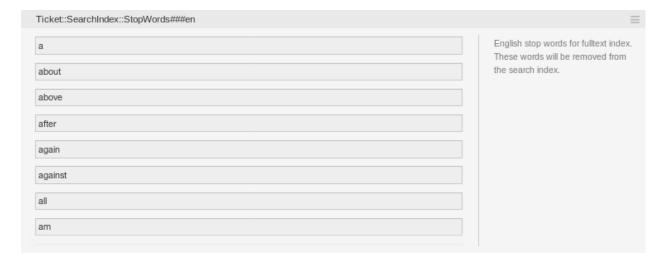English stop words for fulltext index. These words will be removed from the search index.

Fig. 3: `Ticket::SearchIndex::StopWords###en` Setting

There are so-called stop-words defined for some languages. These stop-words will be skipped while creating the search index.

See also:

If your language is not in the system configuration settings or you want to add more words, you can add them to this setting:

- `Ticket::SearchIndex::StopWords###Custom`

6.2 Document Search

OTRS uses Elasticsearch for its document search functionality. For a good introduction into the concepts, installation and usage of Elasticsearch, please follow the Getting Started guide.

6.2.1 Heap Size

Elasticsearch is written in Java and therefore runs in a Java Virtual Machine (JVM) on any cluster node. Such a JVM uses a part of the memory, called *heap*, which size can be configured in configuration file `jvm.options`.

The heap minimum and maximum configurations are by default set to a value of 1 GB and can be modified with the following options:

- `Xms1g`: minimum heap size.
- `Xmx1g`: maximum heap size.

If the `Xms` has a lower value than `Xmx`, the JVM will resize the used heap anytime the current limit is exceeded, until the value of `Xmx` is reached. Such a resizing causes the service to pause until it is finished, which may decrease the speed and reactivity of the search or indexing actions. Therefore it is highly recommended to set those configurations to an equal value.

> **Warning:** If the maximum heap size is exceeded, the related cluster node stops working and might even shutdown the service.

The higher the heap maximum value is set, the more memory can be used by Elasticsearch, which also increases the possible pauses for garbage collection, done by the JVM. Therefore it is recommended to set a value for `Xmx`, that is not higher than 50% of the physical memory.

For more information and good rules of thumb about the heap size, please follow the official documentation.

6.2.2 Disk Allocation

During the run-time of the service, Elasticsearch inspects the available disk space and therefore decides whether to allocate new shards to the related cluster node or even relocate shards away from that particular node. Such behavior will be controlled by the current disk capacity and can be configured in configuration file `elasticsearch.yml`. Enclosed are some important configurations, that come with good default values, but might be important:

cluster.routing.allocation.disk.watermark.low Default value of 85%. If this limit is exceeded, Elasticsearch will not allocate more shards to the related cluster node. The operation of that node is not influenced and data can still be indexed and searched.

cluster.routing.allocation.disk.watermark.high Default value of 90%. If this limit is exceeded, Elasticsearch will try to relocate existing shards to other nodes (if possible), that have enough space available.

cluster.routing.allocation.disk.watermark.flood_stage Default value of 95%. If this limit is exceeded, Elasticsearch will update the configuration of all indices to read-only index blocks `index.blocks.read_only_allow_delete`, that have at least one shard allocated to the related cluster node. Since then, it is not possible to index new data to such indices and restricted to searches and delete actions.

Note: If the flood stage was exceeded and certain indices are configured to read-only mode, such configuration *will not* automatically be changed by Elasticsearch. If the related disks contains enough free space again, due to manual actions, it is needed change the configuration back to normal mode manually.

For more information about disk watermarks and disk-based shard allocation, please follow the official documentation.

6.3 Article Storage

There are two different back end modules for the article storage of phone, email and internal articles. The used article storage can be configured in the setting `Ticket::Article::Backend::MIMEBase::ArticleStorage`.

Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageDB This default module will store attachments in the database. It also works with multiple front end servers, but requires much storage space in the database.

Note: Don't use this with large setups.

Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageFS Use this module to store attachments on the local file system. It is fast, but if you have multiple front end servers, you must make sure the file system is shared between the servers. Place it on an NFS share or preferably a SAN or similar solution.

Note: Recommended for large setups.

You can switch from one back end to the other on the fly. You can switch the back end in the system configuration, and then run this command line utility to put the articles from the database onto the file system or the other way around:

```
otrs> /opt/otrs/bin/otrs.Console.pl Admin::Article::StorageSwitch --target␣
↪ArticleStorageFS
```

You can use the `--target` option to specify the target back end.

Note: The entire process can take considerable time to run, depending on the number of articles you have and the available CPU power and/or network capacity.

If you want to keep old attachments in the database, you can activate the system configuration option `Ticket::Article::Backend::MIMEBase::CheckAllStorageBackends` to make sure OTRS will still find them.

6.4 Archiving Tickets

As OTRS can be used as an audit-proof system, deleting closed tickets may not be a good idea. Therefore we implemented a feature that allows you to archive tickets.

Tickets that match certain criteria can be marked as archived. These tickets are not accessed if you do a regular ticket search or run a generic agent job. The system itself does not have to deal with a huge amount of tickets any longer as only the latest tickets are taken into consideration when using OTRS. This can result in a huge performance gain on large systems.

To use the archive feature:

1. Activate the `Ticket::ArchiveSystem` setting in the system configuration.

2. Define a generic agent job:

    - Click on the *Add Job* button in the *Generic Agent* screen.
    - *Job Settings*: provide a name for the archiving job.
    - *Automatic Execution*: select proper options to schedule this job.
    - *Select Tickets*: it might be a good idea to only archive those tickets in a closed state that have been closed a few months before.
    - *Update/Add Ticket Attributes*: set the field *Archive selected tickets* to *archive tickets*.
    - Save the job at the end of the page.
    - Click on the *Run this task* link in the overview table to see the affected tickets.
    - Click on the *Run Job* button.

    Note: Up to 5000 tickets can be modified by running this job manually.

When you search for tickets, the system default is to search tickets which are not archived.

To search for archived tickets:

1. Open the ticket search screen.

2. Set *Archive search* to *Unarchived tickets* or *All tickets*.

3. Perform the search.

6.5 Tuning the Web Server

The built-in web server of OTRS can handle small and medium setups out of the box. When OTRS serves many users simultaneously, it may be necessary to tweak the web server configuration to increase the number of worker processes, for example.

The web server configuration file is located in `Kernel/WebApp.conf`, and all settings there are documented. The `worker` setting can be increased to deploy more processes for serving HTTP requests on capable servers.

6.6 Caching

OTRS caches a lot of temporary data in `/opt/otrs/var/tmp`. Please make sure that this uses a high performance file system and storage. If you have enough RAM, you can also try to put this directory on a ramdisk like this:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Session::DeleteAll
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Cache::Delete
root> mount -o size=16G -t tmpfs none /opt/otrs/var/tmp
```

Note: Add persistent mount point in `/etc/fstab`.

Warning: This will be a non-permanent storage that will be lost on server reboot. All your sessions (if you store them in the file system) and your cache data will be lost.

See also:

There is also a centralized memcached based cache back end available for purchase from OTRS Group.

6.7 Clustering

For very high loads, it can be required to operate OTRS on a cluster of multiple front end servers. This is a complex task with many pitfalls. Therefore, OTRS Group provides support for clusters in its managed OTRS environment exclusively.