Zammad

Jun 10, 2021
37.3 Performance Tuning ......................................................... 182

38 Configure Database server ................................................. 185

39 Privacy & Data Retention ................................................... 187
   39.1 On-Premises Data ...................................................... 187
   39.2 External Services ...................................................... 188

40 Single Sign-On for Kerberos ................................................ 189
   40.1 Conceptual Overview .................................................. 189
   40.2 Getting Started .......................................................... 190
   40.3 Step 1: Configure Active Directory ................................. 190
   40.4 Step 2: Remove NGINX, Set up Apache + Kerberos ........... 193
   40.5 Step 3: Enable SSO in Zammad ................................. 196
   40.6 Step 4: Configure Client System (Windows Only) .............. 197
   40.7 Troubleshooting ....................................................... 198

41 Reporting Tools (Third party) .............................................. 201
   41.1 Getting Started ........................................................ 201
   41.2 Third Party Reporting Tools known to be working .......... 202
The Zammad documentation consists of three parts:

- Zammad system installation and configuration (this documentation)
- Zammad administration (https://admin-docs.zammad.org)
- Zammad user documentation (https://user-docs.zammad.org)
Do you receive many emails and want to answer them with a team of agents?

You’re going to love Zammad!

Zammad is a web based open source helpdesk/customer support system with many features to manage customer communication via several channels like telephone, facebook, twitter, chat and emails. It is distributed under version 3 of the GNU AFFERO General Public License (GNU AGPLv3).

The code is open source, and available on GitHub!
If you want to install Zammad, you need the following software.

## 2.1 1. Ruby Programming Language

Zammad requires Ruby. All required rubygems like ruby on rails are listed in the Gemfile. The following Ruby version is supported:

Ruby 2.6.6

<table>
<thead>
<tr>
<th>Zammad</th>
<th>Ruby</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.1+</td>
<td>2.6.6</td>
</tr>
<tr>
<td>3.4.0</td>
<td>2.6.5</td>
</tr>
<tr>
<td>3.1 - 3.3</td>
<td>2.5.5</td>
</tr>
<tr>
<td>2.5 - 3.0</td>
<td>2.4.4</td>
</tr>
</tbody>
</table>

## 2.2 2. Supported distributions

Below you can find all distributions Zammad provides packages for.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>7 &amp; 8</td>
</tr>
<tr>
<td>Debian</td>
<td>9 &amp; 10</td>
</tr>
<tr>
<td>OpenSuSE / SLES</td>
<td>Leap 42.3 / 12</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>16.04, 18.04 &amp; 20.04</td>
</tr>
</tbody>
</table>

**Note:** What about my specific distribution?! It’s so cool!

If your distribution is not listed, you can still install Zammad. For this you can either use **Docker-Compose** or **Source** installation.

We try to provide all current distributions that are supported by Packager.io. This means that we can’t always provide support for your favorite system.
2.3 3. Package Dependencies

The below dependencies need to be installed on your system. If you’re using the package install, the packages below will automatically installed with the Zammad-Package.

```
# Debian 9 & 10, Ubuntu 16.04, 18.04 & 20.04
$ apt install libimlib2

# openSUSE
$ zypper install imlib2

# CentOS 7 & 8
$ yum install epel-release
$ yum install imlib2
```

Note:
libimlib2-dev or imlib2-devel are no longer required.
However: If you have to use bundle install for e.g. custom Gems or development, you’ll need to install it!

2.4 4. Database Server

Zammad will store all content in a Database. You can choose between the following database servers:

- MySQL 5.7+
- MariaDB 10.3+
- PostgreSQL 9.1+

Note: We tend to recommend PostgreSQL. For the last 10 years we had the best experience with it.

Zammad requires UTF-8 for its database.

Warning: Required configuration for MySQL/MariaDB:

- Use UTF-8 encoding - utf8mb4 for example will fail!
- Set max_allowed_packet to a value larger than the default of 4 MB (64 MB+ recommended).
2.5 5. Reverse Proxy

In a typical web environment today, you use a reverse proxy to deliver the static content of your application. Only the “expensive” app required HTTP requests are forwarded to the application server.

The following reverse proxies are supported:

- Nginx 1.3+
- Apache 2.2+

2.6 6. Elasticsearch (optional)

Zammad uses Elasticsearch to

1) make search faster
2) support advanced features like reports
3) searching by email attachment contents

This becomes increasingly important as the number of tickets in your system gets larger and larger.

This dependency is optional but strongly recommended; Zammad will work without it, but search performance will be degraded, and some features will be disabled.

**Hint:** If you install Zammad via package manager…

It’s perfectly safe to manually override the Elasticsearch dependency. The appropriate command line flag will depend on your platform (e.g., --force, --ignore-depends, --skip-broken); check your package manager’s manpage to find out.

**Warning:** Please note that if you do not install and use Elasticsearch, the search will be very limited! We recommend using Elasticsearch, as it will boost the usage of Zammad greatly!

**Note:** Starting with Zammad 4.0 you can decide if you want to use elasticsearch or elasticsearch-oss. Please note that CentOS requires elasticsearch.

<table>
<thead>
<tr>
<th>Zammad</th>
<th>Elasticsearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0+</td>
<td>6.5-7.12</td>
</tr>
<tr>
<td>3.4-3.6</td>
<td>5.5–7.9</td>
</tr>
<tr>
<td>3.3</td>
<td>2.4–7.6</td>
</tr>
<tr>
<td>3.2</td>
<td>2.4–7.5</td>
</tr>
<tr>
<td>3.1</td>
<td>2.4–7.4</td>
</tr>
<tr>
<td>2.0–3.0</td>
<td>2.4–5.6</td>
</tr>
</tbody>
</table>

An Elasticsearch plugin is required to index the contents of email attachments: ingest-attachment.
You can run Zammad on bare metal or on a virtual machine. Choose what you prefer.

### 3.1 For Zammad and a database server like PostgreSQL we recommend at least:

- 2 CPU cores
- 4 GB of RAM (+4 GB if you want to run Elasticsearch on the same server)

### 3.2 For optimal performance up to 40 agents:

- 4 CPU cores
- 6 GB of RAM (+6 GB if you want to run Elasticsearch on the same server)

Of course at the end it depends on actual load of concurrent agents and data traffic.

**Note:** We can’t suggest any disk space recommendations, as this highly depends on how you work. Zammad will always try to recognize the same attachments and store it just once.

### 3.3 Performance Tuning

As the number of active users on your system grows, performance will eventually degrade, leading to:

- delays for outgoing email,
- long loading times when viewing or creating tickets,
- stale or out-of-sync search results, or
- stale or out-of-sync ticket overviews.

You may see modest improvements by setting certain environment variables for Performance Tuning, such as `$WEB_CONCURRENCY` or `$ZAMMAD_SESSION_JOBS_CONCURRENT`. 
Note:
Please ensure to meet Zammad's Software requirements before hand.
This page expects administrative permissions, this is why `sudo` is not used.

## 4.1 Prerequisites

### 4.1.1 Additional software dependencies

In addition to already mentioned Package dependencies, some operating systems may require additional packages if not already installed.

**Ubuntu / Debian**

```bash
$ apt install wget apt-transport-https gnu
```

**CentOS**

```bash
$ yum install wget epel-release
```

### 4.1.2 Setup Elasticsearch

Elasticsearch is a dependency of Zammad and needs to be provided before installing Zammad. Please take a look at the following page: Set up Elasticsearch.

### 4.1.3 Ensure correct locale

For Zammad to function correctly, your system has to use the correct locales.

**Ubuntu / Debian**

**CentOS**

**OpenSUSE / SLES**

List your current locale settings.
$ locale |grep "LANG="

If above does not return `<lang_code>.utf8` you can correct this issue as follows.

```
$ apt install locales
$ locale-gen en_US.UTF-8
$ echo "LANG=en_US.UTF-8" > /etc/default/locale
```

List your current locale settings.

```
$ locale |grep "LANG="
```

If above does not return `<lang_code>.utf8` you can correct this issue as follows.

```
$ localectl set-locale LANG=en_US.utf8
```

List your current locale settings.

```
$ localectl status |grep "LC_CTYPE"
```

If above does not return `<lang_code>.utf8` you can correct this issue as follows.

```
$ localectl set-locale LC_CTYPE=en_US.UTF-8
```

**Hint:** By default OpenSUSE uses POSIX as LANG value for the root user. Learn more about this within the OpenSUSE documentation.

This does not affect other users and thus can be ignored.

## 4.2 Add Repository and install Zammad

**Hint:** If you want to use MySQL instead of PostgreSQL, it’s usually enough to have the MySQL server installed on your system already. Some installation managers can’t differentiate and still install Zammad with PostgreSQL. In that case, you’ll have to adapt manually (out of scope of this documentation).

### Add Repository

- **Ubuntu**
  - Debian
  - CentOS
  - OpenSUSE / SLES

**Install Repository Key**

```
$ wget -qO- https://dl.packager.io/srv/zammad/zammad/key | apt-key add -
```

**Ubuntu 16.04**

```
$ wget -O /etc/apt/sources.list.d/zammad.list \ https://dl.packager.io/srv/zammad/zammad/stable/installer/ubuntu/16.04.repo
```
Ubuntu 18.04

```
$ wget -O /etc/apt/sources.list.d/zammad.list \\  
```

Ubuntu 20.04

```
$ wget -O /etc/apt/sources.list.d/zammad.list \\  
```

Install Repository Key

```
$ wget -qO- https://dl.packager.io/srv/zammad/zammad/key | apt-key add -
```

Debian 9

```
$ wget -O /etc/apt/sources.list.d/zammad.list \\  
```

Debian 10

```
$ wget -O /etc/apt/sources.list.d/zammad.list \\  
```

Install Repository Key

```
$ rpm --import https://dl.packager.io/srv/zammad/zammad/key
```

RHEL 7 / CentOS 7

```
$ wget -O /etc/yum.repos.d/zammad.repo \\  
https://dl.packager.io/srv/zammad/zammad/stable/installer/el/7.repo
```

RHEL 8 / CentOS 8

```
$ wget -O /etc/yum.repos.d/zammad.repo \\  
```

Install Repository Key

```
$ rpm --import https://dl.packager.io/srv/zammad/zammad/key
```

SLES 12 / openSUSE 42.x

```
$ wget -O /etc/zypp/repos.d/zammad.repo \\  
https://dl.packager.io/srv/zammad/zammad/stable/installer/sles/12.repo
```

Install Zammad

Ubuntu / Debian

```
$ apt update
$ apt install zammad
```

CentOS

OpenSUSE / SLES

```
$ apt install zammad
```

4.2. Add Repository and install Zammad
Zammad

$ yum install zammad

Due to an issue with packager.io on CentOS 8 you’ll need to correct file permissions for public files.

chown -R 644 /opt/zammad/public/
chmod -R +x /opt/zammad/public/

$ zypper ref
$ zypper install zammad

### 4.3 Firewall & SELinux

Some parts of these steps may not apply to you, feel free to skip them!

#### 4.3.1 SELinux

Ubuntu / Debian / CentOS

OpenSUSE

$ # Allow nginx or apache to access public files of Zammad and communicate
$ chcon -Rv --type=httpd_sys_content_t /opt/zammad/public/
$ setsebool httpd_can_network_connect on -P
$ semanage fcontext -a -t httpd_sys_content_t /opt/zammad/public/
$ restorecon -Rv /opt/zammad/public/
$ chmod -R a+r /opt/zammad/public/

SELinux support on SUSE seems to be in early state, at least for SLES 12 (and Leap 42.x). This is why we won’t cover it in this documentation.

See the documentation for more input if you still wish to continue.

#### 4.3.2 Firewall

**Note:** Below only covers the distributions default firewall. It may not cover your case.

Ubuntu

Debian

CentOS

OpenSUSE

other

$ # Open Port 80 and 443 on your Firewall
$ ufw allow 80
$ ufw allow 443
$ ufw reload
Warning: We’re covering nftables in this part - iptables is discouraged starting from Debian 10 (Buster). Our example uses the input chain, yours may be a different one!

Add the following lines to /etc/nftables.conf or your specific rule file. Ensure to add these lines to your input-chain.

```plaintext
# Open Port 80 and 443 for Zammad
tcp dport { http, https } accept
udp dport { http, https } accept
```

The result should look like the following. Keep in mind that your enviroment could require different / more rules.

```plaintext
#!/usr/local/sbin/nft -f
flush ruleset
table inet filter {
    chain input {
        type filter hook input priority 0; policy drop;
        ct state established,related accept
        tcp dport ssh log accept
        tcp dport { http, https } accept
        udp dport { http, https } accept
    }
    chain forward {
        type filter hook forward priority 0; policy accept;
    }
    chain output {
        type filter hook output priority 0; policy accept;
    }
}
```

To load your new rules, simply run `systemctl reload nftables`.

```
$ # Open Port 80 and 443 on your Firewall
$ firewall-cmd --zone=public --add-service=http --permanent
$ firewall-cmd --zone=public --add-service=https --permanent
$ firewall-cmd --reload
```

If your system does not yet know webserver rules, you can add a new one for your firewall by creating the file `/etc/sysconfig/SuSEfirewall2.d/services/webserver` with this content:

```
## Name: Webserver
## Description: Open ports for HTTP and HTTPs

# space separated list of allowed TCP ports
TCP="http https"
# space separated list of allowed UDP ports
UDP="http https"
```

After that locate `FW_CONFIGURATIONS_EXT` within `/etc/sysconfig/SuSEfirewall2` and add the option `webserver` to the list. The list is seperated by spaces. You may require a different zone, above covers the external zone.
Now ensure to restart the firewall service.

```
systemctl restart SuSEfirewall2
```

If we didn’t cover your distribution or firewall in question, ensure to open ports 80 and 443 (TCP & UDP) beside of the ports you need.

### 4.4 Manage services of Zammad

In general Zammad uses three services - these can be (re)started & stopped with the parent zammad.

```
$ # Zammad service to start all services at once
$ systemctl (status|start|stop|restart) zammad

$ # Zammad’s internal railsserver
$ systemctl (status|start|stop|restart) zammad-web

$ # Zammad’s scheduler - relevant for all delayed- and background jobs
$ systemctl (status|start|stop|restart) zammad-worker

$ # Zammad’s websocket server for session related information
$ systemctl (status|start|stop|restart) zammad-websocket
```

### 4.5 Next steps

With this Zammad technically is ready to go. However, you’ll need to follow the following further steps to access Zammad’s Web-UI and getting started with it.

1. Connect Zammad with Elasticsearch
2. Configure the webserver
3. First steps
4. You may also find Zammad’s Console commands useful

If you expect usage with 5 agents or more you may also want to consider the following pages.

- Performance Tuning
- Configure Database server
5.1 Install from source on MacOS

5.1.1 Prerequisites

- Install Xcode from the App Store, open it -> Xcode menu > Preferences > Downloads -> install command line tools

  $ curl -L https://get.rvm.io | bash -s stable --ruby
  $ source ~/.rvm/scripts/rvm
  $ start new shell -> ruby -v

5.1.2 Get Zammad

  $ test -d ~/zammad/ || mkdir ~/zammad
  $ cd ~/zammad/

5.1.3 Install Zammad

  $ cd zammad-latest
  $ bundle install
  $ sudo ln -s /usr/local/mysql/lib/libmysqlclient.18.dylib /usr/lib/libmysqlclient.18.dylib # if needed!
  $ rake db:create
  $ rake db:migrate
  $ rake db:seed
5.1.4 Database connect

```bash
$ cd zammad-latest
$ cp config/database/database.yml config/database.yml
$ rake db:create
$ rake db:migrate
$ rake db:seed
```

5.1.5 Start Zammad

```bash
$ puma -p 3000 # application web server
$ script/websocket-server.rb start # non blocking websocket server
$ script/scheduler.rb start # generate overviews on demand, just send changed data to browser
```

5.1.6 Visit Zammad in your browser

- http://localhost:3000/#getting_started

**Note:** The source installation is the most difficult installation type of Zammad. If you’re not too experienced with Linux and all that, you may want to use another installation type:

- Install from package
- Install with Docker Compose

**Administrative note**

Please note that we only use `sudo` after direct user changes. In all other situations you can expect `root` being in charge.

---

**Hint:** Looking for MacOS hints? You can find outdated documentation [here](#).

5.2 Prerequisites

5.2.1 Software dependencies

Please ensure that you already provided mentioned *Software requirements*.

Also ensure to provide your database server and web server at this point.
5.2.2 Setup Elasticsearch

Elasticsearch is a dependency of Zammad and needs to be provided before installing Zammad. Please take a look at the following page: Set up Elasticsearch.

5.2.3 Ensure correct locale

For Zammad to function correctly, your system has to use the correct locales.

Ubuntu / Debian

CentOS

OpenSUSE / SLES

List your current locale settings.

```
$ locale |grep "LANG="
```

If above does not return `<lang_code>.utf8` you can correct this issue as follows.

```
$ apt install locales
$ locale-gen en_US.UTF-8
$ echo "LANG=en_US.UTF-8" > /etc/default/locale
```

List your current locale settings.

```
$ locale |grep "LANG="
```

If above does not return `<lang_code>.utf8` you can correct this issue as follows.

```
$ localectl set-locale LANG=en_US.utf8
```

List your current locale settings.

```
$ localectl status |grep "LC_CTYPE"
```

If above does not return `<lang_code>.utf8` you can correct this issue as follows.

```
$ localectl set-locale LC_CTYPE=en_US.UTF-8
```

**Hint:** By default OpenSUSE uses POSIX as LANG value for the root user. Learn more about this within the OpenSUSE documentation.

This does not affect other users and thus can be ignored.
5.2.4 Add user

$ useradd zammad -m -d /opt/zammad -s /bin/bash
$ groupadd zammad

5.3 Installation

5.3.1 Step 1: Get the source

Note: Not all distributions ship wget by default, you may need to install it manually.

Get the latest stable release of Zammad here, or find the initial version at https://ftp.zammad.com.

$ cd /opt
$ wget https://github.com/zammad/zammad/archive/stable.tar.gz
$ tar -xzf stable.tar.gz -C zammad
$ chown -R zammad:zammad zammad/
$ rm -f stable.tar.gz

5.3.2 Step 2: Install dependencies

Note:
Below commands do neither include the database server nor the web server.
We do cover important web server related stuff within Configure the webserver.

Zammad requires specific ruby versions. Adapt the commands below if you install older versions. A list of required versions can be found on the Software requirements page.

Ubuntu
Debian
CentOS
OpenSuSE
other
Install RVM

$ apt update
$ apt install curl git patch build-essential bison zlib1g-dev libssl-dev libxml2-dev
$ libxml2-dev autotools-dev\
$ libxl1-dev libyaml-0-2 autoconf automake libreadline-dev libyaml-dev libtool\
$ libgmp-dev libgdbm-dev libcurses5-dev\
$ pkg-config libffi-dev libimlib2-dev gawk libsqlite3-dev sqlite3 software-\n$ properties-common
$ apt-add-repository -y ppa:rael-gc/rvm

(continues on next page)
Set relevant Environment variables

```
# Set rails environment specific things
$ echo "export RAILS_ENV=production" >> /opt/zammad/.bashrc
$ echo "export RAILS_SERVE_STATIC_FILES=true" >> /opt/zammad/.bashrc
$ echo "rvm --default use 2.6.6" >> /opt/zammad/.bashrc

# Debian, CentOS & OpenSuSE
$ echo "source /usr/local/rvm/scripts/rvm" >> /opt/zammad/.bashrc
# Ubuntu
$ echo "source /usr/share/rvm/scripts/rvm" >> /opt/zammad/.bashrc
```

Install Ruby Environment

```
# Add zammad user to RVM group
$ usermod -a -G rvm zammad

# Install Ruby 2.6.6
$ su - zammad
$ rvm install ruby-2.6.6

# Install bundler, rake and rails
$ rvm use 2.6.6
$ gem install bundler rake rails
```

Install RVM

```
$ apt update
$ apt install curl git patch build-essential bison zlib1g-dev libssl-dev libxml2-dev
  --dev libxml2-dev autotools-dev
  --dev libxml2-dev libyaml-0-2 autoconf automake libreadline-dev libyaml-dev libtool
  --dev libgmp-dev libgdbm-dev libncurses5-dev

$ gpg --keyserver hkp://keys.gnupg.net --recv-keys
  409B6B1796C275462A1703113804BB82D39DC0E3 7D28AF1CF37B13E2069D6956105BD0E739499BDB
$ curl -L https://get.rvm.io | bash -s stable
```

Set relevant Environment variables

```
# Set rails environment specific things
$ echo "export RAILS_ENV=production" >> /opt/zammad/.bashrc
$ echo "export RAILS_SERVE_STATIC_FILES=true" >> /opt/zammad/.bashrc
$ echo "rvm --default use 2.6.6" >> /opt/zammad/.bashrc

# Debian, CentOS & OpenSuSE
$ echo "source /usr/local/rvm/scripts/rvm" >> /opt/zammad/.bashrc
# Ubuntu
$ echo "source /usr/share/rvm/scripts/rvm" >> /opt/zammad/.bashrc
```

5.3. Installation
# Add zammad user to RVM group
$ usermod -a -G rvm zammad

# Install Ruby 2.6.6
$ su - zammad
$ rvm install ruby-2.6.6

# Install bundler, rake and rails
$ rvm use 2.6.6
$ gem install bundler rake rails

Install RVM

$ yum install epel-release
$ yum install patch autoconf automake bison bzip2 gcc-c++ libffi-devel libtool make...
   --patch readline-devel ruby sqlite-devel\
   zlib-devel glibc-headers glibc-devel openssl-devel git imlib2 imlib2-devel

$ gpg --keyserver hkp://keys.gnupg.net --recv-keys
   409B6B1796C275462A1703113804BB82D39DC0E3 7D2BAF1CF37B13E2069D6956105BD0E739499BDB
$ curl -L https://get.rvm.io | bash --s stable

Set relevant Environment variables

# Set rails environment specific things
$ echo "export RAILS_ENV=production" >> /opt/zammad/.bashrc
$ echo "export RAILS_SERVE_STATIC_FILES=true" >> /opt/zammad/.bashrc
$ echo "rvm --default use 2.6.6" >> /opt/zammad/.bashrc

# Debian, CentOS & OpenSuSE
$ echo "source /usr/local/rvm/scripts/rvm" >> /opt/zammad/.bashrc
# Ubuntu
$ echo "source /usr/share/rvm/scripts/rvm" >> /opt/zammad/.bashrc

Install Ruby Environment

# Add zammad user to RVM group
$ usermod -a -G rvm zammad

# Install Ruby 2.6.6
$ su - zammad
$ rvm install ruby-2.6.6

# Install bundler, rake and rails
$ rvm use 2.6.6
$ gem install bundler rake rails

Install RVM

$ zypper install patch autoconf automake bison bzip2 gcc-c++ libffi-devel libtool...
   --make patch readline-devel sqlite3-devel\
   sqlite3 zlib-devel glibc-devel openssl-devel git imlib2 imlib2-devel gdbm-devel...
   --libyaml-devel

(continues on next page)
$ gpg --keyserver hkp://keys.gnupg.net --recv-keys
409B6B1796C275462A1703113804BB82D39DC0E3 7D28AF1CF37B13E2069D6956105BD0E739499BDB
$ curl -L https://get.rvm.io | bash -s stable

Set relevant Environment variables

```
# Set rails environment specific things
$ echo "export RAILS_ENV=production" >> /opt/zammad/.bashrc
$ echo "export RAILS_SERVE_STATIC_FILES=true" >> /opt/zammad/.bashrc
$ echo "rvm --default use 2.6.6" >> /opt/zammad/.bashrc

# Debian, CentOS & OpenSuSE
$ echo "source /usr/local/rvm/scripts/rvm" >> /opt/zammad/.bashrc
# Ubuntu
$ echo "source /usr/share/rvm/scripts/rvm" >> /opt/zammad/.bashrc
```

Install Ruby Environment

```
# Add zammad user to RVM group
$ usermod -a -G rvm zammad

# Install Ruby 2.6.6
$ su - zammad
$ rvm install ruby-2.6.6

# Install bundler, rake and rails
$ rvm use 2.6.6
$ gem install bundler rake rails
```

Other systems than above mentioned are out of scope of this documentation. Please check the rvm documentation on how to install rvm on your system.

After that install the specific required ruby version.

After installing bundler, rake and rails we'll need to install all required gems.
The command depends on the database server you are using.

PostgreSQL (recommended)

MySQL / MariaDB

**Install PostgreSQL Dependencies** Ubuntu / Debian

CentOS

OpenSuSE

```
$ apt install libpq-dev

$ yum install postgresql-libs postgresql-devel

$ zypper install postgresql-devel
```
Install Gems for Zammad

```
$ su - zammad
$ bundle install --without test development mysql
```

Install MySQL/MariaDB Dependencies   Ubuntu / Debian

```
CentOS

OpenSuSE

$ apt install libmariadb-dev

$ yum install mariadb-devel

$ zypper install libmariadb-devel
```

Install Gems for Zammad

```
$ su - zammad
$ bundle install --without test development postgres
```

5.3.3 Step 3: Configure database settings

**Tip:** For easiest usage …

If you provide your Zammad user with database creation permission, you can run the step 4 without adjustment. If you don’t want that, you’ll have to create the database manually.

```
$ cp config/database/database.yml config/database.yml
$ vi config/database.yml
```

Here’s a sample configuration to give you an idea on how your configuration file could look like. Please also have a look at *Configure Database server* for deeper details.

