Modoboa Documentation

Release 1.9.0

Antoine Nguyen

Sep 15, 2017

Contents

1 Overview 3

2 Table of contents 5


modoboa
mail hosting made simple

CHAPTER 1

Overview

Modoboa is a mail hosting and management platform including a modern and simplified Web User Interface designed to work with Postfix and Dovecot.

It is extensible by nature and comes with a lot of additional extensions:

Name	Description	Documentation
modoboa-amavis	A frontend for Amavis	https://modoboa-amavis.readthedocs.io
modoboa-dmarc	A set of tools to use DMARC	https://github.com/modoboa/modoboa-dmarc
modoboa-imap-migration	Migrate mailboxes from an existing server using IMAP (and offlineimap)	https://github.com/modoboa/modoboa-imap-migration
modoboa-pdfcredentials	Generate PDF documents containing account credentials	https://github.com/modoboa/modoboa-pdfcredentials
modoboa-pfxadmin-migrate	A tool to migrate from Postfixadmin	https://github.com/modoboa/modoboa-pfxadmin-migrate
modoboa-postfix-autoreply	Away message editor (postfix compatible)	https://modoboa-postfix-autoreply.readthedocs.io
modoboa-radicale	A frontend for Radicale	https://modoboa-radicale.readthedocs.io
modoboa-sievefilters	A Sieve filters (rules) editor	https://modoboa-sievefilters.readthedocs.io
modoboa-stats	Graphical statistics (message traffic and more)	https://modoboa-stats.readthedocs.io
modoboa-webmail	A simple webmail	https://modoboa-webmail.readthedocs.io

CHAPTER 2

Table of contents

Installation

Recommended way

If you start from scratch and want to deploy a complete mail server, you will love the modoboa installer! It is the easiest and the quickest way to setup a fully functional server (modoboa, postfix, dovecot, amavis and more) on one machine.

Warning: For now, only Debian and CentOS based Linux distributions are supported. We do our best to improve compatibility but if you use another Linux or a UNIX system, you will have to install Modoboa *manually*.

To use it, just run the following commands in your terminal:

```
> git clone https://github.com/modoboa/modoboa-installer
> cd modoboa-installer
> sudo ./run.py <your domain>
```

Wait a few minutes and you're done o/

Manual installation

For those who need a manual installation or who just want to setup a specific part, here are the steps you must follow:

Modoboa

This section describes the installation of the web interface (a Django project).

Prepare the system

First of all, we recommand the following context:

- Use a dedicated system user
- Use a virtualenv to install the application because it will isolate it (and its dependencies) from the rest of your system

The following example illustrates how to realize this (Debian like system):

```
> sudo apt-get install python-virtualenv python-pip
> sudo useradd modoboa
> sudo -i modoboa
> virtualenv env
> source env/bin/activate
(env)> pip install -U pip
```

Modoboa depends on external tools and some of them require compilation so you need a compiler and a few C libraries. Make sure to install the following system packages according to your distribution:

Debian/Ubuntu	CentOS
build-essential, python-dev, libxml2-dev, libxslt-dev, libjpeg-dev, librrd-dev, rrdtool, libffi-dev	gcc, gcc-c++, python-devel, libxml2-devel, libxslt-devel, libjpeg-turbo-devel, rrdtool-devel, rrdtool, libffi-devel

Then, install Modoboa:

```
(env)> pip install modoboa
```

Database

Warning: This documentation does not cover the installation of a database server but only the setup of a functional database that Modoboa will use.

Thanks to Django, Modoboa is compatible with the following databases:

- PostgreSQL
- MySQL / MariaDB
- SQLite

Since the last one does not require particular actions, only the first two ones are described.

PostgreSQL

Install the corresponding Python binding:

```
(env)> pip install psycopg2
```

Then, create a user and a database:

```
> sudo -i postgres
>
```

MySQL / MariaDB

Install the corresponding Python binding:

```
(env)> pip install MySQL-Python
```

Then, create a user and a database:

```
> mysqladmin -u root -p create modoboa
```

Deploy an instance

`modoboa-admin.py`, a command line tool, lets you deploy a *ready-to-use* Modoboa site using only one instruction:

```
(env)> modoboa-admin.py deploy instance --collectstatic \
         --domain <hostname of your server> --dburl default:database-url
```

Note: You can install additional extensions during the deploy process. To do so, use the `--extensions` option which accepts a list of names as argument (`--extensions ext1 ext2 ...`). If you want to install all extensions, just use the `all` keyword like this `--extensions all`.

If you choose to install extensions one at a time, you will have to add their names in settings.py to `MODOBOA_APPS`. Also ensure that you have the line `from modoboa_amavis.settings import *` at the end of this file.

The list of available extensions can be found on the *index page*. Instructions to install them are available on each extensions page.

Note: You can specify more than one database connection using the `--dburl` option. Multiple connections are differentiated by a prefix.

The primary connection must use the `default:` prefix (as shown in the example above). For the amavis extension, use the `amavis:` prefix. For example: `--dburl default:<database url> amavis:<database url>`.

A database url should meet the following syntax `<mysql|postgres>://[user:pass@][host:port]/dbname` **OR** `sqlite:////full/path/to/your/database/file.sqlite`.

The command will ask you a few questions, answer them and you're done.

If you need a **silent installation** (e.g. if you're using Salt-Stack, Ansible or whatever), it's possible to supply the database credentials as commandline arguments.

You can consult the complete option list by running the following command:

```
$ modoboa-admin.py help deploy
```

Cron jobs

A few recurring jobs must be configured to make Modoboa works as expected.

Create a new file, for example `/etc/cron.d/modoboa` and put the following content inside:

```
#
# Modoboa specific cron jobs
#
PYTHON=<PATH TO PYTHON BINARY>
INSTANCE=<PATH TO MODOBOA INSTANCE>

# Operations on mailboxes
*       *       *       *       *       vmail   $PYTHON $INSTANCE/manage.py handle_
↪mailbox_operations

# Sessions table cleanup
0       0       *       *       *       root    $PYTHON $INSTANCE/manage.py␣
↪clearsessions

# Logs table cleanup
0       0       *       *       *       root    $PYTHON $INSTANCE/manage.py cleanlogs

# Logs parsing
*/5     *       *       *       *       root    $PYTHON $INSTANCE/manage.py logparser␣
↪&> /dev/null

# DNSBL checks
*/30    *       *       *       *       root    $PYTHON $INSTANCE/manage.py modo␣
↪check_mx

# Public API communication
0       *       *       *       *       root    $PYTHON $INSTANCE/manage.py␣
↪communicate_with_public_api
```

Now you can continue to the *Web server* section.

Web server

Note: The following instructions are meant to help you get your site up and running quickly. However it is not possible for the people contributing documentation to Modoboa to test every single combination of web server, wsgi server, distribution, etc. So it is possible that **your** installation of uwsgi or nginx or Apache or what-have-you works differently. Keep this in mind.

Apache2

First, make sure that `mod_wsgi` is installed on your server.

Create a new virtualhost in your Apache configuration and put the following content inside:

```
<VirtualHost *:80>
  ServerName <your value>
  DocumentRoot <modoboa_instance_path>

  Alias /media/ <modoboa_instance_path>/media/
  <Directory <modoboa_instance_path>/media>
    Order deny,allow
    Allow from all
  </Directory>
```

```
  Alias /sitestatic/ <modoboa_instance_path>/sitestatic/
  <Directory <modoboa_instance_path>/sitestatic>
    Order deny,allow
    Allow from all
  </Directory>

  WSGIScriptAlias / <modoboa_instance_path>/<instance_name>/wsgi.py

  # Pass Authorization header to enable API usage:
  WSGIPassAuthorization On
</VirtualHost>
```

This is just one possible configuration.

To use mod_wsgi daemon mode, add the two following directives just under `WSGIScriptAlias`:

```
WSGIDaemonProcess example.com python-path=<modoboa_instance>:<virtualenv path>/lib/
↪python2.7/site-packages
WSGIProcessGroup example.com
```

Replace values between <> with yours. If you don't use a virtualenv, just remove the last part of the `WSGIDaemonProcess` directive.

Note: You will certainly need more configuration in order to launch Apache.

Now, you can go the *Dovecot* section to continue the installation.

Nginx

This section covers two different ways of running Modoboa behind Nginx using a WSGI application server. Choose the one you prefer between Green Unicorn or uWSGI.

In both cases, you'll need to download and install nginx.

