ownCloud Administrators Manual

Release 5.0

The ownCloud developers

December 15, 2013

CONTENTS

CHAPTER

ONE

ADMIN DOCUMENTATION

1.1 Introduction

This is the administrators manual for ownCloud, a flexible, open source file sync and share solution. It comprises of the ownCloud server, as well as client applications for Microsoft Windows, Mac OS X and Linux (Desktop Client) and mobile clients for the Android and Apple iOS operating system.

1.1.1 Target audience

This guide is targeted towards people who want to install, administer and optimize ownCloud Server. If you want to learn how to use the Web UI, or how to install clients on the server, please refer to the User Manual or the Desktop Client Manual respectively.

1.1.2 Structure of this document

The next chapters describes how to set up ownCloud Server on different platforms and operating systems, as well as how to update existing installations.

Further chapters will then detail on integrating ownCloud into your existing environment, e.g. how to setup LDAP or how to mount your storage.

1.2 Installation

This chapter will introduce you to the installation of ownCloud in different scenarios.

If you want to just try ownCloud in a virtual machine without any configuration, check the section *Appliances*, where you will find ready-to-use images.

- *Appliances*
- *Linux Distributions*
- *Mac OS X*
- *Windows 7 and Windows Server 2008*
- *Univention Corporate Server*
- *Manual Installation*
- *Other Web Servers*

1.3 Configuration

This chapter covers ownCloud and Webserver configuration.

- *3rd-Party Configuration*
- *Apps Configuration*
- *Automatic Configuration*
- *Database Configuration*
- *Knowledge Base Configuration*
- *Logging Configuration*
- *Mail Configuration*
- *Reverse Proxy Configuration*
- *Custom Mount Configuration*
- *Custom Mount Configuration Web-GUI*
- *Custom User Backend Configuration*
- *User Authentication with LDAP*
- *Serving static files via web server*
- *Background Jobs*
- *Custom Client Configuration*
- *Use Server-Side Encryption*
- *Language Configuration*
- *Maintenance Mode Configuration*
- *Uploading big files > 512MB (as set by default)*

1.4 Maintenance

This chapter covers maintenance tasks such as updating or migrating to a new version.

- *Migrating ownCloud Installations*
- *Updating ownCloud*

1.5 Indices and tables

- *genindex*

CHAPTER

TWO

INSTALLATION

2.1 Appliances

If you are looking for virtual machine images, check the Software Appliances section. The Hardware Appliances section is of interest for people seeking to run ownCloud on appliance hardware (i.e. NAS filers, routers, etc.).

2.1.1 Software Appliances

There are number of pre-made virtual machine-based appliances:

- SUSE Studio, ownCloud on openSuSE, runnable directly from an USB stick.
- Ubuntu charm, ownCloud 4.5
- PCLinuxOS based appliance
- Fedora based appliance

2.1.2 ownCloud on Hardware Appliances

These are tutorials provided by the user communities of the respective appliances:

- QNAP Guide for QNAP NAS appliances
- OpenWrt Guide for the popular embedded distribution for routers and NAS devices.

Todo

Tutorials for running owncloud on Synology and Dreamplug.

2.2 Linux Distributions

This section describes the installation process for different distributions. If there are pre-made packages from ownCloud, you are encouraged to prefer those over the vendor-provided ones, since they usually are more up-to-date.

2.2.1 Archlinux

There are two AUR packages for ownCloud:

- stable version
- development version

2.2.2 openSUSE

Note: ready-to-use SLES and openSUSE RPM packages are available in the openSUSE Build Service ownCloud repository.

---

1. Copy ownCloud to Apache's server directory : **/srv/www/htdocs**
2. Give the web server the necessary permissions:

```
sudo chown -R wwwrun owncloud
```

3. Open the folder in a browser and complete the setup wizard

If have followed the steps above and want to try it out, run this command in a terminal to start Apache if it's not already running:

```
sudo /etc/init.d/apache2 start
```

Go to http://servername/owncloud and walk through the setup.

2.2.3 Fedora

Note: ready-to-use RPM packages are available in the openSUSE Build Service ownCloud repository.

Make sure SELinux is disabled or else the installation process will fail with the following message:

```
Config file (config/config.php) is not writable for the webserver
```

Configure Apache:

1. If you already have a website running from Document Root but would still like to install OwnCloud you can use a Name-based virtual host entry and subdomain.
2. Edit your DNS record following this example: **point owncloud.foo.com > ip.ip.ip.ip**

2.2.4 CentOS 5 & 6

Note: ready-to-use CentOS RPM packages are available in the openSUSE Build Service ownCloud repository.

---

1. Create a new file in **/etc/httpd/conf/** and call it **owncloud.conf**.
2. You can use the following as an example:

```
<IfModule mod_alias.c>
Alias /owncloud /var/www/owncloud/
</IfModule>
<Directory /var/www/owncloud/>
   Options None
   Order allow,deny
   allow from all
</Directory>
<VirtualHost *:80>
    ServerAdmin foo@foofarm.com
    DocumentRoot /var/www/html/owncloud
    ServerName owncloud.foo.com
    ErrorLog logs/owncloud.foo.info-error_log
    CustomLog logs/owncloud.foo.info-access_log common

</VirtualHost>
```

3. Now edit your httpd.conf file which is usually located in `/etc/httpd/conf/httpd.conf`
4. Add the following to the bottom

```
Include /etc/httpd/conf/owncloud.conf
```

5. Restart apache and now when you point your browser to http://owncloud.foo.com it should properly load without affecting http://foo.com

2.2.5 Gentoo

Set up a standard web server (see instructions above). Then change permissions:

```
chown -R apache:apache owncloud
```

Allow .htaccess, modify `/etc/apache2/vhosts.d/00_default_vhost.conf` and make sure it contains the following section

```
<Directory /var/www/localhost/htdocs/owncloud>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order allow,deny
    allow from all
</Directory>
```

2.2.6 PCLinuxOS

Follow the Tutorial ownCloud, installation and setup on the PCLinuxOS web site.

2.2.7 Ubuntu / Debian

Go to the linux package sources page and execute the steps as described there for your distribution.

2.3 Mac OS X

Note: Due to an issue with Mac OS Unicode support, installing ownCloud Server 5.0 on Mac OS is currently not supported.

2.4 Windows 7 and Windows Server 2008

Note: You must move the data directory outside of your public root (See advanced install settings)

This section describes how to install ownCloud on Windows with IIS (Internet Information Services).

It assumes that you have a vanilla, non-IIS enabled Windows machine – Windows 7 or Server 2008. After enabling IIS, the steps are essentially identical for Windows 7 and Windows Server 2008.

For installing ownCloud physical access or a remote desktop connection is required. You should leverage MySQL as the backend database for ownCloud. If you do not want to use MySQL, it is possible to use Postgres or SQLite instead. Microsoft SQL Server is not yet support.

Enabling SSL is not yet covered by this section.

Note: If you make your desktop machine or server available outside of your

LAN, you must maintain it. Monitor the logs, manage the access, apply patches to avoid compromising the system at large.

There are 4 primary steps to the installation, and then a 5th step required for configuring everything to allow files larger than the default 2MB.

1. Install IIS with CGI support – enable IIS on your Windows machine.
2. Install PHP – Grab, download and install PHP.
3. Install MySQL – Setup the MySQL server manager and enable ownCloud to create an instance.
4. Install ownCloud – The whole reason we are here!
5. Configure upload sizes and timeouts to enable large file uploads – So that you can upload larger files.

2.4.1 Activate IIS with CGI Support

Windows 7

1. Go to *Start –> Control Panel –> Programs*.
2. Under Programs and Features, there is link titled *Turn Windows Features on and Off*. Click on it.
3. There is a box labeled Internet Information Services, expand it.
4. Expand World Wide Web Services and all the folders underneath.
5. Select the folders as illustrated in the picture below to get your IIS server up and running.

You do not need an FTP server running, so you should tune that feature off for your server. You definitely need the IIS Management Console, as that is the easiest way to start, stop, restart you server, as well as where you change certificate options and manage items like file upload size. You must check the CGI box under Application Development Features, because CGI is how you enable PHP on IIS.