PostgreSQL

MySQL / MariaDB

```yaml
production:
  adapter: postgresql
  database: zammad
  pool: 50
  timeout: 5000
  encoding: utf8
  username: zammad
  password: changeme
```

**Hint:** You can remove the `password` line if you enable socket based authentication!
production:
  adapter: mysql2
  database: zammad
  pool: 50
  timeout: 5000
  encoding: utf8
  username: zammad
  password: changeme

Hint: If you want to use an existing database server that’s not on the same machine, you can also use host and port to set that up.

5.3.4 Step 4: Initialize your database

Warning: Ensure to do this as zammad user in your Zammad directory!

$ su - zammad
$ rake db:create # SKIP IF you already created zammad database (see tip of step 3)
$ rake db:migrate
$ rake db:seed

5.3.5 Step 5: Pre compile all Zammad assets

$ rake assets:precompile

5.3.6 Step 6: Start Zammad or install as service

Note: Run the following commands as root.

You can start all services by hand or use systemd to start / stop Zammad.

systemd (recommended)
the manual way

$ cd /opt/zammad/script/systemd
$ ./install-zammad-systemd-services.sh

Note: This method is not suitable for production use - you should avoid it.

$ rails s -p 3000 # application web server
$ script/websocket-server.rb start # non blocking websocket server
$ script/scheduler.rb start # generate overviews on demand, just send changed data to browser

(continues on next page)
5.4 Firewall & SELinux

Some parts of these steps may not apply to you, feel free to skip them!

5.4.1 SELinux

Ubuntu / Debian / CentOS

OpenSUSE

$ # Allow nginx or apache to access public files of Zammad and communicate
$ chcon -Rv --type=httpd_sys_content_t /opt/zammad/public/
$ setsebool httpd_can_network_connect on -P
$ semanage fcontext -a -t httpd_sys_content_t /opt/zammad/public/
$ restorecon -Rv /opt/zammad/public/
$ chmod -R a+r /opt/zammad/public/

SELinux support on SUSE seems to be in early state, at least for SLES 12 (and Leap 42.x). This is why we won’t cover it in this documentation.

See the documentation for more input if you still wish to continue.

5.4.2 Firewall

Note: Below only covers the distributions default firewall. It may not cover your case.

Ubuntu

Debian

CentOS

OpenSUSE

other

$ # Open Port 80 and 443 on your Firewall
$ ufw allow 80
$ ufw allow 443
$ ufw reload

Warning: We’re covering nftables in this part - iptables is discouraged starting from Debian 10 (Buster). Our example uses the input chain, yours may be a different one!

Add the following lines to /etc/nftables.conf or your specific rule file. Ensure to add these lines to your input-chain.
# Open Port 80 and 443 for Zammad

tcp dport \{ http, https \} accept
udp dport \{ http, https \} accept

The result should look like the following. Keep in mind that your environment could require different / more rules.

```bash
#!/usr/local/sbin/nft -f
flush ruleset
table inet filter {
    chain input {
        type filter hook input priority 0; policy drop;
        ct state established,related accept
tcp dport ssh log accept
tcp dport \{ http, https \} accept
udp dport \{ http, https \} accept
    }
    chain forward {
        type filter hook forward priority 0; policy accept;
    }
    chain output {
        type filter hook output priority 0; policy accept;
    }
}
```

To load your new rules, simply run `systemctl reload nftables`.

$ # Open Port 80 and 443 on your Firewall
$ firewall-cmd --zone=public --add-service=http --permanent
$ firewall-cmd --zone=public --add-service=https --permanent
$ firewall-cmd --reload

If your system does not yet know webserver rules, you can add a new one for your firewall by creating the file /etc/sysconfig/SuSEfirewall2.d/services/webserver with this content:

```bash
## Name: Webserver
## Description: Open ports for HTTP and HTTPs

TCP="http https"
UDP="http https"
```

After that locate `FW_CONFIGURATIONS_EXT` within /etc/sysconfig/SuSEfirewall2 and add the option `webserver` to the list. The list is seperated by spaces. You may require a different zone, above covers the external zone.

Now ensure to restart the firewall service.

```bash
systemctl restart SuSEfirewall2
```

If we didn’t cover your distribution or firewall in question, ensure to open ports 80 and 443 (TCP & UDP) beside of the ports you need.

5.4. Firewall & SELinux
5.5 Next steps

With this Zammad technically is ready to go. However, you’ll need to follow the following further steps to access Zammad’s Web-UI and getting started with it.

1. Connect Zammad with Elasticsearch
2. Configure the webserver
3. First steps
4. You may also find Zammad’s Console commands useful

If you expect usage with 5 agents or more you may also want to consider the following pages.

- Performance Tuning
- Configure Database server
Zammad’s search function is powered by Elasticsearch, and requires the ingest attachment plugin.

Note: This guide uses the `zammad run` command prefix in command line examples. This prefix is only applicable to package installations (i.e., via apt/yum/zypper, or .deb/.rpm files).

If you installed from source, be sure to omit this prefix and run the bare `rails ...` or `rake ...` commands instead.

### 6.1 Step 1: Installation

Note: To use OSS or not to use...

Starting with Zammad 4.0 our packages allow you to decide whether to use `elasticsearch` or `elasticsearch-oss`.

`elasticsearch-oss` users please use below “direct download” tab for further installation steps.

**Warning:** Above does not apply to CentOS because of compatibility reasons.

**Ubuntu**

**Debian**

**CentOS**

**OpenSUSE**

**Direct Download**

```bash
$ apt install apt-transport-https sudo wget
$ echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | tee -a /etc/apt/...
$ wget -qO- https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
$ apt update
$ apt install elasticsearch
$ /usr/share/elasticsearch/bin/elasticsearch-plugin install ingest-attachment
```
$ apt install apt-transport-https sudo wget
$ echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | tee -a /etc/apt/
→sources.list.d/elastic-7.x.list
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
$ apt update
$ apt install elasticsearch
$ /usr/share/elasticsearch/bin/elasticsearch-plugin install ingest-attachment

$ rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
$ echo "[elasticsearch-7.x]
name=Elasticsearch repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md"| tee /etc/yum.repos.d/elasticsearch-7.x.repo
$ yum install -y elasticsearch
$ /usr/share/elasticsearch/bin/elasticsearch-plugin install ingest-attachment

Find the latest release on the downloads page, or see the installation guide for in-depth instructions. Ensure to also install the fitting (and mandatory!) attachment plugin for elasticsearch.

If you prefer the Open Source version of Elasticsearch, please use the Elasticsearch-OSS download page.

# Install the attachment plugin
$ /usr/share/elasticsearch/bin/elasticsearch-plugin install ingest-attachment

# Increase the virtual memory map limit
$ sysctl -w vm.max_map_count=262144

After you installed Elasticsearch and its attachment plugin, ensure to enable it by default and start it.

$ systemctl start elasticsearch
$ systemctl enable elasticsearch

Note: Docker installations on macOS/Windows:

Setting the vm.max_map_count kernel parameter requires additional steps.
6.2 Step 2: Suggested Configuration

We use the following settings to optimize the performance of our Elasticsearch servers. Your mileage may vary.

```yaml
# /etc/elasticsearch/elasticsearch.yml
# Tickets above this size (articles + attachments + metadata)
# may fail to be properly indexed (Default: 100mb).
#
# When Zammad sends tickets to Elasticsearch for indexing,
# it bundles together all the data on each individual ticket
# and issues a single HTTP request for it.
# Payloads exceeding this threshold will be truncated.
#
# Performance may suffer if it is set too high.
http.max_content_length: 400mb

# Allows the engine to generate larger (more complex) search queries.
# Elasticsearch will raise an error or deprecation notice if this value is too low,
# but setting it too high can overload system resources (Default: 1024).
#
# Available in version 6.6+ only.
indices.query.bool.max_clause_count: 2000
```

Note: For more information on the `indices.query.bool.max_clause_count` setting, see the Elasticsearch 6.6 release notes.

6.3 Step 3: Connect Zammad

Hint: Before proceeding here...

Make sure to install Zammad before running below commands, as this will fail otherwise.

- install from package
- install from source

```bash
# Set the Elasticsearch server address
$ zammad run rails r "Setting.set('es_url', 'http://localhost:9200')"

# Build the search index
$ zammad run rake searchindex:rebuild
```
6.3.1 Optional settings

Authentication
Index namespaceing
File-attachment indexing rules

```
# HTTP Basic
$ zammad run rails r "Setting.set('es_user', '<username>')"
$ zammad run rails r "Setting.set('es_password', '<password>')"
```

Hint: How do I set up authentication on my Elasticsearch server?

Elasticsearch provides many different authentication methods. Some of them may require paid X-Pack, please check the elastic documentation for more information.

Useful when connecting multiple services or Zammad instances to a single Elasticsearch server (to prevent name collisions during indexing).

```
$ zammad run rails r "Setting.set('es_index', Socket.gethostname.downcase + '_zammad')"
```

Zammad supports searching by the contents of file attachments, which means Elasticsearch has to index those, too. Limiting such indexing can help conserve system resources.

```
# Files with these extensions will not be indexed
$ zammad run rails r "Setting.set('es_attachment_ignore',
```

```
# Files larger than this size (in MB) will not be indexed
$ zammad run rails r "Setting.set('es_attachment_max_size_in_mb', 50)"
```

6.4 Appendix

6.4.1 List of Indexed Attributes

Below is a comprehensive list of all object attributes indexed by Elasticsearch. In other words, if you wish to find a ticket, article, or user via the Zammad search box, Elasticsearch can match on any (or all) of the fields below.

<table>
<thead>
<tr>
<th>Table of content</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Ticket</td>
</tr>
<tr>
<td>• Ticket Priority</td>
</tr>
<tr>
<td>• Ticket State</td>
</tr>
<tr>
<td>• Article</td>
</tr>
<tr>
<td>• User</td>
</tr>
<tr>
<td>• Organization</td>
</tr>
</tbody>
</table>
Note: These fields may vary if you created custom fields (objects) in the admin interface.

Warning: Zammad 4.0 introduced breaking changes on the Elasticsearch index.

Hint: Below list contains functionality hints

In order to save space and duplicate information, we’ll provide hints to functions within brackets if applicable.

- **(SLA):** Attributes marked as SLA attribute are only set if the ticket is affected by SLA calculation. Please note that some attributes may not be set if specific conditions are not met.
  
  Also note that some attributes may be reset to null if no longer applicable.

- **note attribute:** Note attributes usually are empty if not specified via console or API.

- **Timestamps:** All timestamps provided by Zammad are UTC by default. This also applies to times provided by Elasticsearch.

**Ticket**

Tip: The following indice contains below mentioned information: *_ticket*

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>article</td>
<td>#{Article Array}</td>
<td>Array with all articles belonging to the ticket</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see Article for more details</td>
</tr>
<tr>
<td>article_count</td>
<td>1</td>
<td>Number of articles within the ticket</td>
</tr>
<tr>
<td>close_at</td>
<td>null, 2021-03-03T14:50:20.673Z</td>
<td>First close time, set once</td>
</tr>
<tr>
<td>close_diff_in_min</td>
<td>null, 239, -5</td>
<td>Depends on close_in_min and tells how many minutes the ticket was closed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>relative to SLAs solution time. (SLA)</td>
</tr>
<tr>
<td>close_escalation_at</td>
<td>null, 2021-03-03T15:50:20.673Z</td>
<td>Time stamp when the ticket would escalate in case solution time is violated. (SLA)</td>
</tr>
<tr>
<td>close_in_min</td>
<td>null, 11</td>
<td>Value in minutes for how long the ticket was open based on business hours. (SLA)</td>
</tr>
</tbody>
</table>

continues on next page
<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>create_article_sender</td>
<td>Contains these attributes:</td>
<td>Sender of the article (System, Agent, Customer)</td>
</tr>
<tr>
<td></td>
<td>• note: null</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• updated_at: [2021-03-03T14:50:20.812Z]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• name: Customer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• created_at: [2021-03-03T14:50:20.812Z]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• updated_by_id: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• id: 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• created_by_id: 1</td>
<td></td>
</tr>
<tr>
<td>create_article_sender_id</td>
<td>1, 2</td>
<td>ID of the user that created the article</td>
</tr>
<tr>
<td>create_article_type</td>
<td>Contains these attributes:</td>
<td>Information of first article type and nature</td>
</tr>
<tr>
<td></td>
<td>• note: null</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• updated_at: [2021-03-03T14:50:20.812Z]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• name: phone, email, web</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• active: true</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• created_at: [2021-03-03T14:50:20.812Z]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• updated_by_id: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• id: 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• created_by_id: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• communication: true, false</td>
<td></td>
</tr>
<tr>
<td>create_article_type_id</td>
<td>5</td>
<td>Type ID of first article</td>
</tr>
<tr>
<td>created_at</td>
<td>[2021-03-24T16:17:27.210Z]</td>
<td>Time stamp of ticket creation</td>
</tr>
<tr>
<td>created_by</td>
<td>#{user object}</td>
<td>Complete Payload of user that created the ticket</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see User for more</td>
</tr>
<tr>
<td>created_by_id</td>
<td>3</td>
<td>User ID that created the ticket</td>
</tr>
<tr>
<td>customer</td>
<td>#{user object}</td>
<td>Complete payload of the customer that created the ticket</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see User for more</td>
</tr>
<tr>
<td>customer_id</td>
<td>8</td>
<td>Customers User ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td>continues on next page</td>
</tr>
</tbody>
</table>
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>escalation_at</td>
<td>null, 2021-03-24T16:28:38.535Z</td>
<td>Time stamp of the next applicable escalation. One of the following attributes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• close_escalation_at</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• first_response_escalation_at</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• update_escalation_at</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(SLA)</td>
</tr>
<tr>
<td>first_response_at</td>
<td>null, 2021-03-24T16:28:38.303Z</td>
<td>Time stamp of the first communication type reaction to the customer (SLA)</td>
</tr>
<tr>
<td>first_response_diff_in_min</td>
<td>null, 10, -6</td>
<td>Depends on first_response_in_min and tells how many minutes the tickets first response took relative to the first response time of your SLA. (SLA)</td>
</tr>
<tr>
<td>first_response_in_min</td>
<td>null, 11</td>
<td>Value in minutes for how long the first response took based on the business hours. (SLA)</td>
</tr>
<tr>
<td>group</td>
<td>#{group object}</td>
<td>Complete payload of the current tickets group</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see Group for more</td>
</tr>
<tr>
<td>group_id</td>
<td>1</td>
<td>ID of the current group</td>
</tr>
<tr>
<td>id</td>
<td>1, 111</td>
<td>ID of the Ticket</td>
</tr>
<tr>
<td>last_contact_agent_at</td>
<td>null, 2021-03-24T16:28:38.303Z</td>
<td>Time stamp of last communication type contact of any agent</td>
</tr>
<tr>
<td>last_contact_at</td>
<td>null, 2021-03-24T16:28:38.303Z</td>
<td>Time stamp of last communication type contact</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Depends on last_contact_agent_at, last_contact_customer_at and “Ticket Last Contact Behaviour” setting</td>
</tr>
<tr>
<td>last_contact_customer_at</td>
<td>null, 2021-03-24T16:28:38.303Z</td>
<td>Time stamp of last communication type contact of customer</td>
</tr>
<tr>
<td>mention_user_ids</td>
<td>[3, 5]</td>
<td>Array with mentioned or subscribed users IDs</td>
</tr>
<tr>
<td>note</td>
<td>null</td>
<td>Note of ticket, only set via console or API</td>
</tr>
<tr>
<td>number</td>
<td>1010138, 202006231010138</td>
<td>Ticket number</td>
</tr>
<tr>
<td>organization</td>
<td>null, #{organization object}</td>
<td>Complete Payload of user that owns the ticket</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see Organization for more</td>
</tr>
<tr>
<td>organization_id</td>
<td>null, 2</td>
<td>ID of the customers organization</td>
</tr>
<tr>
<td>owner</td>
<td>null, #{user object}</td>
<td>Complete Payload of user that owns the ticket</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see User for more</td>
</tr>
<tr>
<td>owner_id</td>
<td>null, 3</td>
<td>User ID of the ticket owner</td>
</tr>
</tbody>
</table>

continues on next page
### Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pending_time</td>
<td>null, 2021-03-24T17:44:06.912Z</td>
<td>Depends on pending states, time stamp for pending time</td>
</tr>
<tr>
<td>preferences</td>
<td>n/a, special information for internal functions</td>
<td>May not be available in your system, contains information for internal system functions</td>
</tr>
<tr>
<td>priority</td>
<td>#{priority object}</td>
<td>Complete Payload of priority of ticket</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see Ticket Priority for more</td>
</tr>
<tr>
<td>priority_id</td>
<td>2</td>
<td>Priority ID of the ticket</td>
</tr>
<tr>
<td>state</td>
<td>#{state object}</td>
<td>Complete Payload of current ticket state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see Ticket State for more</td>
</tr>
<tr>
<td>state_id</td>
<td>1, 4</td>
<td>ID of current ticket state</td>
</tr>
<tr>
<td>tags</td>
<td>[&quot;order&quot;, &quot;americano&quot;]</td>
<td>Array with all attached tags</td>
</tr>
<tr>
<td>time_unit</td>
<td>null, 15</td>
<td>Accounted time units for ticket (total)</td>
</tr>
<tr>
<td>title</td>
<td>Feedback Form, Need help</td>
<td>Title / Subject of Ticket</td>
</tr>
<tr>
<td>type</td>
<td>null</td>
<td>Ticket type (deprecated)</td>
</tr>
<tr>
<td>update_diff_in_min</td>
<td>null, 2021-03-24T16:28:38.303Z</td>
<td>Depends on update_in_min and tells how many minutes the last ticket update took relatively to the update time setting (SLA)</td>
</tr>
<tr>
<td>update_escalation_at</td>
<td>null, 2021-03-24T16:28:38.303Z</td>
<td>Time stamp when the ticket would escalate in case update time is violated. (SLA)</td>
</tr>
<tr>
<td>update_in_min</td>
<td>null, 5, -10</td>
<td>Value in minutes for how long the last ticket update took based on the business hours and update time. (SLA)</td>
</tr>
<tr>
<td>updated_at</td>
<td>2021-03-24T16:28:38.303Z</td>
<td>Last ticket update</td>
</tr>
<tr>
<td>updated_by</td>
<td>#{user object}</td>
<td>Complete Payload of the user that updated the ticket</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see User for more</td>
</tr>
<tr>
<td>updated_by_id</td>
<td>1, 3</td>
<td>User ID that updated the ticket</td>
</tr>
</tbody>
</table>

**Ticket Priority**

**Tip:** The following indice contains below mentioned information: *._ticket_priority*
Table 2: Ticket Priority-Index

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>true, false</td>
<td>Defines if the priority is active (available)</td>
</tr>
<tr>
<td>created_at</td>
<td>2021-03-03T14:50:20.724Z</td>
<td>Creation date of priority</td>
</tr>
<tr>
<td>created_by_id</td>
<td>1</td>
<td>User that created priority</td>
</tr>
<tr>
<td>default_create</td>
<td>false, true</td>
<td>Defines if priority is default priority upon ticket creation</td>
</tr>
<tr>
<td>id</td>
<td>3</td>
<td>ID of priority</td>
</tr>
<tr>
<td>name</td>
<td>3 high</td>
<td>Priority name</td>
</tr>
<tr>
<td>note</td>
<td>null</td>
<td>Note for priority that has been set via console or API</td>
</tr>
<tr>
<td>ui_color</td>
<td>null, high-priority</td>
<td>CSS class for tickets of priority</td>
</tr>
<tr>
<td>ui_icon</td>
<td>null, important</td>
<td>CSS class for ticket icons of priority</td>
</tr>
<tr>
<td>updated_at</td>
<td>2021-03-03T14:50:20.724Z</td>
<td>Date of last change</td>
</tr>
<tr>
<td>updated_by_id</td>
<td>1</td>
<td>User ID of user last updating the priority</td>
</tr>
</tbody>
</table>

**Ticket State**

**Tip:** The following indice contains below mentioned information: *_.ticket_state*
### Table 3: Ticket State-Index

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>true, false</td>
<td>Defines if state is active (available)</td>
</tr>
<tr>
<td>created_at</td>
<td>2021-03-03T14:50:20.694Z</td>
<td>Creation date</td>
</tr>
<tr>
<td>created_by_id</td>
<td>1</td>
<td>User ID that created state</td>
</tr>
<tr>
<td>default_create</td>
<td>false, true</td>
<td>Defines if the state is the default state upon ticket creation</td>
</tr>
<tr>
<td>default_follow_up</td>
<td>false, true</td>
<td>Defines if the state is the default follow up state on ticket follow ups</td>
</tr>
<tr>
<td>id</td>
<td>7</td>
<td>State ID</td>
</tr>
<tr>
<td>ignore_escalation</td>
<td>false, true</td>
<td>Defines if SLA calculation is generally ignored for this state</td>
</tr>
<tr>
<td>name</td>
<td>pending close</td>
<td>State name</td>
</tr>
<tr>
<td>next_state</td>
<td>n/a, #{state object}</td>
<td>Contains all follow up state information if applicable, may not be available depending on the state type</td>
</tr>
<tr>
<td>next_state_id</td>
<td>null, 4</td>
<td>State ID of follow up state</td>
</tr>
<tr>
<td>note</td>
<td>null</td>
<td>Note that has been set via console or API</td>
</tr>
<tr>
<td>state_type</td>
<td>Contains these attributes:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• created_at:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2021-03-03T14:50:20.582Z</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• created_by_id:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• id:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• name: pending action</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• note: null</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• updated_at:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2021-03-03T14:50:20.582Z</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• updated_by_id:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>state_type_id</td>
<td>4</td>
<td>ID of the state type</td>
</tr>
<tr>
<td>updated_at</td>
<td>2021-03-03T14:50:20.694Z</td>
<td>Last update of state</td>
</tr>
<tr>
<td>updated_by_id</td>
<td>1</td>
<td>User ID that updated state last</td>
</tr>
</tbody>
</table>

**Note:** The following indice contains below mentioned information: *ticket*

**Tip:** Articles are part of the ticket index. To reduce complexity we decided to provide it in its own table.
### Table 4: Article-Index

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| body             | Hi,

please send me:

75007 Paris

David Bell | Article body in plain text                                                  |
| cc               | null, alias@domain.tld                                                       | EMail-Addresses set as CC (String)                |
| content_type     | text/html                                                                    | Content type of article                           |
| created_at       | 2021-03-22T03:47:59.290Z                                                     | Time stamp of article creation                    |
| created_by_id    | 10                                                                           | User ID that created the article                  |
| from             | David Bell <david@example.com>                                               | From field of article creator                     |
| id               | 16                                                                           | Internal article ID                               |
| in_reply_to      | null                                                                         | In-Reply-To Header from emails if applicable      |
| internal         | false, true                                                                 | Defines if article is internal                     |
| message_id       | null                                                                         | Message ID of Email if applicable                 |
| origin_by_id     | null                                                                         | User ID or original creator if created on behalf another user |
| preferences      | {}                                                                           | Internal preferences, may be empty, mainly for delivery states |
| references       | null                                                                         | Contains message references                       |
| reply_to         | null                                                                         | Contains reply to header if applicable            |
| sender_id        | 2                                                                            | ID of sender type (Customer, System, Agent)       |
| subject          | My amazing subject                                                          | Article subject                                  |
| ticket_id        | 9                                                                            | Ticket ID the article belongs to                  |
| to               | support@example.com                                                          | EMail address from TO-Header                      |
| type_id          | 1                                                                            | ID of articles Type (phone, email, web, …)        |
| updated_at       | 2021-03-22T03:47:59.290Z                                                     | Last update                                      |
| updated_by_id    | 10                                                                           | User that updated article                         |

### User

**Tip:** The following indice contains below mentioned information: *._user*

### Table 5: User-Index

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>true, false</td>
<td>Defines if user is active</td>
</tr>
</tbody>
</table>
| address          | Bennelong Point

