

---

# Zammad

Feb 03, 2020



<b>1</b>	<b>Zammad</b>	<b>3</b>
<b>2</b>	<b>Software</b>	<b>5</b>
2.1	1. Ruby Programming Language . . . . .	5
2.2	2. Package Dependencies . . . . .	5
2.3	3. Database Server . . . . .	6
2.4	4. Reverse Proxy . . . . .	6
2.5	5. Elasticsearch (optional) . . . . .	6
<b>3</b>	<b>Hardware</b>	<b>9</b>
3.1	For Zammad and a database server like PostgreSQL we recommend at least: . . . . .	9
3.2	For optimal performance up to 40 agents: . . . . .	9
<b>4</b>	<b>Install from source</b>	<b>11</b>
4.1	Generic . . . . .	11
4.2	on Debian 7, 8 / Ubuntu 16.04 / Ubuntu 18.04 (with Nginx & MySQL) . . . . .	13
4.3	on Mac OS 10.8 . . . . .	14
<b>5</b>	<b>Install on CentOS via RPM</b>	<b>17</b>
5.1	Prerequisites . . . . .	17
5.2	Add Zammad-Repo and Install Zammad . . . . .	17
5.3	Go to <a href="http://localhost">http://localhost</a> and you'll see: . . . . .	18
5.4	You can manage the Zammad services manually: . . . . .	18
<b>6</b>	<b>Install on Debian via DEB</b>	<b>21</b>
6.1	Prerequisites . . . . .	21
6.2	Add Zammad DEB repo and install . . . . .	22
6.3	Go to <a href="http://localhost">http://localhost</a> and you'll see: . . . . .	22
6.4	Change your webserver configuration (non localhost connections): . . . . .	22
6.5	You can manage the Zammad services manually: . . . . .	23
<b>7</b>	<b>Install on Ubuntu via DEB</b>	<b>25</b>
7.1	Prerequisites . . . . .	25
7.2	Add Zammad DEB Repo . . . . .	26
7.3	Go to <a href="http://localhost">http://localhost</a> and you'll see: . . . . .	26
7.4	Change your webserver configuration (non localhost connections): . . . . .	26
7.5	You can manage the Zammad services manually: . . . . .	27

<b>8</b>	<b>Install on SUSE via RPM</b>	<b>29</b>
8.1	Install dependencies . . . . .	29
8.2	Add Zammad RPM repo and install . . . . .	29
8.3	Go to <a href="http://localhost">http://localhost</a> and you'll see: . . . . .	30
8.4	Change your webserver configuration (non localhost connections): . . . . .	30
8.5	You can manage the Zammad services manually: . . . . .	30
<b>9</b>	<b>Set up Elasticsearch</b>	<b>33</b>
9.1	Step 1: Installation . . . . .	33
9.2	Step 2: Suggested Configuration . . . . .	35
9.3	Step 3: Connect Zammad . . . . .	35
9.4	Appendix . . . . .	36
<b>10</b>	<b>Install with Docker-Compose</b>	<b>41</b>
10.1	Install Docker Environment . . . . .	41
10.2	Getting started with zammad-docker-compose . . . . .	42
10.3	Go to <a href="http://localhost">http://localhost</a> and you'll see: . . . . .	42
10.4	Maintenance . . . . .	42
<b>11</b>	<b>Install on Kubernetes via Helm</b>	<b>45</b>
<b>12</b>	<b>Installation on Univention Corporate Server via App Center</b>	<b>47</b>
12.1	Prerequisites . . . . .	47
12.2	Installing Zammad . . . . .	47
12.3	First steps you should consider . . . . .	49
12.4	Further information . . . . .	50
<b>13</b>	<b>Updating Zammad</b>	<b>55</b>
13.1	Source update . . . . .	55
13.2	Update with RPM . . . . .	56
13.3	Update with DEB . . . . .	56
13.4	Updating elasticsearch . . . . .	57
<b>14</b>	<b>First steps</b>	<b>59</b>
<b>15</b>	<b>from OTRS</b>	<b>61</b>
15.1	Install plugins on OTRS . . . . .	61
15.2	Import via Browser . . . . .	62
15.3	Import via command line . . . . .	62
15.4	Importing a diff . . . . .	63
15.5	Restarting from scratch . . . . .	64
<b>16</b>	<b>from Zendesk</b>	<b>65</b>
<b>17</b>	<b>Console</b>	<b>67</b>
17.1	Start Zammad's Rails console . . . . .	67
17.2	Working on the console . . . . .	68
<b>18</b>	<b>Start</b>	<b>81</b>
18.1	Source Code . . . . .	81
18.2	Documentation . . . . .	81
<b>19</b>	<b>Branches</b>	<b>83</b>
19.1	Master . . . . .	83
19.2	Develop . . . . .	83
19.3	Stable . . . . .	84

19.4	Stable-X.x	84
<b>20</b>	<b>Packages</b>	<b>85</b>
<b>21</b>	<b>Continuous integration</b>	<b>87</b>
21.1	Internal	87
21.2	External	87
<b>22</b>	<b>Code quality</b>	<b>89</b>
22.1	Rubocop	89
22.2	Codeclimate	89
<b>23</b>	<b>Install with Docker</b>	<b>91</b>
23.1	Run the Docker Container	91
23.2	Go to http://localhost and you'll see:	92
<b>24</b>	<b>Install with Vagrant</b>	<b>93</b>
24.1	Clone the Vagrant file	93
24.2	Run Vagrant	93
24.3	Go to http://localhost:8080 and you'll see:	94
24.4	SSH into the machine	94
24.5	Problems starting the VM?	94
<b>25</b>	<b>Introduction</b>	<b>95</b>
25.1	The API	95
25.2	Authentication	95
25.3	Request Format	96
25.4	Example CURL Requests	96
25.5	Example CURL Requests (for tickets and users)	97
25.6	Example CURL Request on behalf of a different user	97
25.7	Response Format	98
25.8	Response Format (expanded)	98
25.9	Pagination	98
25.10	API clients	99
<b>26</b>	<b>User</b>	<b>101</b>
26.1	me - current user	101
26.2	List	101
26.3	Search	102
26.4	Show	103
26.5	Create	103
26.6	Update	104
26.7	Delete	105
<b>27</b>	<b>Organization</b>	<b>107</b>
27.1	List	107
27.2	Search	108
27.3	Show	108
27.4	Create	109
27.5	Update	109
27.6	Delete	110
<b>28</b>	<b>Group</b>	<b>111</b>
28.1	List	111
28.2	Show	112

28.3	Create . . . . .	112
28.4	Update . . . . .	113
28.5	Delete . . . . .	114
<b>29</b>	<b>Ticket</b>	<b>115</b>
29.1	List . . . . .	115
29.2	Search . . . . .	116
29.3	Show . . . . .	117
29.4	Create . . . . .	117
29.5	Update . . . . .	119
29.6	Delete . . . . .	121
<b>30</b>	<b>Ticket State</b>	<b>123</b>
30.1	List . . . . .	123
30.2	Show . . . . .	124
30.3	Create . . . . .	124
30.4	Update . . . . .	125
30.5	Delete . . . . .	126
<b>31</b>	<b>Ticket Priority</b>	<b>127</b>
31.1	List . . . . .	127
31.2	Show . . . . .	128
31.3	Create . . . . .	128
31.4	Update . . . . .	129
31.5	Delete . . . . .	129
<b>32</b>	<b>Ticket Article</b>	<b>131</b>
32.1	By Ticket . . . . .	131
32.2	Show . . . . .	132
32.3	Create . . . . .	133
<b>33</b>	<b>Online Notification</b>	<b>137</b>
33.1	List . . . . .	137
33.2	Show . . . . .	138
33.3	Update . . . . .	138
33.4	Delete . . . . .	139
33.5	Mark all as read . . . . .	139
<b>34</b>	<b>Object</b>	<b>141</b>
34.1	List . . . . .	141
34.2	Show . . . . .	142
34.3	Create . . . . .	143
34.4	Update . . . . .	144
34.5	Execute Database Migrations . . . . .	146
<b>35</b>	<b>Tags</b>	<b>147</b>
35.1	List . . . . .	147
35.2	Search . . . . .	147
35.3	Add . . . . .	148
35.4	Remove . . . . .	148
35.5	Admin - List . . . . .	148
35.6	Admin - Create . . . . .	149
35.7	Admin - Rename . . . . .	150
35.8	Admin - Delete . . . . .	150

<b>36</b>	<b>User Access Token</b>	<b>151</b>
36.1	List . . . . .	151
36.2	Create . . . . .	152
36.3	Delete . . . . .	152
<b>37</b>	<b>Introduction</b>	<b>155</b>
37.1	Feature list . . . . .	155
<b>38</b>	<b>Push API</b>	<b>157</b>
38.1	How it works . . . . .	157
38.2	Endpoint . . . . .	157
38.3	Events . . . . .	157
<b>39</b>	<b>Backup and Restore</b>	<b>161</b>
39.1	Configuration . . . . .	161
39.2	Create Backup . . . . .	161
39.3	Migrating from another Zammad-Host . . . . .	162
39.4	Restore everything . . . . .	162
39.5	What to do after restoration has been completed . . . . .	163
<b>40</b>	<b>Configure environment variables</b>	<b>165</b>
40.1	Configure IP . . . . .	165
40.2	Configure ports . . . . .	165
40.3	Application Servers . . . . .	165
40.4	Configure Restart Command . . . . .	166
40.5	Configure Zammad to log to STDOUT . . . . .	166
<b>41</b>	<b>Package Repo Files</b>	<b>167</b>
41.1	CentOS 7 . . . . .	167
41.2	Debian 8 . . . . .	167
41.3	Debian 9 . . . . .	168
41.4	Ubuntu 16.04 . . . . .	168
41.5	Ubuntu 18.04 . . . . .	168
41.6	SLES 12 . . . . .	168
41.7	Note . . . . .	168
<b>42</b>	<b>Privacy &amp; Data Retention</b>	<b>169</b>
42.1	On-Premises Data . . . . .	169
42.2	External Services . . . . .	170