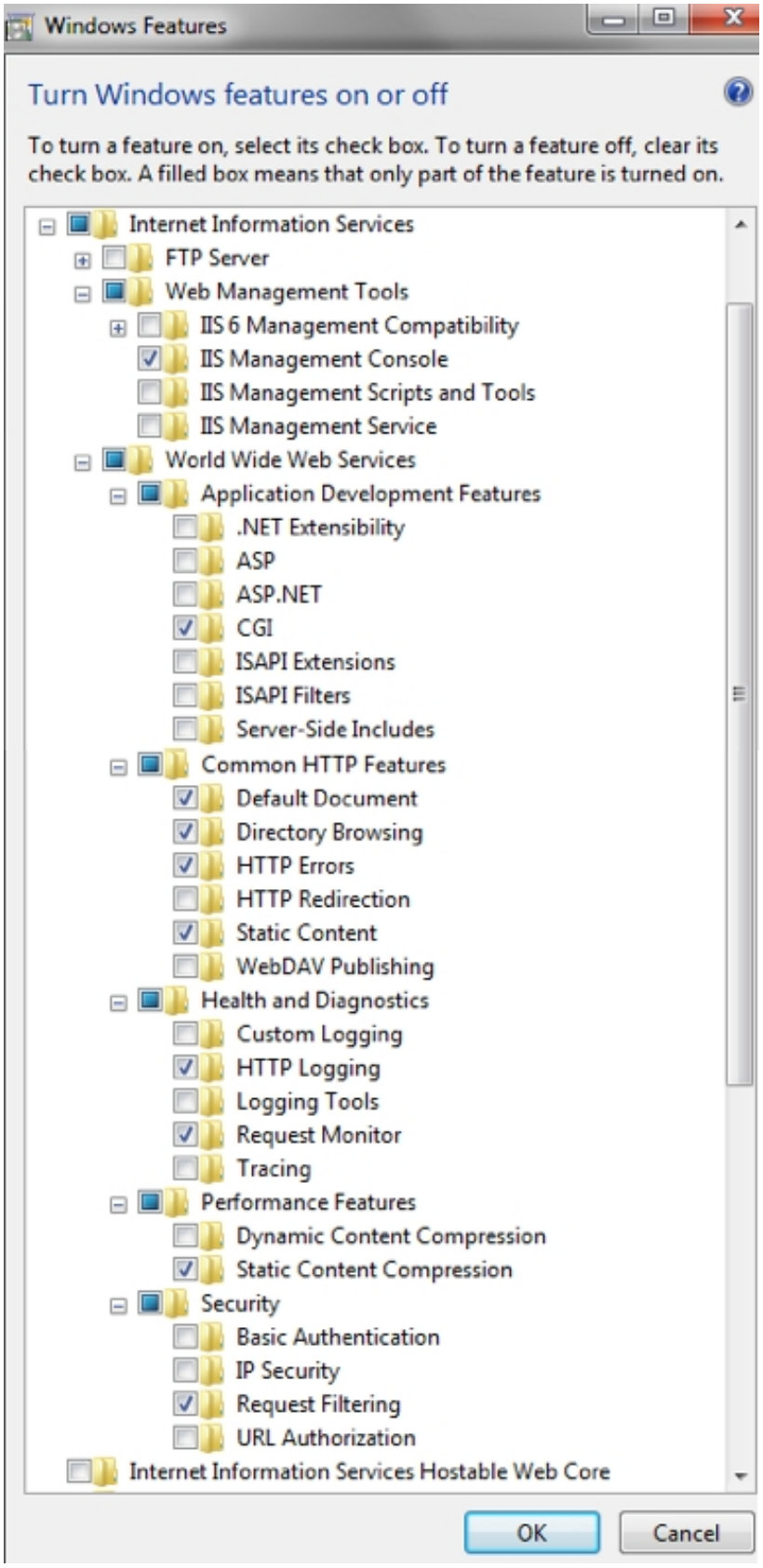


Figure 2.1: Windows Features required for ownCloud on Windows 7

You have to turn off WebDAV publishing or the Windows WebDAV conflicts with the ownCloud WebDAV interface. This might already be turned off for you, just make sure it stays that way. The common HTTP features are the features you would expect from a web server. With the selections on this page, IIS will now serve up a web page for you.

Restart IIS by going to the IIS manager (*Start –> IIS Manager*).

Select your website, and on the far right side is a section titled *Manage Server*. Make sure that the service is started, or click *Start* to start the services selected. Once this is complete, you should be able to go to a web browser and navigate to http://localhost.

This should open the standard IIS 7 splash page, which is just a static image that says your web server is running. Assuming you were able to get the splash page, it is safe to say your web server is now up and running.

Windows Server 2008

1. Go to *Start –> Control Panel –> Programs*.
2. Under Programs and Features, there is link titled *Turn Windows Features on and Off*. Click on it.
3. This will bring up the Server Manager.
4. In the server manager, Click on Roles, and then click Add Roles.
5. Use the *Add Roles Wizard* to add the web server role.
6. Make sure that, at a minimum, the same boxes are checked in this wizard that are checked in the Windows 7 Section. For example, make sure that the CGI box is checked under Application Development Features, and that WebDAV Publishing is turned off. With Remote Desktop Sharing turned on, the detailed role service list looks like the figure "Role Services".
7. Restart IIS by going to the IIS manager (*Start –> IIS Manager*).
8. Select your website, and on the far right side is a section titled Manage server. Make sure that the service is started, or click "Start" to start the services selected.
9. Once this is complete, you should be able to go to a web browser and type *localhost*. This should open the standard IIS 7 splash page, which is just a static image that says your web server is running.Assuming you were able to get the splash page, it is safe to say your web server is now up and running. The next part of this "how to" installs PHP on the server.

2.4.2 Installing PHP

This part is also straightforward, but it is necessary to remind you that this is for IIS only.

1. Go to the following link and grab the PHP installer for version "VC9 Non Thread Safe" 32 or 64 bit based on your system.

Note: If you are using Apache, make sure you grab VC6 instead, lower on the page.

---

2. Once through that login, select the location that is closest to you geographically.
3. Run that install wizard once it is downloaded. Read the license agreement, agree, select an install directory.
4. Then select IIS FastCGI as the install server.
5. Take the default selections for the items to install, and click next. Then click *install*.
6. After a few minutes, PHP will be installed. On to MySQL.

Role Services: 40 installed

Role Service	Status
Web Server	Installed
Common HTTP Features	Installed
Static Content	Installed
Default Document	Installed
Directory Browsing	Installed
HTTP Errors	Installed
HTTP Redirection	Installed
WebDAV Publishing	Not installed
Application Development	Installed
ASP.NET	Installed
.NET Extensibility	Installed
ASP	Installed
CGI	Installed
ISAPI Extensions	Installed
ISAPI Filters	Installed
Server Side Includes	Not installed
Health and Diagnostics	Installed
HTTP Logging	Installed
Logging Tools	Installed
Request Monitor	Installed
Tracing	Installed
Custom Logging	Not installed
ODBC Logging	Not installed
Security	Installed
Basic Authentication	Installed
Windows Authentication	Installed
Digest Authentication	Installed
Client Certificate Mapping Authentication	Installed
IIS Client Certificate Mapping Authentication	Installed
URL Authorization	Installed
Request Filtering	Installed
IP and Domain Restrictions	Installed
Performance	Installed
Static Content Compression	Installed
Dynamic Content Compression	Installed
Management Tools	Installed
IIS Management Console	Installed
IIS Management Scripts and Tools	Installed
Management Service	Installed
IIS 6 Management Compatibility	Installed
IIS 6 Metabase Compatibility	Installed
IIS 6 WMI Compatibility	Installed
IIS 6 Scripting Tools	Installed
IIS 6 Management Console	Installed
FTP Server	Not installed

2.4.3 Installing MySQL

This part installs MySQL on your Windows machine.

1. Point your browser to http://dev.mysql.com/downloads/ and download the latest community edition for your OS – the 32 or 64 bit version. Please download the **MSI Installer** as it will make life easier.
2. Once downloaded, install MySQL (5.5 at the time of writing). Select the Typical installation.
3. When that finishes, check the box to launch the MySQL Instance Configuration Wizard and click Finish.
4. Select a standard configuration, as this will be the only version of MySQL on this machine.
5. Select to install as a windows service, and Check the Launch the MySQL Server Automatically button.
6. Select the modify security settings box on the next page, and enter a password you will remember. You will need this password when you configure ownCloud.
7. Uncheck **enable** root access from remote machines" for security reasons.
8. Click execute, and wait while the instance is created and launched.
9. Click Finish when this is all complete.

Take particular note of your MySQL password, as the user name **root** and the password you select will be necessary later on in the ownCloud installation. As an aside, this link is an excellent resource for questions on how to configure your MySQL instance, and also to configure PHP to work with MySQL. This, however, is not strictly necessary as much of this is handled when you download ownCloud.

More information in this topic can be found in a tutorial on the IIS web site.

2.4.4 Installing ownCloud

1. Download the latest version of ownCloud from http://owncloud.org/download.
2. It will arrive as a tar.bz2 file, and I recommend something like jZip for a free utility to unzip it.
3. Once you have the ownCloud directory unzipped and saved locally, copy it into your wwwroot directory (probably **C:\inetpub\wwwroot**).

Note: You cannot install directly into the directory **wwwroot** from jzip,

---

as only the administrator can unzip into the **wwwroot** directory. If you save it in a different folder, and then move the files into **wwwroot** in windows explorer, it works. This will install ownCloud locally in your root web directory. You can use a subdirectory called owncloud, or whatever you want – the www root, or something else.

4. It is now time to give write access to the ownCloud directory to the ownCloud server: Navigate your windows explorer over to **inetpub/wwwroot/owncloud** (or your installation directory if you selected something different).
5. Right click and select properties. Click on the security tab, and click the button "to change permissions, click edit".
6. Select the "users" user from the list, and check the box "write".
7. Apply these settings and close out.
8. Now open your browser and go to http://localhost/owncloud (or localhost if it is installed in the root www directory). This should bring up the ownCloud configuration page.
9. At this page, you enter your desired ownCloud user name and password for the administrator, and expand the little arrow.

10. Select MySQL as the database, and enter your MySQL database user name,

    password and desired instance name – use the user name and password you setup for MySQL earlier in step 3, and pick any name for the database instance.

Note: The owncloud admin password and the MySQL password CANNOT be the same in any way.

---

11. Click next, and ownCloud should have you logged in as the admin user, and you can get started exploring ownCloud, creating other users and more!

2.4.5 Ensure Proper HTTP-Verb handling

IIS must pass all HTTP and WebDAV verbs to the PHP/CGI handler, and must not try to handle them by itself. If it does, syncing with the Desktop and Mobile Clients will fail. Here is how to ensure your configuration is correct:

1. Open IIS Manager7.
2. In the *Connections* bar, pick your site below *Sites*, or choose the top level entry if you want to modify the machine-wide settings.
3. Choose the *Handler Mappings* feature click *PHP_via_fastCGI*.
4. Choose *Request Restrictions* and find the *Verbs* tab.
5. Ensure *All Verbs* is checked.
6. Click *OK*.
7. Next, choose *Request Filtering* feature from IIS Manager.
8. Ensure that all verbs are permitted (or none are forbidden) in the *Verbs* tab.