Sydney NSW 2000 | Address string                                                              |
| city             | Berlin                                                                       | City string                                      |
| country          | Germany                                                                      | Country string                                   |
| created_at       | 2021-03-22T12:47:56.460Z                                                     | Creation date of user                            |
| created_by_id    | 1                                                                            | User ID that created the user                     |
| department       | IT                                                                            | Department string                                |
| email            | alias@domain.tld                                                             | EMail Address of user, if applicable              |
| fax              | 1234                                                                         | Fax number                                       |
| firstname        | null, John                                                                   | Users first name                                 |

continues on next page
Table 5 – continued from previous page

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>8</td>
<td>Internal User ID</td>
</tr>
<tr>
<td>last_login</td>
<td>null, 2021-03-23T12:47:56.460Z</td>
<td>Updated upon every user login</td>
</tr>
<tr>
<td>lastname</td>
<td>null, Doe</td>
<td>Users last name</td>
</tr>
<tr>
<td>login</td>
<td>auto-1234567, jdoe</td>
<td>Login name, always set and unique, can differ from email</td>
</tr>
<tr>
<td>mobile</td>
<td>***.1232</td>
<td>Mobile phone number</td>
</tr>
<tr>
<td>note</td>
<td>***</td>
<td>Note being available via web, console and API</td>
</tr>
<tr>
<td>organization</td>
<td>#{organization object}</td>
<td>Complete Payload of the organization the user is member of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see Organization for more</td>
</tr>
<tr>
<td>organization_id</td>
<td>3</td>
<td>ID of organization the user is member of</td>
</tr>
<tr>
<td>out_of_office</td>
<td>false, true</td>
<td>Defines if user has activated out of office function</td>
</tr>
<tr>
<td>out_of_office_end_at</td>
<td>null, 2021-03-26</td>
<td>Ending date out of office</td>
</tr>
<tr>
<td>out_of_office_replacement</td>
<td>null, 3</td>
<td>User ID that replaces this user during out of office period</td>
</tr>
<tr>
<td>out_of_office_start_at</td>
<td>null, 2021-03-24</td>
<td>Begin date out of office</td>
</tr>
<tr>
<td>permissions</td>
<td>(Array)</td>
<td>Array with all permissions of the user</td>
</tr>
<tr>
<td>phone</td>
<td>***.0061 2 1234 7777</td>
<td>Phone number of user</td>
</tr>
<tr>
<td>preferences</td>
<td>{}, # { several preference attributes }</td>
<td>Depends on user and situation, may contain notification_config, locale and other internal system information</td>
</tr>
<tr>
<td>role_ids</td>
<td>(Array), [1, 2]</td>
<td>Contains array with role IDs assigned to the user</td>
</tr>
<tr>
<td>street</td>
<td>***</td>
<td>Street</td>
</tr>
<tr>
<td>updated_at</td>
<td>2021-03-25T00:27:52.308Z</td>
<td>Time stamp of last update</td>
</tr>
<tr>
<td>updated_by_id</td>
<td>3</td>
<td>User ID that updated this entry</td>
</tr>
<tr>
<td>verified</td>
<td>false, true</td>
<td>Defines if the user has verified the account</td>
</tr>
<tr>
<td>vip</td>
<td>false, true</td>
<td>Defines if user has VIP state</td>
</tr>
<tr>
<td>web</td>
<td>***.<a href="https://zammad.org">https://zammad.org</a></td>
<td>Web URL of User</td>
</tr>
<tr>
<td>zip</td>
<td>***.10123</td>
<td>ZIP code</td>
</tr>
</tbody>
</table>

**Organization**

**Tip:** The following indice contains below mentioned information: * _organization
Table 6: Organization-Index

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>true, false</td>
<td>Defines if organization is active</td>
</tr>
<tr>
<td>created_at</td>
<td>2021-03-22T12:47:54.807Z</td>
<td>Creation date</td>
</tr>
<tr>
<td>created_by</td>
<td>#{user object}</td>
<td>Complete Payload of the user that created the organization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see User for more</td>
</tr>
<tr>
<td>created_by_id</td>
<td>1</td>
<td>User ID that created the organization</td>
</tr>
<tr>
<td>domain</td>
<td>null, example.com</td>
<td>Organizations domain</td>
</tr>
<tr>
<td>domain_assignment</td>
<td>false, true</td>
<td>Domain assignment depends on domain</td>
</tr>
<tr>
<td>id</td>
<td>1</td>
<td>Organization ID</td>
</tr>
<tr>
<td>members</td>
<td>#{array of user objects}</td>
<td>Array with complete Payload of the users being member of the organization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see User for more</td>
</tr>
<tr>
<td>name</td>
<td>Chrispresso Inc.</td>
<td>Organization name</td>
</tr>
<tr>
<td>note</td>
<td>Manufacturer of individual coffee products.</td>
<td>Note being available via web, console and API</td>
</tr>
<tr>
<td>shared</td>
<td>true, false</td>
<td>Defines if the organization is a sharing one</td>
</tr>
<tr>
<td>updated_at</td>
<td>2021-03-22T12:47:54.807Z</td>
<td>Last update time</td>
</tr>
<tr>
<td>updated_by</td>
<td>#{user object}</td>
<td>Complete Payload of the user that updated the organization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Please see User for more</td>
</tr>
<tr>
<td>updated_by_id</td>
<td>1</td>
<td>User ID that updated the organization</td>
</tr>
</tbody>
</table>

**Group**

**Tip:** The following indice contains below mentioned information: *._group*
<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>true, false</td>
<td>Defines if group is active (available)</td>
</tr>
<tr>
<td>assignment_timeout</td>
<td>null, 30</td>
<td>Time in minutes an agent can be inactive until the ownership is removed</td>
</tr>
<tr>
<td>created_at</td>
<td>2021-03-24T23:55:06.980Z</td>
<td>Time stamp of group creation</td>
</tr>
<tr>
<td>created_by_id</td>
<td>1</td>
<td>User ID that created the group</td>
</tr>
<tr>
<td>email_address</td>
<td>Contains these attributes:</td>
<td>Contains all available information of the group's email address</td>
</tr>
<tr>
<td></td>
<td>• active: true</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• channel_id: 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• created_at: 2021-03-24T23:54:58.187Z</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• created_by_id: 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• email: <a href="mailto:alias@domain.tld">alias@domain.tld</a></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• id: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• note: null</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• realname: Zammad GmbH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• updated_at: 2021-03-24T23:54:58.187Z</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• updated_by_id: 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• preferences: null</td>
<td></td>
</tr>
<tr>
<td>email_address_id</td>
<td>3</td>
<td>ID of email address</td>
</tr>
<tr>
<td>follow_up_assignment</td>
<td>true, false</td>
<td>Defines if owners are still assigned after follow ups</td>
</tr>
<tr>
<td>follow_up_possible</td>
<td>yes, no</td>
<td>Defines if following up on a closed ticket is possible</td>
</tr>
<tr>
<td>id</td>
<td>1</td>
<td>Group ID</td>
</tr>
<tr>
<td>name</td>
<td>Users, Sales</td>
<td>Group name</td>
</tr>
<tr>
<td>note</td>
<td>null</td>
<td>Notes for the group available via web, console and API</td>
</tr>
<tr>
<td>signature</td>
<td>Contains these attributes:</td>
<td>Contains all available information of the group's signature</td>
</tr>
<tr>
<td></td>
<td>• active: true</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• body: &lt;br&gt; #{user.firstname} #{user.lastname}&lt;br&gt;--&lt;br&gt;That Inc</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• created_at: 2021-03-03T14:50:19.775Z</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• created_by_id: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• id: 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• name: default</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• note: null</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• updated_at: 2021-03-03T14:50:19.775Z</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• updated_by_id: 1</td>
<td></td>
</tr>
<tr>
<td>signature_id</td>
<td>1</td>
<td>Signature ID</td>
</tr>
<tr>
<td>updated_at</td>
<td>2021-03-24T23:55:06.980Z</td>
<td>Time stamp of last group update</td>
</tr>
<tr>
<td>updated_by_id</td>
<td>3</td>
<td>User ID that updated group</td>
</tr>
</tbody>
</table>
## CTI Log

**Tip:** The following indice contains below mentioned information: *_cti_log_

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>call_id</td>
<td>00006</td>
<td>Unique Call ID</td>
</tr>
<tr>
<td>comment</td>
<td>&quot;&quot;</td>
<td>Optional comment</td>
</tr>
<tr>
<td>created_at</td>
<td>2021-03-22T11:48:01.703Z</td>
<td>Creation date of Call</td>
</tr>
<tr>
<td>direction</td>
<td>in, out</td>
<td>Call direction</td>
</tr>
<tr>
<td>done</td>
<td>true, false</td>
<td>Defines if call is finished</td>
</tr>
<tr>
<td>duration_talking_time</td>
<td>27</td>
<td>Call duration in seconds</td>
</tr>
<tr>
<td>duration_waiting_time</td>
<td>77</td>
<td>Duration in seconds the caller was waiting for answer</td>
</tr>
<tr>
<td>end_at</td>
<td>2021-03-25T08:49:40.647Z</td>
<td>Time stamp of call end</td>
</tr>
<tr>
<td>from</td>
<td>493055571600</td>
<td>Calling number</td>
</tr>
<tr>
<td>from_comment</td>
<td>null, John, Doe</td>
<td>Display name of calling number if applicable</td>
</tr>
<tr>
<td>from_pretty</td>
<td>+49 30 55571600</td>
<td>Pretty version of from</td>
</tr>
<tr>
<td>id</td>
<td>8</td>
<td>Internal ID of entry</td>
</tr>
<tr>
<td>initialized_at</td>
<td>2021-03-25T08:47:56.753Z</td>
<td>Time stamp of call initialization, usually matches created_at</td>
</tr>
<tr>
<td>preferences</td>
<td>(Array)</td>
<td>Contains internal information if required</td>
</tr>
<tr>
<td>queue</td>
<td>null, 491711234567890</td>
<td>Queue the call was answered in</td>
</tr>
<tr>
<td>start_at</td>
<td>2021-03-25T08:49:13.050Z</td>
<td>Time stamp the call was answered</td>
</tr>
<tr>
<td>state</td>
<td>hangup, voicemail</td>
<td>Last state of call</td>
</tr>
<tr>
<td>to</td>
<td>491711234567890</td>
<td>Dialed number</td>
</tr>
<tr>
<td>to_comment</td>
<td>null, John, Doe</td>
<td>Display name of called number if applicable</td>
</tr>
<tr>
<td>to_pretty</td>
<td>+491711234567890</td>
<td>Pretty version of to</td>
</tr>
<tr>
<td>updated_at</td>
<td>2021-03-25T08:49:40.647Z</td>
<td>Last update of entry</td>
</tr>
</tbody>
</table>

## Chat Session

**Tip:** The following indice contains below mentioned information: *_chat_session_

6.4. Appendix
Table 9: Chat Session-Index

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chat</td>
<td>Contains these attributes:</td>
<td>Contains various preferences of the chat topic in charge</td>
</tr>
<tr>
<td>chat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>active: true</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>block_country: null</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>block_ip: null</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>created_at: 2021-03-03T14:50:22.607Z</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>created_by_id: 1</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>id: 1</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>max_queue: 5</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>name: default</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>note: &quot;&quot;</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>preferences: {}</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>public: false</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>updated_at: 2021-03-03T14:50:22.607Z</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>updated_by_id: 1</td>
<td></td>
</tr>
<tr>
<td>chat</td>
<td>whitelisted_websites: null</td>
<td></td>
</tr>
<tr>
<td>chat_id</td>
<td>1</td>
<td>ID of Chat topic</td>
</tr>
<tr>
<td>created_at</td>
<td>2021-03-25T10:26:24.376Z</td>
<td>Time stamp of chat creation</td>
</tr>
<tr>
<td>created_by_id</td>
<td>null</td>
<td>User that created the chat, place holder, currently always null</td>
</tr>
<tr>
<td>id</td>
<td>1</td>
<td>ID of Chat Session</td>
</tr>
<tr>
<td>messages</td>
<td>(Array) - Array entries contain these attributes:</td>
<td>Array with all messages of chat</td>
</tr>
<tr>
<td>name</td>
<td>null, John Doe</td>
<td>Name agent set for chat user, if applicable</td>
</tr>
<tr>
<td>preferences</td>
<td>Contains these attributes:</td>
<td>Various internal Meta data of the session_id</td>
</tr>
<tr>
<td></td>
<td>dns_name: host.domain.tld</td>
<td></td>
</tr>
<tr>
<td>preferences</td>
<td>_geo_ip: {}</td>
<td></td>
</tr>
<tr>
<td>preferences</td>
<td>participants: Array, [&quot;47118371175780&quot;, &quot;47118371850300&quot;]</td>
<td></td>
</tr>
<tr>
<td>preferences</td>
<td>remote_ip: 192.168.2.19</td>
<td></td>
</tr>
<tr>
<td>preferences</td>
<td>url: <a href="https://zammad.com/en/company/contact">https://zammad.com/en/company/contact</a></td>
<td></td>
</tr>
<tr>
<td>session_id</td>
<td>92f2909631ff1ad5ff4d5d1e046952be8</td>
<td>Uniquely identifies the chat session</td>
</tr>
<tr>
<td>state</td>
<td>closed</td>
<td>Current state of chat session</td>
</tr>
<tr>
<td>tags</td>
<td>(Array), [&quot;order&quot;]</td>
<td>Tags applied to Chat Session by agent, if applicable</td>
</tr>
</tbody>
</table>
Warning: We currently do not support Docker environments in productive use. If you run Zammad on docker, it is fine. But we just support the application!

Note: Docker Compose environments require deeper system know how. If you’re not too familiar with Docker and the way it works, you may want to stick with the package installation instead.

Docker is a container-based software framework for automating deployment of applications. Compose is a tool for defining and running multi-container Docker applications.

This repo is meant to be the starting point for somebody who likes to use dockerized multi-container Zammad in production. The Zammad Docker image uses the stable branch of Zammad’s Git repo.

The Docker images are hosted on Dockerhub.

Warning: Never use the latest tag. Use a tag which has a version attached.

You need at least 4 GB of RAM to run the containers.

7.1 Install Docker Environment

This documentation expects you already have a working Docker Compose environment. You can find the required documentations for these steps below:

- Docker Engine
- Docker Compose
7.2 Getting started with zammad-docker-compose

7.2.1 Docker Compose Environment Variables

Zammad’s Docker Compose supports several environment variables that are not set by default. The best way to provide these is within the file `.env`.

In case our default `docker-compose.yml` is not good enough, please use `docker-compose.override.yml` to provide own changes.

**Docker Compose specific**

- **RESTART**: `always` By default containers will be restarted in case they stopped for whatever reason.
- **VERSION** This variable contains the version tag. Example: `3.6.0-20`
  
  We update this string from time to time, Docker Hub may contain more current tags to the moment you’re pulling.

**Zammad**

- **AUTOWIZARD_JSON**: `''` This variable allows you to provide initial configuration data for your instance. Autowizard JSON is out of scope of this documentation, however this example file should help you.
- **RAILS_TRUSTED_PROXIES**: `['127.0.0.1', '::1']` By default Zammad trusts localhost proxies only.
  
  Only change this option if you know what you’re doing!

**Elasticsearch**

- **ELASTICSEARCH_ENABLED**: `true` Setting this variable to false will allow you to run your Zammad without Elasticsearch. Please note that we strongly advise against doing so.
- **ELASTICSEARCH_HOST**: `zammad-elasticsearch` Provide a host name or address to your external Elasticsearch cluster.
- **ELASTICSEARCH_PORT**: `9200` Provide a different port for Elasticsearch if needed.
- **ELASTICSEARCH_SCHEMA**: `http` By default Elasticsearch is reachable via HTTP.
- **ELASTICSEARCH_NAMESPACE**: `zammad` With this name space all Zammad related indexes will be created. Change this if you're using external clusters.
- **ELASTICSEARCH_REINDEX**: `true` By default the docker-compose will always re-index upon a restart. On big installations this may be troublesome.

  **Warning:** Disabling this setting requires you to re-index your search index manually whenever that’s needed by upgrading to a new Zammad version!

- **ELASTICSEARCH_SSL_VERIFY**: `true` Allows you to let the compose scripts ignore self signed SSL certificates for your Elasticsearch installation if needed.
**Memcached**

**MEMCACHED_HOST**: `zammad-memcached` Provide your own Memcached instance if you already have one existing.

**MEMCACHED_PORT**: `11211` Memcached's default port.

**Nginx**

**NGINX_PORT**: `8080` The port Nginx will listen on.

**NGINX_SERVER_NAME**: `_` By default the Nginx container of Zammad will respond to all requests. You can provide your IP / FQDN if you want to.

**NGINX_SERVER_SCHEME**: `\$scheme` If the Nginx container for Zammad is not the upstream server (aka you're using another proxy in front of nginx) `\$scheme` may be wrong. You can set the correct scheme `http` or `https` if needed.

---

**Tip**: Can’t login because of CSRF token errors?

This usually affects systems with more than one proxy server only. For this to function you may have to tell your web server directly which connection type was used.

---

**Warning**: **Do not** use below options if you're unsure, they may technically be a security issue!

The following options expect HTTPS connections which should be your goal.

* nginx
* apache2

Within your virtual host configuration, locate both directives `proxy_set_header X-Forwarded-Proto` and replace `\$scheme` by `https`.

Within your virtual host configuration just above the first `ProxyPass` directive insert:

```
RequestHeader set X_FORWARDED_PROTO 'https'
RequestHeader set X-Forwarded-Ssl on
```

---

**ZAMMAD_RAILSSERVER_HOST**: `zammad-railsserver` Host name of the rails server container.

**ZAMMAD_RAILSSERVER_PORT**: `3000` Port of Zammad's rails server.

Please also note **Configuration via Environment Variables** in this regard.

**ZAMMAD_WEBSOCKET_HOST**: `zammad-websocket` Host name of Zammad's websocket server.

**ZAMMAD_WEBSOCKET_PORT**: `6042` Port of Zammad's websocket server.

Please also note **Configuration via Environment Variables** in this regard.
Zammad

**PostgreSQL**

**POSTGRESQL_HOST:** `zammad-postgresql` Host name of your PostgreSQL server. Use your own if you already have one.

**POSTGRESQL_PORT:** 5432 Adjust the Port of your PostgreSQL server.

**POSTGRESQL_USER:** `zammad` The database user for Zammad.

**POSTGRESQL_PASS:** `zammad` The password of Zammad's database user.

**POSTGRESQL_DB:** `zammad_production` Zammad's database to use.

**POSTGRESQL_DB_CREATE:** `true` By default we will create the required database.

---

**Note:** On own database servers this setting might be troublesome.

**RSYNC_ADDITIONAL_PARAMS:** `--no-perms --no-owner` By default the compose will copy data without permissions and owners. This may not fit for your storage driver.

---

### 7.2.2 Step 1: Clone GitHub repo

```bash
$ git clone https://github.com/zammad/zammad-docker-compose.git
$ cd zammad-docker-compose
```

### 7.2.3 Step 2: Setting `vm.max_map_count` for Elasticsearch

```bash
$ sysctl -w vm.max_map_count=262144
```

---

**Tip:** Mac OS users please also have a look on Issue 27

### 7.2.4 Step 3: Adjust Environment as needed

In some cases our default environment is not what a docker-compose user is looking for. To remove complexity from this page, we outsourced information on this topic.

See `Docker Compose Environment Variables`

### 7.2.5 Step 4: Start Zammad using DockerHub images

---

**Warning:** Before starting your containers ensure to not use default login data for your Zammad database! See Step 3!

```bash
$ docker-compose up
```
7.3 Next steps

With this Zammad technically is ready to go. However, you’ll need to follow the following further steps to access Zammads Web-UI and getting started with it.

1. Connect Zammad with Elasticsearch
2. Configure the webserver
3. First steps
4. You may also find Zammads Console commands useful

If you expect usage with 5 agents or more you may also want to consider the following pages.

- Performance Tuning
- Configure Database server
Warning: We currently do not support Kubernetes installations in productive use.

Kubernetes (k8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

Helm is the package manager for Kubernetes.

This repo is meant to be the starting point for somebody who likes to use dockerized multi-container Zammad on Kubernetes. The Zammad Docker image uses the stable branch of Zammad’s Git repo.

The used Docker images are hosted on Dockerhub.

You need the Helm binary installed / initialized and at least 4 GB of free RAM in the Kubernetes cluster run the containers.

```
# Add Helm repo
$ helm repo add zammad https://zammad.github.io/zammad-helm

# Install / Upgrade Zammad
$ helm upgrade --install zammad zammad/zammad --namespace=zammad
```
Note: As Zammad is using Docker Compose for Univention Corporate Server, the minimum requirement is UCS 4.3-2 errata 345.

Univention Corporate Server (UCS) is an enterprise server with focus on identity and infrastructure management. With its marketplace called App Center it can easily extended by solutions like Zammad that benefit from integrations with the LDAP directory service and the mail infrastructure.

Click here to learn more about Univention and what it can do for you.

9.1 Prerequisites

To install the Zammad app on UCS, please ensure that you're using at least UCS 4.3-2 errata 345. The basic installation will already meet our requirement. You’ll need the following additional things:

- An email server (no matter if handled via UCS or with an external system) for notifications, as you can’t use sendmail in our Docker setup!
- You should at least have **2 CPU-Cores and 4GB of free RAM**.

Note: Running the Zammad app with less than 4GB free RAM will lead to unexpected errors!

You see, that’s not much - so go a head with the installation.

9.2 Installing Zammad

The app installation itself is quite easy: Just open the App Center within UCS management system and search for Zammad. Press **Install**, accept our license agreement and wait for the installation to finish.
The installation will take about 5-15 minutes, depending on your hardware speed. Please give the installation the needed time and don’t abort. During the automated setup there are some waits for services to come up. Please be patient!

If it’s finished, you can press open - you’ll get to our Zammad Wizard. It helps you with the minimum of information we need. (See First steps you should consider)
9.2.1 Values we automatically change during the UCS-Setup

In order to make the installation as complete and convenient as possible, we're changing the following default values to the following:

<table>
<thead>
<tr>
<th>value</th>
<th>default value</th>
<th>new value</th>
</tr>
</thead>
<tbody>
<tr>
<td>notification sender</td>
<td>Notification Master &lt;noreply@#{config.fqdn}&gt;</td>
<td>Zammad &lt;noreply@{FQDN-of-UCS}&gt;</td>
</tr>
<tr>
<td>maximum email size</td>
<td>10 MB</td>
<td>35 MB</td>
</tr>
<tr>
<td>FQDN</td>
<td>[FQDN-of-UCS]</td>
<td>[FQDN-of-UCS]:10412</td>
</tr>
<tr>
<td>HTTP-Type</td>
<td>&lt;empty&gt;</td>
<td>https</td>
</tr>
<tr>
<td>Allow customer registra-</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>tion</td>
<td>&lt;empty&gt;</td>
<td>Full LDAP-Configuration prepared</td>
</tr>
<tr>
<td>LDAP configuration</td>
<td>&lt;empty&gt;</td>
<td></td>
</tr>
<tr>
<td>LDAP activated</td>
<td>&lt;empty&gt;</td>
<td>false</td>
</tr>
</tbody>
</table>

Note: 2 Please note that the Zammad-LDAP integration is pre filled with authentication data and the group mapping Zammad-Admin to the Admin-Role and Zammad-Agent to the Agent-Role. You can use those security groups. LDAP synchronization is disabled during installation, as activating it would disable the installation wizard of Zammad, which is needed to setup your Zammad instance properly.

Note: 3 Please note that these settings are updated automatically, if you update FQDN and Port settings within the Univention App-Settings.

9.3 First steps you should consider

The most important part is obvious: Run the wizard and insert the information for your admin account.

Warning: If the email address is used within UCS, you need to ensure that your user account within UCS has the needed Admin-Group, as otherwise a LDAP synchronization will downgrade your user account to the setup role!

You can now enter your company name and upload a company logo, if you want to. (the company name is mandatory). The system URL has been set by our installation routine already, you should be good to continue without changing it.

Note: Changing the system URL might lead to broken links within notification mails.

For the notification sender, you should use SMTP, as the Docker container does not come with any sendmail or local MTA. If you choose local MTA, Zammad will not be able to send you any notifications.

The last step offers you to add your first email accounts to Zammad. You're free to skip this step, you can configure your accounts later, as well.

Zammad is now ready to go.
The identity management integration with UCS LDAP directory allows the system administrator to maintain the users at one single point. If you want to take advantage of UCS identity management integration, you need to do the following before hand:

- Add your desired Zammad admin users to the user group Zammad-Admin.
- Add your desired Zammad agents to the user group Zammad-Agent.
- All user accounts that are not covered by the default group mapping, will be added in the Zammad customer role.

You can now go to Admin-Settings -> Integration -> LDAP and simply activate LDAP. The first LDAP synchronization will start shortly thereafter - Zammad will then synchronize user account data with the UCS LDAP directory hourly.

Note: You're free to change the group-role mapping at any time. See Configuring LDAP integration for more information.

9.3.1 Further configuration

The rest of the configuration is pretty straight forward and applies to our default. We split our documentation into two further parts that will be of your interest:

- Admin-Documentation: this documentation holds any information about how to configure Zammad via WebApp.
- User-Documentation: this documentation holds a complete user documentation (how to work with Zammad).

9.4 Further information

The following sub pages might come in handy and help you to understand how the app works or on how to solve an issue.

9.4.1 Univention App-Settings

Note: The App-Settings part is only valid starting with App version 3.1.0-7.

Within the managment interface of Univention, you can change some access related settings:

- FQDN of Zammad (Domain you can access it from)
- Another Port
- Other certificates beside the Univention certificate

Note: Some settings require you to have a combination of the above settings. If the combination of settings is not met, the update script will automatically revert these changes.

This ensures that your Univention Host stays operational. Please do not try to trick the scripts, it might cause outages.

If you're using the default settings, we'll empty the values of the text fields to reduce confusion. This is not a Bug, but a “Feature”.
Using another FQDN

This consists of two settings: a selection and a text field. The default setting of the App is Default UCS-Hostname which will use the FQDN of the Univention-Host (and it's SSL-Certificate).

Note: In order to use custom hostnames, please also ensure to use a custom certificate.

If you set Custom Hostname, you'll need to enter a hostname in the text field below. Ensure that your Univention host can resolve the hostname (and it’s pointing to the host in question!).

Note: We won’t create any DNS entries or certificates during this process.

Using another Port

Currently you can choose between Default Highport and Port 8443. By default, we’ll use Port 10412 for Zammad, if you decide for Port 8443, we’ll handle the firewall steps needed and adjust the vHosts-Port.

Warning: Please ensure that if you choose another Port, that it’s not used already! We do not verify this!

Note: Please note that for technical reasons (how Univention and Zammad work) it’s not possible to use Zammad on Port 443 or within a subdirectory.