Green Unicorn

Firstly, Download and install gunicorn. Then, use the following sample gunicorn configuration (create a new file named `gunicorn.conf.py` inside Modoboa's root dir):

```
backlog = 2048
bind = "unix:/var/run/gunicorn/modoboa.sock"
pidfile = "/var/run/gunicorn/modoboa.pid"
daemon = True
debug = False
workers = 2
logfile = "/var/log/gunicorn/modoboa.log"
loglevel = "info"
```

To start gunicorn, execute the following commands:

```
$ cd <modoboa dir>
$ gunicorn -c gunicorn.conf.py <modoboa dir>.wsgi:application
```

Now the nginx part. Just create a new virtual host and use the following configuration:

```
upstream modoboa {
      server      unix:/var/run/gunicorn/modoboa.sock fail_timeout=0;
}

server {
      listen 443 ssl;
      ssl on;
      keepalive_timeout 70;

      server_name <host fqdn>;
      root <modoboa_instance_path>;

      access_log  /var/log/nginx/<host fqdn>.access.log;
      error_log /var/log/nginx/<host fqdn>.error.log;

      ssl_certificate     <ssl certificate for your site>;
      ssl_certificate_key <ssl certificate key for your site>;

      location /sitestatic/ {
              autoindex on;
      }

      location /media/ {
              autoindex on;
      }

      location / {
              proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
              proxy_set_header Host $http_host;
              proxy_redirect off;
              proxy_set_header X-Forwarded-Protocol ssl;
              proxy_pass http://modoboa;
      }
}
```

If you do not plan to use SSL then change the listen directive to `listen 80;` and delete each of the following directives:

```
ssl on;
keepalive_timeout 70;
ssl_certificate     <ssl certificate for your site>;
ssl_certificate_key <ssl certificate key for your site>;
proxy_set_header X-Forwarded-Protocol ssl;
```

If you do plan to use SSL, you'll have to generate a certificate and a key. This article contains information about how to do it.

Paste this content to your configuration (replace values between <> with yours) and restart nginx.

Now, you can go the *Dovecot* section to continue the installation.

uWSGI

The following setup is meant to get you started quickly. You should read the documentation of both nginx and uwsgi to understand how to optimize their configuration for your site.

The Django documentation includes the following warning regarding uwsgi:

> **Warning:** Some distributions, including Debian and Ubuntu, ship an outdated version of uWSGI that does not conform to the WSGI specification. Versions prior to 1.2.6 do not call close on the response object after handling a request. In those cases the request_finished signal isn't sent. This can result in idle connections to database and memcache servers.

Use uwsgi 1.2.6 or newer. If you do not, you *will* run into problems. Modoboa will fail in obscure ways.

To use this setup, first download and install uwsgi.

Here is a sample nginx configuration:

```
server {
    listen 443 ssl;
    ssl on;
    keepalive_timeout 70;

    server_name <host fqdn>;
    root <modoboa's settings dir>;

    ssl_certificate     <ssl certificate for your site>;
    ssl_certificate_key <ssl certificate key for your site>;

    access_log  /var/log/nginx/<host fqdn>.access.log;
    error_log /var/log/nginx/<host fqdn>.error.log;

    location <modoboa's root url>/sitestatic/ {
            autoindex on;
            alias <location of sitestatic on your file system>;
    }

    # Whether or not Modoboa uses a media directory depends on how
    # you configured Modoboa. It does not hurt to have this.
    location <modoboa's root url>/media/ {
            autoindex on;
            alias <location of media on your file system>;
    }

    # This denies access to any file that begins with
    # ".ht". Apache's .htaccess and .htpasswd are such files. A
    # Modoboa installed from scratch would not contain any such
    # files, but you never know what the future holds.
    location ~ /\.ht {
        deny all;
    }

    location <modoba's root url>/ {
        include uwsgi_params;
        uwsgi_pass <uwsgi port>;
        uwsgi_param UWSGI_SCRIPT <modoboa instance name>.wsgi:application;
        uwsgi_param UWSGI_SCHEME https;
    }
}
```

`<modoboa instance name>` must be replaced by the value you used when you deployed your instance.

If you do not plan to use SSL then change the listen directive to `listen 80;` and delete each of the following

directives:

```
ssl on;
keepalive_timeout 70;
ssl_certificate     <ssl certificate for your site>;
ssl_certificate_key <ssl certificate key for your site>;
uwsgi_param UWSGI_SCHEME https;
```

If you do plan to use SSL, you'll have to generate a certificate and a key. This article contains information about how to do it.

Make sure to replace the <...> in the sample configuration with appropriate values. Here are some explanations for the cases that may not be completely self-explanatory:

<modoboa's settings dir> Where Modoboa's `settings.py` resides. This is also where the `sitestatic` and `media` directories reside.

<modoboa's root url> This is the URL which will be the root of your Modoboa site at your domain. For instance, if your Modoboa installation is reachable at at `https://foo/modoboa` then `<modoboa's root url>` is `/modoboa`. In this case you probably also have to set the `alias` directives to point to where Modoboa's sitestatic and media directories are because otherwise nginx won't be able to find them.

If Modoboa is at the root of your domain, then `<modoboa root url>` is an empty string and can be deleted from the configuration above. In this case, you probably do not need the `alias` directives.

<uwsgi port> The location where uwsig is listening. It could be a unix domain socket or an address:port combination. Ubuntu configures uwsgi so that the port is:

```
unix:/run/uwsgi/app/<app name>/socket
```

where `<app name>` is the name of the application.

Your uwsgi configuration should be:

```
[uwsgi]
# Not needed when using uwsgi from pip
# plugins = python
chdir = <modoboa's top dir>
module = <name>.wsgi:application
master = true
harakiri = 60
processes = 4
vhost = true
no-default-app = true
```

The plugins directive should be turned on if you use a uwsgi installation that requires it. If uwsgi was installed from pip, it does not require it. In the configuration above:

<modoboa's top dir> The directory where `manage.py` resides. This directory is the parent of `<modoboa's settings dir>`

<name> The name that you passed to `modoboa-admin.py deploy` when you created your Modoboa instance.

Now, you can go the *Dovecot* section to continue the installation.

Dovecot

Modoboa requires Dovecot 2+ to work so the following documentation is suitable for this combination.

In this section, we assume dovecot's configuration resides in `/etc/dovecot`, all pathes will be relative to this directory.

Mailboxes

First, edit the `conf.d/10-mail.conf` and set the `mail_location` variable:

```
# maildir
mail_location = maildir:~/.maildir
```

Then, edit the `inbox` namespace and add the following lines:

```
inbox = yes

mailbox Drafts {
  auto = subscribe
  special_use = \Drafts
}
mailbox Junk {
  auto = subscribe
  special_use = \Junk
}
mailbox Sent {
  auto = subscribe
  special_use = \Sent
}
mailbox Trash {
  auto = subscribe
  special_use = \Trash
}
```

With dovecot 2.1+, it ensures all the special mailboxes will be automaticaly created for new accounts.

For dovecot 2.0 and older, use the autocreate plugin.

Operations on the file system

Warning: Modoboa needs to access the `dovecot` binary to check its version. To find the binary path, we use the `which` command first and then try known locations (`/usr/sbin/dovecot` and `/usr/local/sbin/dovecot`). If you installed dovecot in a custom location, please tell us where the binary is by using the `DOVECOT_LOOKUP_PATH` setting (see `settings.py`).

Three operation types are considered:

1. Mailbox creation
2. Mailbox renaming
3. Mailbox deletion

The first one is managed by Dovecot. The last two ones may be managed by Modoboa if it can access the file system where the mailboxes are stored (see *General parameters* to activate this feature).

Those operations are treated asynchronously by a cron script. For example, when you rename an e-mail address through the web UI, the associated mailbox on the file system is not modified directly. Instead of that, a *rename*

order is created for this mailbox. The mailbox will be considered unavailable until the order is executed (see *Postfix configuration*).

Edit the crontab of the user who owns the mailboxes on the file system:

```
$ crontab -u <user> -e
```

And add the following line inside:

```
* * * * * python <modoboa_site>/manage.py handle_mailbox_operations
```

Warning: The cron script must be executed by the system user owning the mailboxes.

Warning: The user running the cron script must have access to the `settings.py` file of the modoboa instance.

The result of each order is recorded into Modoboa's log. Go to *Modoboa* > *Logs* to consult them.