Also, ensure that you did not enable the WebDAV authoring module, since ownCloud needs to be able to handle WebDAV on the application level.

2.4.6 Configuring ownCloud, PHP and IIS for Large File Uploads

Before going too nuts on ownCloud, it is important to do a couple of configuration changes to make this a useful service for you. You will probably want to increase the **max upload size**, for example. The default upload is set to **2MB**, which is too small for even most MP3 files.

To do that, simply go into your **PHP.ini** file, which can be found in your **C:\Program Files (x86)\PHP** folder. In here, you will find a **PHP.ini** file. Open this in a text editor, and look for a few key attributes to change:

- **upload_max_filesize** – change this to something good, like 1G, and you will get to upload much larger files.
- **post_max_size** – also change this size, and make it larger than the max upload size you chose, like 1G.

There are other changes you can make, such as the timeout duration for uploads, but for now you should be all set in the **PHP.ini** file.

Now you have to go back to IIS manager and make one last change to enable file uploads on the web server larger than 30MB.

1. Go to the start menu, and type **iis manager**.
2. Open IIS Manager Select the website you want enable to accept large file uploads.
3. In the main window in the middle double click on the icon **Request filtering**.

4. Once the window is opened you will see a bunch of tabs across the top of the far right,

   Select *Edit Feature Settings* and modify the *Maximum allowed content length (bytes)*

5. In here, you can change this to up to 4.1 GB.

Note: This entry is in BYTES, not KB.

You should now have ownCloud configured and ready for use.

2.5 Univention Corporate Server

Subscribers to the ownCloud Enterprise edition can also integrate with UCS (Univention Corporate Server).

2.5.1 Pre configuration

ownCloud makes use of the UCR, the Univention Configuration Registry. The values are being read during installation, most of them can be changed later, too. Changes done directly via ownCloud are not taken over to UCR. We think we found sane defaults, nevertheless you might have your own requirements. The installation script will listen to the UCR keys listed below. In case you want to override any default setting, simply add the key in question to the UCR and assign your required value.

Key	Default	Description	Introduced
owncloud/directory/data	/var/lib/owncloud	Specifies where the file storage will be placed	2012.0.1
owncloud/db/name	owncloud	Name of the MySQL database. ownCloud will create an own user for it.	2012.0.1
owncloud/user/quota	(empty)	The default quota, when a user is being added. Assign values in human readable strings, e.g. “2 GB”. Unlimited if empty.	2012.0.1
owncloud/user/enabled	0	Wether a new user is allowed to use ownCloud by default.	2012.0.1
owncloud/group/enabled	0	Wether a new group is allowed to be used in ownCloud by default.	2012.4.0.4
owncloud/ldap/base/users	cn=users,$ldap_base	The users-subtree in the LDAP directory. If left blank it will fall back to the LDAP base.	2012.4.0.4
owncloud/ldap/base/groups	cn=groups,$ldap_base	The groups-subtree in the LDAP directory. If left blank it will fall back to the LDAP base.	2012.4.0.4
owncloud/ldap/groupMemberAssoc	uniqueMember	The LDAP attribute showing the group-member relationship. Possible values: uniqueMember, memberUid and member	2012.4.0.4
owncloud/ldap/tls	1	Whether to talk to the LDAP server via TLS.	2012.0.1
owncloud/ldap/disableMainServer	0	Deactivates the (first) LDAP Configuration	5.0.9

Continued on next page

Table 2.1 – continued from previous page

Key	Default	Description	Introduced
owncloud/ldap/cacheTTL	600	Lifetime of the ownCloud LDAP Cache in seconds	5.0.9
owncloud/ldap/UUIDAttribute	(empty)	Attribute that provides a unique value for each user and group entry. Empty value for autodetection.	5.0.9
owncloud/ldap/loginFilter	(&(|(&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (objectClass=sambaSamAccount) (objectClass=simpleSecurityObject) (&(objectClass=person) (objectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0)) (!(uid=*$)) (&(uid=%uid) (ownCloudEnabled=1)))	The LDAP filter that shall be used when a user tries to log in.	2012.0.1
owncloud/ldap/userlistFilter	(&(|(&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (objectClass=sambaSamAccount) (objectClass=simpleSecurityObject) (&(objectClass=person) (objectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0))(!(uid=*$)) (&(ownCloudEnabled=1)))	The LDAP filter that shall be used when the user list is being retrieved (e.g. for sharing)	2012.0.1
owncloud/ldap/groupFilter	(&(objectClass=posixGroup) (ownCloudEnabled=1))	The LDAP filter that shall be used when the group list is being retrieved (e.g. for sharing)	2012.4.0.4
owncloud/ldap/internalNameAttribute	uid	Attribute that should be used to create the user's owncloud internal name	5.0.9
owncloud/ldap/displayName	uid	The LDAP attribute that should be displayed as name in ownCloud	2012.0.1
owncloud/ldap/user/searchAttributes	uid,givenName,sn,description,employeeNumber,mailPrimaryAddress	Attributes taken into consideration when searching for users (comma separated)	5.0.9
owncloud/ldap/user/quotaAttribute	ownCloudQuota	Name of the quota attribute. The default attribute is provided by owncloud-schema.	5.0.9
owncloud/ldap/user/homeAttribute	(empty)	Attribute that should be used to create the user's owncloud internal home folder	5.0.9
owncloud/ldap/group/displayName	cn	The LDAP attribute that should be used as groupname in ownCloud	2012.4.0.4
owncloud/ldap/group/searchAttributes	cn,description, mailPrimaryAddress	Attributes taken into consideration when searching for groups (comma separated)	5.0.9
owncloud/join/users/update	yes	Wether ownCloud LDAP schema should be applied to existing users	2012.0.1

Continued on next page

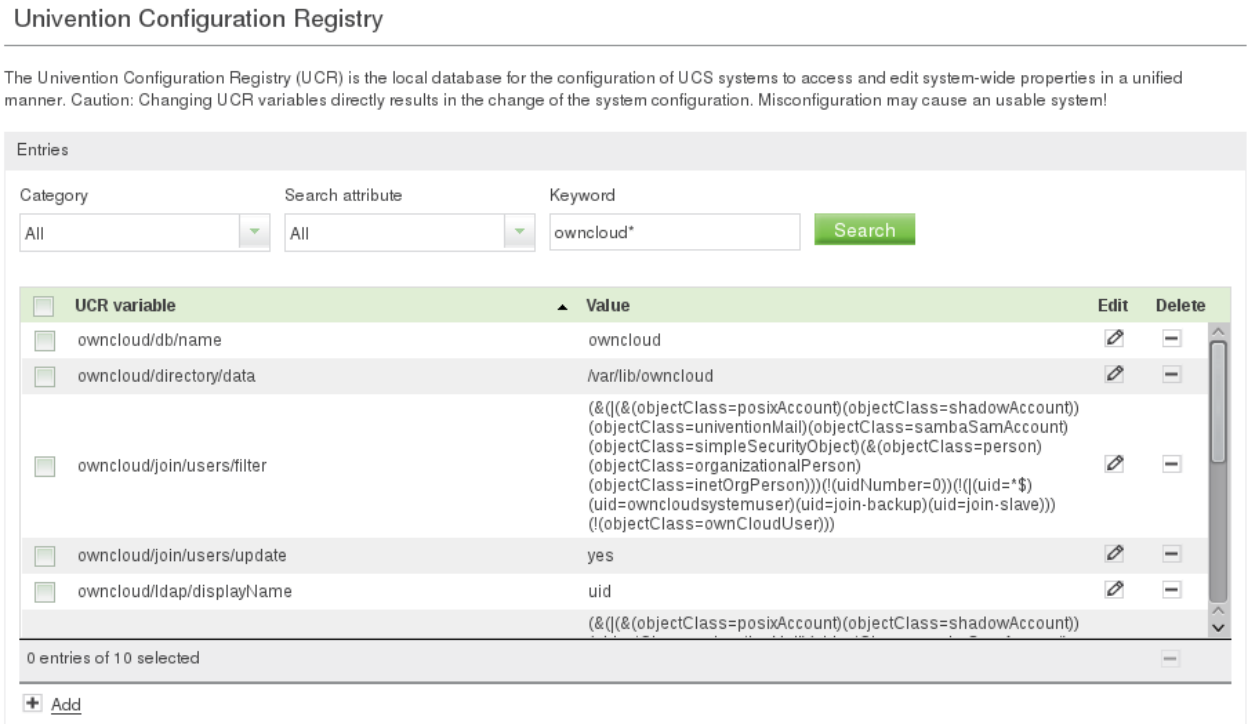
Table 2.1 – continued from previous page

Key	Default	Description	Introduced
owncloud/group/enableDomainUsers	1	Wether the group "Domain Users" shall be enabled for ownCloud on install	2012.4.0.4
owncloud/join/users/filter	(&(|(&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (objectClass=sambaSamAccount) (objectClass=simpleSecurityObject) (&(objectClass=person) (objectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0)) (!(|(uid=*$) (uid=owncloudsystemuser) (uid=join-backup) (uid=join-slave))) (!(objectClass=ownCloudUser)))	Filters, on which LDAP users the ownCloud schema should be applied to. The default excludes system users and already ownCloudUsers.	2012.0.1
owncloud/join/groups/filter	(empty)	Filters which LDAP groups will be en/disabled for ownCloud when running the script /usr/share/owncloud/update-groups.sh	2012.4.0.4

If you want to override the default settings, simply create the key in question in the UCR and assign your required value, for example:

```
ucr set owncloud/user/enabled=1
```

or via UMC:

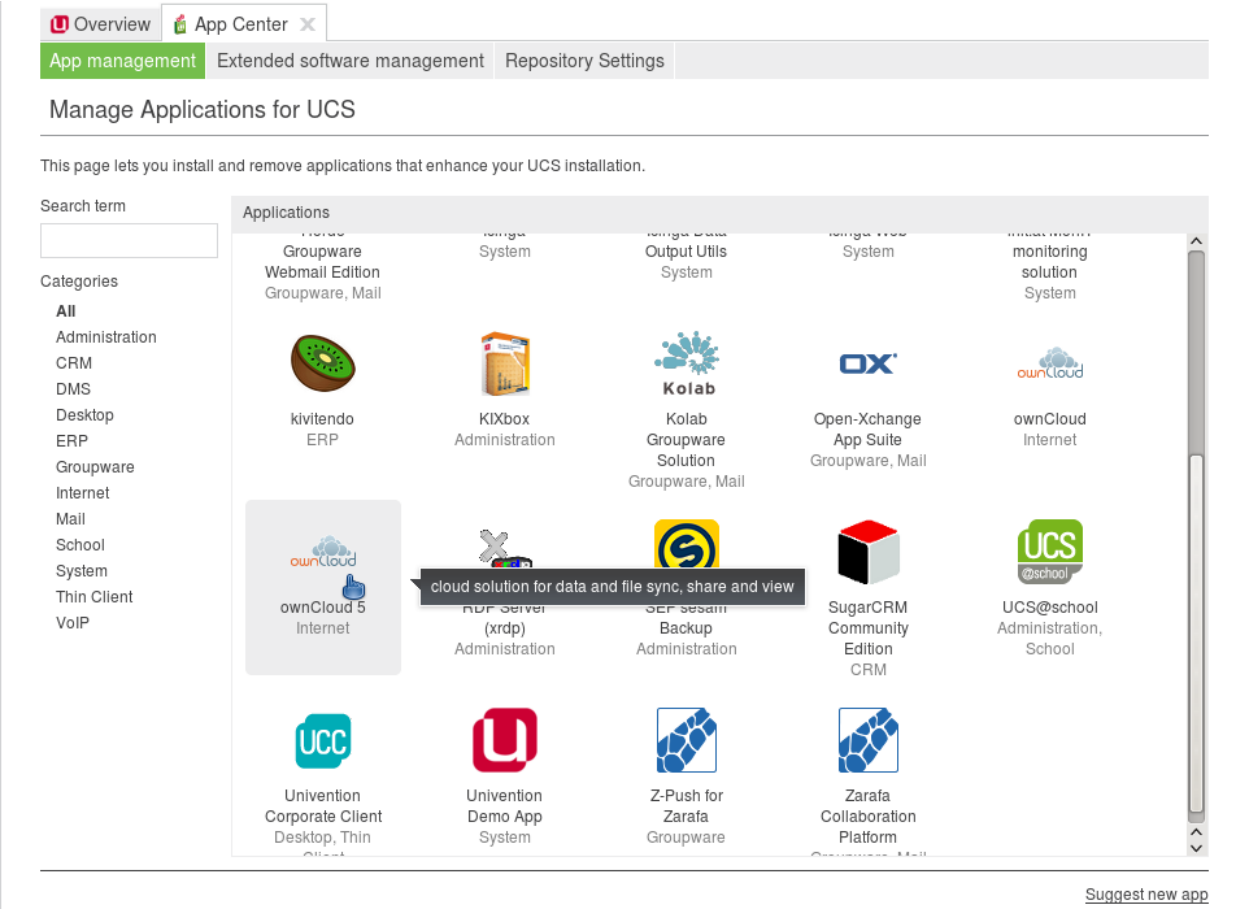


2.5.2 Installation

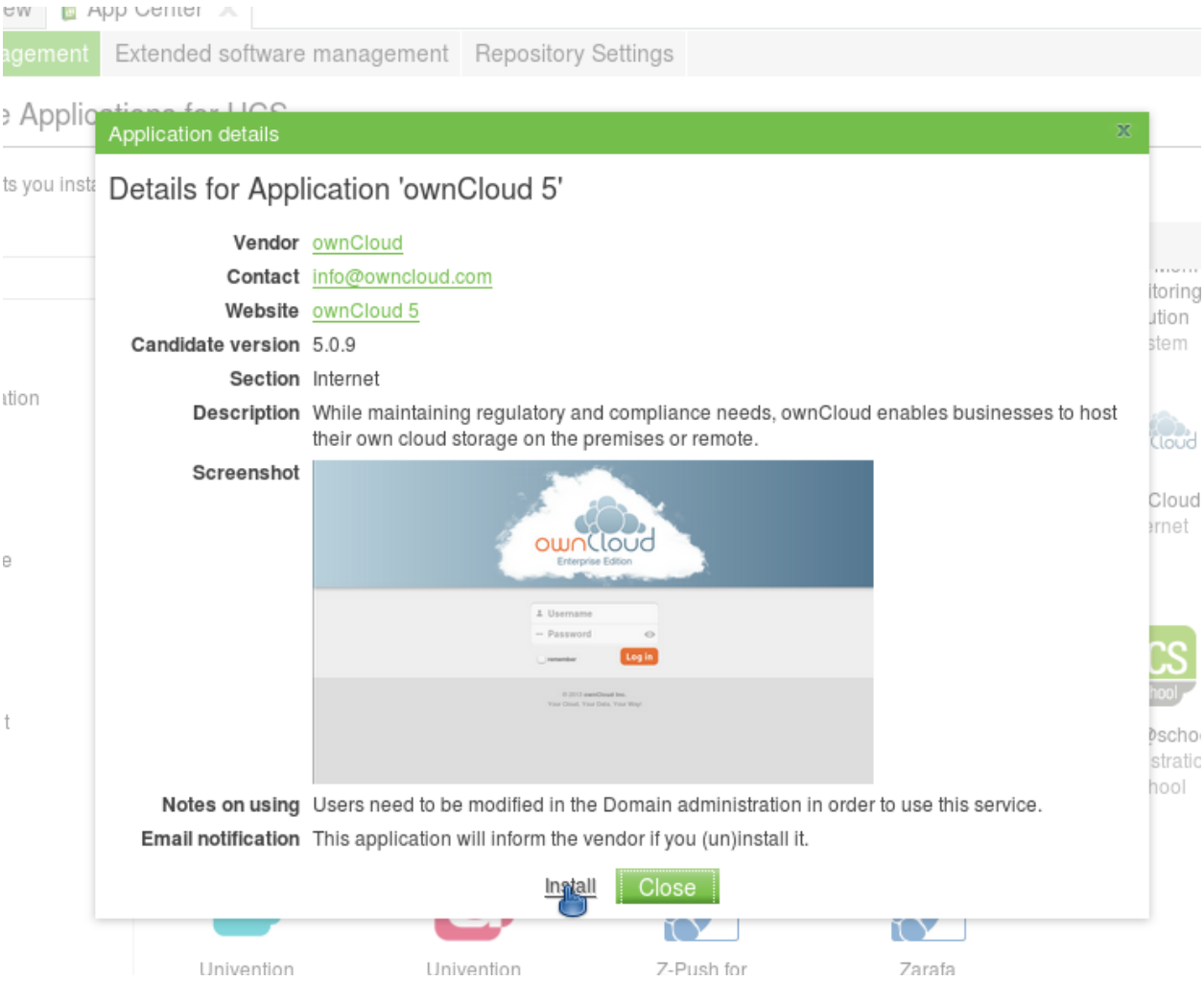
Now, we are ready to install ownCloud. This can be either done through the UCS App Center (recommended) or by downloading the packages.

UCS App Center

Open the Univention Management Console and choose the App Center module. You will see a variety of available applications, including ownCloud.



Click on ownCloud 5 and follow the instructions.



In the UCS App Center, you can also upgrade from ownCloud 4.5 by installing ownCloud 5.0. They are provided as seperate apps. It is only possible to have one version of ownCloud installed.

Manually by download

Download the integration packages from our website and install them from within your download folder (note: the package owncloud-unsupported is optional) via command line:

```
dpkg -i owncloud*.deb
```

ownCloud will be configured to fully work with LDAP.

Reinstallation

When ownCloud was installed before and uninstalled via AppCenter or via command line using apt-get remove, ownCloud can be simply installed again. The old configuration will be used again.

When an older ownCloud was installed and has been purged (only possible via command line using apt-get purge) the old configuration is gone, but data is left. This blocks an installation. You can either install the old version and upgrade to ownCloud 5 or (re)move the old data. This is done by removing the MySQL database "ownCloud" using the command line:

```
mysql -u root -e "DROP DATABASE owncloud" -p`tail /etc/mysql.secret
```

In this case you probably also want to remove the data directory **/var/lib/owncloud** although this is not mandatory.

2.5.3 Postconfiguration (optional)

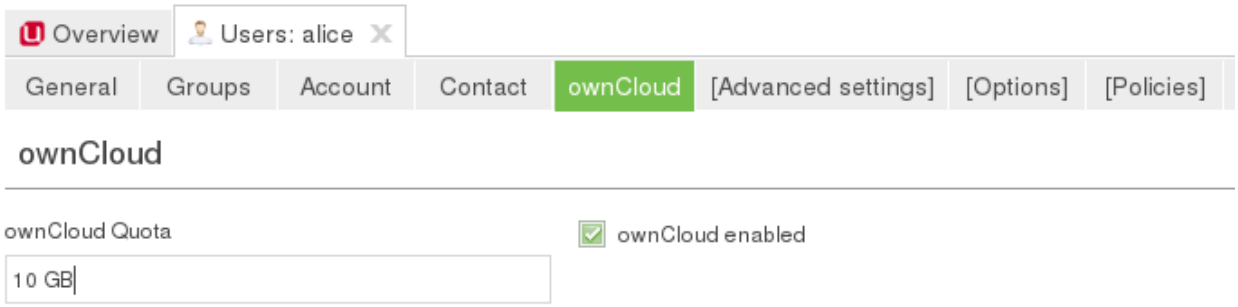
There is only one local admin user "owncloudadmin", you can find his password in `/etc/owncloudadmin.secret`. Use this account, if you want to change basic ownCloud settings.