Using other certificates

By default we’re using the Univention-Host certificate (Univention (default)). In some cases (and especially if you’re using custom hostnames) this might be troublesome.

Note: You can use custom certificates without changing the hostname.

Warning: We’re not verifying if the certificates are valid in any way (e.g. still valid in time and if the hostname is inside). This step might follow, but please be aware that this might lead to certificate issues.

If you choose Let’s Encrypt, please ensure that you already have installed the Let’s Encrypt App (by Univention GmbH) and also already acquired a certificate via it. If you’re applying the settings, we’ll check for the following two files:

/etc/univention/letsencrypt/signed_chain.crt
/etc/univention/letsencrypt/domain.key

If we can’t find these, we’ll revert to the default Univention certificate.

You can also choose to use your very own certificate by selecting Custom Certificate. For this it’s important to know, that we expect the certificate to be within a specific location (/etc/univention/ssl/). Within the two text fields, you’ll need to provide the filenames of your certificate and your certificate-key.
These certificates can be kept in a subfolder. If we cannot find either of the two files, we reset the setting to the default Univention certificate.

### 9.4.2 Running console commands on an Univention-Host

In some cases you might need to access Zammad's rails console on the Univention-Host. For this, you'll need to get the correct container ID first.

Univention will hold this information for you, you can get it like that:

```
$ ucr get appcenter/apps/zammad/container
```

Now where we have our ID, you can run any command from the **Console** section with either:

```
$ docker exec -i "{Container-ID}" rails r "{COMMAND}"
```

or -if you need a console for more commands- by:

```
$ docker exec -i "{Container-ID}" rails c
```

**Note:** Please replace `{Container-ID}` in the above commands by the ID the first command returns. Replace `{COMMAND}` by any rails command Zammad supports.

That's it!

### 9.4.3 Issues you might encounter

Below we have gathered information to problems that might occur in combination with Univention.

**Zammad can't communicate with external systems**

In rare cases (sometimes even right after installation), Zammad won't be able to communicate with e.g. external e-mail servers. Simply restart the Zammad app in the App Center module in the UCS management system and it should be enough to get it back working.

**Zammad communicates a wrong URL within the notifications**

This issue rises because of how the "Getting Started"-Wizard in Zammad works. Even though the wizard reports the correct FQDN-Setting (and we manually set it), it will overwrite the setting after sanitizing what it recognized.

This is an application level issue and a subject to change - you can find more information about this in Issue 2651. To solve this, just go into the Univention app-settings (for Zammad-app) and apply the settings **after** finishing the wizard.
Customers can’t click on the “Knowledge Base”-URL within the customer portal

This currently can’t be fixed, as Zammad is available via one Port only. The issue is described within Issue 2628 and a subject to fix.

**Warning:** Never change any configurations the Zammad-App scripts create and work with! This will lead to unexpected issues and loss of configurations upon update!
UPDATING ZAMMAD

Note: Before updating to a new version, you may want to have a look into the official release notes. These will provide further information on new feature and fixes, but also technical remarks that may be relevant during an upgrade!

Package
Source
Docker Compose

Step 1: Stop Zammad

$ systemctl stop zammad

Step 2: Backup Zammad  See Backup and Restore for more information.

Step 3: Update Zammad  Ubuntu / Debian

CentOS
OpenSUSE / SLES

$ apt update
$ apt upgrade

$ yum update zammad

$ zypper ref
$ zypper up

Warning: The package comes with maintenance scripts that will run regular tasks during updates for you.

However
Do not run Zammad updates unattended and always have a look on the outputs these helper scripts generate. Ignoring said output may lead to incomplete updates that may corrupt data or lead to issues you find way too late.

Step 4: Run required extra steps  Extra steps needed for updates are mentioned in our release news.

Updating Elasticsearch may be relevant in this step.
### Step 5: Log into Zammad
Yes, that’s it!

### Step 1: Ensure dependencies
Before proceeding, double-check that your system environment matches Zammad's requirements.

**Tip:** Ruby version changed?
Please see [Installation part of source code installation](#).

### Step 2: Download Zammad to your system
Get the latest stable release of Zammad [here](#), or find the initial version at [https://ftp.zammad.com](https://ftp.zammad.com).

```
$ cd /opt
$ wget https://github.com/zammad/zammad/archive/stable.tar.gz
$ tar -xzf stable.tar.gz -C zammad
$ chown -R zammad:zammad zammad/
$ rm -f stable.tar.gz
```

### Step 3: Install Gems

```bash
$ su - zammad
$ cd /opt/zammad
$ gem install bundler
```

PostgreSQL

MySQL / MariaDB

```
$ bundle install --without test development mysql
$ bundle install --without test development postgres
```

### Step 4: Stop Zammad services
Stop the application server, websocket server and scheduler.

### Step 5: Upgrade your database

```
$ su - zammad
$ rake db:migrate
$ rake assets:precompile
```

### Step 6: Start Zammad services
Start the application server, websocket server and scheduler.

### Step 7: Log into Zammad
Yes, that’s it!

**Warning:** Updates may require extra steps or introduce breaking changes.
Always check the [upgrade notes](#) first.

**Note:** Incomplete documentation

Sorry, but this documentation part is outdated. We will rework this part later, but can’t tell when yet.

Please feel welcome to provide a pull request if you find spare time!
Zammad

```
$ docker-compose stop
$ git pull
$ docker-compose pull
$ docker-compose up
```

Start Zammad building Docker images locally with development branch

- `GIT_BRANCH=develop docker-compose -f docker-compose-build.yml up`

Recreate locally built images

- `GIT_BRANCH=develop docker-compose -f docker-compose-build.yml build --no-cache`

Open shell in running Zammad image

- `docker-compose exec zammad /bin/bash`

Port compatibility error

- The nginx container may have compatibility problems with other machines or services pointing to port 0.0.0.0:80.
  So to fix this, we’ll just have to modify the file `docker-compose.override.yml` and select different ports

### 10.1 Updating Elasticsearch

**Warning:** Updating Elasticsearch does not automatically update its plugins! This usually isn’t an issue if Zammad is being updated right after Elasticsearch.

If you want to upgrade your elasticsearch installation, please take a look at the [elasticsearch documentation](#) as it will have the most current information for you.

If, for whatever reason, you need to rebuild your search index after upgrading, use:

```
$ zammad run rake searchindex:rebuild
```

**Warning:** This step may fail if Zammad is under heavy load: Elasticsearch locks the indices from deletion if you’re pumping in new data, like receiving a new ticket. (This only applies to single-node deployments, not clusters.)

If it does, try killing Zammad first:

```
$ systemctl stop zammad
$ zammad run rake searchindex:rebuild
$ systemctl start zammad
```
CHAPTER ELEVEN

CONFIGURE THE WEBSERVER

You can find current sample configuration files for your webserver within contrib/ of your Zammad installation.

If you’re using the package installation, Zammad attempts to automatically install a configuration file to your nginx for you.

**Note:** The Zammad installation will not automatically set any host- or server name for you.

**Docker Compose / Kubernetes users**

Please also note the environment information on this page

11.1 Get a ssl certificate (recommended)

Don’t know how to get SSL certificates and install them on a webserver yet? The guide within the tabs below can help you jumping in.

I don’t need that

letsencrypt

public, paid CA

self-signed (discouraged)

You either already know what you’re doing, you’re developing or like the danger.

letsencrypt is an easy and free way to retrieve valid ssl certificates. These certificates are valid for 90 days and can be renewed automatically.

The two most common tools are certbot and acme.sh.

certbot

acme.sh

**Hint:** If not happened automatically, you may need to install the nginx or apache plugin for certbot: python3-certbot-nginx OR python3-certbot-apache

During the first certbot run it will request additional information once. Replace <webserver> in below command by either apache, httpd or nginx and to match your setup.
Certbot will now attempt to issue a certificate for you. If successful, certbot will ask you if you want to [1] not redirect or [2] redirect automatically. You can choose to not redirect if you plan to use the sample configuration of Zammad. If not, select [2] redirect.

From this moment on, certbot will automatically renew your installed certificates if they’re valid for another 30 days or less.

Hint: Not exactly what you’re looking for?
The cerbot documentation has alot more use cases than we cover here.

Hint: The most reliable way is to use the standalone method.

First of all you’ll need to issue your certificate. acme.sh will save this certificate to /root/.acme.sh/<your-domain>/

acme.sh --issue --standalone -d zammad.example.com

It’s not recommended to use the just stored certificates directly. Instead you should install the certificate to a directory of your choice.

We’re using /etc/ssl/private/ in this case, but you can use any directory you like.

Warning: Ensure to adjust value for --reloadcmd as this will ensure that acme.sh reloads your webserver automatically after getting a renewal. Replace <webserver> by either apache2, httpd or nginx

acme.sh --install-cert -d zammad.example.com \  
--cert-file /etc/ssl/private/zammad.example.com.pem \  
--key-file /etc/ssl/private/zammad.example.com.key \  
--fullchain-file /etc/ssl/private/zammad.example.com.full.pem \  
--reloadcmd "systemctl force-reload <webserver>"

From this moment on, acme.sh will automatically renew your installed certificates if they’re valid for another 30 days or less.

Hint: Not exactly what you’re looking for?
The acme.sh documentation has alot more use cases than we cover here.

If you prefer to use certificates from other official CAs than letsencrypt, you can do so as well. Just get your certificate bundle from the source you prefer and continue with Adjusting the webserver configuration.

Note: I’m new to SSL certificates. Where can I get a certificate?
The easiest way to get certificates is to buy an annual subscription through a commercial CA, such as:

• Sectigo (formerly Comodo)
Another way is to use self signed certificates from your own CA. In general you shouldn’t use this option when you have users accessing Zammad that can't verify your certificates.

Beside creating own certificates via e.g. XCA or Microsoft CA, you can also generate a certificate really quick like so:

On any system with openssl installed, you can run below command. Provide the requested information and ensure to provide the fqdn of Zammad when being asked for Common Name (e.g. server FQDN or YOUR name).

```
openssl req -newkey rsa:4096 -nodes -x509 -days 1825
  -keyout key.pem -out certificate.pem
```

Above command creates a certificate that's valid for 5 years. It will write the certificate and private key to the current directory you’re in. If you want to check your certificate you just created, you can use the following command.

```
openssl x509 -text -noout -in certificate.pem
```

**Hint:** Not good enough for you?

If above command is not good enough for you, the OpenSSL documentation is a good place to learn more.

### 11.2 Adjusting the webserver configuration

**Warning:** For a quick start, we’re installing a HTTP configuration. You should never use HTTP connections for authentication - instead, we encourage you to use HTTPS!

If Zammad scripts automatically installed your webserver configuration file, ensure to not rename it. Below we’ll cover HTTPs for above reason.

nginx (default)
apache2
local testing or other proxy servers

**Step 1 - Get a current config file** Copy & overwrite the default zammad.conf by using

```
$ cp /opt/zammad/contrib/nginx/zammad_ssl.conf /etc/nginx/sites-enabled/zammad.conf
```

**Note:**

Your nginx directories may differ, please adjust your commands if needed.

Most common:

- /etc/nginx/conf.d/
- /etc/nginx/vhosts.d/
- /etc/nginx/sites-enabled/
Step 2 - Adjust the config file  Adjust the just copied file with a text editor of your choice (e.g. vi or nano).

Locate any server_name directive and adjust example.com to the subdomain you have chosen for your Zammad instance.

Now you’ll need to adjust the path and file names for your ssl certificates yourobtained on the prior steps. Adjust the following directives to match your setup:

- ssl_certificate (your ssl certificate)
- ssl_certificate_key (the certificates private key)
- ssl_trusted_certificate (the public CA certificate)

Note:  Technically this is not a hard requirement, but recommended!

Hint:  Don’t have a dhparam.pem file yet?
You can easily adapt below example to generate this file. It will improve HTTPs security and thus should be used.

You can find the path by looking at your webserver configuration by looking for:

- ssl_dhparam directive (nginx)
- SSLOpenSSLConfCmd DHParameters directive (apache2)

```bash
$ openssl dhparam -out <path>/dhparam.pem 4096
```

(Optional) - Adjust HTTPs configuration  Our default configuration aims for a broad support of enduser devices. This may not fit your needs - Mozilla has a great ssl-config generator that should help you to meet your requirements!

Step 3 - Save & reload  Reload your nginx systemctl reload nginx to apply your configuration changes.

Step 1 - Ensure required modules are enabled  Zammad requires modules that are not enabled by default. By default use a2enmod (not CentOS) to do so.

```bash
a2enmod
title via configuration file (CentOS)

$ a2enmod proxy proxy_html proxy_http proxy_wstunnel headers ssl
title $ systemctl restart apache2
```

add/uncomment the appropriate LoadModule statements in your Apache config:

```bash
# /etc/httpd/conf/httpd.conf

LoadModule headers_module modules/mod_headers.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_html_module modules/mod_proxy_html.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
```

Don’t forget to restart your apache.
Step 2 - Get a current config file

**Note:** Package installations attempt to copy a `zammad.conf` to your webservers configuration directory. **Do not rename** this file!

Copy & overwrite the default `zammad.conf` by using

```
$ cp /opt/zammad/contrib/apache2/zammad_ssl.conf /etc/apache2/sites-enabled/zammad.conf
```

**Note:**
Your apache directories may differ, please adjust your commands if needed.
Most common:

- `/etc/apache2/conf.d/`
- `/etc/httpd/vhosts.d/`
- `/etc/apache2/sites-available/`

Step 3 - Adjust the config file

Adjust the just copied file with a text editor of your choice (e.g. `vi` or `nano`).

Locate any `ServerName` directive and adjust `example.com` to the subdomain you have chosen for your Zammad instance.

Now you'll need to adjust the path and file names for your ssl certificates you obtained on the prior steps. Adjust the following directives to match your setup:

- `SSLCertificateFile` (your ssl certificate)
- `SSLCertificateKeyFile` (the certificates private key)
- `SSLCertificateChainFile` (the public CA certificate)

**Note:** Technically this is not a hard requirement, but recommended!

**Hint:** Don’t have a dhparam.pem file yet?

You can easily adapt below example to generate this file. It will improve HTTPs security and thus should be used.

You can find the path by looking at your webserver configuration by looking for:

- `ssl_dhparam` directive (nginx)
- `SSLOpenSSLConfCmd DHParameters` directive (apache2)

```
$ openssl dhparam -out <path>/dhparam.pem 4096
```

(Optional) - Adjust HTTPs configuration

Our default configuration aims for a broad support of enduser devices. This may not fit your needs - Mozilla has a great `ssl-config generator` that should help you to meet your requirements!
Zammad

(Optional) - Enable the site

**Hint:** This step mostly depends on your selected folders and most often only affects sites-available folders.

Ubuntu / Debian / openSUSE

CentOS

```
$ a2ensite zammad

$ ln -s /etc/httpd/sites-available/zammad_ssl.conf /etc/httpd/sites-enabled/
```

Also, make sure the following line is present in your Apache configuration:

```
# /etc/apache2/apache2.conf (Ubuntu, Debian, & openSUSE)
# /etc/httpd/conf/httpd.conf (CentOS)

IncludeOptional sites-enabled/*.conf
```

**Step 4 - Save & reload**  Reload your apache systemctl reload apache2 to apply your configuration changes.

Want to test locally first or use a different Proxy we don’t support? The main application (rails server) is listening on http://127.0.0.1:3000.

If you’re using a proxy server, also ensure that you proxy the websockets as well. The websocket server listens on ws://127.0.0.1:6042.

**Tip:** If above ports are used by other applications already you may want to have a look at network options on our environment page.

**Warning:** Do not expose Zammad directly to the internet, as Zammad only provides HTTP!

If you just installed Zammad, you’ll be greeted by our getting started wizard.  You now can continue with *First steps*.

**Hint:** You’re not seeing Zammad’s page but a default landing page of your OS?

Ensure that you did restart your webserver - also check if 000-default.conf or default.conf in your vhost directory possibly overrules your configuration.

Sometimes this is also a DNS resolving issue.

**Tip:** Can’t login because of CSRF token errors?

This usually affects systems with more than one proxy server only. For this to function you may have to tell your web server directly which connection type was used.

**Warning:** Do not use below options if you’re unsure, they may technically be a security issue!
The following options expect HTTPS connections which should be your goal.

nginx

apache2

Within your virtual host configuration, locate both directives `proxy_set_header X-Forwarded-Proto` and replace `$scheme` by `https`.

Within your virtual host configuration just above the first `ProxyPass` directive insert:

```
RequestHeader set X_FORWARDED_PROTO 'https'
RequestHeader set X-Forwarded-Ssl on
```
After successfully installing Zammad you’ll have a couple of options.

- **Migrate from a supported source**
- **Restore Zammad from an existing backup**
- Start from scratch (move on to the next section)

### 12.1 Getting Started Wizard

If you visit Zammad’s web page the first time, you’ll be greeted by its Getting Started Wizard. It will guide you through the first most important things.

**Step 1: Create your very first administrator account** The fields should be fairly self explaining.

**Note:** Zammad does require the following password security by default:

- Password length of 10 or more
- 2 upper and 2 lower characters
- Contains at least one digit

**Step 2: Provide company information** You can upload a custom logo fitting to your company here. The instance address is detected automatically and only required adjustment in case it’s detected wrong.

All of these settings can be changed within Branding and System settings.

**Step 3: E-Mail notification channel** By default Zammad uses sendmail - if that doesn’t fit you can change it to SMTP here.

**Warning:** Zammad uses noreply@<your-fqdn> as sender address by default. SMTP setups might fail - you may want to skip this step with choosing sendmail at this point. You can adjust it later!

**Step 4: Your first email channel (optional)** If you want to start right away, you can connect your email account already.
Warning: Zammad reacts to fetched emails by default. If that’s not what you want, skip this step for now.

Learn more about the email channel within the documentation for email channels. After finishing the wizard you’ll be automatically logged in to the just created account.

12.2 Further Steps

In our opinion the next step order would like below sample. You can skip parts you don’t need or adapt. All parts are described within Zammad’s admin documentation.

1. Configure your required groups
2. Adjust triggers as needed
3. Add postmaster filters if needed
4. Configure SLAs if needed
5. add email / social media channels & signatures
   (go back to group settings to add outgoing email addresses)
6. Add Text Modules
7. Add Organizations
8. Configure roles if needed
9. Consider Third Party logins or LDAP integration for easier logins
10. Add agent accounts (users)
11. Consider backup strategies for Zammad. See Backup and Restore

From point 5 on you’ll be able to work productive in theory.
Hint: Are you still lost?

If above list doesn’t help you or you’ll need to jump in a lot faster, you can also get Workshops with one of our Zammad consultants.
Zammad will migrate the following information:

- Tickets and their Articles
- Groups / Queues
- Organizations
- Agents and Customers (if applicable)

After migrating to Zammad you’ll want to continue with the First steps to configure Zammad. This has to be done after migration.

13.1 Limitations

There might be source dependent limitations which we will be covering on the direct migration pages. However, these limitations count for all migrations:

- Migrations are only possible on new instances
- Zammad can’t migrate object types it doesn’t know, migrations will fail

13.2 Available Migration Options

13.2.1 Migration from OTRS

Limitations

Please note below OTRS specific limitations. These are additional limitations to the general ones listed.

- Password migration works for OTRS >= 3.3 only
  (on older instances a password reset within Zammad will be required)
- If you plan to import a differential migration after, do not change any data in Zammad!
- Only customers of tickets are imported
- Zammad expects your OTRS timestamps to be UTC and won’t adjust them
- If you plan to import a differential after, do not change any data in Zammad!
Prerequisites

Step 1: Install Znuny4OTRS-Repo

This is a dependency of the OTRS migration plugin.

OTRS 6
OTRS 5
OTRS 4
OTRS 3


https://addons.znuny.com/api/addon_repos/public/142/latest

Step 2: Install OTRS migration plugin

OTRS 6
OTRS 5
OTRS 4
OTRS 3


https://addons.znuny.com/api/addon_repos/public/617/latest


Hint: In some cases restarting your webserver may help to solve internal server errors.

Importing OTRS data

via Browser
via Console

Note: If your OTRS installation is rather huge, you might want to consider using the command line version of this feature. This also applies if you experience Timeouts during the migration.
After installing Zammad and configuring your webserver, navigate to your Zammads FQDN in your Browser and follow the migration wizard.

Depending on the size of your OTRS installation this may take a while.

You can get an idea of this process in the migrator video on vimeo.

**Hint:** We have a dedicated page for Zammad's rails console to reduce this pages complexity.

If you miss this at the beginning or you want to re-import again you have to use the command line at the moment.

Stop all Zammad processes and switch Zammad to import mode (no events are fired - e.g. notifications, sending emails, ...)

**Start the migration** Ensure to replace `xxx` with your values.

```
>> Setting.set('import_otrs_endpoint', 'https://xxx/otrs/public.pl?
   →Action=ZammadMigrator')
>> Setting.set('import_otrs_endpoint_key', 'xxx')
>> Setting.set('import_mode', true)
>> Import::OTRS.start
```

**Finish the migration**

```
>> Setting.set('import_mode', false)
>> Setting.set('system_init_done', true)
```

After successfully migrating your OTRS installation, continue with First steps.

**Importing a differential**

**Note:** This is only possible after finishing an earlier OTRS import successful.

In some cases it might be desirable to update the already imported data from OTRS. This is possible with the following commands.

**Run a differential import**

```
>> Setting.set('import_otrs_endpoint', 'http://xxx/otrs/public.pl?
   →Action=ZammadMigrator')
>> Setting.set('import_otrs_endpoint_key', 'xxx')
>> Setting.set('import_mode', true)
>> Setting.set('system_init_done', false)
>> Import::OTRS.diff_worker
```

**Set Zammad back into normal working mode**

```
>> Setting.set('import_mode', false)
>> Setting.set('system_init_done', true)
```

All changes that occurred after your first migration should now also be available within your Zammad installation.
Zammad

Restarting from scratch

Turned wrong at some point? You can find the required commands to reset Zammad in our Dangerzone.

13.2.2 from Zendesk

Note: Incomplete documentation

Sorry, but this documentation part is missing. We will add this part later, but can’t tell when yet.

Please feel welcome to provide a pull request if you find spare time!

Note: Missing a migration source?

If we don’t cover your favorite source yet, you’ll have two options. You can either fiddle around by using Zammads powerful API or drop our sales team a message for a custom development or even migrator sponsoring.

Migrations are available for hosted setups too, contact support for further information!
Zammad uses Ruby on Rails so you can make use of the rails console.

**Warning:** Please double check your commands before running, as some of those commands might cause data loss or damaged tickets! If you’re unsure, use a test system first!

To open the rails console on the shell you have to enter the following commands.

### 14.1 Start Zammad’s Rails console

#### 14.1.1 Running a single command

The following command will allow you to run a single command, without running a shell (e.g. for automation).

**Note:** Replace `{COMMAND}` with your command you want to run.

**Tip:** If you enter a `p` in front of your command (e.g. like `rails r 'p Delayed::Job.count'`), you’ll actually receive a printed output (without you won’t!).

```bash
# package installation
$ zammad run rails r '{COMMAND}'

# source installation
$ rails r '{COMMAND}'
```

#### 14.1.2 Running several commands in a shell

The following command will provide you a rails console, you can run several commands inside it.

```bash
# package installation
$ zammad run rails c

# source installation
$ rails c
```
14.2 Working on the console

Here’s a topic list for quick jumping and better overview.

14.2.1 Getting and Updating Zammad-Settings

Note: Please note that this is not a full setting list, if you’re missing settings, feel free to ask over at the Community.

Get ticket_hook setting

This will give you the Ticket hook that you’ll find inside the [] in front of the ticket number. By default this will be Ticket# - you shouldn’t change this setting in a productive system.

```
>> Setting.get('ticket_hook')
```

Get fqdn setting

Get the current FQDN-Setting of Zammad and, if needed, adjust it.

```
>> Setting.get('fqdn')               # Get FQDN
>> Setting.set('fqdn', 'new.domain.tld') # Set a new FQDN
```

Find storage_provider setting

The following command returns a list of available settings for storage_provider (for attachments).

```
>> Setting.find_by(name: 'storage_provider')
```

Set storage_provider Setting

Change the storage_provider if needed.

```
>> Setting.set('storage_provider', 'DB')  # Change Attachment-Storage to database
>> Setting.get('storage_provider')        # get the current Attachment-Storage
```

Configuring Elasticsearch

If your elasticsearch installation changes, you can use the following commands to ensure that Zammad still can access elasticsearch.

```
>> Setting.set('es_url', 'http://127.0.0.1:9200')  # Change elasticsearch URL
to poll
>> Setting.set('es_user', 'elasticsearch')      # Change elasticsearch user
  (e.g. for authentication)
>> Setting.set('es_password', 'zammad')         # Change the elasticsearch
  password for authentication
```

(continues on next page)
Use the OTRS importer from the shell

If needed, you can configure and run the OTRS-Import from console.

```
>> Setting.set('import_otrs_endpoint_key', 'xxx')
>> Setting.set('import_mode', true)
>> Import::OTRS.start
```

Enable proxy

Zammad needs to use a proxy for network communication? Set it here.

```
>> Setting.set('proxy', 'proxy.example.com:3128')
>> Setting.set('proxy_username', 'some user')
>> Setting.set('proxy_password', 'some pass')
```

14.2.2 Advanced customization settings

On this page you can find some settings that you won’t find within the Zammad UI. Those settings might come in handy as it can change Zammad’s behavior.