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>)



# CHAPTER 1

---

## Zammad

---

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 2.5.5

**Warning:** We changed our Ruby dependency with Zammad 3.1. Earlier Zammad-Versions require Ruby 2.4.4.

### 2.2 2. 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.

**Note:** The below package dependency was added with Zammad 2.9 which improves image previews.

**Warning:** Please note that upgrading from Zammad 2.8 and earlier might fail, because your system does not satisfy the new dependencies. Below installation commands will help you out (you can update normally afterwards)

```
# Debian 8 & 9, Ubuntu 16.04 & 18.04  
$ apt-get install libimlib2 libimlib2-dev
```

(continues on next page)

(continued from previous page)

```
# openSUSE
$ zypper install imlib2 imlib2-devel

# CentOS 7
$ yum install imlib2 imlib2-devel
```

## 2.3 3. Database Server

Zammad will store all content in an RDBMS. You can choose between the following products:

- MySQL 5.6+
- MariaDB 10.0+
- PostgreSQL 9.1+

---

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

---

**Warning: Required configuration for MySQL/MariaDB:**

- Use UTF8 encoding. utf8mb4 for example will fail.
- Set `max_allowed_packet` to a value larger than the default of 4 MB (64 MB+ recommended).

## 2.4 4. 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+

## 2.5 5. Elasticsearch (optional)

---

**Note:** Package install will insist on installing elasticsearch, you can break dependencies during install if needed.

---

**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!

For excellent search performance we use Elasticsearch. The following Elasticsearch versions are supported:

- Elasticsearch 5.5 with `mapper-attachments` plugin

- Elasticsearch 5.6, 6.x & 7.x with `ingest-attachment` plugin

**Warning:** Please note that Elasticsearch 6.x and 7.x support came with Zammad 3.1. If you try to use Elasticsearch newer than 5.6.x on Zammad 3.0 and earlier, your search index will **not work**.

**Warning:** Please note that we will be dropping Elasticsearch support prior 5.5.x on future releases.





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.

---



## 4.1 Generic

### 4.1.1 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*.

### 4.1.2 1. Install Zammad on your system

You can directly download Zammad from <https://ftp.zammad.com/> or use the direct URL to get the latest stable release via <https://ftp.zammad.com/zammad-latest.tar.gz>

```
$ sudo useradd zammad -m -d /opt/zammad -s /bin/bash
$ cd /opt
$ sudo wget https://ftp.zammad.com/zammad-latest.tar.gz
$ sudo tar -xzf zammad-latest.tar.gz -C zammad
$ sudo su - zammad
```

### 4.1.3 2. Install all dependencies

Please note that a working ruby 2.5.5 environment is needed.

```
zammad@host $ gem install bundler rake rails
```

**For PostgreSQL (note, the option says “without ... mysql”)**

```
zammad@host $ bundle install --without test development mysql
```

### For MySQL (note, the option says “without ... postgres”)

```
zammad@host $ bundle install --without test development postgres
```

#### 4.1.4 3. Configure your databases

```
zammad@host $ cp config/database/database.yml config/database.yml
zammad@host $ vi config/database.yml
```

#### 4.1.5 4. Initialize your database

```
zammad@host $ export RAILS_ENV=production
zammad@host $ export RAILS_SERVE_STATIC_FILES=true
zammad@host $ rake db:create
zammad@host $ rake db:migrate
zammad@host $ rake db:seed
```

#### 4.1.6 5. Change directory to zammad (if needed) and start services:

```
zammad@host $ rake assets:precompile
```

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

#### Starting all servers manually

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

#### Starting servers with Systemd

```
zammad@host $ cd scripts/systemd
zammad@host $ sudo ./install-zammad-systemd-services.sh
```

#### 4.1.7 6. Go to <http://localhost:3000> and you'll see:

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

#### Reset a Zammad installation (for a fresh start after testing)

Please note: this actions will delete all existing data! Dont use it on a production system.

```

zammad@host $ sudo systemctl stop zammad
zammad@host $ rake db:drop
zammad@host $ rake db:create
zammad@host $ rake db:migrate
zammad@host $ rake db:seed
zammad@host $ sudo systemctl start zammad

```

## 4.2 on Debian 7, 8 / Ubuntu 16.04 / Ubuntu 18.04 (with Nginx & MySQL)

### 4.2.1 Prerequisites

```

$ apt-get update
$ apt-get install curl git-core patch build-essential bison zlib1g-dev libssl-dev
↳ libxml2-dev libxml2-dev sqlite3 libsqlite3-dev autotools-dev libxslt1-dev libyaml-0-
↳ 2 autoconf automake libreadline6-dev libyaml-dev libtool libgmp-dev libgdbm-dev
↳ libncurses5-dev pkg-config libffi-dev libmysqlclient-dev mysql-server nginx gawk
↳ libimlib2-dev

```

### 4.2.2 Add User

```

$ useradd zammad -m -d /opt/zammad -s /bin/bash
$ echo "export RAILS_ENV=production" >> /opt/zammad/.bashrc

```

### 4.2.3 Create MySQL user zammad (for Debian: upgrade MySQL to v5.6+ before, see: <http://dev.mysql.com/downloads/repo/apt/>)

```

$ mysql --defaults-extra-file=/etc/mysql/debian.cnf -e "CREATE USER 'zammad'@
↳ 'localhost' IDENTIFIED BY 'Your_Pass_Word'; GRANT ALL PRIVILEGES ON zammad_prod.*
↳ TO 'zammad'@'localhost'; FLUSH PRIVILEGES;"

```

### 4.2.4 Get Zammad

```

$ su - zammad
$ curl -O https://ftp.zammad.com/zammad-latest.tar.gz
$ tar -xzf zammad-latest.tar.gz
$ rm zammad-latest.tar.gz

```

### 4.2.5 Install environment

```

$ gpg --keyserver hkp://keys.gnupg.net --recv-keys
↳ 409B6B1796C275462A1703113804BB82D39DC0E3
$ curl -L https://get.rvm.io | bash -s stable
$ source /opt/zammad/.rvm/scripts/rvm
$ echo "source /opt/zammad/.rvm/scripts/rvm" >> /opt/zammad/.bashrc

```

(continues on next page)

(continued from previous page)

```
$ echo "rvm --default use 2.5.5" >> /opt/zammad/.bashrc
$ rvm install 2.5.5
$ gem install bundler
```

### 4.2.6 Install Zammad

```
$ bundle install --without test development postgres
$ cp config/database/database.yml config/database.yml
```

- insert mysql user, pass & change adapter to mysql2 & change database to zammad\_prod

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

```
$ rake db:create
$ rake db:migrate
$ rake db:seed
$ rake assets:precompile
```

### 4.2.7 Start Zammad

```
$ rails s -p 3000 &>> log/zammad.log &
$ script/websocket-server.rb start &>> log/zammad.log &
$ script/scheduler.rb start &>> log/zammad.log &
```

### 4.2.8 Create Nginx Config & restart Nginx

```
$ exit
$ cp /opt/zammad/contrib/nginx/zammad.conf /etc/nginx/sites-available/zammad.conf
```

- change servername “localhost” to your domain if your’re not testing locally

```
$ vi /etc/nginx/sites-available/zammad.conf
$ ln -s /etc/nginx/sites-available/zammad.conf /etc/nginx/sites-enabled/zammad.conf
$ systemctl restart nginx
```

### 4.2.9 Go to <http://localhost> and you’ll see:

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

## 4.3 on Mac OS 10.8

### 4.3.1 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
```

### 4.3.2 Get Zammad

```
$ test -d ~/zammad/ || mkdir ~/zammad
$ cd ~/zammad/
$ curl -L -O https://ftp.zammad.com/zammad-latest.tar.bz2 | tar -xj
```

### 4.3.3 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
```

### 4.3.4 Database connect

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

### 4.3.5 Start Zammad

```
$ 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
```

### 4.3.6 Visit Zammad in your browser

- [http://localhost:3000/#getting\\_started](http://localhost:3000/#getting_started)





---

## Install on CentOS via RPM

---

---

**Note:** Currently we support RHEL7 & CentOS7.

---

### 5.1 Prerequisites

#### 5.1.1 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 Add Zammad-Repo and Install Zammad

```
$ sudo yum -y install epel-release wget
$ sudo wget -O /etc/yum.repos.d/zammad.repo https://dl.packager.io/srv/zammad/zammad/
↳ stable/installer/el/7.repo
$ sudo yum -y install zammad
```

#### 5.2.1 SeLinux & Firewalld

On Centos SeLinux & Firewalld are possibly enabled. To get everything work you have to use the following commands:

```
$ setsebool httpd_can_network_connect on -P
$ firewall-cmd --zone=public --add-service=http --permanent
$ firewall-cmd --zone=public --add-service=https --permanent
$ firewall-cmd --reload
```

### 5.3 Go to `http://localhost` and you'll see:

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

#### 5.3.1 Change your webserver configuration (non localhost connections):

Add your fully qualified domain name or public IP to server name directive in your web server configuration and restart your web server. The installer will give you a hint where Zammad's web server config file is located.