Authentication

To make the authentication work, edit the `conf.d/10-auth.conf` and uncomment the following line at the end:

```
#!include auth-system.conf.ext
!include auth-sql.conf.ext
#!include auth-ldap.conf.ext
#!include auth-passwdfile.conf.ext
#!include auth-checkpassword.conf.ext
#!include auth-vpopmail.conf.ext
#!include auth-static.conf.ext
```

Then, edit the `conf.d/auth-sql.conf.ext` file and add the following content inside:

```
passdb sql {
  driver = sql
  # Path for SQL configuration file, see example-config/dovecot-sql.conf.ext
  args = /etc/dovecot/dovecot-sql.conf.ext
}

userdb sql {
  driver = sql
  args = /etc/dovecot/dovecot-sql.conf.ext
}
```

Make sure to activate only one backend (per type) inside your configuration (just comment the other ones).

Edit the `dovecot-sql.conf.ext` and modify the configuration according to your database engine.

MySQL users

```
driver = mysql

connect = host=<mysqld socket> dbname=<database> user=<user> password=<password>
```

```
default_pass_scheme = CRYPT

password_query = SELECT email AS user, password FROM core_user WHERE email='%Lu' and␣
↪is_active=1

 user_query = SELECT '<mailboxes storage directory>/%Ld/%Ln' AS home, <uid> as uid,
↪<gid> as gid, concat('*:bytes=', mb.quota, 'M') AS quota_rule FROM admin_mailbox mb␣
↪INNER JOIN admin_domain dom ON mb.domain_id=dom.id WHERE mb.address='%Ln' AND dom.
↪name='%Ld'

iterate_query = SELECT email AS user FROM core_user
```

PostgreSQL users

```
driver = pgsql

connect = host=<postgres socket> dbname=<database> user=<user> password=<password>

default_pass_scheme = CRYPT

password_query = SELECT email AS user, password FROM core_user u INNER JOIN admin_
↪mailbox mb ON u.id=mb.user_id INNER JOIN admin_domain dom ON mb.domain_id=dom.id␣
↪WHERE u.email='%Lu' AND u.is_active AND dom.enabled

user_query = SELECT '<mailboxes storage directory>/%Ld/%Ln' AS home, <uid> as uid,
↪<gid> as gid, '*:bytes=' || mb.quota || 'M' AS quota_rule FROM admin_mailbox mb␣
↪INNER JOIN admin_domain dom ON mb.domain_id=dom.id WHERE mb.address='%Ln' AND dom.
↪name='%Ld'

iterate_query = SELECT email AS user FROM core_user
```

SQLite users

```
driver = sqlite

connect = <path to the sqlite db file>

default_pass_scheme = CRYPT

password_query = SELECT email AS user, password FROM core_user u INNER JOIN admin_
↪mailbox mb ON u.id=mb.user_id INNER JOIN admin_domain dom ON mb.domain_id=dom.id␣
↪WHERE u.email='%Lu' AND u.is_active=1 AND dom.enabled=1

user_query = SELECT '<mailboxes storage directory>/%Ld/%Ln' AS home, <uid> as uid,
↪<gid> as gid, ('*:bytes=' || mb.quota || 'M') AS quota_rule FROM admin_mailbox mb␣
↪INNER JOIN admin_domain dom ON mb.domain_id=dom.id WHERE mb.address='%Ln' AND dom.
↪name='%Ld'

iterate_query = SELECT email AS user FROM core_user
```

Note: Replace values between <> with yours.

LMTP

Local Mail Transport Protocol is used to let Postfix deliver messages to Dovecot.

First, make sure the protocol is activated by looking at the `protocols` setting (generally inside `dovecot.conf`). It should be similar to the following example:

```
protocols = imap pop3 lmtp
```

Then, open the `conf.d/10-master.conf`, look for `lmtp` service definition and add the following content inside:

```
service lmtp {
  # stuff before
  unix_listener /var/spool/postfix/private/dovecot-lmtp {
    mode = 0600
    user = postfix
    group = postfix
  }
  # stuff after
}
```

We assume here that Postfix is *chrooted* within `/var/spool/postfix`.

Finally, open the `conf.d/20-lmtp.conf` and modify it as follows:

```
protocol lmtp {
  postmaster_address = postmaster@<domain>
  mail_plugins = $mail_plugins quota sieve
}
```

Replace `<domain>` by the appropriate value.

Note: If you don't plan to apply quota or to use filters, just adapt the content of the `mail_plugins` setting.

Quota

Modoboa lets adminstrators define per-domain and/or per-account limits (quota). It also lists the current quota usage of each account. Those features require Dovecot to be configured in a specific way.

Inside `conf.d/10-mail.conf`, add the `quota` plugin to the default activated ones:

```
mail_plugins = quota
```

Inside `conf.d/10-master.conf`, update the `dict` service to set proper permissions:

```
service dict {
  # If dict proxy is used, mail processes should have access to its socket.
  # For example: mode=0660, group=vmail and global mail_access_groups=vmail
  unix_listener dict {
    mode = 0600
    user = <user owning mailboxes>
    #group =
  }
}
```

Inside `conf.d/20-imap.conf`, activate the `imap_quota` plugin:

```
protocol imap {
  # ...

  mail_plugins = $mail_plugins imap_quota

  # ...
}
```

Inside `dovecot.conf`, activate the quota SQL dictionary backend:

```
dict {
  quota = <driver>:/etc/dovecot/dovecot-dict-sql.conf.ext
}
```

Inside `conf.d/90-quota.conf`, activate the *quota dictionary* backend:

```
plugin {
  quota = dict:User quota::proxy::quota
}
```

It will tell Dovecot to keep quota usage in the SQL dictionary.

Finally, edit the `dovecot-dict-sql.conf.ext` file and put the following content inside:

```
connect = host=<db host> dbname=<db name> user=<db user> password=<password>
# SQLite users
# connect = /path/to/the/database.db

map {
  pattern = priv/quota/storage
  table = admin_quota
  username_field = username
  value_field = bytes
}
map {
  pattern = priv/quota/messages
  table = admin_quota
  username_field = username
  value_field = messages
}
```

PostgreSQL users

Database schema update

The `admin_quota` table is created by Django but unfortunately it doesn't support `DEFAULT` constraints (it only simulates them when the ORM is used). As PostgreSQL is a bit strict about constraint violations, you must execute the following query manually:

```
db=> ALTER TABLE admin_quota ALTER COLUMN bytes SET DEFAULT 0;
db=> ALTER TABLE admin_quota ALTER COLUMN messages SET DEFAULT 0;
```

Trigger

As indicated on Dovecot's wiki, you need a trigger to properly update the quota.

A working copy of this trigger is available on Modoboa's website.

Download this file and copy it on the server running postgres. Then, execute the following commands:

```
$ su - postgres
$ psql [modoboa database] < /path/to/modoboa_postgres_trigger.sql
$ exit
```

Replace `[modoboa database]` by the appropriate value.

Forcing recalculation

For existing installations, *Dovecot* (> 2) offers a command to recalculate the current quota usages. For example, if you want to update all usages, run the following command:

```
$ doveadm quota recalc -A
```

Be carefull, it can take a while to execute.

ManageSieve/Sieve

Modoboa lets users define filtering rules from the web interface. To do so, it requires *ManageSieve* to be activated on your server.

Inside `conf.d/20-managesieve.conf`, make sure the following lines are uncommented:

```
protocols = $protocols sieve

service managesieve-login {
  # ...
}

service managesieve {
  # ...
}

protocol sieve {
  # ...
}
```

Messages filtering using Sieve is done by the LDA.

Inside `conf.d/15-lda.conf`, activate the `sieve` plugin like this:

```
protocol lda {
  # Space separated list of plugins to load (default is global mail_plugins).
  mail_plugins = $mail_plugins sieve
}
```

Finally, configure the `sieve` plugin by editing the `conf.d/90-sieve.conf` file. Put the follwing caontent inside:

```
plugin {
  # Location of the active script. When ManageSieve is used this is actually
  # a symlink pointing to the active script in the sieve storage directory.
  sieve = ~/.dovecot.sieve

  #
  # The path to the directory where the personal Sieve scripts are stored. For
  # ManageSieve this is where the uploaded scripts are stored.
  sieve_dir = ~/sieve
}
```

Restart Dovecot.

Now, you can go to the *Postfix* section to finish the installation.

Last-login tracking

To update the `last_login` attribute of an account after a succesful IMAP or POP3 login, you can configure a post-login script.

Open `conf.d/10-master.conf` add the following configuration (`imap` and `pop3` services are already defined, you just need to update them):

```
service imap {
  executable = imap postlogin
}

service pop3 {
  executable = pop3 postlogin
}

service postlogin {
  executable = script-login /usr/local/bin/postlogin.sh
  user = modoboa
  unix_listener postlogin {
  }
}
```

Then, you must create a script named `/usr/local/bin/postlogin.sh`. According to your database engine, the content will differ.