**Note:** Please note that this is not a full command list, if you’re missing commands, feel free to ask over at the Community.

Send all outgoing E-Mails to a BCC-Mailbox

This option allows you to send all outgoing E-Mails (not notifications) to a specific mailbox. Please note that this shouldn’t be a mailbox you’re importing already! This will apply to all groups and is a global setting.

```
>> Setting.set('system_bcc', 'alias@domain.tld')
```

You can easily check the current BCC-Setting by running the following:

```
>> Setting.get('system_bcc')
```

14.2. Working on the console
Zammad

Activate counter on grouped overviews

This is a hidden setting which you can only set via Command-Line. This will globally enable a ticket number value in each heading for grouped elements.

```ruby
>> Setting.set('ui_table_group_by_show_count', true)  # enable counter on grouped overviews
>> Setting.set('ui_table_group_by_show_count', false)  # disable counter on grouped overviews
>> Setting.get('ui_table_group_by_show_count')  # get current setting ('nil' is false)
```

Default Ticket type on creation

Zammad allows you to define the default article type upon Ticket creation. By default this will be an incoming phone call. You can choose between phone-in (incoming call, default), phone-out (outgoing call) and email-out (Sending an E-Mail out).

```ruby
>> Setting.set('ui_ticket_create_default_type', 'email-out')
```

To check what setting is set currently, simply run

```ruby
>> Setting.get('ui_ticket_create_default_type')
```

Adding a warning to the ticket creation process

If in case you need to give your agent a note or warning during ticket creation, you can do so with the below command. You can use three different warnings for Incoming Calls : "phone-in=>"", Outgoing Calls : "phone-out=>"" and Outgoing E-Mails : "email-out=>"".

```ruby
>> Setting.set('ui_ticket_create_notes', {:"phone-in=>""You're about to note a incoming phone call."", :"phone-out=>""You're about to note an outgoing phone call."", :"email-out=>""You're going to send out an E-Mail."})
```

Note: You can use those three sub-settings independently, if you e.g. don't need a warning on incoming calls, simply leave out :"phone-in=>"" out of the setting. The setting itself is done within an array ({ }).

To check what's currently set, you can use:

```ruby
>> Setting.get('ui_ticket_create_notes')
```
Sample of the above setting:

Show E-Mail-Address of customer on customer selection (Ticket-Creation)

By default Zammad will not display the E-Mail-Addresses of customers. The below option allows you to change this behavior.

```ruby
>> Setting.set('ui_user_organization_selector_with_email', true)
```

Get the current state of this setting with:

```ruby
>> Setting.get('ui_user_organization_selector_with_email')
```

Change Font-Settings for outgoing HTML-Mails

**Note:** Some Clients (like Outlook) might fallback to other Settings while it might work for other Clients.

The below setting allows you to adjust Zammad's email font setting. This setting does not require a service restart.

```ruby
>> Setting.set("html_email_css_font", "font-family:'Helvetica Neue', Helvetica, Arial,, Geneva, sans-serif; font-size: 12px;")
```

If you want to check the current setting, you can simply run the below code.

```ruby
>> Setting.get('html_email_css_font')
```

### 14.2.3 Working on user information

**Note:** Please note that this is not a full command list, if you're missing commands, feel free to ask over at the Community.

**Find user**

In order to work on user information or to check for specific information, you'll need to find it first.

```ruby
>> User.find(4) # We already know the ID of the user
>> User.find_by(email: 'your@email') # Searching for the user by his E-Mail-Address
>> User.find_by(login: 'john.doe') # Searching for the user by his login
```
Re-activate a locked user account

It sometimes happens that a user locks himself out by wildly trying the wrong password multiple times. Depending on your maximum failing login count (default: 10 times), Zammad might lock the account. The user can’t login any more (forever) if he doesn’t change the password or you reset the counter.

```
>> u = User.find(**USERID**)
>> u.login_failed = 0
>> u.save!
```

You can also double check if the account is locked by running the following (result needs to be 1 above your limit, so 11 for the default of 10 failing logins)

```
>> User.find(**USERID**).login_failed
```

Change / Update E-Mail-Adress of User

If needed, you can simply change the E-Mail-Address of the user.

**Note:** Please note that the login attribute is not affected by this and Zammad thus might show different information within the UI.

```
>> u = User.find(**USERID**)
>> u.email = 'user@example.com'
>> u.save!
```

You need to find the User-ID of the user first for this.

Change / Update Login name of User

Change the user name of the user (e.g. if you want to login with a shorter username instead of a mail address)

```
>> u = User.find(**USERID**)
>> u.login = 'user@example.com'
>> u.save!
```

You need to find the User-ID of the user first for this.

Set admin rights for user

Don’t have access to Zammad anymore? Grant yourself or another user administrative rights.

```
>> u = User.find_by(email: 'you@example.com')
>> u.roles = Role.where(name: ['Agent', 'Admin'])
>> u.save!
```
**Set password for user**

You or the user did forget his password? No problem! Simply reset it by hand if needed.

`User.find_by(email: 'you@example.com').update!(password: 'your_new_password')`

---

### 14.2.4 Working with ticket information

**Note:** Please note that this is not a full command list, if you’re missing commands, feel free to ask over at the Community.

**Get the RAW mail that Zammad fetched**

The following command will help you to check on received emls Zammad fetched. This comes in handy if you delete Mails upon fetching and you need to check the eml itself.

To get the first article’s eml, you can use the following command. In our example the ticket number in question is 101234

`Ticket.find_by(number: '101234').articles.first.as_raw.content`

If needed, you can also get the raw content of later articles (you’ll need to find the correct article though). Again, we expect 101234 to be our ticket number. In the first step we get all article IDs of the ticket, from the list we get, we can then get the articles content.

```ruby
>> Ticket.find_by(number: '101234').article_ids
=> [4, 3, 2]
>> Ticket::Article.find(3).as_raw.content
```

**Note:** If you just use `Ticket::Article.find(3)` you can see further information (like who sent the mail, when we fetched it, ...).

**Update all tickets of a specific customer**

**Warning:** Please note that this action can be expensive in resource terms, if you have many tickets, this might slow down Zammad.

```
>> Ticket.where(customer_id: 4).update_all(customer_id: 1)
```
### Change priority

The following commands will enable you to change the naming of priorities. If you set `.default_create` to `true` you can manipulate what Zammad will use as default priority.

```ruby
>> priority2 = Ticket::Priority.find(2)
>> priority2.name = '2-high'
>> priority2.default_create = true
>> priority2.save!
```

### Get ticket state types

This will show all Ticket States needed for creating new states.

**Note:** Missing States you just created? You might want to use `Ticket::State.all` to display all states for Tickets.

```ruby
>> Ticket::StateType.all
```

### Add new ticket state

**Note:** You can use `ignore_escalation: true`, to ignore possible SLA escalations (pending reminder and pending close use that by default).

#### Non-Pending states

A state that’s not a pending state (e.g. open, closed). Just replace 'open' by whatever you need (like closed).

```ruby
>> Ticket::State.create_or_update(
    name: 'Developing',
    state_type: Ticket::StateType.find_by(name: 'open'),
    created_by_id: 1,
    updated_by_id: 1,
  )
```

#### Pending reminders

A pending reminder state that will send a reminder notification to the agent if the time has been reached.

```ruby
>> Ticket::State.create_or_update(
    name: 'pending customer feedback',
    state_type: Ticket::StateType.find_by(name: 'pending reminder'),
    ignore_escalation: true,
    created_by_id: 1,
    updated_by_id: 1,
  )
```
Pending Action

A pending action that will change to another state if “pending till” has been reached.

```ruby
Ticket::State.create_or_update(
  name: 'pending and reopen',
  state_type: Ticket::StateType.find_by(name: 'pending action'),
  ignore_escalation: true,
  next_state: Ticket::State.find_by(name: 'open'),
  created_by_id: 1,
  updated_by_id: 1,
)
```

Add a date and time picker (pending till) for pending states

To add the time picker (pending till) to the new pending state, you’ll need to execute the following code:

```ruby
attribute = ObjectManager::Attribute.get(
  object: 'Ticket',
  name: 'pending_time',
)
attribute.data_option[:required_if][:state_id] = Ticket::State.by_category(:pending).pluck(:id)
attribute.data_option[:shown_if][:state_id] = Ticket::State.by_category(:pending).pluck(:id)
attribute.save!
```

Note: In enhanced cases you might want do define the state_id on your own. In this case just pick the returned state_id from Ticket::State.by_category(:pending).pluck(:id) and use them with attribute.data_option[:required_if][:state_id] = {state_id(s)} and attribute.data_option[:shown_if][:state_id] = {state_id(s)} directly. Don’t forget to save!

Make new states available to UI

Before being able to use the new states within the WebApp, you need to run the following commands to make them available.

WARNING: Please do not replace anything below, state_id is a named attribute which is correct and shall not be replaced!

```ruby
attribute = ObjectManager::Attribute.get(
  object: 'Ticket',
  name: 'state_id',
)
attribute.data_option[:filter] = Ticket::State.by_category(:viewable).pluck(:id)
attribute.screens[:create_middle]['ticket.agent'][:filter] = Ticket::State.by_category(:viewable_agent_new).pluck(:id)
```
Limit available states for customers

By default Zammad allows customers to change Ticket states to open and closed. If this does not meet your requirements, you can adjust this at anytime. The below example shows how to restrict your customer to only close tickets if needed:

```ruby
>> attribute = ObjectManager::Attribute.get(
    object: 'Ticket',
    name: 'state_id',
)
>> attribute.screens['edit']['ticket.customer']['filter'] = Ticket::State.where(name: ['closed']).pluck(:id)
>> attribute.save!
```

Hint: If you want to allow several different states for customers, you need to provide the state names as array - like so: ['closed', 'open', 'my-amazing-state'] (instead of ['closed']).

You can check the current active states that customers can set like so:

```ruby
>> ObjectManager::Attribute.get(
    object: 'Ticket',
    name: 'state_id',
).screens['edit']['ticket.customer']['filter']
```

The above will return one or more IDs - if you're not sure which state they belong to, you can check the state name with the following command. (Ensure to replace {ID} with your returned ID(s))

```ruby
>> Ticket::State.find({ID}).name
```

14.2.5 Working with ticket articles

Note: Please note that this is not a full command list, if you're missing commands, feel free to ask over at the Community.
Count Public “Notes” toward SLAs

Normally, notes don’t count toward service-level agreements. Use the following command to include publicly-visible notes when tracking SLA compliance. (Internal notes cannot be made to apply toward SLAs.)

Note: By default, customers are not notified when public notes are added to a ticket. Set up a trigger if you wish to change this behavior.

Warning: Changing this setting will disable the option to delete public notes.

```
>> Ticket::Article::Type.find_by(name: 'note').update!(communication: true)  # Enable SLA to count notes as communication
>> Ticket::Article::Type.find_by(name: 'note').update!(communication: false)  # Enable SLA to ignore notes as communication
```

14.2.6 Working with groups

Note: Please note that this is not a full command list, if you’re missing commands, feel free to ask over at our Community.

To open the rails console on the shell you have to enter the following commands.

Find group

```
>> Group.find_by(name: 'Users').follow_up_possible
```

14.2.7 Working with chat logs

Hint: To find out how to do something not listed below, post your question on the community boards.

Removing IP address logs

Use the following command to remove all IP address records from closed chats that haven’t been updated in the last seven days:

```
>> Chat::Session.where(state: 'closed').where('updated_at < ?', 7.days.ago).each do |session|
  next if session.preferences['remote_ip'].blank?
  session.preferences.delete('geo_ip')
  session.preferences.delete('remote_ip')
  session.save!(touch: false)
end
```
14.2.8 Other useful commands

Note: Please note that this is not a full command list, if you’re missing commands, feel free to ask over at the Community.

**Fetch mails**

The below command will do a manual fetch of mail channels. This will also show errors that might appear within that process.

```ruby
>> Channel.fetch
```

**Reprocess unprocessable mails**

When Zammad encounters a mail it cannot parse (e.g. due to a parser bug or a malformed message), it will store the mail in `tmp/unprocessable_mail/<ID>.eml`, give up on attempting to parse the mail, and will warn on the monitoring page that there are unprocessed mails.

To force Zammad to reattempt to parse those mails, run the following command:

```ruby
>> Channel::EmailParser.process_unprocessable_mails
```

In case of a malformed message (e.g. an invalid email address in one of the header fields), you may need to manually edit the mail before Zammad can process it.

If Zammad fails to process the message, it will remain in the `tmp/unprocessable_mail` folder; otherwise it will be removed after it has been parsed successfully.

**Add translation**

This comes in handy if you e.g. added a new state that you need to translate for several languages.

```ruby
>> Translation.create_if_not_exists( :locale => 'de-de', :source => "New", :target => "Neu", format: 'string', created_by_id: 1, updated_by_id: 1 )
```

**Translating attributes**

By default Zammad will not translate custom attributes. With the following code you can enable translation. This will translate the attribute display name and the display names of values (if it’s a value field). For this to work, just replace `{attribute-name}` against the name of your attribute.

```ruby
>> attribute = ObjectManager::Attribute.find_by(name: '{attribute-name}')
>> attribute.data_option[:translate] = true  # set this to false to disable translation
>> attribute.save!
```

Note: Translating value display names works for the following attribute types:

- Boolean
• Select
• Tree Select

If you’re translating the display name of e.g. an Integer-attribute, this works as well!

**Fill a test system with test data**

**Warning:** Don’t run this in a productive environment! This can slow down Zammad and is hard to revert if you create much!

The below command will add 50 agents, 1000 customers, 20 groups, 40 organizations, 5 new overviews and 100 tickets. You can always use 0 to not create specific items. Zammad will create random “fill data”.

```ruby
>> FillDB.load(agents: 50, customers: 1000, groups: 20, organizations: 40, overviews: 5, tickets: 100,
           )
```

**14.2.9 Deleting Records**

**Danger:** The commands listed here cause **irrecoverable data loss**! Only proceed if you know what you’re doing and you have a backup!

**Note:** The list of commands below is not exhaustive. If you can’t find what you’re looking for here, you are encouraged to ask the community.

**Deleting Tickets (and their articles)**

```ruby
# Delete a ticket (specified by database ID)
>> Ticket.find(4).destroy

# Delete all tickets
>> Ticket.destroy_all

# Keep some tickets (specified by database ID); delete the rest
>> tickets_to_keep = [1, 2, 3]
>> Ticket.where.not(id: tickets_to_keep).destroy_all
```
Deleting Customers

**Warning:** Customers **may not** be deleted while they have tickets remaining in the system.

As such, the examples below will delete not only the specified customers, but all tickets associated with them, as well.

**Step 1: Select customers by email address**

```ruby
>> customer_emails = %w[customer@example.com customer@example.org]

```

**Step 2: Preview affected users & tickets**

```ruby
>> puts customers.map { |user| <<~PREVIEW }.

  Customer #{user.fullname}/#{user.id}/#{user.email} has #{Ticket.where(customer_id: user.id).count} tickets #{Ticket.where(customer_id: user.id).pluck(:number)}

PREVIEW
```

**Step 3: Proceed with deletion**

```ruby
>> customers.find_each do |user|

  puts %{Preparing deletion of customer "#{user.fullname}" (and #{Ticket.where(customer_id: user.id).count} associated tickets)}

  Ticket.where(customer: user).find_each do |ticket|

    puts " Deleting ticket ##{ticket.number}..."
    ticket.destroy

  end

  puts " Removing references for user with email #{user.email}..."
  ActivityStream.where(created_by_id: user.id).update_all(created_by_id: 1)
  History.where(created_by_id: user.id).update_all(created_by_id: 1)
  Ticket::Article.where(created_by_id: user.id).update_all(created_by_id: 1)
  Ticket::Article.where(updated_by_id: user.id).update_all(updated_by_id: 1)
  Store.where(created_by_id: user.id).update_all(created_by_id: 1)
  StatsStore.where(created_by_id: user.id).update_all(created_by_id: 1)
  Tag.where(created_by_id: user.id).update_all(created_by_id: 1)
  OnlineNotification.find_by(user_id: user.id)&.destroy!

  puts " Deleting #{user.fullname}..."
  user.destroy

end
```
Deleting Organizations

Note: Deleting an organization does not delete associated customers.

Step 1: Select organizations

```ruby
# by "active" status
>> organizations = Organization.where(active: false)

# by name
>> organizations = Organization.where(name: 'Acme')

# by partial match on notes
>> organizations = Organization.where('note LIKE ?', '%foo%')
```

Step 2: Preview affected organizations

```ruby
>> puts organizations.map { |org| "ORGANIZATION #{org.name}" }.join("\n")
```

Step 3: Proceed with deletion

```ruby
>> organizations.each do |org|
  puts %{Preparing deletion of organization "#{org.name}"...}
  org.members.each do |member|
    puts " Removing #{member.fullname} from organization..."
    member.update!(organization_id: nil)
  end
  puts " Deleting #{org.name}..."
  org.destroy
end
```

Deleting System Records

```ruby
# Remove all online notifications
>> OnlineNotification.destroy_all

# Remove all entries from the Activity Stream (dashboard)
>> ActivityStream.destroy_all

# Remove entries for all recently viewed objects (tickets, users, organizations)
>> RecentView.destroy_all

# Remove all history information from tickets, users and organizations (dangerous!)
>> History.destroy_all
```
Zammad

Reset Zammad installation

**Hint:** Below commands are incomplete intentionally, error outputs will hint you through! The following operations will cause data loss and for development / testing only. Don’t forget to stop Zammad before trying to drop the database!

```
$ rake db:drop
$ rake db:create
$ rake db:migrate
$ rake db:seed
```
We would be glad if you contribute to Zammad. You can do this in several ways. Contributions are mainly done by forking one of our repos on GitHub and creating a pull request with your changes.

All repos can be found at https://github.com/zammad

15.1 Source Code

The Zammad source code can be found on GitHub at https://github.com/zammad/zammad

15.2 Documentation

Do you want to contribute to the Zammad documentation?

Open a new GitHub pull request at https://github.com/zammad/zammad-documentation with your changes.

The Zammad documentation is hosted on readthedocs.org. You can read it there at https://docs.zammad.org or browse the files via GitHub which also renders the used ReStructuredText markup.

15.2.1 ReStructuredText markup

If you like to edit the docs use the ReStructuredText markup language. Information about this language can be found at:

- http://docutils.sourceforge.net/docs/user/rst/quickref.html

Thanks!

Zammad Team
The Zammad main repo at https://github.com/zammad/zammad has several branches

16.1 Master

- Current unreleased development state of next stable minor release
- Bug fixes of current stable version are added here
- Is the branch where features work correctly
- Could be used for production environment by experienced users
- If current stable version is 1.1.0 this will become 1.1.1

16.2 Develop

- Default GitHub branch
- Current unreleased development state of next major release
- Is the first instance where all features are being developed
- This branch will have open issues
- If current stable version is 1.1.0 this will become 1.2.0
- Unstable!
- Should not be used in production environment!

16.3 Stable

- Current stable release
- Can be used for production
- Stable bugfixes will be merged from master when new stable minor version will be released
16.4 Stable-X.x

- There will be several more stable branches because we'll support the last three major versions of Zammad
- If current stable version is 1.2.0 then the name of the branch is stable-1.2 and there also would be stable-1.1 and stable-1.0
Zammad packages are built on packager.io.

You can find all Zammad packages at https://packager.io/gh/zammad/zammad

Builds of new packages are triggered with every push to our GitHub repo

If you fork the Zammad repo you can use packager.io to get builds for your fork

Just change the file “.pkgr.yml” to fit your needs
CONTINUOUS INTEGRATION

All pushes to our main repo at https://github.com/zammad/zammad will trigger build tests. We use internal build tests on internal and external continuous integration platforms.

18.1 Internal

• Done on our private continuous integration platform

18.2 External

18.2.1 Travis-CI

• You can find the build test results at https://travis-ci.org/zammad/zammad
• If you fork the Zammad repo you’re able to also use travis-ci.org to get your builds tested
• Just change the file “.travis.yml” to fit your needs
• Current build test status is:

18.3 Local

18.3.1 RSpec

To run tests locally in the test environment, you need to first ensure the test DB is in the expected state:

```
RAILS_ENV=test bundle exec rake db:drop db:create zammad:ci:test:prepare
```

For tests that use compiled front end assets, make sure the latest state of the code is considered:

```
RAILS_ENV=test bundle exec rake assets:precompile
```

Finally, running a single test can be done via the following command:

```
bundle exec rspec spec/system/ticket/zoom_spec.rb
```

Note that it’s also possible to run a specific test case by including the line number in the command:
Chapter 18. Continuous integration
• All pushes to our main repo at https://github.com/zammad/zammad will be checked by several libraries and services

19.1 Rubocop

• Code is being checked by Rubocop
• http://rubocop.readthedocs.io

19.2 Codeclimate

• Code is also being checked on https://codeclimate.com.
• You can find the results at https://codeclimate.com/github/zammad/zammad
• If you fork the Zammad repo you can use codeclimate.com to check your code
• Just change the file “.codeclimate.yml” to fit your needs
Docker is a container-based software framework for automating deployment of applications. Our Docker image is a single container based application designed to have Zammad up and running fast for testing purposes.

Please note that this is a non persistent storage container and all Zammad data is lost when you’re stopping the container.

If you like to run Docker in production environment try our Docker-compose version: Install with Docker Compose.

Your Docker environment needs to be up and running.

You can find the image at https://hub.docker.com/r/zammad/zammad/

You need at least 4 GB of RAM to run the container.

### 20.1 Run the Docker Container

Docker run will run a command in a new container, -i attaches stdin and stdout, -t allocates a tty.

#### 20.1.1 Set vm.max_map_count for Elasticsearch

```
$ sysctl -w vm.max_map_count=262144
```

**Tip:** For Mac OS: https://github.com/zammad/zammad-docker/issues/27#issuecomment-455171752

#### 20.1.2 Run docker container

```
$ docker container run -ti --rm --name zammad -p 80:80 zammad/zammad
```

That’s it! You’re now using a bash shell inside of a Zammad docker container using the develop branch of the GitHub repo.

To disconnect or detach from the shell without exiting, use the escape sequence Ctrl-p + Ctrl-q.
20.2 Go to http://localhost and you’ll see:

- “Welcome to Zammad!”, there you need to create your admin user.
Vagrant is a tool for building complete development environments. With an easy-to-use workflow and focus on automation, Vagrant lowers development environment setup time, increases development/production parity, and makes the “works on my machine” excuse a relic of the past.

Be aware that Vagrant is meant for developers and therefore uses our unstable packages from the “develop” branch on GitHub.

Let’s begin using Vagrant! First be sure that a Vagrant provider is installed. You can use “Virtual Box” from https://www.virtualbox.org.

21.1 Clone the Vagrant file

$ git clone git@github.com:zammad/zammad-vagrant.git
$ cd zammad-vagrant

21.2 Run Vagrant

21.2.1 For stable branch package

$ PACKAGER_REPO=stable vagrant up --provision

21.2.2 For develop branch package

$ vagrant up --provision

That’s it! You’re now running Zammad in a Vagrant environment.
21.3 Go to http://localhost:8080 and you’ll see:

- “Welcome to Zammad!”, there you need to create your admin user and invite other agents.

21.4 SSH into the machine

After “vagrant up”

$ vagrant ssh

After this you can switch to root user via:

$ sudo -i

21.5 Problems starting the VM?

If you get errors like:

Bringing machine 'default' up with 'virtualbox' provider...
=> default: Checking if box 'centos/7' is up to date...
=> default: VirtualBox VM is already running.

Use the following commands to fix it:

$ vboxmanage controlvm Zammad poweroff
Zammad is a web based open source helpdesk/ticket system with many features to manage customer communication via several channels like telephone, facebook, twitter, chat and e-mails.

This chapter describes the Zammad API v1.

### 22.1 The API

Zammad provides a REST/JSON API. Its endpoints are documented with the HTTP method for the request and a partial resource:

```
GET /api/v1/users
```

The full URL looks like:

```
https://your_zammad/api/v1/users
```

Curly braces {} indicate values you have to supply for the URL:

```
GET /api/v1/users/{id}
```

### 22.2 Authentication

Zammad supports three different authentication methods for API.