In the installation process a virtual host is set up (Apache is required therefore). If you want to modify the settings, edit `/etc/apache2/sites-available/owncloud` and restart the web server. You might want to do it to enable HTTPS connections. Besides that, you can edit the **.htaccess-File in /var/www/owncloud/**. In the latter file there are also the PHP limits for file transfer specified.

2.5.4 Using ownCloud

If you decided to enable every user by default to use ownCloud, simply open up http://myserver.com/owncloud/ and log in with your LDAP credentials and enjoy.

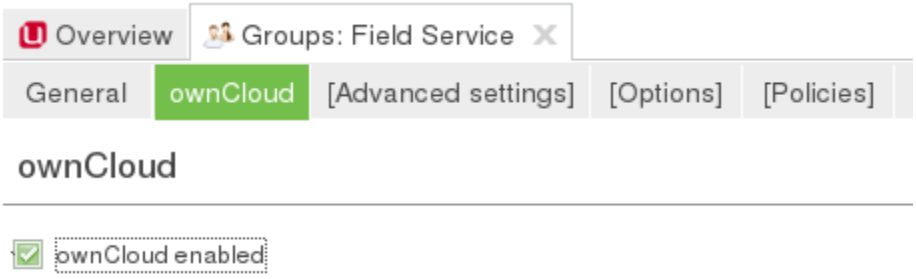
If you did not, go to the UMC and enable the users who shall have access (see picture below). Then, login at http://myserver.com/owncloud/ with your LDAP credentials.



Updating users can also be done by the script `/usr/share/owncloud/update-users.sh` . It takes the following UCR variables as parameters: **owncloud/user/enabled** for enabling or disabling, **owncloud/user/quota** as the Quota value and **owncloud/join/users/filter** as LDAP filter to select the users to update.

Groups 2012.4.0.4

Since ownCloud Enterprise 2012.4.0.4 group support is enabled. Groups, that are activated for ownCloud usage, can be used to share files to instead of single users, for example. It is also important to note, that users can only share within groups where they belong to. Groups can be enabled and disabled via UCM as shown in the screen shot below.



Another way to enable or disable groups is to use the script `/usr/share/owncloud/update-groups.sh`. Currently, it takes an argument which can be 1=enable groups or 0=disable groups. The filter applied is being taken from the UCR variable **owncloud/join/groups/filter**. In case it is empty, a message will be displayed.

2.6 Manual Installation

If you do not want to use packages, here is how you setup ownCloud on from scratch using a classic LAMP (Linux, Apache, MySQL, PHP) setup:

2.6.1 Prerequisites

To run ownCloud, your webserver must have the following installed:

- php5 (>= 5.3)
- php5-gd
- php-xml-parser

And as *optional* dependencies:

- php5-intl
- php5-sqlite (>= 3)
- php5-mysql
- php5-pgsql (or php-pgsql depending on your distro)
- smbclient
- php5-curl
- curl
- libcurl3

You have to install at least one of php5-sqlite, php5-pgsql or php5-mysql, depending on which of the three database systems (SQLite, PostgreSQL or MySQL) you want to use and activate its PDO module in the **php.ini**.

smbclient is only used if you want to mount SMB shares to your ownCloud. The curl packages are needed for some apps (e.g. http user authentication)

Commands for Ubuntu and Debian (run as root):

```
apt-get install apache2 php5 php5-gd php-xml-parser php5-intl
apt-get install php5-sqlite php5-mysql php5-pgsql smbclient curl libcurl3 php5-curl
```

If you are running Ubuntu 10.04 LTS you will need to update your PHP from this PHP PPA:

```
sudo add-apt-repository ppa:ondrej/php5
sudo apt-get update
sudo apt-get install php5
```

Todo

Document other distributions.

You don't need any WebDAV support of your webserver (i.e. apache's mod_webdav) to access your ownCloud data via WebDAV, ownCloud has a WebDAV server built in. In fact, you should make sure that any built-in WebDAV module of

your webserver is disabled (at least for the ownCloud directory), as it can interfere with ownCloud's built-in WebDAV support.

2.6.2 Extract ownCloud and Copy to Your Webserver

```
tar -xjf path/to/downloaded/owncloud-x.x.x.tar.bz2
cp -r owncloud /path/to/your/webserver
```

2.6.3 Set the Directory Permissions

The owner of your webserver must own the apps/, data/ and config/ directories in your ownCloud install. You can do this by running the following command for the apps, data and config directories.

For debian based distros like Ubuntu, Debian or Linux Mint and Gentoo use:

```
chown -R www-data:www-data /path/to/your/owncloud/install/data
```

For ArchLinux use:

```
chown -R http:http /path/to/your/owncloud/install/data
```

Fedora users should use:

```
chown -R apache:apache /path/to/your/owncloud/install/data
```

Note: The **data/** directory will only be created after setup has run (see below) and is not present by default in the tarballs.

2.6.4 Enable .htaccess and mod_rewrite if Running Apache

If you are running the apache webserver, it is recommended that you enable **.htaccess** files as ownCloud uses them to enhance security and allows you to use webfinger. To enable .htaccess files you need to ensure that **AllowOverride** is set to **All** in the **Directory /var/www/** section of your virtual host file. This is usually in `/etc/apache2/sites-enabled/000-default`. If your distro supports **a2enmod** run the following commands:

```
a2enmod rewrite
```

In distros that do not come with a2enmod the `/etc/httpd/httpd.conf` needs to be changed to enable **mod_rewrite**

Then restart apache. For Ubuntu systems (or distros using updstart) use:

```
service apache2 restart
```

For systemd systems (fedora, ArchLinux, Fedora, OpenSuse) use:

```
systemctl restart httpd.service
```

In order for the maximum upload size to be configurable, the .htaccess file in the ownCloud folder needs to be made writable by the server.

2.6.5 Follow the Install Wizard

Open your web browser and navigate to your ownCloud instance. If you are installing ownCloud on the same machine as you will access the install wizard from, the url will be: http://localhost/ (or http://localhost/owncloud).

For basic installs we recommend SQLite as it is easy to setup (ownCloud will do it for you). For larger installs you should use MySQL or PostgreSQL. Click on the Advanced options to show the configuration options. You may enter admin credentials and let ownCloud create its own database user, or enter a preconfigured user. If you are not using apache as the webserver, please set the data directory to a location outside of the document root. See the advanced install settings.

2.6.6 Test your Installation

Login and start using ownCloud. Check your web servers errror log. If it shows error, you might have missed a dependency or hit a bug with your particular configuration.

If you plan on using the Webfinger app and your ownCloud installation is not in the webroot then you'll have to manually link `/var/www/.well-known` to `/path/to/your/owncloud/.well-known`.

2.7 Other Web Servers

The most popular server choice for ownCloud is Apache, which is why it is also the combinations tested best. However, it is also possible to run ownCloud on other web servers. This section does not cover Microsoft Internet Information Services (IIS), it is covered in the *Windows 7 and Windows Server 2008* section.

2.7.1 Nginx Configuration

- You need to insert the following code into **your nginx config file.**
- Adjust **server_name**, **root**, **ssl_certificate** and **ssl_certificate_key** to suit your needs.
- Make sure your SSL certificates are readable by the server (see http://wiki.nginx.org/HttpSslModule).

```
server {
        listen 80;
        server_name cloud.example.com;
        return 301 https://$server_name$request_uri;  # enforce https
}

server {
        listen 443 ssl;
        server_name cloud.example.com;

        ssl_certificate /etc/ssl/nginx/cloud.example.com.crt;
        ssl_certificate_key /etc/ssl/nginx/cloud.example.com.key;

        # Path to the root of your installation
        root /var/www/;

        client_max_body_size 10G; # set max upload size
        fastcgi_buffers 64 4K;

        rewrite ^/caldav(.*)$ /remote.php/caldav$1 redirect;
        rewrite ^/carddav(.*)$ /remote.php/carddav$1 redirect;
```

```
        rewrite ^/webdav(.*)$ /remote.php/webdav$1 redirect;

        index index.php;
        error_page 403 /core/templates/403.php;
        error_page 404 /core/templates/404.php;

        location = /robots.txt {
            allow all;
            log_not_found off;
            access_log off;
        }

        location ~ ^/(data|config|\.ht|db_structure\.xml|README) {
                deny all;
        }

        location / {
                # The following 2 rules are only needed with webfinger
                rewrite ^/.well-known/host-meta /public.php?service=host-meta last;
                rewrite ^/.well-known/host-meta.json /public.php?service=host-meta-json last;

                rewrite ^/.well-known/carddav /remote.php/carddav/ redirect;
                rewrite ^/.well-known/caldav /remote.php/caldav/ redirect;

                rewrite ^(/core/doc/[^\/]+/)$ $1/index.html;

                try_files $uri $uri/ index.php;
        }

        location ~ ^(.+?\.php)(/.*)?$ {
                try_files $1 = 404;

                include fastcgi_params;
                fastcgi_param SCRIPT_FILENAME $document_root$1;
                fastcgi_param PATH_INFO $2;
                fastcgi_param HTTPS on;
                fastcgi_pass 127.0.0.1:9000;
                # Or use unix-socket with 'fastcgi_pass unix:/var/run/php5-fpm.sock;'
        }

        # Optional: set long EXPIRES header on static assets
        location ~* ^.+\.(jpg|jpeg|gif|bmp|ico|png|css|js|swf)$ {
                expires 30d;
                # Optional: Don't log access to assets
                access_log off;
        }

}
```

Note: You can use Owncloud without SSL/TLS support, but we strongly encourage you not to do that:

- Remove the server block containing the redirect
- Change **listen 443 ssl** to **listen 80;**
- Remove **ssl_certificate** and **ssl_certificate_key**.
- Remove **fastcgi_params HTTPS on;**

Note: If you want to effectively increase maximum upload size you will also have to modify your **php-fpm configuration** (**usually at /etc/php5/fpm/php.ini**) and increase **upload_max_filesize** and **post_max_size** values. You'll need to restart php5-fpm and nginx services in order these changes to be applied.