PostgreSQL

```
#!/bin/sh

psql -c "UPDATE core_user SET last_login=now() WHERE username='$USER'" > /dev/null

exec "$@"
```

MySQL

```
#!/bin/sh

DBNAME=XXX
DBUSER=XXX
DBPASSWORD=XXX

echo "UPDATE core_user SET last_login=now() WHERE username='$USER'" | mysql -u
↪$DBUSER -p$DBPASSWORD $DBNAME

exec "$@"
```

Postfix

This section gives an example about building a simple virtual hosting configuration with *Postfix*. Refer to the official documentation for more explanation.

Map files

You first need to create configuration files (or map files) that will be used by Postfix to lookup into Modoboa tables.

To automaticaly generate the requested map files and store them in a directory, run the following command:

```
> cd <modoboa_instance_path>
> python manage.py generate_postfix_maps --destdir <directory>
```

`<directory>` is the directory where the files will be stored. Answer the few questions and you're done.

Configuration

Use the following configuration in the `/etc/postfix/main.cf` file (this is just one possible configuration):

```
# Stuff before
virtual_transport = lmtp:unix:private/dovecot-lmtp

relay_domains =
virtual_mailbox_domains = <driver>:/etc/postfix/sql-domains.cf
virtual_alias_domains = <driver>:/etc/postfix/sql-domain-aliases.cf
virtual_alias_maps = <driver>:/etc/postfix/sql-aliases.cf

relay_domains = <driver>:/etc/postfix/sql-relaydomains.cf
transport_maps =
    <driver>:/etc/postfix/sql-spliteddomains-transport.cf
    <driver>:/etc/postfix/sql-relaydomains-transport.cf

smtpd_recipient_restrictions =
      # ...
      check_recipient_access
          <driver>:/etc/postfix/sql-maintain.cf
          <driver>:/etc/postfix/sql-relay-recipient-verification.cf
      permit_mynetworks
      reject_unauth_destination
      reject_unverified_recipient
      # ...
```

```
smtpd_sender_login_maps =
      <driver>:/etc/postfix/sql-sender-login-mailboxes.cf
      <driver>:/etc/postfix/sql-sender-login-aliases.cf
      <driver>:/etc/postfix/sql-sender-login-mailboxes-extra.cf

smtpd_sender_restrictions =
      reject_sender_login_mismatch

# Stuff after
```

Replace `<driver>` by the name of the database you use.

Restart Postfix.

Extensions

Only few commands are needed to add a new extension to your setup.

In case you use a dedicated user and/or a virtualenv, do not forget to use them:

```
> sudo -u <modoboa_user> -i
> source <virtuenv_path>/bin/activate
```

Then, run the following commands:

```
> pip install <EXTENSION>==<VERSION>
> cd <modoboa_instance_dir>
> python manage.py migrate
> python manage.py collectstatic
```

Then, restart your web sever.

Upgrade

Modoboa

Warning: The new version you are going to install may need to modify your database. Before you start, make sure to backup everything!

Most of the time, upgrading your installation to a newer Modoboa version only requires a few actions. In every case, you will need to apply the general procedure first and then check if the version you are installing requires specific actions.

In case you use a dedicated user and/or a virtualenv, do not forget to use them:

```
> sudo -u <modoboa_user> -i
> source <virtuenv_path>/bin/activate
```

Then, run the following commands:

```
> pip install modoboa==<VERSION>
> cd <modoboa_instance_dir>
```

```
> python manage.py migrate
> python manage.py collectstatic
```

Once done, check if the version you are installing requires *Specific instructions*.

Finally, restart your web server.

Sometimes, you might need to upgrade postfix map files too. To do so, just run the `generate_postfix_maps` command on the same directory than the one used for installation (`/etc/postfix` by default).

Make sure to use root privileges and run the following command:

```
> python manage.py generate_postfix_maps --destdir <directory>
```

Then, reload postfix.

Extensions

If a new version is available for an extension you're using, it is recommanded to install it. Upgrading an extensions is pretty and the procedure is almost the same than the one used for Modoboa.

In case you use a dedicated user and/or a virtualenv, do not forget to use them:

```
> sudo -i <modoboa_user>
> source <virtuenv_path>/bin/activate
```

Then, run the following commands:

```
> pip install <EXTENSION>==<VERSION>
> cd <modoboa_instance_dir>
> python manage.py migrate
> python manage.py collectstatic
```

Finally, restart your web server.

It is a generic upgrade procedure which will be enough most of the time but it is generally a good idea to check the associated documentation.

Specific instructions

1.9.0

If you want to manage inactive accounts, look at *Cleaning inactive accounts*.

1.8.3

Edit the `settings.py` file and replace the following line:

```
BASE_DIR = os.path.dirname(os.path.dirname(__file__))
```

by:

```
BASE_DIR = os.path.realpath(os.path.dirname(os.path.dirname(__file__)))
```

1.8.0

Modoboa now relies on Django's builtin password validation system to validate user passwords, instead of `django-passwords`.

Remove `django-passwords` from your system:

```
> sudo -u <modoboa_user> -i
> source <virtuenv_path>/bin/activate
> pip uninstall django-passwords
```

Edit the `settings.py` file and remove the following content:

```
# django-passwords

PASSWORD_MIN_LENGTH = 8

PASSWORD_COMPLEXITY = {
    "UPPER": 1,
    "LOWER": 1,
    "DIGITS": 1
}
```

Add the following lines:

```
# Password validation rules
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.
↪UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
    {
        'NAME': 'modoboa.core.password_validation.ComplexityValidator',
        'OPTIONS': {
            'upper': 1,
            'lower': 1,
            'digits': 1,
            'specials': 0
        }
    },
]
```

1.7.2

API documentation has evolved (because of the upgrade to Django Rest Framework 3.6) and CKeditor is now embedded by default (thanks to the `django-ckeditor` package). Some configuration changes are required.

Edit your `settings.py` file and apply the following modifications:

- Update the `INSTALLED_APPS` variable as follows:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.sites',
    'django.contrib.staticfiles',
    'reversion',
    'ckeditor',
    'ckeditor_uploader',
    'rest_framework',
    'rest_framework.authtoken',
)
```

- Update the `REST_FRAMEWORK` variable as follows:

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.TokenAuthentication',
        'rest_framework.authentication.SessionAuthentication',
    ),
}
```

- Remove the `SWAGGER_SETTINGS` variable
- Add the following content

```
# CKeditor

CKEDITOR_UPLOAD_PATH = "uploads/"

CKEDITOR_IMAGE_BACKEND = "pillow"

CKEDITOR_RESTRICT_BY_USER = True

CKEDITOR_BROWSE_SHOW_DIRS = True

CKEDITOR_ALLOW_NONIMAGE_FILES = False

CKEDITOR_CONFIGS = {
    'default': {
        'allowedContent': True,
        'toolbar': 'Modoboa',
        'width': None,
        'toolbar_Modoboa': [
            ['Bold', 'Italic', 'Underline'],
            ['JustifyLeft', 'JustifyCenter', 'JustifyRight', 'JustifyBlock'],
            ['BidiLtr', 'BidiRtl', 'Language'],
            ['NumberedList', 'BulletedList', '-', 'Outdent', 'Indent'],
            ['Undo', 'Redo'],
            ['Link', 'Unlink', 'Anchor', '-', 'Smiley'],
            ['TextColor', 'BGColor', '-', 'Source'],
            ['Font', 'FontSize'],
            ['Image', ],
            ['SpellChecker']
        ],
    },
```

```
}
```

Don't forget to run the following command:

```
> python manage.py collectstatic
```

1.7.1

If you used 1.7.0 for a fresh installation, please run the following commands:

```
> sudo -u <modoboa_user> -i
> source <virtuenv_path>/bin/activate
> cd <modoboa_instance_dir>
> python manage.py load_initial_data
```

1.7.0

This version requires Django >= 1.10 so you need to make some modifications. It also brings internal API changes which are not backward compatible so installed extensions must be upgraded too.

First of all, deactivate all installed extensions (edit the `settings.py` file and comment the corresponding lines in `MODOBOA_APPS`).