#### 5.3.2 nginx

**Warning:** Please **do not rename** the webserver config file for nginx or apache. The update process will re create it, if it does not exist!

```
# /etc/nginx/conf.d/zammad.conf

server {
    listen 80;

    # replace 'localhost' with your fqdn if you want to use zammad from remote
    server_name localhost;
```

### 5.4 You can manage the Zammad services manually:

#### 5.4.1 Zammad

```
$ sudo systemctl status zammad
$ sudo systemctl stop zammad
$ sudo systemctl start zammad
$ sudo systemctl restart zammad
```

#### 5.4.2 Only web application server

```
$ sudo systemctl status zammad-web
$ sudo systemctl stop zammad-web
$ sudo systemctl start zammad-web
$ sudo systemctl restart zammad-web
```

#### 5.4.3 Only worker process

```
$ sudo systemctl status zammad-worker
$ sudo systemctl stop zammad-worker
$ sudo systemctl start zammad-worker
$ sudo systemctl restart zammad-worker
```

#### 5.4.4 Only websocket server

```
$ sudo systemctl status zammad-websocket  
$ sudo systemctl stop zammad-websocket  
$ sudo systemctl start zammad-websocket  
$ sudo systemctl restart zammad-websocket
```



---

## Install on Debian via DEB

---

---

**Note:** Currently we support Debian 8, 9 and 10

---

### 6.1 Prerequisites

Be sure to use an UTF-8 locale or PostgreSQL will not install.

#### 6.1.1 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](#) .

#### 6.1.2 Check locale

```
$ locale
```

If there is nothing with UTF-8 in the name shown like “LANG=en\_US.UTF-8” you have to set a new locale.

#### 6.1.3 Set locale

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

## 6.2 Add Zammad DEB repo and install

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

### 6.2.1 For Debian 8

```
$ sudo wget -O /etc/apt/sources.list.d/zammad.list https://dl.packager.io/srv/zammad/  
→zammad/stable/installer/debian/8.repo
```

### 6.2.2 For Debian 9

```
$ sudo wget -O /etc/apt/sources.list.d/zammad.list https://dl.packager.io/srv/zammad/  
→zammad/stable/installer/debian/9.repo
```

### 6.2.3 For Debian 10

```
$ sudo wget -O /etc/apt/sources.list.d/zammad.list https://dl.packager.io/srv/zammad/  
→zammad/stable/installer/debian/10.repo
```

```
$ sudo apt-get update  
$ sudo apt-get install zammad
```

## 6.3 Go to <http://localhost> and you'll see:

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

## 6.4 Change your webserver configuration (non localhost connections):

Add your fully qualified domain name or public IP to server name directive in your web server configuration and restart your web server. The installer will give you a hint where Zammad's web server config file is located.

### 6.4.1 nginx

**Warning:** Please **do not rename** the webserver config file for nginx or apache. The update process will re create it, if it does not exist!

```
# /etc/nginx/sites-enabled/zammad.conf  
  
server {  
    listen 80;
```

(continues on next page)

(continued from previous page)

```
# replace 'localhost' with your fqdn if you want to use zammad from remote
server_name localhost;
```

## 6.5 You can manage the Zammad services manually:

### 6.5.1 Zammad

```
$ sudo systemctl status zammad
$ sudo systemctl stop zammad
$ sudo systemctl start zammad
$ sudo systemctl restart zammad
```

### 6.5.2 only web application server

```
$ sudo systemctl status zammad-web
$ sudo systemctl stop zammad-web
$ sudo systemctl start zammad-web
$ sudo systemctl restart zammad-web
```

### 6.5.3 only worker process

```
$ sudo systemctl status zammad-worker
$ sudo systemctl stop zammad-worker
$ sudo systemctl start zammad-worker
$ sudo systemctl restart zammad-worker
```

### 6.5.4 only websocket server

```
$ sudo systemctl status zammad-websocket
$ sudo systemctl stop zammad-websocket
$ sudo systemctl start zammad-websocket
$ sudo systemctl restart zammad-websocket
```





---

## Install on Ubuntu via DEB

---

---

**Note:** We currently support Ubuntu 16.04 LTS and 18.04 LTS.

---

### 7.1 Prerequisites

Be sure to use an UTF-8 locale or PostgreSQL will not install.

#### 7.1.1 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](#) .

#### 7.1.2 Check locale

```
$ locale
```

If there is nothing with UTF-8 in the name shown like “LANG=en\_US.UTF-8” you have to set a new locale.

#### 7.1.3 Set locale

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

## 7.2 Add Zammad DEB Repo

### 7.2.1 Ubuntu 16.04

```
$ wget -qO- https://dl.packager.io/srv/zammad/zammad/key | sudo apt-key add -  
$ sudo wget -O /etc/apt/sources.list.d/zammad.list \  
https://dl.packager.io/srv/zammad/zammad/stable/installer/ubuntu/16.04.repo
```

### 7.2.2 Ubuntu 18.04

```
$ wget -qO- https://dl.packager.io/srv/zammad/zammad/key | sudo apt-key add -  
$ sudo wget -O /etc/apt/sources.list.d/zammad.list \  
https://dl.packager.io/srv/zammad/zammad/stable/installer/ubuntu/18.04.repo
```

### 7.2.3 Install on Ubuntu (16.04 and 18.04)

```
$ sudo apt-get update  
$ sudo apt-get install zammad
```

Note: You might need to apt-get install wget apt-transport-https for the above instructions to work.

## 7.3 Go to <http://localhost> and you'll see:

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

## 7.4 Change your webserver configuration (non localhost connections):

Add your fully qualified domain name or public IP to server name directive in your web server configuration and restart your web server. The installer will give you a hint where Zammad's web server config file is located.

### 7.4.1 nginx

**Warning:** Please **do not rename** the webserver config file for nginx or apache. The update process will re create it, if it does not exist!

```
# /etc/nginx/sites-enabled/zammad.conf  
  
server {  
    listen 80;  
  
    # replace 'localhost' with your fqdn if you want to use zammad from remote  
    server_name localhost;
```

---

## 7.5 You can manage the Zammad services manually:

### 7.5.1 Zammad

```
$ sudo systemctl status zammad
$ sudo systemctl stop zammad
$ sudo systemctl start zammad
$ sudo systemctl restart zammad
```

### 7.5.2 Only web application server

```
$ sudo systemctl status zammad-web
$ sudo systemctl stop zammad-web
$ sudo systemctl start zammad-web
$ sudo systemctl restart zammad-web
```

### 7.5.3 Only worker process

```
$ sudo systemctl status zammad-worker
$ sudo systemctl stop zammad-worker
$ sudo systemctl start zammad-worker
$ sudo systemctl restart zammad-worker
```

### 7.5.4 Only websocket server

```
$ sudo systemctl status zammad-websocket
$ sudo systemctl stop zammad-websocket
$ sudo systemctl start zammad-websocket
$ sudo systemctl restart zammad-websocket
```



---

## Install on SUSE via RPM

---

---

**Note:** Currently we support SLES 12 and OpenSUSE with versions 42.2 and 42.3

---

**Warning:** OpenSUSE LEAP 15.0 hasn't been tested yet, but should work as well.

## 8.1 Install dependencies

### 8.1.1 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* .

### 8.1.2 nginx on SLES12

```
$ sudo zypper addrepo "http://nginx.org/packages/sles/12" "nginx"
```

## 8.2 Add Zammad RPM repo and install

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

### 8.3 Go to `http://localhost` and you'll see:

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

Note: Make sure that the firewall is not blocking port 80 (configure firewall via “yast firewall” or stop it via “systemctl stop SuSEfirewall2”).

### 8.4 Change your webserver configuration (non localhost connections):

Add your fully qualified domain name or public IP to server name directive in your web server configuration and restart your web server. The installer will give you a hint where Zammad’s web server config file is located.

#### 8.4.1 nginx

**Warning:** Please **do not rename** the webserver config file for nginx or apache. The update process will re create it, if it does not exist!

```
# /etc/nginx/sites-enabled/zammad.conf

server {
    listen 80;

    # replace 'localhost' with your fqdn if you want to use zammad from remote
    server_name localhost;
```

### 8.5 You can manage the Zammad services manually:

#### 8.5.1 Zammad

```
$ sudo systemctl status zammad
$ sudo systemctl stop zammad
$ sudo systemctl start zammad
$ sudo systemctl restart zammad
```

#### 8.5.2 Only web application server

```
$ sudo systemctl status zammad-web
$ sudo systemctl stop zammad-web
$ sudo systemctl start zammad-web
$ sudo systemctl restart zammad-web
```

### 8.5.3 Only worker process

```
$ sudo systemctl status zammad-worker
$ sudo systemctl stop zammad-worker
$ sudo systemctl zammad-worker
$ sudo systemctl restart zammad-worker
```

### 8.5.4 Only websocket server

```
$ sudo systemctl status zammad-websocket
$ sudo systemctl stop zammad-websocket
$ sudo systemctl start zammad-websocket
$ sudo systemctl restart zammad-websocket
```





---

## Set up Elasticsearch

---

Zammad's search function is powered by Elasticsearch, and requires one of:

- Elasticsearch 5.5 (with the [mapper attachments plugin](#))
- Elasticsearch 5.6 or above (with the [ingest attachment plugin](#))

**Warning:** Versions below 5.5 may continue to work for the time being, but are officially deprecated. Support will be dropped in upcoming releases.