2.7.2 Lighttpd Configuration

This assumes that you are familiar with installing PHP application on lighttpd.

It is important to note that the **.htaccess** files used by ownCloud to protect the **data** folder are ignored by lighttpd, so you have to secure it by yourself, otherwise your **owncloud.db** database and user data are publicly readable even if directory listing is off. You need to add two snippets to your lighttpd configuration file:

Disable access to data folder:

```
$HTTP["url"] =~ "^/owncloud/data/" {
     url.access-deny = ("")
   }
```

Disable directory listing:

```
$HTTP["url"] =~ "^/owncloud($|/)" {
     dir-listing.activate = "disable"
   }
```

2.7.3 Yaws Configuration

This should be in your **yaws_server.conf**. In the configuration file, the **dir_listings = false** is important and also the redirect from **/data** to somewhere else, because files will be saved in this directory and it should not be accessible from the outside. A configuration file would look like this

```
<server owncloud.myserver.com/>
        port = 80
        listen = 0.0.0.0
        docroot = /var/www/owncloud/src
        allowed_scripts = php
        php_handler = <cgi, /usr/local/bin/php-cgi>
        errormod_404 = yaws_404_to_index_php
        access_log = false
        dir_listings = false
        <redirect>
                /data == /
        </redirect>
</server>
```

The apache **.htaccess** file that comes with ownCloud is configured to redirect requests to nonexistent pages. To emulate that behaviour, you need a custom error handler for yaws. See this github gist for further instructions on how to create and compile that error handler.

2.7.4 Hiawatha Configuration

Add **WebDAVapp = yes** to the ownCloud virtual host. Users accessing WebDAV from MacOS will also need to add **AllowDotFiles = yes**.

Disable access to data folder:

```
UrlToolkit {
    ToolkitID = denyData
    Match ^/data DenyAccess
}
```

2.7.5 PageKite Configuration

You can use this PageKite how to to make your local ownCloud accessible from the internet using PageKite.

2.7.6 Open Wrt

Here you can find a **'tutorial for open Wrt'_**

CHAPTER

THREE

CONFIGURATION

3.1 User Authentication with LDAP

ownCloud ships an LDAP backend, which allows full use of ownCloud for user logging in with LDAP credentials including:

- LDAP group support
- File sharing with users and groups
- Access via WebDAV and of course ownCloud Desktop Client
- Versioning, external Storages and all other ownCloud Goodies

To connect to an LDAP server the configuration needs to be set up properly. Once the LDAP backend is activated (Settings→Apps, choose **LDAP user and group backend**, click on **Enable**) the configuration can be found on Settings→Admin. Read on for a detailed description of the configuration fields.

3.1.1 Basic Settings

The basic settings are all you need. However, if you have a larger directory, custom requirements or need to connect to Active Directory (AD) you want to have a look on the advanced settings afterwards. The basic part allows you to set up a working connection to your LDAP server and use it with ownCloud.

Note that a hint will be shown on the right hand side, when hovering with the mouse over an input field. This gives you more context information while filling out the settings.

Settings Details

Server configuration: ownCloud can be configured to connect to multiple LDAP servers. Using this control you can pick a configuration you want to edit or add a new one. The button **Delete Configuration** deletes the current configuration.

- *Example: 1. Server*

Host: The host name of the LDAP server. It can also be a **ldaps://** URI, for instance.

- Example: *directory.my-company.com*

Base DN: The base DN of LDAP, from where all users and groups can be reached. Separated Base DNs for users and groups can be set in the Advanced tab. Nevertheless, this field is mandatory.

- Example: *dc=my-company,dc=com*

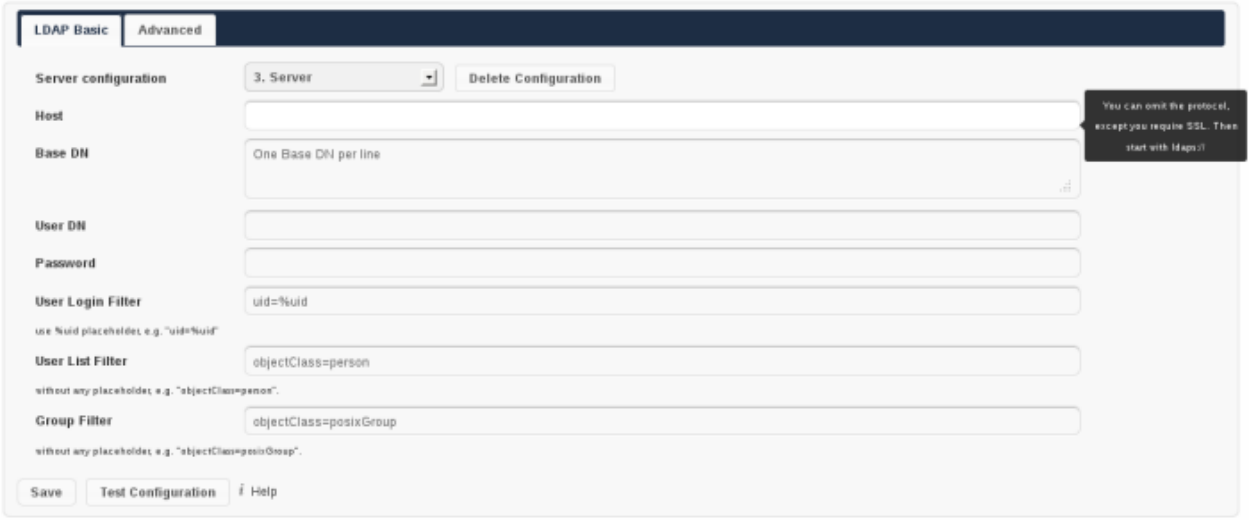


Figure 3.1: LDAP Basic Settings

User DN: The name as DN of a user who is able to do searches in the LDAP directory. Let it empty for anonymous access. It is recommended to have a special system user for ownCloud.

- Example: *uid=owncloudsystemuser,cn=sysusers,dc=my-company,dc=com*

Password: The password for the user given above. Empty for anonymous access.

User Login Filter: The filter to use when a users tries to login. Use **%uid** as placeholder for the user name. Note, that login applies this filter only, but not User List Filter. This may change in future.

- Example (allows login with user name and email address): *(|(uid=%uid)(email=$uid))*

User List Filter: The filter to use when a search for users will be executed.

- Example: *objectClass=posixAccount*

Group Filter: The filter to use when a search for groups will be executed. In case you do not want to use LDAP groups in ownCloud, leave it empty.

- Example: *objectClass=posixGroup*

3.1.2 Advanced Settings

In the LDAP Advanced settings section you can define options, that are less common to set. They are not needed for a working connection, unless you use a non-standard Port, e.g. It can also have a positive effect on the performance to specify distinguished bases for user and group searches.

The Advanced Settings are structured into three parts: * Connection Settings * Directory Settings * Special Attributes

Connection Settings

Configuration Active: Enables or Disables the current configuration. Disabled configuration will not connect to the LDAP server.

- Example: *[X]*

Port: The port on which to connect to the LDAP server.

Figure 3.2: LDAP Advanced Settings

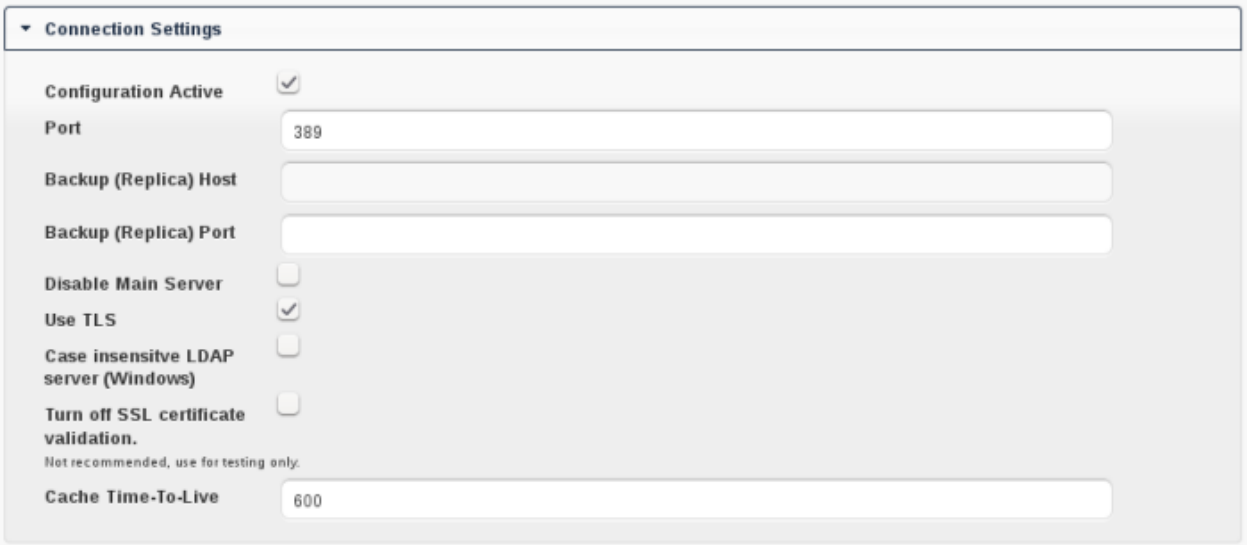


Figure 3.3: LDAP Advanced Settings, section Connection Settings

- Example: *389*

Backup (Replica) Host: A backup server can be defined here. ownCloud tries to connect to the backup server automatically, when the main host (as specified in basic settings) cannot be reached. It is import that the backup server is a replica of the main server, because the object UUIDs must match.