Edit the `urls.py` file of your local instance and replace its content by the following one:

```
from django.conf.urls import include, url

urlpatterns = [
    url(r'', include('modoboa.urls')),
]
```

Edit the `settings.py` and apply the following changes:

- Add `'modoboa.parameters'` to `MODOBOA_APPS`:

```
MODOBOA_APPS = (
    'modoboa',
    'modoboa.core',
    'modoboa.lib',
    'modoboa.admin',
    'modoboa.relaydomains',
    'modoboa.limits',
    'modoboa.parameters',
    # Modoboa extensions here.
)
```

- Add `'modoboa.core.middleware.LocalConfigMiddleware'` to `MIDDLEWARE_CLASSES`:

```
MIDDLEWARE_CLASSES = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.locale.LocaleMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
```

```
    'modoboa.core.middleware.LocalConfigMiddleware',
    'modoboa.lib.middleware.AjaxLoginRedirect',
    'modoboa.lib.middleware.CommonExceptionCatcher',
    'modoboa.lib.middleware.RequestCatcherMiddleware',
)
```

- Modoboa used to provide a custom authentication backend (`modoboa.lib.authbackends.SimpleBackend`) but it has been removed. Replace it as follows:

```
AUTHENTICATION_BACKENDS = (
    # Other backends before...
    'django.contrib.auth.backends.ModelBackend',
)
```

- Remove `TEMPLATE_CONTEXT_PROCESSORS` and replace it by:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.template.context_processors.i18n',
                'django.template.context_processors.media',
                'django.template.context_processors.static',
                'django.template.context_processors.tz',
                'django.contrib.messages.context_processors.messages',
                'modoboa.core.context_processors.top_notifications',
            ],
            'debug': False,
        },
    },
]
```

Run the following commands (load virtualenv if you use one):

```
> sudo -u <modoboa_user> -i
> source <virtuenv_path>/bin/activate
> cd <modoboa_instance_dir>
> python manage.py migrate
> python manage.py collectstatic
```

Finally, upgrade your extensions and reactivate them.

Name	Version
modoboa-amavis	1.1.0
modoboa-dmarc	1.0.0
modoboa-imap-migration	1.1.0
modoboa-pdfcredentials	1.1.0
modoboa-postfix-autoreply	1.2.0
modoboa-radicale	1.1.0
modoboa-sievefilters	1.1.0
modoboa-stats	1.1.0
modoboa-webmail	1.1.0

Command line shortcuts:

```
$ pip install modoboa-amavis==1.1.0
$ pip install modoboa-dmarc==1.0.0
$ pip install modoboa-imap-migration==1.1.0
$ pip install modoboa-pdfcredentials==1.1.0
$ pip install modoboa-postfix-autoreply==1.2.0
$ pip install modoboa-radicale==1.1.0
$ pip install modoboa-sievefilters==1.1.0
$ pip install modoboa-stats==1.1.0
$ pip install modoboa-webmail==1.1.0
```

And please make sure you use the latest version of the `django-versionfield2` package:

```
$ pip install -U django-versionfield2
```

Notes about quota changes and resellers

Reseller users now have a quota option in Resources tab. This is the quota that a reseller can share between all its domains.

There are two quotas for a domain in the new version:

1. Quota &
2. Default mailbox quota.

[1]. Quota: quota shared between mailboxes This quota is shared between all the mailboxes of this domain. This value cannot exceed reseller's quota and hence cannot be 0(unlimited) if reseller has finite quota.

[2]. Default mailbox quota: default quota applied to mailboxes This quota is the default quota applied to new mailboxes. This value cannot exceed Quota[1] and hence cannot be 0(unlimited) if Quota[1] is finite.

1.6.1

First of all, update postfix map files as follows:

```
> python manage.py generate_postfix_maps --destdir <path> --force-overwrite
```

Then, modify postfix's configuration as follows:

```
smtpd_sender_login_maps =
    <driver>:<path>/sql-sender-login-mailboxes.cf
    <driver>:<path>/sql-sender-login-aliases.cf
    <driver>:<path>/sql-sender-login-mailboxes-extra.cf
```

Replace <driver> and <path> by your values.

Finally, reload postfix.

This release also deprecates some internal functions. As a result, several extensions has been updated to maintain the compatibility. If you enabled the notification service, you'll find the list of available updates directly in your Modoboa console.

For the others, here is the list:

Name	Version
modoboa-amavis	1.0.10
modoboa-postfix-autoreply	1.1.7
modoboa-radicale	1.0.5
modoboa-stats	1.0.9

Command line shortcut:

```
$ pip install modoboa-amavis==1.0.10
$ pip install modoboa-postfix-autoreply==1.1.7
$ pip install modoboa-radicale==1.0.5
$ pip install modoboa-stats==1.0.9
```

1.6.0

Warning: You have to upgrade extensions due to *core.User* model attribute change (*user.group* to *user.role*). Otherwise, you will have an internal error after upgrade. In particular: modoboa-amavisd, modoboa-stats, modoboa-postfix-autoreply are concerned.

An interesting feature brougth by this version is the capability to make different checks about MX records. For example, Modoboa can query main DNSBL providers for every defined domain. With this, you will quickly know if one the domains you manage is listed or not. To activate it, add the following line to your crontab:

```
*/30 * * * * <optional_virtualenv_path/>python <modoboa_instance_dir>/manage.py modo␣
↪check_mx
```

The communication with Modoboa public API has been reworked. Instead of sending direct synchronous queries (for example to check new versions), a cron job has been added. To activate it, add the following line to your crontab:

```
0 * * * * <optional_virtualenv_path/>python <modoboa_instance_dir>/manage.py␣
↪communicate_with_public_api
```

Please also note that public API now uses TLS so you must update your configuration as follows:

```
MODOBOA_API_URL = 'https://api.modoboa.org/1/'
```

Finally, it is now possible to declare additional sender addresses on a per-account basis. You need to update your postfix configuration in order to use this functionality. Just edit the `main.cf` file and change the following parameter:

```
smtpd_sender_login_maps =
    <driver>:/etc/postfix/sql-sender-login-mailboxes.cf
    <driver>:/etc/postfix/sql-sender-login-aliases.cf
    <driver>:/etc/postfix/sql-sender-login-mailboxes-extra.cf
```

1.5.0

The API has been greatly improved and a documentation is now available. To enable it, add `'rest_framework_swagger'` to the `INSTALLED_APPS` variable in `settings.py` as follows:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.sites',
    'django.contrib.staticfiles',
    'reversion',
    'rest_framework.authtoken',
    'rest_framework_swagger',
)
```

Then, add the following content into `settings.py`, just after the `REST_FRAMEWORK` variable:

```
SWAGGER_SETTINGS = {
    "is_authenticated": False,
    "api_version": "1.0",
    "exclude_namespaces": [],
    "info": {
        "contact": "contact@modoboa.com",
        "description": ("Modoboa API, requires a valid token."),
        "title": "Modoboa API",
    }
}
```

You're done. The documentation is now available at the following address:

http://<your instance address>/docs/api/

Finally, if you find a `TEMPLATE_CONTEXT_PROCESSORS` variable in your `settings.py` file, make sure it looks like this:

```
TEMPLATE_CONTEXT_PROCESSORS = global_settings.TEMPLATE_CONTEXT_PROCESSORS + [
    'modoboa.core.context_processors.top_notifications',
]
```

1.4.0

Warning: Please make sure to use Modoboa 1.3.5 with an up-to-date database before an upgrade to 1.4.0.

Warning: Do not follow the regular upgrade procedure for this version.

Some extension have been moved back into the main repository. The main reason for that is that using Modoboa without them doesn't make sense.

First of all, you must rename the following applications listed inside the `MODOBOA_APPS` variable:

Old name	New name
modoboa_admin	modoboa.admin
modoboa_admin_limits	modoboa.limits
modoboa_admin_relaydomains	modoboa.relaydomains

Then, apply the following steps:

1. Uninstall old extensions:

```
$ pip uninstall modoboa-admin modoboa-admin-limits modoboa-admin-relaydomains
```

2. Install all extension updates using pip (check the *Modoboa* > *Information* page)
3. Manually migrate database:

```
$ cd <instance_dir>
$ python manage.py migrate auth
$ python manage.py migrate admin 0001 --fake
$ python manage.py migrate admin
$ python manage.py migrate limits 0001 --fake
$ python manage.py migrate relaydomains 0001 --fake
$ python manage.py migrate
```

4. Finally, update static files:

```
$ python manage.py collectstatic
```

This version also introduces a REST API. To enable it:

1. Add `'rest_framework.authtoken'` to the `INSTALLED_APPS` variable
2. Add the following configuration inside `settings.py`:

```
# Rest framework settings

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.TokenAuthentication',
    ),
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.IsAuthenticated',
    )
}
```

3. Run the following command:

```
$ python manage.py migrate
```

1.3.5

To enhance security, Modoboa now checks the *strength of user passwords <https://github.com/dstufft/django-passwords>_*.