#### 22.2.1 HTTP Basic Authentication (username/password)

The username/password must be provided as HTTP header in the HTTP call. The Zammad admin can enable/disable the authentication method in the admin interface. Read more about HTTP basic authentication on Wikipedia.

```
$ curl -u {username}:{password} https://your_zammad/api/v1/users
```
22.2.2 HTTP Token Authentication (access token)

The access token must be provided as HTTP header in the HTTP call. Each user needs to create its own access token in the user preferences. The Zammad admin can enable/disable the authentication method in the admin interface.

```
$ curl -H "Authorization: Token token={your_token}" https://your_zammad/api/v1/users
```

22.2.3 OAuth2 (token access)

The Zammad API supports OAuth2 authorization. In order to create OAuth2 tokens for an external application, the Zammad user needs to create an application in the admin interface. The access token then has to be given within the HTTP header:

```
$ curl -H "Authorization: Bearer {your_token}" https://your_zammad/api/v1/users
```

22.3 Request Format

Zammad uses JSON for its API, so you need to set a “Content-Type: application/json” in each HTTP call. Otherwise the response will be text/html.

```
POST /api/v1/users/{id} HTTP/1.1
Content-Type: application/json

{
    "name":"some name",
    "organization_id": 123,
    "note":"some note"
}
```

22.4 Example CURL Requests

```
# Get information
$ curl -u test@zammad.com:test123 https://xxx.zammad.com/api/v1/tickets/3

# Put information
$ curl -u test@zammad.com:test123 -H "Content-Type: application/json" -X PUT -d '{ json: { "data": } }' https://xxx.zammad.com/api/v1/tickets/3

# Post information
$ curl -u test@zammad.com:test123 -H "Content-Type: application/json" -X POST -d '{ json: "data" }' https://xxx.zammad.com/api/v1/tickets/3
```
# 22.5 Example CURL Requests (for tickets and users)

## Create a new ticket

```
$ curl -u test@zammad.com:test123 -H "Content-Type: application/json" -X POST -d "{
    "title": "Help me!",
    "group": "Users",
    "article": {
        "subject": "some subject",
        "body": "some message",
        "type": "note",
        "internal": false,
        "customer": "email_of_existing_customer@example.com",
        "note": "some note"
    }
} https://xxx.zammad.com/api/v1/tickets
```

## Search for tickets (with contains "some message")

```
$ curl -u test@zammad.com:test123 'https://xxx.zammad.com/api/v1/tickets/search?
    query=some+message&limit=10&expand=true'
```

## Search for tickets (for tickets with state new and open)

```
$ curl -u test@zammad.com:test123 'https://xxx.zammad.com/api/v1/tickets/search?
    query=state: new%20OR%20state: open&limit=10&expand=true'
```

## Create a new user

```
curl -u test@zammad.com:test123 -H "Content-Type: application/json" -X POST -d '{
    "firstname": "Bob",
    "lastname": "Smith",
    "email": "email_of_customer@example.com",
    "roles": [
        "Customer"
    ],
    "password": "some_password"
} https://xxx.zammad.com/api/v1/users
```

## Create a new user (with welcome email)

```
$ curl -u test@zammad.com:test123 -H "Content-Type: application/json" -X POST -d '{
    "firstname": "Bob",
    "lastname": "Smith",
    "email": "email_of_customer@example.com",
    "roles": [
        "Customer"
    ],
    "password": "some_password",
    "invite": true
} https://xxx.zammad.com/api/v1/users
```

## Search for users

```
$ curl -u test@zammad.com:test123 'https://xxx.zammad.com/api/v1/users/search?
    query=smith&limit=10&expand=true'
```

### Hint: For more search examples regarding searching, please see this page.

---

## 22.6 Example CURL Request on behalf of a different user

It is possible to do a request on behalf of a different user. If you have your own application and you want to create a ticket for the customer without the information that the api user has created this ticket then you can transfer the target user with the request to create the ticket on behalf of the customer user:

```
$ curl -u test@zammad.com:test123 -H "Content-Type: application/json" -H "X-On-Behalf-Of: user-login" -X POST -d '{"title": "Help me!",
"group": "Users",
"article": {
    "subject": "some subject",
    "body": "some message",
    "type": "note",
    "internal": false,
    "customer": "email_ofExisting_customer@example.com",
    "note": "some note"
} https://xxx.zammad.com/api/v1/tickets
```

The value of the header has to contain one of the following values:

- user id
- user login
The value types will be checked in a cascade and the first detected user by id, login or email will be used for the request action.

This functionality can be used for any type of action.

Requirements for the feature:

- Authenticated user must have `admin.user` permissions
- Feature is available since Zammad version 2.4

### 22.7 Response Format

If a response is successful, an HTTP status code in the 200 or 300 range will be returned. If an item has been created or updated, all new attributes will be returned (also server side generated attributes like created_at and updated_at):

```
Status: 201 Created
Content-Type:application/json; charset=utf-8
{
    "id": 123,
    "name":"some name",
    "organization_id": 123,
    "note":"some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
}
```

### 22.8 Response Format (expanded)

If you want to retrieve expanded information for a request (e.g. the organization attribute), you just need to add an `expand=true` to the request URL:

```
GET /api/v1/users/{id}?expand=true HTTP/1.1
```

will return the following structure, expanded by “organization”:

```
Status: 200 Ok
Content-Type:application/json; charset=utf-8
{
    "id": 123,
    "name":"some name",
    "organization_id": 123,
    "organization": "Some Organization Name",
    "note":"some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
}
```
22.9 Pagination

All resources support pagination:

```
GET /api/v1/users?expand=true&page=1&per_page=5 HTTP/1.1
```

will return five records beginning with first record of all:

```
Status: 200 Ok
Content-Type: application/json; charset=utf-8

[
  {
    "id": 1,
    "name": "some name 1",
    "organization_id": 123,
    "organization": "Some Organization Name",
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
  },
  {
    "id": 2,
    "name": "some name 2",
    "organization_id": 345,
    "organization": "Some Other Organization Name",
    "note": "some note",
    "updated_at": "2016-08-17T07:55:42.221Z",
    "created_at": "2016-08-16T09:112:42.221Z"
  },
  ...
]
```

22.10 API clients

- Ruby Client - https://github.com/zammad/zammad-api-client-ruby
- PHP Client - https://github.com/zammad/zammad-api-client-php
- Python Client - https://pypi.org/project/zammad-py/
- .NET Client - https://github.com/Asesjix/Zammad-Client
- Android API-Client - https://github.com/KirkBushman/zammad-android

**Note:** Please note that this is a API client only, it’s no “ready to use” App.
23.1 me - current user

Required permission:

• any (only valid authentication)

Request:

GET /api/v1/users/me

Response:

Status: 200 Ok

{
    "id": 123,
    "firstname": "Bob",
    "lastname": "Smith",
    "email": "bob@smith.example.com",
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z",
    ...
}

23.2 List

Required permission:

• ticket.agent or admin.user (can read all users)

• any (can only read its own user if exists)

Request:

GET /api/v1/users

Response:
23.3 Search

Required permission:

- ticket.agent or admin.user (can read all users)

Request:

GET /api/v1/users/search?query=what&limit=10

Note: As of Zammad 2.6 parameters (sort_by=some_row and order_by=asc or desc) can also be used for sorting.

Response:

Status: 200 Ok

[  
  {  
    "id": 123,  
    "firstname": "Bob",  
    "lastname": "Smith",  
    "email": "bob@smith.example.com",  
    "note": "some note",  
    "updated_at": "2016-08-16T07:55:42.119Z",  
    "created_at": "2016-08-16T07:55:42.119Z",  
  },  
  {  
    "id": 124,  
    "firstname": "Martha",  
    "lastname": "Braun",  
    "email": "marta@braun.example.com",  
    "note": "some note",  
    "updated_at": "2016-08-16T07:55:42.119Z",  
    "created_at": "2016-08-16T07:55:42.119Z",  
  },  
]  

(continues on next page)
23.4 Show

Required permission:

- ticket.agent or admin.user (can read all users)
- customer with same organization (can read all users of same organization)
- any (can only read it’s own user if exists)

Request:

GET /api/v1/users/{id}

Response:

Status: 200 Ok

```json
{
    "id": 123,
    "firstname": "Bob",
    "lastname": "Smith",
    "email": "bob@smith.example.com",
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z",
    ...
}
```

23.5 Create

Required permission:

- admin.user
- ticket.agent (can not set roles/role_ids and not set groups/group_ids - roles.default_at_signup roles will get assigned automatically)
- any - until user_create_account is disabled (can not set roles/role_ids and not set groups/group_ids - roles.default_at_signup roles will get assigned automatically)
Request:

POST /api/v1/users

{
  "firstname": "Bob",
  "lastname": "Smith",
  "email": "bob@smith.example.com",
  "organization": "Some Organization Name",
  ...
}

Response:

Status: 201 Created

{
  "id": 123,
  "firstname": "Bob",
  "lastname": "Smith",
  "email": "bob@smith.example.com",
  "organization_id": 123,
  "organization": "Some Organization Name",
  "note": "some note",
  "updated_at": "2016-08-16T07:55:42.119Z",
  "created_at": "2016-08-16T07:55:42.119Z",
  ...
}

23.6 Update

Required permission:

- admin.user
- ticket.agent (can only update customer accounts and not set roles/role_ids and not set groups/group_ids - already assigned attributes will not changed)

Request:

PUT /api/v1/users/{id}

{
  "firstname": "Bob",
  "lastname": "Smith",
  "email": "bob@smith.example.com",
  "organization": "Some Other Organization Name",
  ...
}

Response:
23.7 Delete

Required permission:

- admin.user (only if no references in history tables and tickets exist)

Request:

```
DELETE /api/v1/users/{id}
```

Response:

```
Status: 200 Ok
{}
```


24.1 List

Required permission:

- ticket.agent or admin.organization (can read all organizations)
- any (can only read its own organization if exists)

Request:

GET /api/v1/organizations

Response:

Status: 200 Ok

[
  {
    "id": 123,
    "name": "Org 1",
    "shared": true,
    "active": true,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
  },
  {
    "id": 124,
    "name": "Org 2",
    "shared": false,
    "active": true,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
  }
]
24.2 Search

Required permission:

- ticket.agent or admin.organization (can read all organization)

Request:

```
GET /api/v1/organizations/search?query=what&limit=10
```

Note: As of Zammad 2.6 parameters (sort_by=some_row and order_by=asc or desc) can also be used for sorting.

Response:

```
Status: 200 Ok
[
  {
    "id": 123,
    "name": "Org 1",
    "shared": true,
    "active": true,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
  },
  {
    "id": 124,
    "name": "Org 2",
    "shared": false,
    "active": true,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
  }
]
```

24.3 Show

Required permission:

- ticket.agent or admin.organization (can read all organizations)
- any (can only read its own user if exists)

Request:

```
GET /api/v1/organizations/{id}
```

Response:

```
Status: 200 Ok
{
(continues on next page)
```
24.4 Create

Required permission:

- admin.organization

Request:

```json
POST /api/v1/organizations
{
"name": "Org 1",
"shared": true,
"active": true,
"note": "some note",
"updated_at": "2016-08-16T07:55:42.119Z",
"created_at": "2016-08-16T07:55:42.119Z"
}
```

Response:

```json
{
"id": 123,
"name": "Org 1",
"shared": true,
"active": true,
"note": "some note",
"updated_at": "2016-08-16T07:55:42.119Z",
"created_at": "2016-08-16T07:55:42.119Z"
}
```

24.5 Update

Required permission:

- admin.organization

Request:

```bash
PUT /api/v1/organizations/{id}
```
24.6 Delete

Required permission:
- admin.organization (only if no references in history tables and tickets exist)

Request:
```
DELETE /api/v1/organization/{id}
```

Response:
```
Status: 200 Ok
{}
```
25.1 List

Required permission:

- admin.group (can read all groups)

Request:

GET /api/v1/groups

Response:

Status: 200 Ok

[

{
   "id": 123,
   "name": "Group 1",
   "signature_id": 123,
   "email_address_id": 123,
   "assignment_timeout": 180,
   "follow_up_possible": "yes",
   "follow_up_assignment": true,
   "active": true,
   "note": "some note",
   "updated_at": "2016-08-16T07:55:42.119Z",
   "created_at": "2016-08-16T07:55:42.119Z"
},

{
   "id": 124,
   "name": "Group 2",
   "signature_id": 123,
   "email_address_id": 123,
   "assignment_timeout": 180,
   "follow_up_possible": "no",
   "follow_up_assignment": false,
   "active": true,
   "note": "some note",
   "updated_at": "2016-08-16T07:55:42.119Z",
   "created_at": "2016-08-16T07:55:42.119Z"
}]

(continues on next page)
25.2 Show

Required permission:

- admin.group (can read all groups)

Request:

```
GET /api/v1/groups/{id}
```

Response:

```
Status: 200 Ok
{
  "id": 123,
  "name": "Group 1",
  "signature_id": 123,
  "email_address_id": 123,
  "assignment_timeout": 180,
  "follow_up_possible": "yes",
  "follow_up_assignment": true,
  "active": true,
  "note": "some note",
  "updated_at": "2016-08-16T07:55:42.119Z",
  "created_at": "2016-08-16T07:55:42.119Z"
}
```

25.3 Create

Required permission:

- admin.group

Request:

```
POST /api/v1/groups
```

```
Response:

Status: 201 Created

{
  "id": 123,
  "name": "Group 1",
  "signature_id": 123,
  "email_address_id": 123,
  "assignment_timeout": 180,
  "follow_up_possible": "yes",
  "follow_up_assignment": true,
  "active": true,
  "note": "some note",
  "updated_at": "2016-08-16T07:55:42.119Z",
  "created_at": "2016-08-16T07:55:42.119Z"
}

25.4 Update

Required permission:
- admin.group

Request:

PUT /api/v1/groups/{id}

{
  "id": 123,
  "name": "Group 1",
  "signature_id": 123,
  "email_address_id": 123,
  "assignment_timeout": 180,
  "follow_up_possible": "yes",
  "follow_up_assignment": true,
  "active": true,
  "note": "some note"
}

Response:

Status: 200 Ok

{
  "id": 123,
  "name": "Group 1",
  "signature_id": 123,
  "email_address_id": 123,
  "assignment_timeout": 180,
  "follow_up_possible": "yes",
  "follow_up_assignment": true,
  "active": true,
  "note": "some note"
}
"email_address_id": 123,
"assignment_timeout": 180,
"follow_up_possible": "yes",
"follow_up_assignment": true,
"active": true,
"note": "some note",
"updated_at": "2016-08-16T07:55:42.119Z",
"created_at": "2016-08-16T07:55:42.119Z"
}

25.5 Delete

Required permission:

• admin.group (only if no references in history tables and tickets exist)

Request:

DELETE /api/v1/groups/{id}

Response:

Status: 200 Ok

{}`
26.1 List

Required permission:
- ticket.agent (access to all ticket in allocated groups)
- ticket.customer (access to all ticket with customer_id ** current_user.id || organization_id ** current_user.organization_id)

Request:

```
GET /api/v1/tickets
```

Response:

```
[{
    "id": 123,
    "title": "Help me!",
    "group_id": 1,
    "state_id": 1,
    "priority_id": 2,
    "customer_id": 2,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z",
    ...
}
]
```

```
[{
    "id": 124,
    "title": "Just want to ask for support",
    "state_id": 2,
    "priority_id": 2,
    "customer_id": 2,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z",
    ...
}
]```
26.2 Search

Required permission:

- `ticket.agent` (access to all ticket in allocated groups)
- `ticket.customer` (access to all ticket with `customer_id == current_user.id` or `organization_id == current_user.organization_id`)

Request:

```
GET /api/v1/tickets/search?query=what&limit=10
```

Note: As of Zammad 2.6 parameters (sort_by=some_row and order_by=asc or desc) can also be used for sorting.

Response:

```
Status: 200 Ok

[
  {
    "id": 123,
    "title": "Help me!",
    "group_id": 1,
    "state_id": 1,
    "priority_id": 2,
    "customer_id": 2,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z",
    ...
  },
  {
    "id": 124,
    "title": "Just want to ask for support",
    "state_id": 2,
    "priority_id": 2,
    "customer_id": 2,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z",
    ...
  },
  ...
]
```
26.3 Show

Required permission:

- ticket.agent (access to all ticket in allocated groups)
- ticket.customer (access to all ticket with customer_id == current_user.id || organization_id == current_user.organization_id)

Request:

```
GET /api/v1/tickets/{id}
```

Response:

```
Status: 200 Ok

{
  "id": 123,
  "title": "Help me!",
  "group_id": 1,
  "state_id": 1,
  "priority_id": 2,
  "customer_id": 2,
  "note": "some note",
  "updated_at": "2016-08-16T07:55:42.119Z",
  "created_at": "2016-08-16T07:55:42.119Z",
  ...
}
```

26.4 Create

Required permission:

- ticket.agent (create in all allocated groups)
- ticket.customer

Request:

```
POST /api/v1/tickets

{
  "title": "Help me!",
  "group": "Users",
  "customer": "email_of_existing_customer@example.com",
  "article": {
    "subject": "some subject",
    "body": "some message",
    "type": "note",
    "internal": false
  },
  "note": "some note",
}
```

(continues on next page)
Response:

**Status: 201 Created**

```json
{
  "id": 123,
  "title": "Help me!",
  "group_id": 1,
  "state_id": 1,
  "priority_id": 2,
  "customer_id": 2,
  ...
  "note": "some note",
  "updated_at": "2016-08-16T07:55:42.119Z",
  "created_at": "2016-08-16T07:55:42.119Z"
}
```

For more article attributes have a look into “Ticket Article”.

If you want to include attachments of the first article, the payload looks like:

**Request:**

```json
POST /api/v1/tickets

{
  "title": "Help me!",
  "group": "Users",
  "article": {
    "subject": "some subject",
    "body": "some message",
    "attachments": [
      {
        "filename": "some_file1.txt",
        "data": "content in base64",
        "mime-type": "text/plain"
      },
      {
        "filename": "some_file2.txt",
        "data": "content in base64",
        "mime-type": "text/plain"
      }
    ],
    "note": "some note",
    ...
  }
}
```

If you want to add inline images, just use data URIs in HTML markup:

**Request:**

```json
POST /api/v1/tickets

{
  "title": "Help me!",
  "group": "Users",
  "article": {
    "subject": "some subject",
    "body": "some message",
    "attachments": [
      {
        "filename": "some_file1.txt",
        "data": "content in base64",
        "mime-type": "text/plain"
      },
      {
        "filename": "some_file2.txt",
        "data": "content in base64",
        "mime-type": "text/plain"
      }
    ],
    "note": "some note",
    ...
  }
}
```
POST /api/v1/tickets
{
  "title": "Help me!",
  "group": "Users",
  "article": {
    "content_type": "text/html",
    "subject": "some subject",
    "body": "<b>some</b> message witn inline image <img src="data:image/jpeg;base64,ABCDEFGHIJKLMNOPQRSTUVWXYZ">",
  },
  "note": "some note",
  ...
}

If you want to use or create a custom user by email address at ticket creation, you can do with “guess:customer@example.com” in the customer_id attribute:

Request:
POST /api/v1/tickets
{
  "title": "Help me!",
  "group": "Users",
  "customer_id": "guess:customer@example.com",
  "note": "some note",
  ...
}

26.5 Update

Required permission:
- ticket.agent (access to all ticket in allocated groups)
- ticket.customer (access to all ticket with customer_id ** current_user.id || organization_id ** current_user.organization_id)

Request:
PUT /api/v1/tickets/{id}
{
  "id": 123,
  "title": "Help me!",
  "group": "Users",
  "state": "open",
  "priority": "3 high",
  "article": {
    "subject": "some subject of update",
    "body": "some message of update"
  },
  ...
}
(continues on next page)
Response:

```
{  
    "id": 123,  
    "title": "Help me!",  
    "group_id": 1,  
    "state_id": 1,  
    "priority_id": 2,  
    "note": "some note",  
    "updated_at": "2016-08-16T07:55:42.119Z",  
    "created_at": "2016-08-16T07:55:42.119Z"
}
```

If you want to include attachments of the article, the payload looks like:

Request:

```
PUT /api/v1/tickets/{id}

{  
    "id": 123,  
    "title": "Help me!",  
    "group": "Users",  
    "article": {  
        "subject": "some subject",  
        "body": "some message",  
        "attachments": [  
            {  
                "filename": "some_file1.txt",  
                "data": "content in base64",  
                "mime-type": "text/plain"
            },  
            {  
                "filename": "some_file2.txt",  
                "data": "content in base64",  
                "mime-type": "text/plain"
            }
        ]
    },  
    "note": "some note",  
    ...
}
```

If you want to add inline images, just use data URIs in HTML markup:

Request:

136 Chapter 26. Ticket
PUT /api/v1/tickets/{id}
{
    "id": 123,
    "title": "Help me!",
    "group": "Users",
    "article": {
        "content_type": "text/html",
        "subject": "some subject",
        "body": "<b>some</b> message with inline image <img src="data:image/jpeg;base64,ABCDEFGHIJKLMNOPQRSTUVWXYZ">"
    },
    "note": "some note",
    ...
}

26.6 Delete

Required permission:

• admin

Request:

DELETE /api/v1/tickets/{id}

Response:

Status: 200 Ok
{}

26.6. Delete
### 27.1 List

Required permission:

- `admin.object` (can read all ticket states)
- `ticket.agent` (can read all ticket states)
- `ticket.customer` (can read all ticket states)

Request:

```bash
GET /api/v1/ticket_states
```

Response:

```
Status: 200 Ok

[

{
    "id": 123,
    "name": "Ticket State 1",
    "state_type_id": 1,
    "next_state_id": null,
    "ignore_escalation": true,
    "active": true,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
}

{
    "id": 124,
    "name": "Ticket State 2",
    "state_type_id": 2,
    "next_state_id": 4,
    "ignore_escalation": false,
    "active": true,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
}
```

27.2 Show

Required permission:

- admin.object (can read all ticket states)
- ticket.agent (can read all ticket states)
- ticket.customer (can read all ticket states)

Request:

```plaintext
GET /api/v1/ticket_states/{id}
```

Response:

```json
Status: 200 Ok
{
  "id": 123,
  "name": "Ticket State 1",
  "state_type_id": 1,
  "next_state_id": null,
  "ignore_escalation": true,
  "active": true,
  "note": "some note",
  "updated_at": "2016-08-16T07:55:42.119Z",
  "created_at": "2016-08-16T07:55:42.119Z"
}
```

27.3 Create

Required permission:

- admin.object

Request:

```plaintext
POST /api/v1/ticket_states
{
  "name": "Ticket State 1",
  "state_type_id": 1,
  "next_state_id": null,
  "ignore_escalation": true,
  "active": true,
  "note": "some note"
}
```

Response:

```json
Status: 201 Created
{
(continues on next page)
```
27.4 Update

Required permission:

• admin.object

Request:

```
PUT /api/v1/ticket_states/{id}

{
  "id": 123,
  "name": "Ticket State 1",
  "state_type_id": 1,
  "next_state_id": null,
  "ignore_escalation": true,
  "active": true,
  "note": "some note",
  "updated_at": "2016-08-16T07:55:42.119Z",
  "created_at": "2016-08-16T07:55:42.119Z"
}
```

Response:

```
Status: 200 Ok

{
  "id": 123,
  "name": "Ticket State 1",
  "state_type_id": 1,
  "next_state_id": null,
  "ignore_escalation": true,
  "active": true,
  "note": "some note",
  "updated_at": "2016-08-16T07:55:42.119Z",
  "created_at": "2016-08-16T07:55:42.119Z"
}
```
27.5 Delete

Required permission:

- admin.object (only if no references in history tables and tickets exist)

Request:

```
DELETE /api/v1/ticket_states/{id}
```

Response:

```
Status: 200 Ok
{}
```
28.1 List

Required permission:

- admin.object (can read all ticket states)
- ticket.agent (can read all ticket states)
- ticket.customer (can read all ticket states)

Request:

```plaintext
GET /api/v1/ticket_priorities
```

Response:

```
Status: 200 Ok

[
  {
    "id": 123,
    "name": "Ticket Priority 1",
    "active": true,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
  },
  {
    "id": 124,
    "name": "Ticket Priority 2",
    "active": true,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
  }
]
```
28.2 Show

Required permission:
- admin.object (can read all ticket states)
- ticket.agent (can read all ticket states)
- ticket.customer (can read all ticket states)

Request:

```
GET /api/v1/ticket_priorities/{id}
```

Response:

```
Status: 200 Ok

{
    "id": 123,
    "name": "Ticket Priority 1",
    "active": true,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
}
```

28.3 Create

Required permission:
- admin.object

Request:

```
POST /api/v1/ticket_priorities
```

```json
{
    "name": "Ticket Priority 1",
    "active": true,
    "note": "some note"
}
```

Response:

```
Status: 201 Created

{
    "id": 123,
    "name": "Ticket Priority 1",
    "active": true,
    "note": "some note",
    "updated_at": "2016-08-16T07:55:42.119Z",
    "created_at": "2016-08-16T07:55:42.119Z"
}
```
## 28.4 Update

Required permission:

- admin.object

Request:

```
PUT /api/v1/ticket_priorities/{id}

{
  "id": 123,
  "name": "Ticket Priority 1",
  "active": true,
  "note": "some note"
}
```

Response:

```
Status: 200 Ok

{
  "id": 123,
  "name": "Ticket Priority 1",
  "active": true,
  "note": "some note",
  "updated_at": "2016-08-16T07:55:42.119Z",
  "created_at": "2016-08-16T07:55:42.119Z"
}
```

## 28.5 Delete

Required permission:

- admin.object (only if no references in history tables and tickets exist)

Request:

```
DELETE /api/v1/ticket_priorities/{id}
```

Response:

```
Status: 200 Ok

{}
```
29.1 By Ticket

Required permission:

- ticket.agent (access to related ticket)
- ticket.customer (access to related ticket with customer_id == current_user.id || organization_id == current_user.organization_id)

Request:

```plaintext
GET /api/v1/ticket_articles/by_ticket/{ticketId}
```

Response:

```plaintext
Status: 200 Ok
[
{
  "id": 3,
  "ticket_id": 3,
  "from": "Bob Smith",
  "to": "",
  "cc": "",
  "subject": "some subject",
  "body": "huhuhuu<br>huhuhuu<br>huhuhuu<br><br>"
  },
{
  "id": 4,
  "ticket_id": 3,
  "from": "Bob Smith",
  "to": "",
  "cc": "",
  "subject": "some subject",
  "body": "huhuhuu<br>huhuhuu<br>huhuhuu<br><br>"
}
```

(continues on next page)
29.2 Show

Required permission:

- ticket.agent (access to related ticket)
- ticket.customer (access to related ticket with customer_id ** current_user.id || organization_id ** current_user.organization_id)

Request:

```
GET /api/v1/ticket_articles/{id}
```

Response:

```
Status: 200 Ok

{
  "id": 3,
  "ticket_id": 3,
  "from": "Bob Smith",
  "to": "",
  "cc": "",
  "subject": "some subject",
  "body": "huhuhuu<br>huhuhuu<br>huhuhuu<br>huhuhuu<br><br>",
  "content_type": "text/html",
  "type": "note",
  "internal": false,
  "attachments": [
    {
      "id": 123,
      "filename": "some_file1.txt",
      "preferences": {
        "Mime-Type": "text/plain"
      }
    },
    {
      "id": 124,
      "filename": "some_file2.txt",
      "preferences": {
        "Mime-Type": "text/plain"
      }
    }
  ]
}
```
29.3 Create

Required permission:

- ticket.agent (access to related ticket)
- ticket.customer (access to related ticket with customer_id == current_user.id || organization_id == current_user.organization_id)

Request:

```json
POST /api/v1/ticket_articles
{
  "ticket_id": 3,
  "to": "",
  "cc": "",
  "subject": "some subject",
  "body": "huhuhuu<br>huhuhuu<br>huhuhuu<br><br>",
  "content_type": "text/html",
  "type": "note",
  "internal": false,
  "time_unit": "12"
}
```

Response:

```
Status: 201 Created
{
  "id": 3,
  "ticket_id": 3,
  "from": "Bob Smith",
  "to": "",
  "cc": "",
  "subject": "some subject",
  "body": "huhuhuu<br>huhuhuu<br>huhuhuu<br><br>",
  "content_type": "text/html",
  "type": "note",
  "internal": false,
  "time_unit": "12.0"
  "created_at": "2016-10-19T10:07:12.011Z",
  "updated_at": "2017-01-18T12:45:53.420Z",
  ...
}
```
If you want to include attachments of articles, the payload looks like:

Request:

```
POST /api/v1/ticket_articles
{
    "ticket_id": 3,
    "to": "",
    "cc": "",
    "subject": "some subject",
    "body": "huhuhuu<br>huhuhuu<br>huhuhuu<br><br>",
    "content_type": "text/html",
    "type": "note",
    "internal": false,
    "time_unit": "12",
    "attachments": [
        {
            "filename": "some_file1.txt",
            "data": "content in base64",
            "mime-type": "text/plain"
        },
        {
            "filename": "some_file2.txt",
            "data": "content in base64",
            "mime-type": "text/plain"
        }
    ]
}
```

Response:

```
Status: 201 Created
{
    "id": 3,
    "from": "Bob Smith",
    "to": "",
    "cc": "",
    "subject": "some subject",
    "body": "huhuhuu<br>huhuhuu<br>huhuhuu<br><br>",
    "content_type": "text/html",
    "type": "note",
    "internal": false,
    "time_unit": "12.0"
    "attachments": [
        {
            "id": 123,
            "filename": "some_file1.txt",
            "preferences": {
                "Mime-Type": "text/plain"
            }
        },
        {
        
        }
    ]
}
```
To download attachments you need to call “GET /api/v1/ticket_attachment/#{ticket_id}/#{article_id}/#{id}”.