- Example: *directory2.my-company.com*

Backup (Replica) Port: The port on which to connect to the backup LDAP server. If no port is given, but a host, then the main port (as specified above) will be used.

- Example: *389*

Disable Main Server: You can manually override the main server and make ownCloud only connect to the backup server. It may be handy for planned downtimes.

- Example: *[ ]*

Use TLS: Whether to use TLS encrypted connection to the LDAP server. This will be ignored when "ldaps://" protocol is specified in the host entries.

- Example: *[ ]*

Case insensitive LDAP server (Windows): Whether the LDAP server is running on a Windows Host

- Example: *[ ]*

Turn off SSL certificate validation: Turns of check of valid SSL certificates. Use it – if needed – for testing, only!

- Example: *[ ]*

Cache Time-To-Live: A cache is introduced to avoid unnecessary LDAP traffic, for example lookups check whether the users exists on every page request or WebDAV interaction. It is also supposed to speed up the Admin → User page or list of users to share with, once it is populated. Saving the configuration empties the cache (changes are not necessary). The time is given in seconds.

Note that almost every PHP request would require to build up a new connection to the LDAP server. If you require a most up-to-dateness it is recommended not to totally switch off the cache, but define a minimum life time of 15s.

- Example (10 min): *600*

Directory Settings

User Display Name Field: The attribute that should be used as display name in ownCloud. Prior to ownCloud 5 it was used as internal user name. This is not the case anymore. It also means that display names are not permanent in ownCloud, i.e. if the attribute's value changes in LDAP, it changes in ownCloud too. Display names to not need to be unique, but you rather want to specify a more or less unique attribute here to avoid confusion.

- Example: *displayName*

Base User Tree: The base DN of LDAP, from where all users can be reached. It needs to be given completely despite to the Base DN from the Basic settings. You can specifiy multiple base trees, one in each line.

- Example:

cn=programmers,dc=my-company,dc=com
cn=designers,dc=my-company,dc=com

Figure 3.4: LDAP Advanced Settings, section Directory Settings

User Search Attributes: These attributes are used when a search for users with a search string is done. This happens, for instance, in the share dialogue. By default the user display name attribute as specified above is being used. Multiple attributes can be given, one in each line.

- Example:

  displayName
  mail

Group Display Name Field: The attribute that should be used as ownCloud group name. ownCloud allows a limited set of characters (a-zA-Z0-9.-_@), every other character will be replaced in ownCloud. Once a group name is assigned, it will not be changed, i.e. changing this value will only have effect to new LDAP groups.

- Example: *cn*

Base Group Tree: The base DN of LDAP, from where all groups can be reached. It needs to be given completely despite to the Base DN from the Basic settings. You can specifiy multiple base trees, one in each line.

- Example:

  cn=barcelona,dc=my-company,dc=com
  cn=madrid,dc=my-company,dc=com

Group Search Attributes: These attributes are used when a search for groups with a search string is done. This happens, for instance, in the share dialogue. By default the group display name attribute as specified above is being used. Multiple attributes can be given, one in each line.

- Example:

cn
description

Group Member association: The attribute that is used to indicate group memberships, i.e. the attribute used by LDAP groups to refer to their users.

- Example: *uniquemember*

Special Attributes

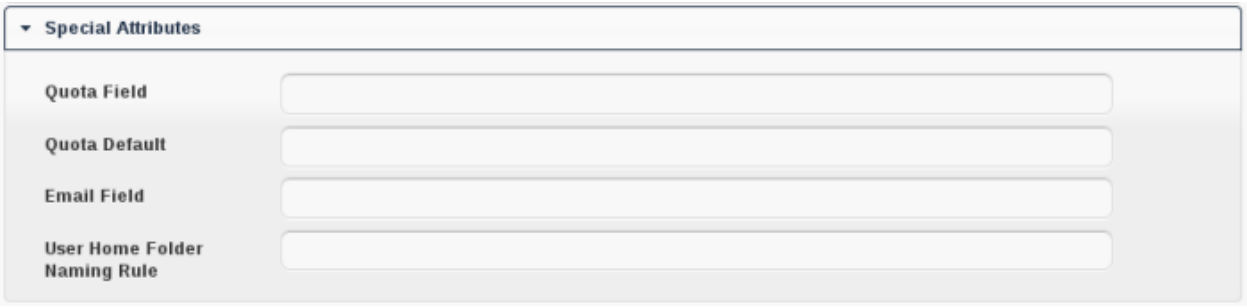


Figure 3.5: LDAP Advanced Settings, section Special Attributes

Quota Field: ownCloud can read an LDAP attribute and set the user quota according to its value. Specify the attribute here, otherwise keep it empty. The attribute shall return human readable values, e.g. "2 GB".

- Example: *ownCloudQuota*

Quota Default: Override ownCloud default quota for LDAP users who do not have a quota set in the attribute given above.

- Example: *15 GB*

Email Field: ownCloud can read an LDAP attribute and set the user email there from. Specify the attribute here, otherwise keep it empty.

- Example: *mail*

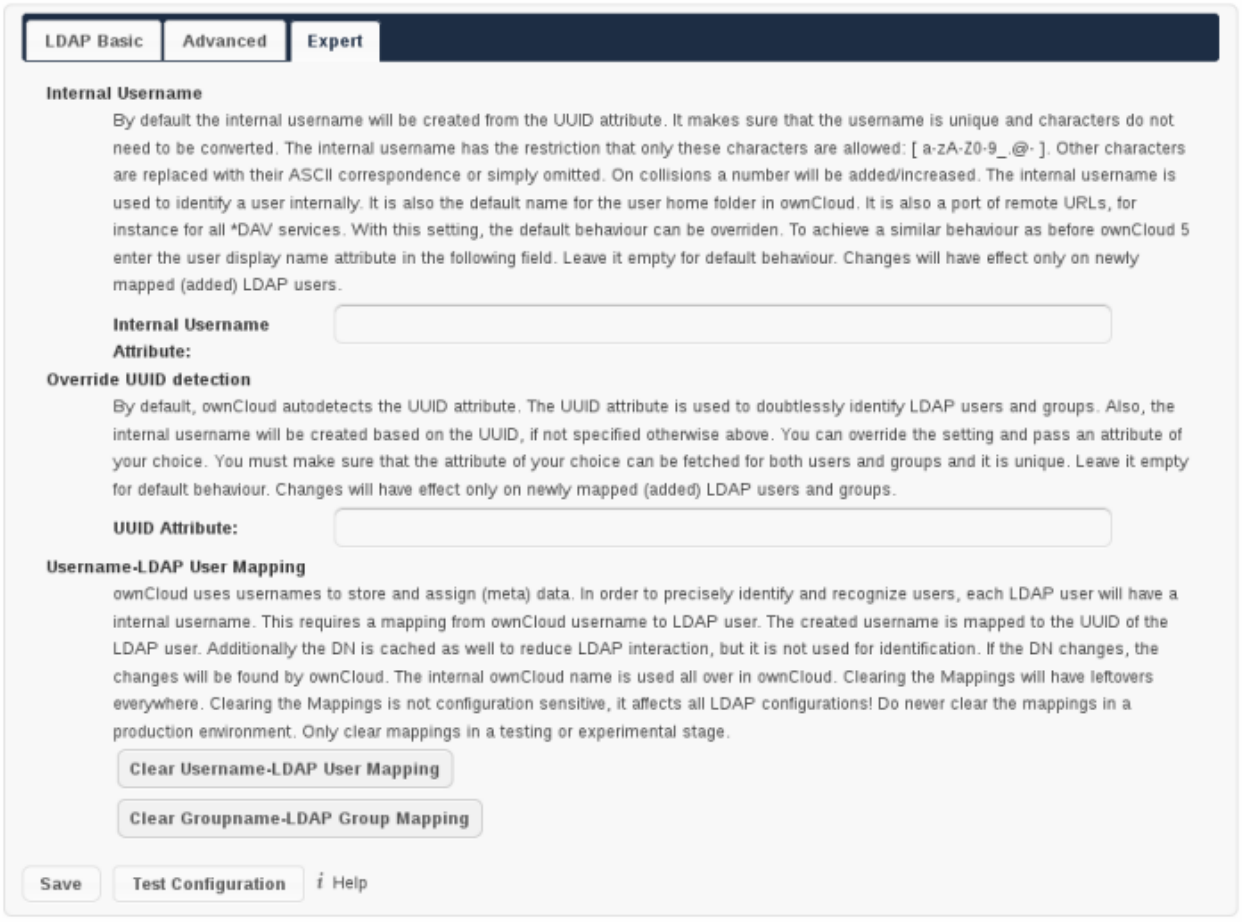
User Home Folder Naming Rule: By default, the ownCloud creates the user directory, where all files and meta data are kept, according to the ownCloud user name. You may want to override this setting and name it after an attribute's value. The attribute given can also return an absolute path, e.g. `/mnt/storage43/alice`. Leave it empty for default behavior.

- Example: *cn*

3.1.3 Expert Settings (>= ownCloud 5.0.7)

In the Expert Settings fundamental behavior can be adjusted to your needs. The configuration should be done before starting production use or when testing the installation.