To use this feature, add the following configuration into the `settings.py` file:

```
# django-passwords

PASSWORD_MIN_LENGTH = 8
```

```
PASSWORD_COMPLEXITY = {
    "UPPER": 1,
    "LOWER": 1,
    "DIGITS": 1
}
```

1.3.2

Modoboa now uses the *atomic requests* mode to preserve database consistency (reference).

To enable it, update the `DATABASES` variable in `settings.py` as follows:

```
DATABASES = {
    "default": {
        # stuff before...
        "ATOMIC_REQUESTS": True
    },
    "amavis": {
        # stuff before...
        "ATOMIC_REQUESTS": True
    }
}
```

1.3.0

This release does not bring awesome new features but it is a necessary bridge to the future of Modoboa. All extensions now have their own git repository and the deploy process has been updated to reflect this change.

Another important update is the use of Django 1.7. Besides its new features, the migration system has been reworked and is now more robust than before.

Before we begin with the procedure, here is a table showing old extension names and their new name:

Old name	New package name	New module name
modoboa.extensions.admin	modoboa-admin	modoboa_admin
modoboa.extensions.limits	modoboa-admin-limits	modoboa_admin_limits
modoboa.extensions.postfix_autoreply	modoboa-postfix-autoreply	modoboa_postfix_autoreply
modoboa.extensions.postfix_relay_domains	modoboa-admin-relaydomains	modoboa_admin_relaydomains
modoboa.extensions.radicale	modoboa-radicale	modoboa_radicale
modoboa.extensions.sievefilters	modoboa-sievefilters	modoboa_sievefilters
modoboa.extensions.stats	modoboa-stats	modoboa_stats
modoboa.extensions.webmail	modoboa-webmail	modoboa_webmail

Here are the required steps:

1. Install the extensions using pip (look at the second column in the table above):

   ```
   $ pip install <the extensions you want>
   ```

2. Remove `south` from `INSTALLED_APPS`
3. Rename old extension names inside `MODOBOA_APPS` (look at the third column in the table above)
4. Remove `modoboa.lib.middleware.ExtControlMiddleware` from `MIDDLEWARE_CLASSES`
5. Change `DATABASE_ROUTERS` to:

```
DATABASE_ROUTERS = ["modoboa_amavis.dbrouter.AmavisRouter"]
```

6. Run the following commands:

```
$ cd <modoboa_instance_dir>
$ python manage.py migrate
```

7. Reply `yes` to the question
8. Run the following commands:

```
$ python manage.py load_initial_data
$ python manage.py collectstatic
```

9. The cleanup job has been renamed in Django, so you have to modify your crontab entry:

```
- 0 0 * * * <modoboa_site>/manage.py cleanup
+ 0 0 * * * <modoboa_site>/manage.py clearsessions
```

1.2.0

A new notification service let administrators know about new Modoboa versions. To activate it, you need to update the `TEMPLATE_CONTEXT_PROCESSORS` variable like this:

```
from django.conf import global_settings

TEMPLATE_CONTEXT_PROCESSORS = global_settings.TEMPLATE_CONTEXT_PROCESSORS + (
  'modoboa.core.context_processors.top_notifications',
)
```

and to define the new `MODOBOA_API_URL` variable:

```
MODOBOA_API_URL = 'http://api.modoboa.org/1/'
```

The location of external static files has changed. To use them, add a new path to the `STATICFILES_DIRS`:

```
# Additional locations of static files
STATICFILES_DIRS = (
  # Put strings here, like "/home/html/static" or "C:/www/django/static".
  # Always use forward slashes, even on Windows.
  # Don't forget to use absolute paths, not relative paths.
  "<path/to/modoboa/install/dir>/bower_components",
)
```

Run the following commands to define the hostname of your instance:

```
$ cd <modoboa_instance_dir>
$ python manage.py set_default_site <hostname>
```

If you plan to use the Radicale extension:

1. Add `'modoboa.extensions.radicale'` to the `MODOBOA_APPS` variable
2. Run the following commands:

```
$ cd <modoboa_instance_dir>
$ python manage.py syncdb
```

Warning: You also have to note that the `sitestatic` directory has moved from `<path to your site's dir>` to `<modoboa's root url>` (it's probably the parent directory). You have to adapt your web server configuration to reflect this change.

Configuration

Online parameters

Modoboa provides online panels to modify internal parameters. There are two available levels:

- Application level: global parameters, define how the application behaves. Available at *Modoboa* > *Parameters*
- User level: per user customization. Available at *User* > *Settings* > *Preferences*

Regardless level, parameters are displayed using tabs, each tab corresponding to one application.

General parameters

The *admin* application exposes several parameters, they are presented below:

Name	Tab	Description	Default value
Authentication type	General	The backend used for authentication	Local
Default password scheme	General	Scheme used to crypt mailbox passwords	crypt
Rounds	General	Number of rounds (only used by *sha256crypt* and *sha512crypt*). Must be between 1000 and 999999999, inclusive.	70000
Secret key	General	A key used to encrypt users' password in sessions	random value
Sender address	General	Email address used to send notifications.	
Enable communication	General	Enable communication with Modoboa public API	yes
Check new versions	General	Automatically checks if a newer version is available	yes
Send statistics	General	Send statistics to Modoboa public API (counters and used extensions)	yes
Top notifications check interval	General	Interval between two top notification checks (in seconds)	30
Maximum log record age	General	The maximum age in days of a log record	365
Items per page	General	Number of displayed items per page	30
Default top redirection	General	The default redirection used when no application is specified	user-prefs
Enable MX checks	Admin	Check that every domain has a valid MX record	yes
Valid MXs	Admin	A list of IP or network address every MX should match. A warning will be sent if a record does not respect this it.	
Enable DNSBL checks	Admin	Check every domain against major DNSBL providers	yes
Handle mailboxes on filesystem	Admin	Rename or remove mailboxes on the filesystem when they get renamed or removed within Modoboa	no
Mailboxes owner	Admin	The UNIX account who owns mailboxes on the filesystem	vmail
Default domain quota	Admin	Default quota (in MB) applied to freshly created domains with no value specified. A value of 0 means no quota.	0
Automatic account removal	Admin	When a mailbox is removed, also remove the associated account	no
Automatic domain/mailbox creation	Admin	Create a domain and a mailbox when an account is automatically created	yes

Note: If you are not familiar with virtual domain hosting, you should take a look at postfix's documentation. This How to also contains useful information.

Note: A random secret key will be generated each time the *Parameters* page is refreshed and until you save parameters at least once.

Note: Specific LDAP parameters are also available, see *LDAP authentication*.

Media files

Modoboa uses a specific directory to upload files (ie. when the webmail is in use) or to create ones (ex: graphical statistics). This directory is named `media` and is located inside modoboa's installation directory (called `modoboa_site` in this documentation).

To work properly, the system user which runs modoboa (`www-data`, `apache`, whatever) must have write access to this directory.

Customization

Custom logo

You have the possibility to use a custom logo instead of the default one on the login page.

To do so, open the `settings.py` file and add a `MODOBOA_CUSTOM_LOGO` variable. This variable must contain the relative URL of your logo under `MEDIA_URL`. For example:

```
MODOBOA_CUSTOM_LOGO = os.path.join(MEDIA_URL, "custom_logo.png")
```

Then copy your logo file into the directory indicated by `MEDIA_ROOT`.

Host configuration

Note: This section is only relevant when Modoboa handles mailboxes renaming and removal from the filesystem.

To manipulate mailboxes on the filesystem, you must allow the user who runs Modoboa to execute commands as the user who owns mailboxes.

To do so, edit the `/etc/sudoers` file and add the following inside:

```
<user_that_runs_modoboa> ALL=(<mailboxes owner>) NOPASSWD: ALL
```

Replace values between <> by the ones you use.

Time zone and language

Modoboa is available in many languages.

To specify the default language to use, edit the `settings.py` file and modify the `LANGUAGE_CODE` variable:

```
LANGUAGE_CODE = 'fr' # or 'en' for english, etc.
```

Note: Each user has the possibility to define the language he prefers.

In the same configuration file, specify the timezone to use by modifying the `TIME_ZONE` variable. For example:

```
TIME_ZONE = 'Europe/Paris'
```

Sessions management

Modoboa uses Django's session framework to store per-user information.

Few parameters need to be set in the `settings.py` configuration file to make Modoboa behave as expected:

```
SESSION_EXPIRE_AT_BROWSER_CLOSE = False # Default value
```

This parameter is optional but you must ensure it is set to `False` (the default value).