If you want to add inline images, just use data URIs in HTML markup:

Request:

```json
POST /api/v1/ticket_articles
{
  "ticket_id": 3,
  "to": "",
  "cc": "",
  "subject": "some subject",
  "body": 
    "<b>some</b> message with inline image
    <img src="data:image/jpeg;base64,ABCDEFGHIJKLMNOPQRSTUVWXYZ">

  "content_type": "text/html",
  "type": "note",
  "internal": false,
  "time_unit": "12"
}
```

Response:

```
Status: 201 Created
{
  "id": 3,
  "ticket_id": 3,
  "from": "Bob Smith",
  "to": "",
  "cc": "",
  "subject": "some subject",
  "body": "huhuhuu<br>huhuhuu<br>huhuhuu<br>
  <img src="data:image/jpeg;base64,ABCDEFGHIJKLMNOPQRSTUVWXYZ">

  "content_type": "text/html",
  "type": "note",
  "internal": false,
  "time_unit": "12.0"
  "attachments": [
    {
      "id": 123,
      "filename": "44.262871107@zammad.example.com",
```

(continues on next page)
To download attachments you need to call “GET /api/v1/ticket_attachment/#{ticket_id}/#{article_id}/#{id}”.

If you want to create a phone ticket on behalf for a specific customer, use origin_by_id:

Required permission:
  • ticket.agent (access to related ticket)

Request:

```json
POST /api/v1/ticket_articles
{
  "ticket_id": 3,
  "origin_by_id": 5,
  "to": "",
  "cc": "",
  "subject": "some subject",
  "body": "<b>some</b> message with inline image <img src="data:image/jpeg;base64,ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  "content_type": "text/html",
  "sender": "Customer",
  "type": "phone",
  "internal": false,
  "time_unit": "12"
}
```
30.1 List

Required permission:

- authenticated user (content of notifications depends on user permissions)

Request:

GET /api/v1/online_notifications

Response:

Status: 200 Ok

[  
  {
   "id": 123,
   "o_id": 628,
   "object": "Ticket",
   "type": "escalation",
   "seen": true,
   "updated_at": "2016-08-16T07:55:42.119Z",
   "updated_by_id": 123,
   "created_at": "2016-08-16T07:55:42.119Z",
   "created_at_id": 123
  },
  {
   "id": 124,
   "o_id": 629,
   "object": "Ticket",
   "type": "update",
   "seen": false,
   "updated_at": "2016-08-16T07:55:47.119Z",
   "updated_by_id": 123,
   "created_at": "2016-08-16T07:55:42.119Z",
   "created_at_id": 123
  },
  {
   "id": 125,
   "o_id": 630,
   "object": "Ticket",
   (continues on next page)
30.2 Show

Required permission:
- authenticated user (content of notifications depends on user permissions)

Request:
GET /api/v1/online_notifications/{id}

Response:

Status: 200 Ok

{
  "id": 123,
  "o_id": 628,
  "object": "Ticket",
  "type": "escalation",
  "seen": true,
  "updated_at": "2016-08-16T07:55:42.119Z",
  "updated_by_id": 123,
  "created_at": "2016-08-16T07:55:42.119Z",
  "created_at_id": 123
}

30.3 Update

Required permission:
- admin.object

Request:
PUT /api/v1/online_notifications/{id}

{
  "seen": true,
}

Response:
Status: 200 Ok

```json
{
  "id": 123,
  "o_id": 628,
  "object": "Ticket",
  "type": "escalation",
  "seen": true,
  "updated_at": "2016-08-16T07:55:42.119Z",
  "updated_by_id": 123,
  "created_at": "2016-08-16T07:55:42.119Z",
  "created_at_id": 123
}
```

30.4 Delete

Required permission:

- authenticated user (content of notifications depends on user permissions)

Request:

```
DELETE /api/v1/online_notifications/{id}
```

Response:

```
Status: 200 Ok
{}
```

30.5 Mark all as read

Required permission:

- authenticated user (content of notifications depends on user permissions)

Request:

```
POST /api/v1/online_notifications/mark_all_as_read
```

Response:

```
Status: 200 Ok
{}
```
31.1 List

Required permission:

• admin (access to admin interface)

Request:

GET /api/v1/object_manager_attributes

Response:

Status: 200 Ok

[ 
  
  { 
    "id":49,
    "name":"anrede",
    "display":"Anrede",
    "data_type":"select",
    "data_option":{
      "options":{
        "Mr":"Mr",
        "Ms":"Ms",
        "Company":"Company"
      },
      "default":"Mr",
      "null":true,
      "maxlength":255,
      "nulloption":true
    },
    "data_option_new":{
    },
    "editable":true,
    "active":true,
    "screens":{
      "create":{
        "Customer":{
          "shown":true,
          "required":false
        }
      }
    }
  } 
]
31.2 Show

Required permission:

- admin (access to admin interface)

Request:

```
GET /api/v1/object_manager_attributes/:id
```

Response:

Status: 200 Ok

```
{
  "id":49,
  "name":"anrede",
  "display":"Anrede",
  "data_type":"select",
  "data_option":{
    "options":{}
  }
}
```
"Mr":"Mr",
"Ms":"Ms",
"Company":"Company"
},
"default":"Mr",
"null":true,
"maxlength":255,
"nulloption":true
},
"data_option_new":{

},
"editable":true,
"active":true,
"screens":{
"create":{
"Customer":{
"shown":true,
"required":true
}
},
"edit":{
"Customer":{
"shown":true
},
"Agent":{
"shown":true
}
},
"create_middle":{
"Agent":{
"shown":true
}
}
},
"to_create":false,
"to_migrate":false,
"to_delete":false,
"to_config":false,
"position":1550,
"created_by_id":3,
"updated_by_id":3,
"created_at":"2017-01-13T16:19:23.116Z",
"updated_at":"2017-01-17T11:16:13.298Z",
"object":"Ticket"}
31.3 Create

Required permission:

- admin (access to admin interface)

Request:

```
POST /api/v1/object_manager_attributes
```

Response:

```
Status: 200 Ok

{
   "name":"product",
   "object":"Ticket",
   "display":"Produkt",
   "active":true,
   "data_type":"select",
   "data_option":{
      "options":{
         "wert1":"anzeige1",
         "wert2":"anzeige12"
      }
   },
   "screens":{
      "create_middle":{
         "Customer":{
            "shown":true,
            "item_class":"column"
         },
         "Agent":{
            "shown":true,
            "item_class":"column"
         }
      },
      "edit":{
         "Customer":{
            "shown":true
         },
         "Agent":{
            "shown":true
         }
      }
   }
}
```
31.4 Update

Required permission:

- admin (access to admin interface)

Request:

```
PUT /api/v1/object_manager_attributes/:id
```

Response:

```
{   "id":49,   "name":"anrede",   "display":"Anrede",   "data_type":"select",   "data_option":{     "options":{       "Mr":"Mr",       "Ms":"Ms",       "Company":"Company"     },     "default":"Mr",     "null":true,     "maxlength":255,     "nulloption":true   },   "data_option_new":{   },   "editable":true,   "active":true,   "screens":{     "create":{       "Customer":{         "shown":true,         "required":true       }     },     "edit":{       "Customer":{         "shown":true       },       "Agent":{         "shown":true       }     },     "create_middle":{       "Agent":{         "shown":true       }     }   }
```

(continues on next page)
31.5 Execute Database Migrations

Required permission:

- admin (access to admin interface)

Request:

```
POST /api/v1/object_manager_attributes_execute_migrations
```

Response:

```
Status: 200 Ok
{}
```
Warning: API-Endpoints for Tags have changed with Zammad 3.5!

### 32.1 List

Required permission: `ticket.agent` or `admin.tag`

GET-Request sent: `/api/v1/tags?object=Ticket&o_id={ticket-id}`

Sample response:

```json
# HTTP-Code 200 OK
{
    "tags": [
        "americano",
        "complaint"
    ]
}
```

### 32.2 Search

Required permission: `ticket.agent` or `admin.tag`

GET-Request sent: `/api/v1/tag_search?term={tag-name}`

**Hint:** Zammad will return all tags that contain your search phrase.

Sample response:

```json
# HTTP-Code 200 OK
[
    {
        "id": 1,
        "value": "americano"
    },
    {
        "id": 2,
        "value": "complaint"
    }
]
```

(continues on next page)
32.3 Add

Required permission: ticket.agent or admin.tag

POST-Request sent: /api/v1/tags/add

```
{
   "item": "{tag-name}",
   "object": "Ticket",
   "o_id": {ticket-id}
}
```

**Hint:** This will create the tag if it doesn’t exist and the user has permission to do so.

Response:

```
# HTTP-Code 201 Created
true
```

32.4 Remove

Required permission:

- ticket.agent or admin.tag

DELETE-Request sent: /api/v1/tags/remove

```
{
   "item": "{tag-name}",
   "object": "Ticket",
   "o_id": "{ticket-id}"
}
```

Response:

```
# HTTP-Code 201 Created
true
```
### 32.5 Admin - List

Required permission: admin.tag

GET-Request sent: /api/v1/tag_list

Sample response:

```json
# HTTP-Code 200 OK
[
  {
    "id": 1,
    "name": "americano",
    "count": 0
  },
  {
    "id": 2,
    "name": "complaint",
    "count": 0
  },
  {
    "id": 3,
    "name": "viennese melange",
    "count": 0
  }
]
```

### 32.6 Admin - Create

Required permission: admin.tag

POST-Request sent: /api/v1/tag_list

```
{
  "name": "tag 5"
}
```

Response:

```
# HTTP-Code 200 OK
{}
```

### 32.7 Admin - Rename

Required permission: admin.tag

PUT-Request sent: /api/v1/tag_list/{tag-id}

```
{
  "name": "order"
}
```
Response:

```
# HTTP-Code 200 OK
{}
```

### 32.8 Admin - Delete

Required permission: admin.tag

DELETE-Request sent: /api/v1/tag_list/{tag-id}

Response:

```
# HTTP-Code 200 OK
{}
```
### 33.1 List

Required permission:

- `user_preferences.access_token`

Request:

```
GET /api/v1/user_access_token
```

Response:

```json
{
  "tokens": [
    {
      "id": 1,
      "label": "some user access token",
      "preferences": {
        "permission": ["cti.agent","ticket.agent"]
      },
      "last_used_at": null,
      "expires_at": null,
      "created_at": "2018-07-11T08:18:56.947Z"
    },
    {
      "id": 2,
      "label": "some user access token 2",
      "preferences": {
        "permission": ["ticket.agent"]
      },
      "last_used_at": null,
      "expires_at": null,
      "created_at": "2018-07-11T08:18:56.947Z"
    }
  ],
  "permissions": [
    {
      "id": 1,
      "name": "admin"
    }
  ]
}
```
33.2 Create

Required permission:

- user_preferences.access_token

Request:

```plaintext
POST /api/v1/user_access_token

{
  "label": "some test",
  "permission": ["cti.agent","ticket.agent"],
  "expires_at": null
}
```

Response:

```plaintext
Status: 200 Ok

{
  "name": "new_token_only_shown_once"
}
```

33.3 Delete

Required permission:

- user_preferences.access_token

Request:

```plaintext
DELETE /api/v1/user_access_token/{id}
```
Response:

```
Status: 200 Ok
{}
```
In many use cases, agents work in connection customer conversations over the phone.

It is a great relief when the telephone system (PBX) is integrated with Zammad, which makes processes with agents more effective.

The goal of the document is to provide the necessary API documentation to enable PBX vendors to easily integrate with Zammad.

### 34.1 Feature list

**Inbound**
- Caller identification based on the CallerID (open a customer profile with just one click)
- Display of open and closed tickets of a customer in a special overview. This overview should also give the possibility to create a ticket for the given customer.
- Intelligent mapping of CallerIDs with direct (e.g. directly at the contact) and not direct (e.g. telephone numbers from the signature)
- Caller Journal (which calls have been made and which have been handled and which require a callback)
- Blocking of CallerIDs (already during the call)
- Support to allow an agent to set a DND - like state
- Overview of agents who currently handle a call

**Outbound**
- Direct dialling of the customer telephone number and indexing of the call
- Set the outbound caller ID based on the line phone number (e.g. set sender caller id based on country of destination caller id)
- if supported by the PBX/telephone system
35.1 How it works

Events can be transferred in realtime from the telephone system to the Zammad CTI Push API (REST API) via a generic interface.

Depending on the event, Zammad offers various functions to quickly and easily identify callers and the corresponding tickets, for example, or to provide a caller log. Or to modify the incoming or outgoing call.

35.2 Endpoint

The endpoint of your Zammad CTI Push API looks like http://localhost:3000/api/v1/cti/:token and can be found in Zammad -> Admin -> Integrations -> CTI (generic) -> Endpoint

35.3 Events

Zammad supports the following three events (newCall, hangup and answer) in version 2.x.

**Event: newCall**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>“newCall”</td>
</tr>
<tr>
<td>from</td>
<td>The calling number (e.g. “493055571600” or “anonymous”)</td>
</tr>
<tr>
<td>to</td>
<td>The called number (e.g. “491711234567890”)</td>
</tr>
<tr>
<td>direction</td>
<td>The direction of the call (either “in” or “out”)</td>
</tr>
<tr>
<td>callId</td>
<td>A unique alphanumeric identifier to match events to specific calls (max. 250 characters)</td>
</tr>
<tr>
<td>user[]</td>
<td>The user(s) realname involved. It is the name of the calling user when direction is “out”, or of the users receiving the call when direction is “in”. Group calls may be received by multiple users. In that case a “user[]” parameter is set for each of these users. It is always “user[]” (not “user”), even if only one user is involved.</td>
</tr>
<tr>
<td>queue</td>
<td>The queue name (e.g. helpdesk). This field is optional.</td>
</tr>
</tbody>
</table>

You can simulate this POST request and test your server with a CURL command:
$ curl -X POST --data "event=newCall&from=493055571600&to=491711234567890&direction=in&callId=123456&user[]=Alice&user[]=Bob" http://localhost:3000/api/v1/cti/:token

The response (optional)

After sending the POST request to Zammad, your PBX can accept an JSON response to determine what to do (e. g. for `direction=in` to block the caller or for `direction=out` to set a caller id).

Zammad currently supports the following responses for incoming calls:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reject</td>
<td>Reject call or pretend to be busy (depending on your settings in Zammad)</td>
</tr>
</tbody>
</table>

Example 1: Reject call signaling busy

```
{
    "action": "reject",
    "reason": "busy"
}
```

Zammad currently supports the following responses for outgoing calls:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dial</td>
<td>To set the caller id (depending on your settings in Zammad). Number need to be in E.164 format.</td>
</tr>
</tbody>
</table>

Example 1: Set custom caller id for outgoing call

```
{
    "action": "dial",
    "callerId": "493055571642",
    "number": "491711234567890"
}
```

Event: hangup

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>“hangup”</td>
</tr>
<tr>
<td>callId</td>
<td>Same as in newCall-event for a specific call</td>
</tr>
<tr>
<td>cause</td>
<td>The cause for the hangup event (see</td>
</tr>
<tr>
<td>from</td>
<td>The calling number (e.g. “493055571600” or “anonymous”)</td>
</tr>
<tr>
<td>to</td>
<td>The called number (e.g. “491711234567890”)</td>
</tr>
<tr>
<td>direction</td>
<td>The direction of the call (either “in” or “out”)</td>
</tr>
<tr>
<td>answeringNumber</td>
<td>The number which was answering</td>
</tr>
</tbody>
</table>

You can simulate this POST request and test your server with a CURL command:

$ curl -X POST --data "event=hangup&cause=normalClearing&callId=123456&from=493055571600&to=491711234567890&direction=in&answeringNumber=4921199999999" http://localhost:3000/api/v1/cti/:token

Hangup causes: For these reasons, hangups may occur because of these causes:
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>normalClearing</td>
<td>One of the parties hung up after the call was established.</td>
</tr>
<tr>
<td>busy</td>
<td>The called party was busy</td>
</tr>
<tr>
<td>cancel</td>
<td>The caller hung up before the called party picked up</td>
</tr>
<tr>
<td>noAnswer</td>
<td>The called party rejected the call (e.g. through a DND setting)</td>
</tr>
<tr>
<td>congestion</td>
<td>The called party could not be reached</td>
</tr>
<tr>
<td>notFound</td>
<td>The called number does not exist or called party is offline</td>
</tr>
<tr>
<td>forwarded</td>
<td>The call was forwarded to a different party</td>
</tr>
</tbody>
</table>

**Event: answer**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>“answer”</td>
</tr>
<tr>
<td>callId</td>
<td>Same as in newCall-event for a specific call</td>
</tr>
<tr>
<td>user</td>
<td>Name of the user who answered this call. Only incoming calls can have this parameter</td>
</tr>
<tr>
<td>from</td>
<td>The calling number (e.g. “492111234567” or “anonymous”)</td>
</tr>
<tr>
<td>to</td>
<td>The called number (e.g. “491711234567890”)</td>
</tr>
<tr>
<td>direction</td>
<td>The direction of the call (either “in” or “out”)</td>
</tr>
<tr>
<td>answeringNumber</td>
<td>The number of the answering destination. Useful when redirecting to multiple destinations</td>
</tr>
</tbody>
</table>

You can simulate this POST request and test your server with a CURL command:

```
$ curl -X POST --data "event=answer&callId=123456&user=John+Doe&from=493055571600&
   to=491711234567890&direction=in&answeringNumber=21199999999" http://localhost:3000/api/
   v1/cti/:token
```
Zammad contains simple backup & restore scripts that can be executed via command line or cron job. You can find the scripts in the /opt/zammad/contrib/backup directory.

**Warning:** You’ll need to rename the config file for the backup before you can use this script!

### 36.1 Configuration

- Rename /opt/zammad/contrib/backup/config.dist to /opt/zammad/contrib/backup/config
- Configure backup path in /opt/zammad/contrib/backup/config if you want. The default backup path is /var/tmp/zammad_backup (needs to be created!)
- If needed, you can also adjust the variable HOLD_DAYS to any value you need. Default value here is 10 backups before the oldest backup is deleted.

**Note:** Please note that the Backup script always creates a Full-Dump of /opt/zammad and a Full-Dump of your database. If your Zammad installation is rather big, you might need to ensure you have enough space.

### 36.2 Create Backup

Creating a Backup is done very easy, you can just call the following to backup your Zammad-Instance. You can also run this as a cronjob to have a regular backup:

```
$ cd /opt/zammad/contrib/backup
$ ./zammad_backup.sh
```

**Note:** Please note that you should run the cronjob as User zammad (ensure this user can write to the backup-directory). If you’re using the root user, you might want to consider the following issues “Permission issue for Backup” and “Backup script asks for password”.

**Warning:** If you plan on migrating your Zammad-Installation to another system, ensure to stop Zammad before creating a Backup. Other wise, data might change! You can do this with:

```
$ systemctl disable zammad && systemctl stop zammad
```
36.3 Migrating from another Zammad-Host

Migration between different Zammad installations is very easy. Before you migrate, please ensure the following requirements are met:

- The Zammad-Version on the destination system has to be the same or newer
- You can’t mix database types (postgresql or MySQL), as this needs conversion of your dump (which the script does not perform)
  - We can offer you Dump-Migrations from MySQL to postgresql and postgresql to MySQL if need to change the database for whatever reason, as a commercial service.
- Ensure you have enough free space on your drive (at least double as the size of your Dump!)
- If not source code installation: You need a fresh Zammad installation

If above requirements are met, you can continue with restoring.

Hint: Source code installations have to supply the old ruby version that has been used earlier. This is because the restore script partly uses Zammad commands which will fail if you don’t!

You can find a version list on the Software page.

36.4 Restore everything

# Change into the folder of Zammad's backup-script:
$ cd /opt/zammad/contrib/backup

36.4.1 With menu for choosing backup date

# When called without arguments, Zammad will show you a list of available backups.
$ ./zammad_restore.sh

36.4.2 With command line argument for backup date

Warning: Only use the following option if you know what you’re doing! The following command will overwrite existing data without further prompts!

# When called with a timestamp argument (matching the backup's filename),
# Zammad will proceed immediately to restoring the specified backup.
$ ./zammad_restore.sh 20170507121848
36.5 What to do after restoration has been completed

36.5.1 When migrated from a self hosted Zammad system

**Note:** This step is only needed, if one of the following points is met:

- The source and destination Zammad-Version are not the same
- The Zammad-installation is not a source code installation
- The Zammad-Backup is not an Export from Hosted-Setup

If no points affect you, just continue with *the things you need to do after migration on every system*.

If your versions differ, it might happen, that your Zammad-Service will not start cleanly. You can update your installation.