---

**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.

---

### 9.1 Step 1: Installation

**Direct Download** Find the latest release on the [downloads page](#), or see the [installation guide](#) for in-depth instructions. Then,

```
# Install the attachment plugin
$ sudo /usr/share/elasticsearch/bin/elasticsearch-plugin install ingest-
↪attachment # for 5.6+
$ sudo /usr/share/elasticsearch/bin/elasticsearch-plugin install mapper-
↪attachments # for 5.5

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

and start Elasticsearch.

---

**Note:** Docker installations on macOS/Windows:

Setting the `vm.max_map_count` kernel parameter requires [additional steps](#).

---

### CentOS 7

```
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"| sudo tee /etc/yum.repos.d/elasticsearch-7.x.repo
yum install -y java-1.8.0-openjdk elasticsearch
sudo /usr/share/elasticsearch/bin/elasticsearch-plugin install ingest-
↳attachment
systemctl start elasticsearch
systemctl enable elasticsearch
```

### Debian 8

```
apt-get install apt-transport-https sudo wget
echo "deb http://ftp.debian.org/debian jessie-backports main" | sudo tee
↳-a /etc/apt/sources.list.d/debian-backports.list
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" |
↳sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-
↳key add -
apt-get update
apt-get install -t jessie-backports openjdk-8-jre
apt-get install elasticsearch
sudo /var/lib/dpkg/info/ca-certificates-java.postinst configure
sudo /usr/share/elasticsearch/bin/elasticsearch-plugin install ingest-
↳attachment
systemctl restart elasticsearch
systemctl enable elasticsearch
```

### Debian 9

```
apt-get install apt-transport-https sudo wget
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" |
↳sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-
↳key add -
apt-get update
apt-get install openjdk-8-jre elasticsearch
sudo /usr/share/elasticsearch/bin/elasticsearch-plugin install ingest-
↳attachment
systemctl restart elasticsearch
systemctl enable elasticsearch
```

### Ubuntu 16.04 & 18.04

```

apt-get install apt-transport-https sudo wget
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | \
↳ sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-
↳ key add -
apt-get update
apt-get install openjdk-8-jre elasticsearch
sudo /usr/share/elasticsearch/bin/elasticsearch-plugin install ingest-
↳ attachment
systemctl restart elasticsearch
systemctl enable elasticsearch

```

## 9.2 Step 2: Suggested Configuration

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

```

# /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](#).

## 9.3 Step 3: Connect Zammad

```

# 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

```

## 9.3.1 Optional settings

### Authentication

```
# 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?**

For HTTP Basic auth, try this nginx reverse proxy config.

Elasticsearch also supports authentication via its X-Pack paid subscription service. Consult the official Elasticsearch guides for more details.

---

### Index namespacing

```
# 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')"
```

### File-attachment indexing rules

```
# 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', [ '.png', '.jpg
↳ ', '.jpeg', '.mpeg', '.mpg', '.mov', '.bin', '.exe', '.box', '.mbox' ]
↳ )"
#
# Files larger than this size (in MB) will not be indexed
$ zammad run rails r "Setting.set('es_attachment_max_size_in_mb', 50)"
```

## 9.4 Appendix

### 9.4.1 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.

---

**Note:** These fields may vary if you created custom fields (objects) in the admin interface.

---

#### Ticket

Field	Sample Value	Description
article	Article	Article Hash, which includes all articles stored on a ticket
article_count	1	Count of articles
close_at	null	First close time, after create
close_diff_in_min	null	Business hours in minutes within or above the specified SLA for closing
close_escalation_at	null	Time stamp of the escalation if the SLA of the closing time has been violated
close_in_min	null	Business hours in minutes it took to close the ticket
create_article_sender	Customer	Who has created the first article (Agent, Customer)
create_article_sender_id	2	Sender id of the first article (Agent/Customer)
create_article_type	web	Article type for the first article (note, email, phone...)
create_article_type_id	11	Article type ID for the first article (note, email, phone...)
created_at	2017-08-03T14:21:38.701Z	Created timestamp (DateTime, UTC)
created_by	User	User details of the user who created the ticket
created_by_id	13	User id of user who created the ticket
customer	User	Customer details
customer_id	13	User id of the current customer (assigned to ticket)
escalation_at	null	Next first escalation date (nearest close_escalation_at, first_response_escalation_at)
first_response_at	null	Time stamp of the first reaction to the customer (DateTime, UTC)
first_response_diff_in_min	null	Business hours in minutes within or above the specified SLA for the first response
first_response_escalation_at	null	Time stamp of the escalation if the SLA of the first reaction time has been violated
first_response_in_min	null	Business hours in minutes it took to send initial response to customer.
group	Sales	Current ticket group (Sales, Support...)
group_id	1	Current ticket group id
id	19	Ticket id
last_contact_agent_at	null	Last contact to customer from agent, timestamp (DateTime, UTC)
last_contact_at	2017-08-03T14:21:38.701Z	Last contact timestamp (DateTime, UTC)
last_contact_customer_at	2017-08-03T14:21:38.701Z	Last contact from a customer, timestamp (DateTime, UTC)
note	null	Internal note for ticket
number	61019	The uniq ticket number
organization_id	null	Id of the organization of a given customer
owner	User	Current owner (agent)
owner_id	1	User id of owner
pending_time	null	Current pending time (DateTime, UTC)
preferences		Sub Hash for special information
priority	2 normal	Ticket priority
priority_id	2	ID of the currently set priority
state	new	Ticket state (new, open...)
state_id	1	Ticket state id for available ticket states (new, open...)
time_unit	null	Accounted time units for this ticket
title	Feedback Form	Ticket title
type	null	Ticket Type (deprecated)
update_diff_in_min	null	Business hours in minutes within or above the specified SLA for updating
update_escalation_at	null	Time stamp of the last update reaction to the customer (DateTime, UTC)
update_in_min	null	Business hours in minutes it took to send the last update response to customer
updated_at	2017-08-03T14:21:38.701Z	Last update timestamp (DateTime, UTC)
updated_by	User	User who updated the ticket
updated_by_id	13	User id of user who updated the ticket

## Article

Field	Sample Value	Description
attachment.title	file1.txt	File name
attachment.content	Hello world	File Content
attachment.keywords	keyword	File Keywords
attachment.content	Max	File Author
body	:)	Content of the article
cc	null	Content of the optional cc field
content_type	text/plain	Content type
created_at	2017-08-03T14:21:38.000Z	Article create date (DateTime, UTC)
created_by	See User	Who has created the article
created_by_id	13	Who (UserID) has created the article
from	Christopher Miller via <order@chrispresso.com>	Sender address of the article Sender address of the article
id	19	internal (DB) article id
in_reply_to	null	Content of reply to field
internal	FALSE	Is article visible for customer
message_id	null	Message ID (if article was an email)
message_id_md5	null	internal message id MD5 Checksum
origin_by_id	null	For which real user (UserID) the article creation has been
preferences	{ }	Hash for additional information
references	null	Email references header
reply_to	null	Content of the reply to field
sender	Customer	Who is the sender (Customer, Agent)
sender_id	2	Which type of user has created the article (Agent, Customer)
subject	Feedback Form	Article subject
ticket_id	19	referencing ticket ID
to	null	Content of the to field
type	web	Article type (phone, email, web...)
type_id	11	Article type id (phone, email, web...)
updated_at	2017-08-03T14:21:38.701Z	Update time of the article (DateTime, UTC)
updated_by	See User	Who has updated the article
updated_by_id	13	Who (UserID) has updated the article

## User

Field	Sample Value	Description
active	TRUE	is activ (boolean)
address		User Adress
city		User City
country		User Country
created_at	2017-07-26T21:21:28.000Z	User creation date (DateTime, UTC)
created_by_id	1	ID of user who created the current user
department		User Department
email	<a href="mailto:chris@chrispresso.com">chris@chrispresso.com</a>	User E-Mail
fax		User Fax
firstname	Christopher	User Firstname
id	3	Internal id (database, autincrement)
last_login	2017-07-26T21:23:15.019Z	User last login (DateTime, UTC)
lastname	Miller	User Lastname
login	<a href="mailto:chris@chrispresso.com">chris@chrispresso.com</a>	User Login
mobile		User Mobile
note		internal note
organization	Chrispresso Inc	Orgnaization name of the current user
organization_id	2	ID which links to the organization name
phone		User Phone
street		User Street
updated_at	2017-07-27T15:04:47.270Z	Last update date (DateTime, UTC)
updated_by_id	3	ID of user who updated the current user
verified	FALSE	is verified (boolean)
vip	FALSE	Is VIP (boolean)
web		User Web Url
zip		User ZIP





---

## Install with Docker-Compose

---

**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!

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](#).

---

**Tip:** 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.

## 10.1 Install Docker Environment

Your Docker environment needs to be up and running and you need to have docker-compose installed.

### 10.1.1 Docker

- <https://docs.docker.com/engine/installation/>

### 10.1.2 Docker-Compose

- <https://docs.docker.com/compose/install/>

## 10.2 Getting started with zammad-docker-compose

### 10.2.1 Clone GitHub repo

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

### 10.2.2 Setting `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>

---

### 10.2.3 Start Zammad using DockerHub images

- `docker-compose up`

## 10.3 Go to `http://localhost` and you'll see:

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

## 10.4 Maintenance

### 10.4.1 Updating Zammad

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

### 10.4.2 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`

### 10.4.3 Open shell in running Zammad image

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

#### 10.4.4 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



---

## Install on Kubernetes via Helm

---

**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

# Install / Upgrade Zammad
$ helm upgrade --install zammad zammad/zammad --namespace=zammad
```



---

## Installation on Univention Corporate Server via App Center

---

---

**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 be 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.](#)

### 12.1 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 ahead with the installation.

### 12.2 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 screenshot shows the Zammad App Center interface. At the top, there is a navigation bar with a home icon, a 'ZAMMAD' tab, a search bar, and notification icons. Below the navigation bar, the page title 'Zammad' is displayed. The main content area features the Zammad logo, a brief description of the software, and a prominent green 'INSTALL' button under the heading 'Install this App'. To the right, there is a 'More information' section with details about the provider (Zammad GmbH), contact (support@zammad.com), license (Free commercial use), version (3.0.0-5), and support options. Below this, there is an 'App Center Badges' section with a small icon.