The default configuration file provided by the `modoboa-admin.py` command is properly configured.

External authentication

LDAP

Modoboa supports external LDAP authentication using the following extra components:

- Python LDAP client
- Django LDAP authentication backend

If you want to use this feature, you must first install those components:

```
$ pip install python-ldap django-auth-ldap
```

Then, all you have to do is to modify the `settings.py` file. Add a new authentication backend to the *AUTHENTICATION_BACKENDS* variable, like this:

```
AUTHENTICATION_BACKENDS = (
  'modoboa.lib.authbackends.LDAPBackend',
  'django.contrib.auth.backends.ModelBackend',
)
```

Finally, go to *Modoboa* > *Parameters* > *General* and set *Authentication type* to LDAP.

From there, new parameters will appear to let you configure the way Modoboa should connect to your LDAP server. They are described just below:

Name	Description	Default value
Server address	The IP address of the DNS name of the LDAP server	local-host
Server port	The TCP port number used by the LDAP server	389
Use a secure connection	Use an SSL/TLS connection to access the LDAP server	no
Authentication method	Choose the authentication method to use	Direct bind
User DN template (direct bind mode)	The template used to construct a user's DN. It should contain one placeholder (ie. `%(user)s`)	
Bind BN	The distinguished name to use when binding to the LDAP server. Leave empty for an anonymous bind	
Bind password	The password to use when binding to the LDAP server (with 'Bind DN')	
Search base	The distinguished name of the search base	
Search filter	An optional filter string (e.g. '(objectClass=person)'). In order to be valid, it must be enclosed in parentheses.	(mail=%(user)s)
Password attribute	The attribute used to store user passwords	user-Pass-word
Active Directory	Tell if the LDAP server is an Active Directory one	no
Administrator groups	Members of those LDAP Posix groups will be created ad domain administrators. Use ';' characters to separate groups.	
Group type	The type of group used by your LDAP directory.	Posix-Group
Groups search base	The distinguished name of the search base used to find groups	
Domain/mailbox creation	Automatically create a domain and a mailbox when a new user is created just after the first successful authentication. You will generally want to disable this feature when the relay domains extension is in use	yes

If you need additional parameters, you will find a detailled documentation here.

Once the authentication is properly configured, the users defined in your LDAP directory will be able to connect to Modoboa, the associated domain and mailboxes will be automatically created if needed.

The first time a user connects to Modoboa, a local account is created if the LDAP username is a valid email address. By default, this account belongs to the *SimpleUsers* group and it has a mailbox.

To automatically create domain administrators, you can use the **Administrator groups** setting. If a LDAP user belongs to one the listed groups, its local account will belong to the *DomainAdmins* group. In this case, the username is not necessarily an email address.

Users will also be able to update their LDAP password directly from Modoboa.

Note: Modoboa doesn't provide any synchronization mechanism once a user is registered into the database. Any modification done from the directory to a user account will not be reported to Modoboa (an email address change for example). Currently, the only solution is to manually delete the Modoboa record, it will be recreated on the next user login.

SMTP

It is possible to use an existing SMTP server as an authentication source. To enable this feature, edit the `settings.py` file and change the following setting:

```
AUTHENTICATION_BACKENDS = (
    'modoboa.lib.authbackends.SMTPBackend',
    'django.contrib.auth.backends.ModelBackend',
)
```

SMTP server location can be customized using the following settings:

```
AUTH_SMTP_SERVER_ADDRESS = 'localhost'
AUTH_SMTP_SERVER_PORT = 25
AUTH_SMTP_SECURED_MODE = None  # 'ssl' or 'starttls' are accepted
```

Database maintenance

Cleaning the logs table

Modoboa logs administrator specific actions into the database. A clean-up script is provided to automatically remove oldest records. The maximum log record age can be configured through the online panel.

To use it, you can setup a cron job to run every night:

```
0 0 * * * <modoboa_site>/manage.py cleanlogs
#
# Or like this if you use a virtual environment:
# 0 0 * * * <virtualenv path/bin/python> <modoboa_site>/manage.py cleanlogs
```

Cleaning the session table

Django does not provide automatic purging. Therefore, it's your job to purge expired sessions on a regular basis.

Django provides a sample clean-up script: `django-admin.py clearsessions`. That script deletes any session in the session table whose `expire_date` is in the past.

For example, you could setup a cron job to run this script every night:

```
0 0 * * * <modoboa_site>/manage.py clearsessions
#
# Or like this if you use a virtual environment:
# 0 0 * * * <virtualenv path/bin/python> <modoboa_site>/manage.py clearsessions
```

Cleaning inactive accounts

Thanks to *Last-login tracking*, it is now possible to monitor inactive accounts. An account is considered inactive if no login has been recorded for the last 30 days (this value can be changed through the admin panel).

A management command is available to disable or delete inactive accounts. For example, you could setup a cron job to run it every night:

```
0 0 * * * <modoboa_site>/manage.py clean_inactive_accounts
#
# Or like this if you use a virtual environment:
# 0 0 * * * <virtualenv path/bin/python> <modoboa_site>/manage.py clean_inactive_
↪accounts
```

The default behaviour is to disable accounts. You can delete them using the `--delete` option.

Moving to Modoboa

You have an existing platform and you'd like to move to Modoboa, the following tools could help you.

From postfixadmin

A dedicated command allows you to convert an existing postfixadmin database to a Modoboa one. Consult the documentation to know more about the process.

Using CSV files

Modoboa allows you to import any object (domain, domain alias, mailbox and alias) using a simple CSV file encoded using **UTF8**. Each line corresponds to a single object and must respect one of the following format:

```
domain; <name: string>; <quota: integer>; <default mailbox quota: integer>; <enabled:␣
↪boolean>
domainalias; <name: string>; <targeted domain: string>; <enabled: boolean>
relaydomain; <name: string>; <target host: string>; <target port: integer>; <service:␣
↪string>; <enabled: boolean>; <verify recipients: boolean>
account; <loginname: string>; <password: string>; <first name: string>; <last name:␣
↪string>; <enabled: boolean>; <group: string>; <address: string>; <quota: integer>; [
↪<domain: string>, ...]
alias; <address: string>; <enabled: boolean>; <recipient: string>; ...
```

Boolean fields accept the following values: `true`, `1`, `yes`, `y` (case insensitive). Any other value will be evaluated as false.

Warning: The order does matter. Objects are created sequencially so a domain must be created before its mailboxes and aliases and a mailbox must created before its alias(es).

To actually import such a file:

```
> sudo -u <modoboa_user> -i
> source <virtualenv_path>/bin/activate
> cd <modoboa_instance_dir>
> python manage.py modo import <your file>
```

Available options can be listed using the following command:

```
> python manage.py modo import -h
```

REST API

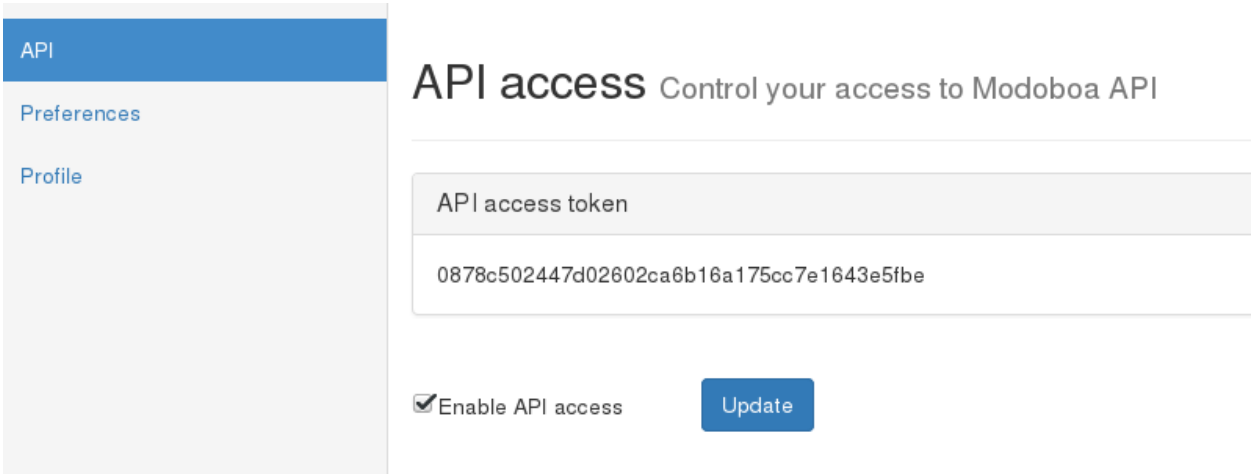
To ease the integration with external sources (software or other), Modoboa provides a REST API.