If you receive the following, you can workaround your problem with reinstalling Zammad (example on Debian, other Operating systems might differ)

```bash
root@zammad:~ # apt-get update && apt install zammad
Reading package lists... Done
Building dependency tree
Reading state information... Done
zammad is already the newest version (x.x.x-xxxxxx.xxxxxx.stretch).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

The following will uninstall and install Zammad without resolving dependencies:

**Debian, Ubuntu**

```
$ dpkg -r --force-depends zammad
$ apt install zammad
```

**openSuSe**

```
$ zypper remove -R zammad
$ zypper install zammad
```

36.5.2 Things you need to do after migration on every system

**Note:** This does not apply to Docker images, as the following settings should be applied upon every start.

**Warning:** For Zammad-Versions 2.9 and earlier, please run a change owner on your Zammad folder. Default-Installations should be fine with `chown -R zammad:zammad /opt/zammad/` (Source code installations might differ). Please restart Zammad after the change-owner command `systemctl restart zammad`.

Before you can use Zammad and all it's features, you'll need to ensure your Searchindex is up and running. If you didn't install elasticsearch yet, now’s a good time. If you already did, ensure to configure the ES-URL (if migrated) and also run a reindex.
You can find further information on how to do that on the following page: *Set up Elasticsearch.*
CONFIGURATION VIA ENVIRONMENT VARIABLES

Use these environment variables to configure Zammad’s behavior at runtime.

Note: What’s an environment variable, and how do I “use” it?

Unfortunately, that question has a very long answer that goes beyond the scope of this article. How you set environment variables will depend on how you installed Zammad (e.g., source, package, or Docker).

But for package installations, here’s a short answer:

```
# set OPTION to "value"
$ zammad config:set OPTION=value
$ systemctl restart zammad

# unset OPTION
$ zammad config:unset OPTION
$ systemctl restart zammad
```

To learn more, do some googling on environment variables and the shell environment (or execution environment) in Unix.

37.1 General Options

**APP_RESTART_CMD** The command Zammad will use to automatically restart the server after changes have been made in the Object Manager. (E.g., "systemctl restart zammad")

If this is undefined, you will have to restart manually after making changes in the Object Manager.

Default: unset

**RAILS_LOG_TO_STDOUT** Print output directly to standard output instead of /var/log/zammad/production.log.

**Warning:** On package installations, this setting can be overwritten during update.

Use enabled to turn this option on only until the next update. Use true to turn it on permanently.

Default: unset
37.2 Network Options

**ZAMMAD_BIND_IP** The IP address that the web server is bound to.

Default: 0.0.0.0

**ZAMMAD_RAILS_PORT** The port that the web server is exposed on.

Default: 3000

**ZAMMAD_WEBSOCKET_PORT** The port that the websocket server is exposed on.

Default: 6042

**Note:** Remember to update your webserver config to reflect any changes you make here.

37.3 Performance Tuning

**Warning:** Each of these settings comes with its own tradeoffs.

There are no “recommended values” here; the optimal configuration will depend on your system’s resources and typical application load.

Proceed with caution; when adjusting any of these settings, there is a point at which performance will begin to degrade rather than improve, or other problems will begin to crop up.

**WEB_CONCURRENCY** How many instances of the application server to keep open at a time.

Increasing this can reduce loading times when too many users are on Zammad at once.

Default: unset

**ZAMMAD_SESSION_JOBS_CONCURRENT** How many instances of the session worker to run at a time.

Increasing this can speed up background jobs (like the scheduler) when too many users are on Zammad at once. Generally speaking, it should only be useful to adjust this setting if you have more than 40 active users at a time.

**Warning:** Session workers can be extremely CPU-intensive.

In some cases, they can reach 100% CPU utilization on their own. Increasing this setting is safer on systems with more cores.

Default: unset

**Tip:** How can I find out how many users are currently on Zammad?

```
$ zammad run rails r "p Sessions.list.uniq.count"
```
Hint: Above settings *may* consume all available database connections. Please consider the *database server configuration* section for more.
Note: Parts of this page also applies to both supported database servers. We can't provide a complete how to and will only enlighten the relevant parts for Zammad.

Within `database.yml` (config/ directory) you can define the allowed pool size. By default each Zammad process takes up to 50 connections (pool: 50). This should be fairly enough for every use case. If you experience database connection timeouts or similar pool errors, this usually indicates to other issues that are relevant to your PostgreSQL.

Note: Below only affects PostgreSQL-Servers. All relevant steps for MySQL are mentioned on Software because they're relevant before installation.

Below you can the locations of the relevant PostgreSQL configuration files to adjust. Keep in mind that versions may differ from your setup - adapt where needed.

Ubuntu / Debian
CentOS / openSuSE
other

```
/etc/postgresql/(10|11|12)/main/postgresql.conf
/var/lib/pgsql/data/postgresql.conf
```

Can't find your configuration files? You can run the following command to get the path:

```
$ sudo -u postgres psql -c 'SHOW config_file'
```

Adjust **max_connections** (mandatory) Zammad will take up to 200 connections by default, with below command you can raise this limit fairly high.

```
# Raise maximum allowed number of connections
$ sed -i "/max_connections/c\max_connections = 2000" <postgresql-configuration-file>

# Apply changes by restarting postgresql and Zammad (in this order)
$ systemctl restart postgresql zammad
```

Adjust PostgreSQL for bigger instances (optional)
Warning: Check below settings first and ensure your system is able to provide the requirements! Below settings are what we found to be useful, everything else is out of scope of this documentation!

# Caching improvements
$ sed -i "/shared_buffers/c\shared_buffers = 2GB" <postgresql-configuration-file>
$ sed -i "/temp_buffers/c\temp_buffers = 256MB" <postgresql-configuration-file>
$ sed -i "/work_mem/c\work_mem = 10MB" <postgresql-configuration-file>
$ sed -i "/max_stack_depth/c\max_stack_depth = 5MB" <postgresql-configuration-file>

# Apply changes by restarting postgresql and Zammad (in this order)
$ systemctl restart postgresql zammad
How long does Zammad hold onto user data? How can I manage its user data retention behavior?

39.1 On-Premises Data

The following kinds of data are stored locally on the production system:

**Tickets and users**  By default, Zammad never automatically deletes tickets or users.

To enable automatic deletion of tickets after a given interval, use the scheduler. To manually delete users and all their associated tickets (e.g., in compliance with a “Right to Forget” request under the GDPR), use the console.

**Note:** The ability to delete users via the admin panel is planned for a future release of Zammad.

**Chat sessions**  Once a chat session has been marked closed, it is scheduled for automatic deletion 3 months later.

IP address logs for chat sessions can be manually deleted by following the directions here.

**CTI caller log**  The caller log shows only the 60 most recent entries. Each entry in the caller log is automatically deleted after 12 months.

**Log files**  Zammad writes log files to disk (typically under /opt/zammad/log/).

Package installations will set up a separate system utility called logrotate to rename and archive (or rotate) log files on a nightly basis and remove old logs after 14 days.

If installing from source, it is strongly recommended to configure logrotate or a similar log management utility; Zammad will not purge old logs on its own.

**User sessions**  Zammad maintains session information about every user currently logged in.

This information is automatically purged when a user logs out, and can be viewed or manually deleted via the admin panel (under System → Sessions). Users may also delete their own session information via the user preferences menu, under Device.

Session information includes IP address (and possibly geographic location), browser, time of original login, and time of last visit.
39.2 External Services

Zammad utilizes third-party web services for certain functions, meaning that user data may occasionally be sent or exposed to third parties. These functions can be individually disabled in the admin panel under Settings → System → Services.

Note: By default, the third-party services that Zammad relies on are mostly ones hosted and managed by the Zammad Foundation itself, but Zammad can be extended to interface with other services instead.

The source code for these third-party service integrations can be found here.

Images  No private images or personally-identifying information are stored on images.zammad.com.

The Images service caches publicly-available images from sources like Gravatar and serves them to the Zammad application as user avatars and organization logos. These images are discovered using MD5 digests of user email addresses and organization domain names. User avatars are cached for 7 days; organization logos are cached for 30 days.

GeoCalendar  No user information is stored or cached on geo.zammad.com.

As part of its service-level agreement (SLA) functionality, Zammad requires detailed, localized calendar information (e.g., to set the time zone and accommodate national holidays and daylight savings time). The GeoCalendar service is used to retrieve this information.

GeoIP  No user information is stored or cached on geo.zammad.com.

One of Zammad’s security features is to track user sessions based on the user’s browser and country of origin. Suspicious login activity from a different browser or country may trigger Zammad to dispatch an alert email to the affected user. The GeoIP service is used to associate IP addresses to a geographic origin.

Geolocation  Since Zammad’s geolocation service relies on Google’s Geocoding API, its use is subject to the Google Privacy Policy.

Zammad uses geolocation to associate tickets with locations to support map-style ticket overviews, which display tickets as points on a map rather than items in a list.
This guide will discuss how to set up single sign-on using Microsoft Active Directory.

**Note:** SSO can only be configured on self-hosted installations.

Fig. 1: As of Zammad 3.5, enabling SSO adds a new button to the sign-in page.

### 40.1 Conceptual Overview

Like every other web application out there, Zammad has its own logic for signing users up, storing their passwords, authenticating them, and managing their sessions.

If your IT department keeps its own user identity store (like Active Directory), Zammad’s SSO support allows you to leverage that existing auth system so that anyone with an account on your local intranet will 1) automatically have an account in Zammad and 2) be able to log in with a single click.

**Note:** If you don’t have this IT infrastructure but still want one-click login, see Third-Party Authentication for alternatives.

### 40.1.1 How does it work?

Once enabled, single sign-on activates an endpoint at `https://your.zammad.host/auth/sso`. When the Zammad server receives a GET request at this endpoint with a valid username in **any one of the following**:

- an X-Forwarded-User request header
- a `REMOTE_USER` web server environment variable
- a `HTTP_REMOTE_USER` web server environment variable

it creates a new session for that user.

**Note:** Wait. SSO allows you to sign in with only a username?

In principle, yes.

**How is that okay?**
In this guide, we configure our web server (Apache) to intercept all requests to the /auth/sso endpoint. Instead of forwarding them to Zammad, Apache initiates a three-sided login process (Kerberos authentication) between the itself, the user, and the Active Directory server.

If Active Directory doesn’t recognize the user or their password, Zammad never sees the request, and the session is never created.

**What does this all mean?**

It means there are many ways you could set up SSO—you don’t need to follow this guide or even use Active Directory or Kerberos—but if you don’t know what you’re doing, you’re going to end up with a **massive** security hole.

### 40.2 Getting Started

**Hint:** Too busy to handle it on your own?

We’ve got you covered. Our experts offer custom-tailored workshops to get your team up and running fast and with confidence. Just drop us a line!

You will need:

- a Microsoft Active Directory environment with
  - root access
  - support for AES 256-bit encryption
- a Zammad host with
  - root access
  - a fully-qualified domain name (FQDN)
- some familiarity with system administration (e.g., Apache configuration)

For best results, set up **LDAP integration** to make sure your Active Directory and Zammad user accounts are always in sync.

### 40.3 Step 1: Configure Active Directory

In the Kerberos authentication scheme, the **authentication server** (Active Directory) needs to maintain shared secrets with the **service** (Zammad). To make this possible, we need to register a **service principal name** (SPN) for Zammad on Active Directory.

**Note:** These directions have been confirmed on Windows Server 2016.
40.3.1 1a. Create a service account

You may use an existing service account if you have one. Admin privileges are not required; a normal user account will do.

![Zammad Service Properties](image)

Fig. 2: Select “This account supports Kerberos AES 256 bit encryption” under Properties > Account > Account options.
40.3.2 1b. Register an SPN for Zammad

Note: Replace the following placeholders in the command below:

- `<zammad-host>` Zammad FQDN
- `<service-acct>` Service account logon name
- `<service-acct-pwd>` Service account password
- `<domain>` Windows domain
- `<master-domain-controller>` Master domain controller IP/FQDN

```bash
$ setspn -s HTTP/<zammad-host> <service-acct>
$ ktpass /princ <service-acct>@<domain> \
   /mapuser <service-acct> \
   /crypto AES256-SHA1 \
   /ptype KRB5_NT_PRINCIPAL \
   /pass <service-acct-pwd> -SetPass +DumpSalt \
   /target <master-domain-controller> \
   /out zammad.keytab
```

40.3.3 1c. Note the secret key and version number

The output of the command above contains important data for Step 2e below:

```bash
Using legacy password setting method
Failed to set property 'servicePrincipalName' to 'HTTP/<zammad-host>' on Dn 'CN=Zammad,…
...Service,DC=<domain>,DC=<tld>': 0x13.
WARNING: Unable to set SPN mapping data.
If <service-acct> already has an SPN mapping installed for HTTP/<zammad-host>, this is…
...no cause for concern.
Building salt with principalname HTTP/<zammad-host> and domain <domain> (encryption type…
...18)...
Hashing password with salt "<domain><service-acct>".
Key created.
Output keytab to zammad.keytab:
Keytab version: 0x502
keysize 67 <service-acct>@<domain> ptype 1 (KR5_NT_PRINCIPAL) vno 3 etype 0x12 (AES256-…
...SHA1) keylen 32 (0x5ee827c30c736dd4095c9cbe146ebc21664151d6aabd61be8f7fb1)
```

On the last line, take note of:

- **the secret key** in parentheses at the end (0x5ee827...)
- **the secret key version number** preceded by vno (3)
40.4  Step 2: Remove NGINX, Set up Apache + Kerberos

Next, the Zammad host must be configured to support Kerberos (and to accept auth credentials provided by the Active Directory server).

In most cases, you would have to recompile NGINX from source with an extra module to enable Kerberos support. To get around this, we will use Apache, which offers Kerberos support through a plug-in module instead.

**Note:** All commands in this section must be run as root (or with sudo).

### 40.4.1 2a. Turn off NGINX

```
$ systemctl stop nginx  # turn off nginx
$ systemctl disable nginx  # keep it off after reboot
```

**Warning:** This will take your Zammad instance offline until Apache is fully configured and running.

If you wish to minimize downtime, you can save this step for last; just bear in mind that Apache will not start if the port it wants to listen on is being used by NGINX.

If for any reason you can’t complete this tutorial, simply turn off Apache and restore NGINX:

```
$ systemctl stop apache2
$ systemctl disable apache2
$ systemctl enable nginx
$ systemctl start nginx
```

### 40.4.2 2b. Pre-Configure Apache

This documentation expects a already working Apache configuration. Please see [Configure the webserver](#) before continuing.

### 40.4.3 2c. Install further Apache dependencies

Ubuntu / Debian
CentOS
OpenSUSE

```
$ apt update
$ apt install krb5-user libapache2-mod-auth-kerb

$ yum install krb5-workstation mod_auth_kerb

$ zypper ref
$ zypper install krb5-client apache2-mod_auth_kerb
```
40.4.4 2d. Enable Apache modules

SSO requires modules that are not enabled by default. By default you can use `a2enmod` to do so.

```bash
$ a2enmod auth_kerb rewrite
$ systemctl restart apache2
```

add/uncomment the appropriate `LoadModule` statements in your Apache config:

```bash
# /etc/httpd/conf/httpd.conf
LoadModule auth_kerb_module /usr/lib/apache2/modules/mod_auth_kerb.so
LoadModule rewrite_module modules/mod_rewrite.so
```

40.4.5 2e. Configure Kerberos

Kerberos realm configuration is how you tell the Zammad server how to reach the domain controller (Active Directory server).

**Note:** Replace the following placeholders in the sample config below:

- `<domain>` Windows domain
- `<domain-controller>` Domain controller IP/FQDN(s)
- `<master-domain-controller>` Master domain controller IP/FQDN
  (must not be read-only, but can be the same as `<domain-controller>`)
40.4.6 2f. Generate keytab

Apache needs a Kerberos keytab (key table) to manage its shared secrets with the domain controller. 

Note: Replace the following placeholders in the commands below:

- `<zammad-host>` Zammad FQDN
- `<domain>` Windows domain
- `<secret-key>` Secret key (omit the leading 0x)
- `<vno>` Secret key version number

The secret key and version number were found in Step 1: Configure Active Directory above.

$ ktutil
ktutil: addent -key -p HTTP/<zammad-host> -k <vno> -e aes256-cts
Key for HTTP/<zammad-host>@<domain> (hex): <secret-key>

ktutil: list  # confirm the entry was added successfully
slot KVNO Principal
--- ---- ---------------------------------------------------------------
1 3 HTTP/<zammad-host>@<domain>

ktutil: wkt /root/zammad.keytab  # write keytab to disk
ktutil: quit

Then, place the keytab in the Apache config directory and set the appropriate permissions:

Ubuntu, Debian, openSUSE

CentOS

$ mv /root/zammad.keytab /etc/apache2/
$ chown www-data:www-data /etc/apache2/zammad.keytab
$ chmod 400 /etc/apache2/zammad.keytab

$ mv /root/zammad.keytab /etc/httpd/
$ chown apache:apache /etc/httpd/zammad.keytab
$ chmod 400 /etc/httpd/zammad.keytab
40.4.7 2g. Configure Apache

Add the following directive to the end of the virtual host configuration file to create your Kerberos SSO endpoint at /auth/sso:

```
<LocationMatch "/auth/sso" |>SSLRequireSSL
    AuthType Kerberos
    AuthName "Your Zammad"
    KrbMethodNegotiate On
    KrbMethodK5Passwd On
    KrbAuthRealms <domain>
    KrbLocalUserMapping on # strips @REALM suffix from REMOTE_USER_

    KrbServiceName HTTP/<zammad-host>@<domain>
    Krb5KeyTab /etc/apache2/zammad.keytab # Ubuntu, Debian, & openSUSE
    Krb5KeyTab /etc/httpd/zammad.keytab # CentOS
    require valid-user

    RewriteEngine On
    RewriteCond %{LA-U:REMOTE_USER} (.+)
    RewriteRule . - [E=RU:%1,NS]
    RequestHeader set X-Forwarded-User "%{RU}e" env=RU
</LocationMatch>
```

40.4.8 2g. Restart Apache to apply changes

```
$ systemctl restart apache2
```

40.5 Step 3: Enable SSO in Zammad

Next, enable “Authentication via SSO” in Zammad’s Admin Panel under Settings > Security > Third-Party Applications:

Note: On older versions of Zammad, visit https://your.zammad.host/auth/sso to sign in.
40.6 Step 4: Configure Client System (Windows Only)

For the full SSO experience (i.e., for passwordless, one-click sign-in), Zammad users must:

1. be on the Active Directory server’s local intranet; and
2. modify their network settings for the Zammad host to be treated as a local intranet server.

**Tip:** This setting can be centrally managed across the entire intranet using a group policy object (GPO).
1. Add your Zammad FQDN in Internet Options under **Security > Local Intranet > Sites > Advanced**.
2. Select “Require server verification (https:) for all sites in this zone”.
3. Under **Security level for this zone > Custom level... > Settings > User Authentication > Logon**, select “Automatic logon only in Intranet Zone”.

**Note:** This option cannot be centrally managed because it is set in the browser rather than Windows Settings.

1. Enter `about:config` in the address bar. Click **Accept the risk and continue**.
2. Search for the `network.negotiate-auth.trusted-uris` option.
3. Double-click to edit, then add your Zammad FQDN.
4. Restart Firefox to apply your changes.

![Fig. 5: Enter about:config in the address bar to access advanced settings in Firefox.](image)

### 40.7 Troubleshooting

- Are all relevant FQDNs/hostnames reachable from your Active Directory and Zammad servers (including each other’s)?
- Are the system clocks of your Active Directory and Zammad servers synchronized within five minutes of each other? (Kerberos is a time-sensitive protocol.)

#### 40.7.1 Errors in Apache Logs

**Tip:** Try raising your Apache log level temporarily.

Add `LogLevel debug` to your virtual host configuration, then restart the service to apply the changes.

“**an unsupported mechanism was requested**” Does your Active Directory service account have **Kerberos AES 256-bit encryption** enabled?

If for some reason your server does not support AES 256-bit encryption, the LDAP Wiki has more information about Kerberos encryption types.

“**failed to verify krb5 credentials: Key version is not available**” Did you use the exact **version number** (vno) provided by `ktpass` when **generating your keytab**?

Try generating it again, just to be sure.

“**unspecified GSS failure. Minor code may provide more information (, No key table entry found for HTTP/FQDN@DOMAIN)**” Does the **service name** you provided to `setspn` exactly match the one you used when **generating your keytab**?

Try generating it again, just to be sure.

“**No key table entry found for HTTP/FQDN@DOMAIN**” Does your virtual host configuration’s `KrbServiceName` setting exactly match the **service name** you provided to `setspn`?

This setting is case-sensitive.
“Warning: received token seems to be NTLM, which isn't supported by the Kerberos module. Check your IE configuration”
   
   Is your Zammad host accessible at an FQDN? This error may indicate that you configured your Zammad host as a numeric IP address instead.

“Cannot decrypt ticket for HTTP/FQDN@DOMAIN” Did you make sure to change the password on your Active Directory service account after enabling 256-bit AES encryption?

   And did you make sure to register the SPN (with ktpass) and generate your keytab (with ktutil) after changing your password?
CHAPTER FORTYONE

REPORTING TOOLS (THIRD PARTY)

This guide will discuss how to set up third party reporting tools with Zammad.

Fig. 1: Use third party reporting tools to boost your reporting capabilities.

41.1 Getting Started

You will need

• A instance of the reporting tool of your choice (hosted or self-hosted)
• (read) access to your Elasticsearch index

Warning: Never expose Elasticsearch to the public if you’re not sure how to do it. Especially never without authentication! Zammad stores very sensitive information within the Elasticsearch Index.

• (a Zammad 4+ instance that supports your use case)

Note: Limitations

Please note that this guide expects all requirements to be up and running. We will not cover core configurations of each tool. Please also note that we can’t support you with configuration of your specific third party tool.
Specific use cases

You may have specific use cases which we can’t cover in this documentation. The following sub pages and also our List of Indexed Attributes should provide enough information to help you!

41.2 Third Party Reporting Tools known to be working

41.2.1 Grafana

Grafana allows you to query, visualize and alert on metrics your Zammad installation stores within the Elasticsearch indexes.

Overview

Quickly jump to…

- Setting up required data sources
- The Dashboards

You will need

- A Grafana 7.5+ instance (hosted or self hosted)
- Worldmap panel plugin
- (read) access to your Elasticsearch index

Warning: Never expose Elasticsearch to the public if you’re not sure how to do it. Especially never without authentication! Zammad stores very sensitive information within the Elasticsearch Index.

- (a Zammad 4+ instance that supports your use case)
Note: Limitations

Please note that this guide expects all requirements to be up and running. We will not cover core configurations of each tool. Please also note that we can’t support you with configuration of your specific third party tool.

Specific use cases

You may have specific use cases which we can’t cover in this documentation. The following sub pages and also our List of Indexed Attributes should provide enough information to help you!

Setting up required data sources

Hint: You may not need all data sources

Before we start: The data sources always follow the same scheme. We reduced below information to name, time field name and index name. Everything else relies on your environment and is out of our scope.

Note: Please replace zammad_production_ with your fitting prefix.

ES - Chat Sessions:
- Index name: zammad_production_chat_session
- Time field name: created_at

ES - CTI Log:
- Index name: zammad_production_cti_log
- Time field name: start_at

ES - Ticket Articles:
- Index name: zammad_production_ticket
- Time field name: article.created_at

ES - Tickets by closed_at:
- Index name: zammad_production_ticket
- Time field name: close_at

ES - Tickets by created_at:
- Index name: zammad_production_ticket
- Time field name: created_at

ES - Tickets by first_response_at:
- Index name: zammad_production_ticket
- Time field name: first_response_at

With above data sources you basically have everything you need to start building your own dashboards.

Tip: Not sure about your index names?
- Querying your Elasticsearch like below
# Replace localhost:9200 with the IP/URL of your setup if needed

```bash
$ curl http://localhost:9200/_aliases?pretty=true
```

will return a list that looks similar to the following:

```json
{
    "zammad_production_knowledge_base_translation" : {
        "aliases" : { }
    },
    "zammad_production_ticket_priority" : {
        "aliases" : { }
    },
    "zammad_production_stats_store" : {
        "aliases" : { }
    },
    "zammad_production_organization" : {
        "aliases" : { }
    },
    "zammad_production_cti_log" : {
        "aliases" : { }
    },
    "zammad_production_group" : {
        "aliases" : { }
    },
    "zammad_production_knowledge_base_answer_translation" : {
        "aliases" : { }
    },
    "zammad_production_ticket" : {
        "aliases" : { }
    },
    "zammad_production_ticket_state" : {
        "aliases" : { }
    },
    "zammad_production_chat_session" : {
        "aliases" : { }
    },
    "zammad_production_user" : {
        "aliases" : { }
    },
    "zammad_production_knowledge_base_category_translation" : {
        "aliases" : { }
    }
}
```
The Dashboards

If you want to get inspired, you can also use our sample dashboards as mentioned below. These dashboards can also be found on GitHub.

Importing an existing Dashboard

Navigate to → Import and either upload the json file you received or use the grafana.com ID. During importing you can provide a dashboard name and folder. You’ll also be asked to map the data sources to your environment. If you used our data source names above, you can simply search for the same name.

Fig. 2: Importing existing dashboards by ID

Ticket statistics

Tip: Grafana.com ID: 14222

This dashboard provides graphs for:

- ticket opening and closing
- created articles
- ticket SLA (in time and violation) per type

It also contains specific ticket and article meta information:

- ticket group distribution

---

2 Some values are not available as time series information. This means we can only display the last value of the field in question.

3 Requires SLA function to be active. Negative values indicate SLA violations.
• sender ratio (e.g. Customer / Agent)
• article type ratio (e.g. email, phone)
• article content type
• escalation ratios
• average first response, update time and close time
• top 10
  – organization of ticket customer
  – ticket customers
  – ticket owners
  – average accounted time on ticket
  – ticket tags
• last 10 escalated tickets

**Required data sources:**

• ES - Ticket Articles
• ES - Tickets by created_at
• ES - Tickets by closed_at

**Chat-Session statistics**

**Tip:** Grafana.com ID: 14224

---

**This dashboard provides graphs for:**

1 Specific reference IDs are not the same on every instance and thus the panel may not work or show incorrect data. Check the panels description on how to find our the relations on your system.
• Chat session creations

**It also contains specific chat session meta information:**

• top 10
  – chat tags
  – chat agents
  – chat exit pages
  – city origins
• chat topic ratio
• average number of messages within chat-sessions
• average chatting time
• World map with chat origin countries

**Required data sources:**

• ES - Chat Sessions

**CTI-Log statistics**

**Tip:** Grafana.com ID: 14223

**This dashboard provides graphs for:**

• number of calls per direction (in / out)

**It also contains specific chat session meta information:**

• call ratio (in / out)
• average waiting time
• average talking time
• top 10
  – callers (in)
  – call answerers (in)

Required data sources:
• ES - CTI Log

Note: Your favorite tool is not available?
Worry not, if it does support Elasticsearch Indexes, you may be good to go! See List of Indexed Attributes for available indexes.