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 )

The screenshot shows the Zammad App Center interface after installation. The navigation bar is identical to the previous screenshot. The main content area now features the Zammad logo and a 'First steps' section. The 'First steps' section includes a welcome message, a link to the Zammad Community for help, and instructions on how to use Zammad with a service contract. Below this, there is an 'After installation' section with a link to the documentation on how to start with UCS. At the bottom of the 'First steps' section, there are two buttons: 'APP SETTINGS' and 'UNINSTALL'. To the right, the 'Use this App' section now features a prominent green 'OPEN' button. The 'More information' section remains the same as in the previous screenshot.



## 12.2.1 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 1: Changes values during installation

value	default value	new value
notification sender	Notification Master <noreply@#{config.fqdn}>	Zammad <noreply@{FQDN-of-UCS}> <sup>3</sup>
maximum email size	10 MB	35 MB
FQDN	{FQDN-of-UCS}	{FQDN-of-UCS}:10412 <sup>3</sup>
HTTP-Type	<empty>	https
Allow customer registration	true	false
LDAP configuration	<empty>	Full LDAP-Configuration prepared <sup>2</sup>
LDAP activated	<empty>	false

**Note:**<sup>2</sup> 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:**<sup>3</sup> Please note that these settings are updated automatically, if you update FQDN and Port settings within the *Univention App-Settings*.

## 12.3 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.

---

### 12.3.1 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).

## 12.4 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.

### 12.4.1 Univention App-Settings

---

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

---

Within the management 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.

### 12.4.2 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!

### 12.4.3 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:** Please backup your Zammad instance before update! You can learn how to back up Zammad on *Backup and Restore*.

---

### 13.1 Source update

#### 13.1.1 1. Download Zammad to your system

You can directly download Zammad from <https://ftp.zammad.com/> or use the direct URL to get the latest stable release via <https://ftp.zammad.com/zammad-latest.tar.gz>

```
$ cd /opt
$ sudo wget https://ftp.zammad.com/zammad-latest.tar.gz
$ sudo tar -C zammad -xzf zammad-latest.tar.gz
$ sudo chown -R zammad /opt/zammad
$ sudo su - zammad
```

#### 13.1.2 2. Install all dependencies

```
zammad@host $ cd zammad
zammad@host $ gem install bundler
```

- For PostgreSQL (note, the option says “without ... mysql”):

```
zammad@host $ bundle install --without test development mysql
```

- For MySQL (note, the option says “without ... postgres”):

```
zammad@host $ bundle install --without test development postgres
```

### 13.1.3 3. Stop Zammad services

Stop the application server, websocket server and scheduler.

### 13.1.4 4. Upgrade your database

```
zammad@host $ export RAILS_ENV=production
zammad@host $ export RAILS_SERVE_STATIC_FILES=true # only if you use no HTTP reverse_
↳proxy
zammad@host $ rake db:migrate
zammad@host $ rake assets:precompile
```

### 13.1.5 5. Start Zammad services

Start the application server, websocket server and scheduler.

### 13.1.6 6. Go and login to Zammad

## 13.2 Update with RPM

### 13.2.1 1. Stop Zammad

```
$ sudo systemctl stop zammad
```

### 13.2.2 3. Update Zammad

```
$ sudo yum update zammad
```

**Note:** The package will automatically execute maintenance tasks like database changes and will restart Zammad for you.

### 13.2.3 4. Start Zammad

```
$ sudo systemctl start zammad
```

### 13.2.4 5. Go and log in to Zammad

## 13.3 Update with DEB

**Note:** Please backup your Zammad instance before update!



### 13.3.1 1. Stop Zammad

```
$ sudo systemctl stop zammad
```

### 13.3.2 3. Update Zammad

```
$ apt-get update  
$ apt-get upgrade
```

**Note:** The package will automatically execute maintenance tasks like database changes and will restart Zammad for you.

### 13.3.3 4. Start Zammad

```
$ sudo systemctl start zammad
```

### 13.3.4 5. Go and log in to Zammad

## 13.4 Updating 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.



# CHAPTER 14

---

## First steps

---

After installing Zammad, open <http://localhost:3000> with your browser and follow the installation wizard.

You can find further information about Zammad's configuration on our [Admin-Documentation](#) (for Zammad-Administration within the WebApp).

Basically you need to do the following things:

- Create an admin account
- Set your system basics like hostname and logo
- Connect channels like an email account
- Invite agents and customers to work with you in Zammad



## 15.1 Install plugins on OTRS

---

**Note:** Currently only passwords of OTRS  $\geq$  3.3 can be reused in Zammad! Passwords that were stored in another format than the default SHA2 are not possible to use. Users then have to use the password reset procedure.

---

### 15.1.1 Install Znuny4OTRS-Repo

This is a dependency of the OTRS migration plugin

- On OTRS 6:
  - [https://addons.znuny.com/api/addon\\_repos/public/1029/latest](https://addons.znuny.com/api/addon_repos/public/1029/latest)
- On OTRS 5:
  - [https://addons.znuny.com/api/addon\\_repos/public/615/latest](https://addons.znuny.com/api/addon_repos/public/615/latest)
- On OTRS 4:
  - [https://addons.znuny.com/api/addon\\_repos/public/309/latest](https://addons.znuny.com/api/addon_repos/public/309/latest)
- On OTRS 3:
  - [https://addons.znuny.com/api/addon\\_repos/public/142/latest](https://addons.znuny.com/api/addon_repos/public/142/latest)

### 15.1.2 Install OTRS migration plugin

- OTRS 6:
  - [https://addons.znuny.com/api/addon\\_repos/public/1085/latest](https://addons.znuny.com/api/addon_repos/public/1085/latest)
- OTRS 5:

- [https://addons.znuny.com/api/addon\\_repos/public/617/latest](https://addons.znuny.com/api/addon_repos/public/617/latest)
- OTRS 4:
  - [https://addons.znuny.com/api/addon\\_repos/public/383/latest](https://addons.znuny.com/api/addon_repos/public/383/latest)
- OTRS 3.1 - 3.3:
  - [https://addons.znuny.com/api/addon\\_repos/public/287/latest](https://addons.znuny.com/api/addon_repos/public/287/latest)

## 15.2 Import via Browser

---

**Note:** If your OTRS installation is rather huge, you might want to consider using the command line version of this feature.

---

After installing Zammad, open <http://localhost:3000> with your browser and follow the installation wizard. From there you're able to start the migration from OTRS.

See the Video at [this site](#) .

## 15.3 Import via command line

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,...)

### 15.3.1 If you installed the Zammad DEB or RPM package

```
$ zammad run rails c
```

### 15.3.2 If you installed from source

```
$ su zammad
$ cd /opt/zammad
$ rails c
```

### 15.3.3 Extending import time for big installations (optional)

Optional, if you're having a bigger installation or running in timeouts like: `Delayed::Worker.max_run_time` is only 14400 seconds (4 hours) you need to do the following:

#### For importing via console

- open the file `config/initializers/delayed_jobs_settings_reset.rb` and add the following at the end of it:

```
>> Delayed::Worker.max_run_time = 7.days
```

- Restart the Zammad-Service (systemctl restart zammad)

### For importing via browser (not recommended on big installations)

Run below in a Zammad console and ensure to not close it during import:

```
>> Delayed::Worker.max_run_time = 7.days
```

**Note:** The above setting is only valid for the lifetime of the Zammad rails console. If you close the console, the change is reset to the default value.

## 15.3.4 Enter the following commands in the rails console

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

After the import is done switch Zammad back to non-import mode and mark the system initialization as done.

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

Start all Zammad processes again. Done.

## 15.4 Importing a diff

**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.

### 15.4.1 Enter the following commands in the rails console

```
>> 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
```

After the import is done switch Zammad back to non-import mode and mark the system initialization as done.

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

Start all Zammad processes again. Done.

## 15.5 Restarting from scratch

First make sure all Zammad processes are stopped. After that reset your database.