Every installed instance comes with a ready-to-use API and a documentation. You will find them using the following url patterns:

- API: *http://<hostname>/api/v1/*
- Documentation: *http://<hostname>/docs/api/*

An example of this documentation is available on the official demo.

Using this API requires an authentication and for now, only a token based authentication is supported. To get a valid token, log-in to your instance with a super administrator, go to *Settings* > *API* and activate the API access. Press the Update button and wait until the page is reloaded, the token will be displayed.



To make valid API calls, every requests you send must embed this token within an Authorization HTTP header like this:

```
Authorization: Token <YOUR_TOKEN>
```

and the content type of those requests must be `application/json`.

How to contribute

Contributions are always welcome. If you want to submit a patch, please respect the following rules:

- Open a pull request on the appropriate repository
- Respect PEP8
- Document your patch and respect PEP 257
- Add unit tests and make sure the global coverage does not decrease

If all those steps are validated, your contribution will generally be integrated.

Table of contents

Useful tips

You would like to work on Modoboa but you don't know where to start? You're at the right place! Browse this page to learn useful tips.

Prepare a virtual environment

A virtual environment is a good way to setup a development environment on your machine.

Note: `virtualenv` is available on all major distributions, just install it using your favorite packages manager.

To do so, run the following commands:

```
$ virtualenv <path>
$ source <path>/bin/activate
$ git clone https://github.com/modoboa/modoboa.git
$ cd modoboa
$ python setup.py develop
$ pip install -r dev-requirements.txt
```

The `develop` command creates a symbolic link to your local copy so any modification you make will be automatically available in your environment, no need to copy them.

Deploy an instance for development

Warning: Make sure to *create a database* before running this step. The format of the database url is also described in this page.

Now that you have a *running environment*, you're ready to deploy a test instance:

```
$ cd <path>
$ modoboa-admin.py deploy --dburl default:<database url> --hostname localhost --devel␣
↪instance
$ python manage.py runserver
```

You're ready to go! You should be able to access Modoboa at `http://localhost:8000` using `admin:password` as credentials.

Manage static files

Modoboa uses bower (thanks to django-bower) to manage its CSS and javascript dependencies.

Those dependencies are listed in a file called `dev_settings.py` located inside the `<path_to_local_copy>/modoboa/core` directory.

If you want to add a new dependency, just complete the `BOWER_INSTALLED_APPS` parameter and run the following command:

```
$ python manage.py bower install
```

It will download and store the required files into the <path_to_local_copy>/modoboa/bower_components directory.

Test your modifications

If you deployed a specific instance for your development needs, you can run the tests suite as follows:

```
> python manage.py test modoboa.core modoboa.lib modoboa.admin modoboa.limits modoboa.
↪relaydomains
```

Otherwise, you can run the tests suite from the repository using tox.

Start a basic Modoboa instance

From the repository, run the following command to launch a simple instance with a few fixtures:

```
> tox -e serve
```

You can use admin/password to log in.

Build the documentation

If you need to modify the documenation and want to see the result, you can build it as follows:

```
> tox -e doc
> firefox .tox/doc/tmp/html/index.html
```

FAQ

bower command is missing in manage.py

bower command is missing in *manage.py* if you don't use the `--devel` option of the `modoboa-admin.py deploy` command.

To fix it, regenerate your instance or update your `settings.py` file manually. Look at `devmode` in https://github.com/tonioo/modoboa/blob/master/modoboa/core/commands/templates/settings.py.tpl

Create a new plugin

Introduction

Modoboa offers a plugin API to expand its capabilities. The current implementation provides the following possibilities:

- Expand navigation by adding entry points to your plugin inside the GUI
- Access and modify administrative objects (domains, mailboxes, etc.)
- Register callback actions for specific events

Plugins are nothing more than Django applications with an extra piece of code that integrates them into Modoboa. The `modo_extension.py` file will contain a complete description of the plugin:

- Admin and user parameters
- Custom menu entries

The communication between both applications is provided by Django signals.

The following subsections describe the plugin architecture and explain how you can create your own.

The required glue

To create a new plugin, just start a new django application like this (into Modoboa's directory):

```
$ python manage.py startapp
```

Then, you need to register this application using the provided API. Just copy/paste the following example into the `modo_extension.py` file of the future extension:

```
from modoboa.core.extensions import ModoExtension, exts_pool


class MyExtension(ModoExtension):
    """My custom Modoboa extension."""

    name = "myext"
    label = "My Extension"
    version = "0.1"
    description = "A description"
    url = "myext_root_location" # optional, name is used if not defined

    def load(self):
        """This method is called when Modoboa loads available and activated plugins.

        Declare parameters and register events here.
        """
        pass

    def load_initial_data(self):
        """Optional: provide initial data for your extension here."""
        pass

exts_pool.register_extension(MyExtension)
```

Once done, simply add your extension's module name to the `MODOBOA_APPS` variable located inside `settings.py`. Finally, run the following commands:

```
$ python manage.py migrate
$ python manage.py load_initial_data
$ python manage.py collectstatic
```

Parameters

A plugin can declare its own parameters. There are two levels available:

- 'Global' parameters : used to configure the plugin, editable inside the *Admin* > *Settings* > *Parameters* page
- 'User' parameters : per-user parameters (or preferences), editable inside the *Options* > *Preferences* page

Playing with parameters

Parameters are defined using Django forms and Modoboa adds two special forms you can inherit depending on the level of parameter(s) you want to add:

- `modoboa.parameters.forms.AdminParametersForm`: for general parameters
- `modoboa.parameters.forms.UserParametersForm`: for user parameters

To register new parameters, add the following line into the `load` method of your plugin class:

```
from modoboa.parameters import tools as param_tools
param_tools.registry.add(
    LEVEL, YourForm, ugettext_lazy("Title"))
```

Replace `LEVEL` by `"global"` or `"user"`.

Custom role permissions

Modoboa uses Django's internal permission system. Administrative roles are nothing more than groups (`Group` instances).

An extension can add new permissions to a group by listening to the `extra_role_permissions` signal. Here is an example:

```
from django.dispatch import receiver
from modoboa.core import signals as core_signals

PERMISSIONS = {
    "Resellers": [
        ("relaydomains", "relaydomain", "add_relaydomain"),
        ("relaydomains", "relaydomain", "change_relaydomain"),
        ("relaydomains", "relaydomain", "delete_relaydomain"),
        ("relaydomains", "service", "add_service"),
        ("relaydomains", "service", "change_service"),
        ("relaydomains", "service", "delete_service")
    ]
}

@receiver(core_signals.extra_role_permissions)
def extra_role_permissions(sender, role, **kwargs):
   """Add permissions to the Resellers group."""
   return constants.PERMISSIONS.get(role, [])
```

Extending admin forms

The forms used to edit objects (account, domain, etc.) through the admin panel are composed of tabs. You can extend them (ie. add new tabs) in a pretty easy way thanks to custom signals.

Account

To add a new tab to the account edition form, define new listeners (handlers) for the following signals:

- `modoboa.admin.signals.extra_account_forms`

- `modoboa.admin.signals.get_account_form_instances`
- `modoboa.admin.signals.check_extra_account_form` (optional)

Example:

```
from django.dispatch import receiver
from modoboa.admin import signals as admin_signals


@receiver(admin_signals.extra_account_forms)
def extra_account_form(sender, user, account, **kwargs):
    return [
        {"id": "tabid", "title": "Title", "cls": MyFormClass}
    ]

@receiver(admin_signals.get_account_form_instances)
def fill_my_tab(sender, user, account, **kwargs):
    return {"id": my_instance}
```

Domain

To add a new tab to the domain edition form, define new listeners (handlers) for the following signals:

- `modoboa.admin.signals.extra_domain_forms`
- `modoboa.admin.signals.get_domain_form_instances`

Example:

```
from django.dispatch import receiver
from modoboa.admin import signals as admin_signals


@receiver(admin_signals.extra_domain_forms)
def extra_account_form(sender, user, domain, **kwargs):
    return [
        {"id": "tabid", "title": "Title", "cls": MyFormClass}
    ]

@receiver(admin_signals.get_domain_form_instances)
def fill_my_tab(sender, user, domain, **kwargs):
    return {"id": my_instance}
```

Contributors

- Antidot
- Bearstech
- Dalnix