### 15.5.1 If you installed the Zammad DEB or RPM package

```
$ zammad run rake db:drop
$ zammad run rake db:create
$ zammad run rake db:migrate
$ zammad run rake db:seed
```

### 15.5.2 If you installed from source

```
$ rake db:drop
$ rake db:create
$ rake db:migrate
$ rake db:seed
```

After that your DB is reset and you can start the import right over.



## CHAPTER 16

---

from Zendesk

---

Sorry, this still needs to be added :-)

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.



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.

## 17.1 Start Zammad's Rails console

### 17.1.1 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!).

---

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

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

### 17.1.2 Running several commands in a shell

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

```
# package installation
$ zammad run rails c

# source installation
$ rails c
```

## 17.2 Working on the console

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

### 17.2.1 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 our [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\_rprovider 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
>> Setting.set('es_index', Socket.gethostname + '_zammad') # Change the index name
>> Setting.set('es_attachment_ignore', %w[.png .jpg .jpeg .mpeg .mpg .mov .bin .exe .
↳box .mbox]) # A list of ignored file extensions (they will not be indexed)
>> Setting.set('es_attachment_max_size_in_mb', 50) # Limit the Attachment-
↳Size to push to your elasticsearch index
```

## Use the OTRS importer from the shell

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

```
>> Setting.set('import_otrs_endpoint', 'http://xxx/otrs/public.pl?
↳Action=ZammadMigrator')
>> 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')
```

## 17.2.2 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 Zammads behavior.

---

**Note:** Please note that this is not a full command list, if you're missing commands, feel free to ask over at our [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')
```

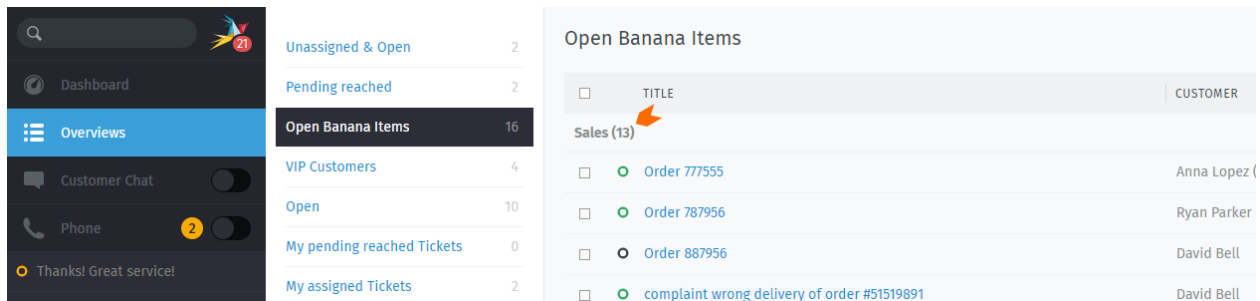
You can easily check the current BCC-Setting by running the following:

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

### 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.

```
>> 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)
```



The screenshot shows the Zammad web interface. On the left is a dark sidebar with navigation options: Dashboard, Overviews (selected), Customer Chat, and Phone. The main content area displays a list of ticket categories with counts: Unassigned & Open (2), Pending reached (2), Open Banana Items (16), VIP Customers (4), Open (10), My pending reached Tickets (0), and My assigned Tickets (2). The 'Open Banana Items' category is expanded to show a table with columns 'TITLE' and 'CUSTOMER'. The table lists several tickets, including 'Sales (13)', 'Order 777555', 'Order 787956', 'Order 887956', and 'complaint wrong delivery of order #51519891'. An orange arrow points to the 'Sales (13)' heading.

### Default Ticket type on creation

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

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

To check what setting is set currently, simply run

```
>> 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"=>" ".

```
>> 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:

```
>> 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.

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

Get the current state of this setting with:

```
>> 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 Zammads email font setting. This setting does not require a service restart.

```
>> 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.

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

## 17.2.3 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 our Community.

---

### Find user

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

```
>> 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-Address 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@exmaple.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@exmaple.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')
```

## 17.2.4 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 our [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 articles 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.

```
>> Ticket.find_by(number: '101234').articles_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.

```
>> 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.

---

```
>> 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).

```
>> 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.

```
>> 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.

```
>> 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:

```
>> 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 to 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!

```
>> 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)
>> attribute.screens[:create_middle]['ticket.customer'][:filter] = Ticket::State.by_
↳category(:viewable_customer_new).pluck(:id)
>> attribute.screens[:edit]['ticket.agent'][:filter] = Ticket::State.by_
↳category(:viewable_agent_edit).pluck(:id)
>> attribute.screens[:edit]['ticket.customer'][:filter] = Ticket::State.by_
↳category(:viewable_customer_edit).pluck(:id)
>> attribute.save!
```

## 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:

```
>> 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:

```
>> 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))

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

### 17.2.5 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
```

### 17.2.6 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
```

## 17.2.7 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 our [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.

```
>> Channel.fetch
```

### Add translation

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

```
>> 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.

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

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

### 17.2.8 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)

```
# 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

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

>> customers = User.joins(roles: :permissions).where(email: customer_emails,
↳roles: { active: true }, permissions: { name: 'ticket.customer', active: true }
↳).where.not(id: 1)
```

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

```
>> puts customers.map { |user| <<~PREVIEW }.join("\n")
      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**

```
>> 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}..."
  ActiveSupport.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**

```
# 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**

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

**Step 3: Proceed with deletion**

```
>> 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

```
# 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
```



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>

## 18.1 Source Code

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

## 18.2 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](https://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.

### 18.2.1 ReStructuredText markup

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

- <http://www.sphinx-doc.org/en/stable/rest.html>
- <http://docutils.sourceforge.net/docs/user/rst/quickref.html>
- [http://docs.readthedocs.io/en/latest/\\_themes/sphinx\\_rtd\\_theme/demo\\_docs/source/demo.html](http://docs.readthedocs.io/en/latest/_themes/sphinx_rtd_theme/demo_docs/source/demo.html)

Thanks!

Zammad Team



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

### 19.1 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

### 19.2 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!

## 19.3 Stable

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

## 19.4 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

## CHAPTER 20

---

### Packages

---

- 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.

### 21.1 Internal

- Done on our private continuous integration platform

### 21.2 External

#### 21.2.1 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](https://travis-ci.org) to get your builds tested
- Just change the file “.travis.yml” to fit your needs
- Current build test status is:





- All pushes to our main repo at <https://github.com/zammad/zammad> will be checked by several libraries and services

### 22.1 Rubocop

- Code is being checked by Rubocop
- <http://rubocop.readthedocs.io>

### 22.2 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



---

## Install with Docker

---

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.

## 23.1 Run the Docker Container

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

### 23.1.1 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>

---

### 23.1.2 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.

### 23.2 Go to <http://localhost> and you'll see:

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

---

## Install with Vagrant

---

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>.

### 24.1 Clone the Vagrant file

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

### 24.2 Run Vagrant

#### 24.2.1 For stable branch package

```
$ PACKAGER_REPO=stable vagrant up --provision
```

#### 24.2.2 For develop branch package

```
$ vagrant up --provision
```

That’s it! You’re now running Zammad in a Vagrant environment.

## 24.3 Go to <http://localhost:8080> and you'll see:

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

## 24.4 SSH into the machine

After “vagrant up”

```
$ vagrant ssh
```

After this you can switch to root user via:

```
$ sudo -i
```

## 24.5 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.

## 25.1 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}
```

## 25.2 Authentication

Zammad supports three different authentication methods for API.

### 25.2.1 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 [here]([https://en.wikipedia.org/wiki/Basic\\_access\\_authentication](https://en.wikipedia.org/wiki/Basic_access_authentication)).

```
$ curl -u {username}:{password} https://your_zammad/api/v1/users
```

## 25.2.2 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
```

## 25.2.3 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
```

## 25.3 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"
}
```

## 25.4 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
```



## 25.5 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](#).

## 25.6 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_of_existing_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
- user email

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

## 25.7 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"
}
```

## 25.8 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"
}
```

## 25.9 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"
  },
  ...
]
```

## 25.10 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.



### 26.1 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",
  ...
}
```

### 26.2 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:

```
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",
    ...
  },
]
```

## 26.3 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",

```

(continues on next page)

(continued from previous page)

```
...
},
{
  "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",
  ...
},
]
```

## 26.4 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

{
  "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",
  ...
}
```

## 26.5 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",
  ...
}
```

## 26.6 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:

```
Status: 200 Ok

{
  "id": 123,
```

(continues on next page)



(continued from previous page)

```
"firstname": "Bob",
"lastname": "Smith",
"email": "bob@smith.example.com",
"organization_id": 124,
"organization": "Some Other Organization Name",
"note": "some note",
"updated_at": "2016-08-16T07:55:42.119Z",
"created_at": "2016-08-16T07:55:42.119Z",
...
}
```

## 26.7 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
```

```
{}
```



## 27.1 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"
  }
]
```

(continues on next page)

(continued from previous page)

```
},  
]
```

## 27.2 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"  
  },  
]
```

## 27.3 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
```

```
{
  "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"
}
```

## 27.4 Create

Required permission:

- admin.organization

Request:

```
POST /api/v1/organizations
```

```
{
  "name": "Org 1",
  "shared": true,
  "active": true,
  "note": "some note"
}
```

Response:

```
Status: 201 Created
```

```
{
  "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"
}
```

## 27.5 Update

Required permission:

- admin.organization

Request:

```
PUT /api/v1/organizations/{id}
```

```
{
```

(continues on next page)

(continued from previous page)

```
"id": 123,  
"name": "Org 1",  
"shared": true,  
"active": true,  
"note": "some note"  
}
```

**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"  
}
```

## 27.6 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  
  
{}
```

## 28.1 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,
```

(continues on next page)

(continued from previous page)

```
"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"
},
]
```

## 28.2 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"
}
```

## 28.3 Create

Required permission:

- admin.group

Request:

```
POST /api/v1/groups

{
  "name": "Group 1",
  "signature_id": 123,
  "email_address_id": 123,
  "assignment_timeout": 180,
```

(continues on next page)



(continued from previous page)

```
"follow_up_possible": "yes",
"follow_up_assignment": true,
"active": true,
"note": "some note"
}
```

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"
}
```

## 28.4 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,
```

(continues on next page)

(continued from previous page)

```
"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"  
}
```

## 28.5 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  
  
{}
```

## 29.1 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:

```
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,
```

(continues on next page)

(continued from previous page)

```
"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",
...
},
]
```

## 29.2 Search

Required permission:

- ticket.agent (access to all ticket in allocated groups)
- ticket.customer (access to all ticket with customer\_id \*\* current\_user.id || 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",
    ...
  },
]
```

## 29.3 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",
  ...
}
```

## 29.4 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",
  ...
}
```

Response:

```
Status: 201 Created

{
  "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:

```
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:

```
POST /api/v1/tickets

{
  "title": "Help me!",
  "group": "Users",
  "article": {
    "content_type": "text/html",
    "subject": "some subject",
```

(continues on next page)

(continued from previous page)

```

    "body": "<b>some</b> message with inline image <img src=\"data:image/jpeg;base64,
↪ABCDEF==\">"
  },
  "note": "some note",
  ...
}

```

If you want to use or create a customer 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",
  ...
}

```

## 29.5 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"
  },
  ...
}

```

Response:

```

Status: 200 Ok

{
  "id": 123,
  "title": "Help me!",

```

(continues on next page)

(continued from previous page)

```
"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:

```
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,
↵ABCDEF==\">"
  },
  "note": "some note",
  ...
}
```



## 29.6 Delete

Required permission:

- admin

Request:

```
DELETE /api/v1/tickets/{id}
```

Response:

```
Status: 200 Ok
```

```
{}
```



### 30.1 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:

```
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,
```

(continues on next page)

(continued from previous page)

```
"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"
},
]
```

## 30.2 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_states/{id}
```

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"
}
```

## 30.3 Create

Required permission:

- admin.object

Request:

```
POST /api/v1/ticket_states

{
  "name": "Ticket State 1",
  "state_type_id": 1,
  "next_state_id": null,
  "ignore_escalation": true,
```

(continues on next page)

(continued from previous page)

```
"active": true,  
"note": "some note"  
}
```

**Response:**

```
Status: 201 Created  
  
{  
  "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"  
}
```

## 30.4 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"  
}
```

**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"  
}
```

## 30.5 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
```

```
{}
```

### 31.1 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:

```
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"
  }
]
```

(continues on next page)

(continued from previous page)

```
  },  
]  
]
```

## 31.2 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"  
}
```

## 31.3 Create

Required permission:

- admin.object

Request:

```
POST /api/v1/ticket_priorities  
  
{  
  "name": "Ticket Priority 1",  
  "active": true,  
  "note": "some note"  
}
```

Response:

```
Status: 201 Created  
  
{  
  "id": 123,  
  "name": "Ticket Priority 1",  
}
```

(continues on next page)



(continued from previous page)

```
"active": true,  
"note": "some note",  
"updated_at": "2016-08-16T07:55:42.119Z",  
"created_at": "2016-08-16T07:55:42.119Z"  
}
```

## 31.4 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"  
}
```

## 31.5 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  
  
{}
```



## 32.1 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:

```
GET /api/v1/ticket_articles/by_ticket/{ticketId}
```

Response:

```
Status: 200 Ok

[
  {
    "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,
    ...
    "updated_at": "2016-08-15T07:55:42.119Z",
    "created_at": "2016-08-15T07:55:42.119Z"
  },
  {
```

(continues on next page)

(continued from previous page)

```
"id": 4,
"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,
...
"updated_at": "2016-08-16T07:55:42.119Z",
"created_at": "2016-08-16T07:55:42.119Z"
},
]
```

## 32.2 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><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",

```

(continues on next page)

(continued from previous page)

```
    "preferences": {
      "Mime-Type": "text/plain"
    }
  ],
  "created_at": "2016-10-19T10:07:12.011Z",
  "updated_at": "2017-01-18T12:45:53.420Z",
  ...
}
```

## 32.3 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:

```
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"
      }
    },
    {
      "id": 124,
      "filename": "some_file2.txt",
      "preferences": {
        "Mime-Type": "text/plain"
      }
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    }
  }
],
"created_at": "2016-10-19T10:07:12.011Z",
"updated_at": "2017-01-18T12:45:53.420Z",
...
}

```

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:

```

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,
↵ABCDEF==\">"
  "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"
  "attachments": [
    {
      "id": 123,
      "filename": "44.262871107@zammad.example.com",
      "preferences": {
        "Mime-Type": "image/jpeg",
        "Content-ID": "44.262871107@zammad.example.com",
        "Content-Disposition": "inline"
      }
    }
  ]
},
"created_at": "2016-10-19T10:07:12.011Z",

```

(continues on next page)

(continued from previous page)

```
"updated_at": "2017-01-18T12:45:53.420Z",  
...  
}
```

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:

```
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,  
↪ ABCDEFG==\">",  
  "content_type": "text/html",  
  "sender": "Customer",  
  "type": "phone",  
  "internal": false,  
  "time_unit": "12"  
}
```



### 33.1 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,
```

(continues on next page)

(continued from previous page)

```
"created_at": "2016-08-16T07:55:47.119Z",
"created_at_id": 123
},
{
  "id": 125,
  "o_id": 630,
  "object": "Ticket",
  "type": "create",
  "seen": false,
  "updated_at": "2016-08-16T07:57:49.119Z",
  "updated_by_id": 123,
  "created_at": "2016-08-16T07:57:49.119Z",
  "created_at_id": 123
},
]
```

## 33.2 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
}
```

## 33.3 Update

Required permission:

- admin.object

Request:

```
PUT /api/v1/online_notifications/{id}
```

```
{
```

(continues on next page)

(continued from previous page)

```
"seen": true,  
}
```

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  
}
```

## 33.4 Delete

Required permission:

- authenticated user (content of notifications depends on user permissions)

Request:

```
DELETE /api/v1/online_notifications/{id}
```

Response:

```
Status: 200 Ok  
  
{}
```

## 33.5 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  
  
{}
```



## 34.1 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":{
  },
  },
],
```

(continues on next page)

(continued from previous page)

```
"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"
},
...
]
```

## 34.2 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",
```

(continues on next page)

(continued from previous page)

```

"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"
}

```

### 34.3 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": "anzeigel",
      "wert2": "anzeigel2"
    }
  },
  "screens": {
    "create_middle": {
      "Customer": {
        "shown": true,
        "item_class": "column"
      },
      "Agent": {
        "shown": true,
        "item_class": "column"
      }
    },
    "edit": {
      "Customer": {
        "shown": true
      },
      "Agent": {
        "shown": true
      }
    }
  }
}
```

## 34.4 Update

### Required permission:

- admin (access to admin interface)

### Request:

```
PUT /api/v1/object_manager_attributes/:id
```

### Response:

```
Status: 200 Ok
```

(continues on next page)



(continued from previous page)

```
{
  "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"
}
```

## 34.5 Execute Database Migrations

Required permission:

- admin (access to admin interface)

Request:

```
POST /api/v1/object_manager_attributes_execute_migrations
```

Response:

```
Status: 200 Ok
```

```
{}
```

### 35.1 List

Required permission:

- ticket.agent or admin.tag

Request:

```
GET /api/v1/tags?object=Ticket&o_id=10
```

Response:

```
Status: 200 Ok
```

```
{
  "tags": [
    "tag 1",
    "tag 2",
    "tag 3"
  ]
}
```

### 35.2 Search

Required permission:

- ticket.agent or admin.tag

Request:

```
GET /api/v1/tag_search?term=tag
```

Response:

---

```
Status: 200 Ok
```

```
[
  {
    "id": 7,
    "value": "tag 1"
  },
  {
    "id": 8,
    "value": "tag 2"
  },
  {
    "id": 9,
    "value": "tag 3"
  }
]
```

### 35.3 Add

Required permission:

- ticket.agent or admin.tag

Request:

```
GET /api/v1/tags/add?object=Ticket&o_id=10&item=tag+4
```

Response:

```
Status: 200 Ok
```

```
true
```

### 35.4 Remove

Required permission:

- ticket.agent or admin.tag

Request:

```
GET /api/v1/tags/remove?object=Ticket&o_id=10&item=tag+4
```

Response:

```
Status: 200 Ok
```

```
true
```

### 35.5 Admin - List

Required permission:

- admin.tag

**Request:**

```
GET /api/v1/tag_list
```

**Response:**

```
Status: 200 Ok
```

```
[
  {
    "id": 7,
    "name": "tag 1",
    "count": 1
  },
  {
    "id": 8,
    "name": "tag 2",
    "count": 1
  },
  {
    "id": 9,
    "name": "tag 3",
    "count": 1
  },
  {
    "id": 11,
    "name": "tag 4",
    "count": 0
  },
  {
    "id": 6,
    "name": "test",
    "count": 0
  }
]
```

## 35.6 Admin - Create

**Required permission:**

- admin.tag

**Request:**

```
POST /api/v1/tag_list
```

```
{
  name: "tag 5"
}
```

**Response:**

```
Status: 200 Ok
```

```
{}
```

## 35.7 Admin - Rename

Required permission:

- admin.tag

Request:

```
PUT /api/v1/tag_list/{id}

{
  id: 6,
  name: "tag 5"
}
```

Response:

```
Status: 200 Ok

{}
```

## 35.8 Admin - Delete

Required permission:

- admin.tag

Request:

```
DELETE /api/v1/tag_list/{id}
```

Response:

```
Status: 200 Ok

{}
```

---

## User Access Token

---

### 36.1 List

Required permission:

- user\_preferences.access\_token

Request:

```
GET /api/v1/user_access_token
```

Response:

```
Status: 200 Ok

{
  "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,
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```
    "created_at": "2018-07-11T08:18:56.947Z"
  }
],
"permissions": [
  {
    id: 1,
    name: "admin",
    note: "Admin Interface",
    preferences: {},
    active: true,
    ...
  },
  {
    id: 2,
    name: "admin.user",
    note: "Manage Users",
    preferences: {},
    active: true,
    ...
  },
  ...
]
```

## 36.2 Create

Required permission:

- user\_preferences.access\_token

Request:

```
POST /api/v1/user_access_token

{
  "label": "some test",
  "permission": ["cti.agent", "ticket.agent"],
  "expires_at": null
}
```

Response:

```
Status: 200 Ok

{
  "name": "new_token_only_shown_once"
}
```

## 36.3 Delete

Required permission:

- user\_preferences.access\_token



**Request:**

```
PUT /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.

## 37.1 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



### 38.1 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.

### 38.2 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

### 38.3 Events

Zammad supports the following three events (newCall, hangup and answer) in version 2.x.

**Event: newCall**

At-tribute	Description
event	"newCall"
from	The calling number (e.g. "493055571600" or "anonymous")
to	The called number (e.g. "491711234567890")
di- rec- tion	The direction of the call (either "in" or "out")
cal- Id	A unique alphanumeric identifier to match events to specific calls (max. 250 characters)
user[]	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.
queue	The queue name (e. g. helpdesk). This field is optional.

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:

Action	Description
reject	Reject call or pretend to be busy (depending on your settings in Zammad)

### Example 1: Reject call signaling busy

```
{
  "action": "reject",
  "reason": "busy"
}
```

Zammad currently supports the following responses for outgoing calls:

Action	Description
dial	To set the caller id (depending on your settings in Zammad). Number need to be in E.164 format.

### Example 1: Set custom caller id for outgoing call

```
{
  "action": "dial",
  "callerId": "493055571642",
  "number": "491711234567890"
}
```

### Event: hangup

Attribute	Description
event	“hangup”
callId	Same as in newCall-event for a specific call
cause	The cause for the hangup event (see
from	The calling number (e.g. “493055571600” or “anonymous”)
to	The called number (e.g. “491711234567890”)
direction	The direction of the call (either “in” or “out”)
answeringNumber	The number which was answering

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:

Attribute	Description
normalClearing	One of the parties hung up after the call was established.
busy	The called party was busy
cancel	The caller hung up before the called party picked up
noAnswer	The called party rejected the call (e.g. through a DND setting)
congestion	The called party could not be reached
notFound	The called number does not exist or called party is offline
forwarded	The call was forwarded to a different party

### Event: answer

Attribute	Description
event	“answer”
callId	Same as in newCall-event for a specific call
user	Name of the user who answered this call. Only incoming calls can have this parameter
from	The calling number (e.g. “492111234567” or “anonymous”)
to	The called number (e.g. “491711234567890”)
direction	The direction of the call (either “in” or “out”)
answeringNumber	The number of the answering destination. Useful when redirecting to multiple destinations

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
```





---

## Backup and Restore

---

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!

### 39.1 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.

---

### 39.2 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
```

### 39.3 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 databae 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 above requirements are met, you can continue with restoring.

### 39.4 Restore everything

```
# Change into the folder of Zammads backup-script:  
$ cd /opt/zammad/contrib/backup
```

#### 39.4.1 With menu for choosing backup date

```
# When called without arguments, Zammad will show you a list of available backups.  
$ ./zammad_restore.sh
```

#### 39.4.2 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
```

## 39.5 What to do after restoration has been completed

### 39.5.1 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)

```
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
```

### 39.5.2 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*.



---

## Configure environment variables

---

If you're using the DEB or RPM packages you can change Zammads environment variables by the following commands.

### 40.1 Configure IP

```
$ zammad config:set ZAMMAD_BIND_IP=0.0.0.0
$ systemctl restart zammad
```

### 40.2 Configure ports

Please note that you also have to reconfigure Nginx when changing the ports!

```
$ zammad config:set ZAMMAD_RAILS_PORT=3000
$ zammad config:set ZAMMAD_WEBSOCKET_PORT=6042
$ systemctl restart zammad
```

### 40.3 Application Servers

Per default one application server will get started. If you have more http requests (user sessions) you need to increase the amount of your application server. The typical problem is long waiting times in the web interface for opening or editing tickets.

```
$ zammad config:set WEB_CONCURRENCY=3
$ systemctl restart zammad
```

### 40.4 Configure Restart Command

If you need to make changes (creating objects) to Zammad, it can be necessary to restart the service. This can be done manually or automatic. If you like to use the automatic way you need to set an special environment variable.

Note: you might need to adjust the value for `APP_RESTART_CMD` if you have / need a different command to restart your Zammad on your installation.

```
$ zammad config:set APP_RESTART_CMD="systemctl restart zammad"
```

### 40.5 Configure Zammad to log to STDOUT

If you want to log to STDOUT instead of the production-logfile (`/var/log/zammad/production.log`) you can set it:

```
$ zammad config:set RAILS_LOG_TO_STDOUT=true
```

To reset this back to logfile logging run:

```
$ zammad config:set RAILS_LOG_TO_STDOUT=
```

---

**Note:** This applies to package installations: Do not set it to enabled, because we'll then unset the variable upon Update! Using `true` is update safe.

---

# CHAPTER 41

---

## Package Repo Files

---

Recently (24 Jul 2017) our packaging service provider (packager.io: <http://www.packager.io/>) improved its package distribution which makes it necessary that you update your Zammad repo file (e. g. `/etc/apt/sources.list.d/zammad.list` or `/etc/yum.repos.d/zammad.repo`) on your operating system.

If you're using an old repo file, you will not be able to update Zammad.

For more background information see: <https://blog.packager.io/posts/24-change-of-repository-urls>

Please use the following commands to update your Zammad repo file:

### 41.1 CentOS 7

```
$ sudo yum -y install epel-release wget
$ sudo wget -O /etc/yum.repos.d/zammad.repo https://dl.packager.io/srv/zammad/zammad/
↪stable/installer/el/7.repo
$ sudo yum install zammad
```

### 41.2 Debian 8

```
$ sudo apt-get install wget
$ wget -qO- https://dl.packager.io/srv/zammad/zammad/key | sudo apt-key add -
$ sudo wget -O /etc/apt/sources.list.d/zammad.list https://dl.packager.io/srv/zammad/
↪zammad/stable/installer/debian/8.repo
$ sudo apt-get update
$ sudo apt-get install zammad
```

## 41.3 Debian 9

```
$ sudo apt-get install wget
$ wget -qO- https://dl.packager.io/srv/zammad/zammad/key | sudo apt-key add -
$ sudo wget -O /etc/apt/sources.list.d/zammad.list https://dl.packager.io/srv/zammad/
↪zammad/stable/installer/debian/9.repo
$ sudo apt-get update
$ sudo apt-get install zammad
```

## 41.4 Ubuntu 16.04

```
$ sudo apt-get install wget
$ wget -qO- https://dl.packager.io/srv/zammad/zammad/key | sudo apt-key add -
$ sudo wget -O /etc/apt/sources.list.d/zammad.list https://dl.packager.io/srv/zammad/
↪zammad/stable/installer/ubuntu/16.04.repo
$ sudo apt-get update
$ sudo apt-get install zammad
```

## 41.5 Ubuntu 18.04

```
$ sudo apt-get install wget
$ wget -qO- https://dl.packager.io/srv/zammad/zammad/key | sudo apt-key add -
$ sudo wget -O /etc/apt/sources.list.d/zammad.list \
  https://dl.packager.io/srv/zammad/zammad/stable/installer/ubuntu/18.04.repo
$ sudo apt-get update
$ sudo apt-get install zammad
```

## 41.6 SLES 12

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

## 41.7 Note

If you're using an old repo file, you will get error messages like these:

```
E: Failed to fetch https://deb.packager.io/gh/zammad/zammad/dists/xenial/stable/
↪binary-amd64/Packages Writing more data than expected (7831 > 1153)
E: Some index files failed to download. They have been ignored, or old ones used.
↪instead.
```

```
Paket zammad-1.5.0-1500965473.2be861e2.centos7.x86_64.rpm not signed
```



---

## Privacy & Data Retention

---

How long does Zammad hold onto user data? How can I manage its user data retention behavior?

### 42.1 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.

## 42.2 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](https://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](https://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](https://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.