Internal Username: The internal username is the identifier in ownCloud for LDAP users. By default it will be created from the UUID attribute. By using the UUID attribute it is made sure that the username is unique and characters do not need to be converted. The internal username has the restriction that only these characters are allowed: [a-zA-Z0-9_.@-]. Other characters are replaced with their ASCII correspondence or are simply omitted.


LDAP Basic
Advanced
Expert
Internal Username
By default the internal username will be created from the UUID attribute. It makes sure that the username is unique and characters do not need to be converted. The internal username has the restriction that only these characters are allowed: [a-zA-Z0-9_.@-]. Other characters are replaced with their ASCII correspondence or simply omitted. On collisions a number will be added/increased. The internal username is used to identify a user internally. It is also the default name for the user home folder in ownCloud. It is also a port of remote URLs, for instance for all *DAV services. With this setting, the default behaviour can be overriden. To achieve a similar behaviour as before ownCloud 5 enter the user display name attribute in the following field. Leave it empty for default behaviour. Changes will have effect only on newly mapped (added) LDAP users.
Internal Username Attribute:
Override UUID detection
By default, ownCloud autodetects the UUID attribute. The UUID attribute is used to doubtlessly identify LDAP users and groups. Also, the internal username will be created based on the UUID, if not specified otherwise above. You can override the setting and pass an attribute of your choice. You must make sure that the attribute of your choice can be fetched for both users and groups and it is unique. Leave it empty for default behaviour. Changes will have effect only on newly mapped (added) LDAP users and groups.
UUID Attribute:
Username-LDAP User Mapping
ownCloud uses usernames to store and assign (meta) data. In order to precisely identify and recognize users, each LDAP user will have a internal username. This requires a mapping from ownCloud username to LDAP user. The created username is mapped to the UUID of the LDAP user. Additionally the DN is cached as well to reduce LDAP interaction, but it is not used for identification. If the DN changes, the changes will be found by ownCloud. The internal ownCloud name is used all over in ownCloud. Clearing the Mappings will have leftovers everywhere. Clearing the Mappings is not configuration sensitive, it affects all LDAP configurations! Do never clear the mappings in a production environment. Only clear mappings in a testing or experimental stage.
Clear Username-LDAP User Mapping
Clear Groupname-LDAP Group Mapping
Save
Test Configuration
Help

The LDAP backend ensures that there are no duplicate internal usernames in ownCloud, i.e. that it is checking all other activated user backends (including local ownCloud users). On collisions a random number (between 1000 and 9999) will be attached to the retrieved value. For example, if "alice" exists, the next username may be "alice_1337".

The internal username is also the default name for the user home folder in ownCloud. It is also a part of remote URLs, for instance for all *DAV services. With this setting the default behaviour can be overriden. To achieve a similar behaviour as before ownCloud 5 enter the user display name attribute in the following field.

Leave it empty for default behaviour. Changes will have effect only on newly mapped (added) LDAP users.

- Example: *uid*

Override UUID detection By default, ownCloud autodetects the UUID attribute. The UUID attribute is used to doubtlessly identify LDAP users and groups. Also, the internal username will be created based on the UUID, if not specified otherwise above.

You can override the setting and pass an attribute of your choice. You must make sure that the attribute of your choice can be fetched for both users and groups and it is unique. Leave it empty for default behaviour. Changes will have effect only on newly mapped (added) LDAP users and groups. It also will have effect when a user's or group's DN changes and an old UUID was cached: It will result in a new user. Because of this, the setting should be applied before putting ownCloud in production use and cleaning the bindings (see below).

The default behaviour does not differ from ownCloud 4.5. You do not want to change this after upgrading from ownCloud 4.5 unless you update the mapping tables yourself.

- Example: *cn*

Username-LDAP User Mapping ownCloud uses the usernames as key to store and assign data. In order to precisely identify and recognize users, each LDAP user will have a internal username in ownCloud. This requires a mapping from ownCloud username to LDAP user. The created username is mapped to the UUID of the LDAP user. Additionally the DN is cached as well to reduce LDAP interaction, but it is not used for identification. If the DN changes, the change will be detected by ownCloud by checking the UUID value.

The same is valid for groups.

The internal ownCloud name is used all over in ownCloud. Clearing the Mappings will have leftovers everywhere. Do never clear the mappings in a production environment. Only clear mappings in a testing or experimental stage.

Clearing the Mappings is not configuration sensitive, it affects all LDAP configurations!

3.1.4 Testing the configuration

In this version we introduced the **Test Configuration** button on the bottom of the LDAP settings section. It will always check the values as currently given in the input fields. You do not need to save before testing. By clicking on the button, ownCloud will try to bind to the ownCloud server with the settings currently given in the input fields. The response will look like this:

In case the configuration fails, you can see details in ownCloud's log, which is in the data directory and called **owncloud.log** or on the bottom the **Settings** → **Admin page**. Unfortunately it requires a reload – sorry for the inconvenience.

In this case, Save the settings. You can check if the users and groups are fetched correctly on the Settings → Users page.

Figure 3.6: Failure

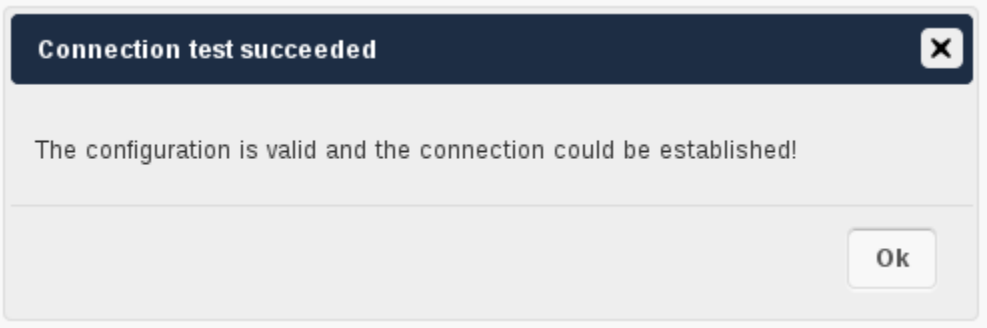


Figure 3.7: Success

3.1.5 Troubleshooting, Tips and Tricks

3.1.6 SSL Certificate Verification (LDAPS, TLS)

A common mistake with SSL certificates is that they may not be known to PHP. If you have trouble with certificate validation make sure that

- you have the certificate of the server installed on the ownCloud server
- the certificate is announced in the system's LDAP configuration file (usually */etc/ldap/ldap.conf* on Linux, *C:\openldap\sysconf\ldap.conf* or *C:\ldap.conf* on Windows) using a **TLS_CACERT /path/to/cert** line.
- Using LDAPS, also make sure that the port is correctly configured (by default 686)

3.1.7 Microsoft Active Directory

In case you want to connect to a Windows AD, you must change some values in the Advanced tab.

- The default login filter will not work with AD. Use "samaccountname=%uid" instead.
- The default in User Display Name Field will not work with Active Directory.
- The Group Member association must be set to "member (AD)"
- Check Case insensitive LDAP server (Windows)

3.1.8 Duplicating Server Configurations

In case you have a working configuration and want to create a similar one or "snapshot" configurations before modifying them you can do the following:

1. Go to the **LDAP Basic** tab
2. On **Server Configuration** choose *Add Server Configuration*
3. Answer the question *Take over settings from recent server configuration?* with *yes*.
4. (optional) Switch to **Advanced** tab and uncheck **Configuration Active** in the *Connection Settings*, so the new configuration is not used on Save
5. Click on **Save**

Now you can modify the configuration and enable it if you wish.

3.1.9 ownCloud LDAP Internals

Some parts of how the LDAP backend works are described here. May it be helpful.

3.1.10 User and Group Mapping

In ownCloud the user or group name is used to have all relevant information in the database assigned. To work reliably a permanent internal user name and group name is created and mapped to the LDAP DN and UUID. If the DN changes in LDAP it will be detected, there will be no conflicts.

Those mappings are done in the database table ldap_user_mapping and ldap_group_mapping. The user name is also used for the user's folder (except something else is specified in *User Home Folder Naming Rule*), which contains files and meta data.

As of ownCloud 5 internal user name and a visible display name are separated. This is not the case for group names, yet, i.e. group cannot be altered.

That means that your LDAP configuration should be good and ready before putting it into production. The mapping tables are filled early, but as long as you are testing, you can empty the tables any time. Do not do this in production. If you want to rename a group, be very careful. Do not rename the user's internal name.

3.1.11 Caching

For performance reasons a cache has been introduced to ownCloud. He we store all users and groups, group memberships or internal userExists-requests. Since ownCloud is written in PHP and each and every page request (also done by Ajax) loads ownCloud and would execute one or more LDAP queries again, you do want to have some of those queries cached and save those requests and traffic. It is highly recommended to have the cache filled for a small amount of time, which comes also very handy when using the sync client, as it is yet another request for PHP.

3.1.12 Handling with Backup Server

When ownCloud is not able to contact the main server, he will be treated as offline and no connection attempts will be done for the time specified in **Cache Time-To-Live**. If a backup server is configured, it will be connected instead. If you plan a maintained downtime, check **Disable Main Server** for the time being to avoid unnecessary connection attempts every now and then.

3.2 Background Jobs

A system like ownCloud sometimes requires tasks to be done on a regular base without blocking the user interface. For that purpose you, as a system administrator, can define background jobs which make it possible to execute tasks without any need of user interaction, e.g. database clean-ups etc. For the sake of completeness it is worth to know that additionally background jobs can also be defined by installed apps.

3.2.1 Parameters

In the admin settings menu you can configure how cron-jobs should be executed. You can choose between the following options:

- AJAX
- Webcron
- Cron

3.2.2 Cron-Jobs

OwnCloud requires various automated background jobs to be run. There are three methods to achieve this. The default way is AJAX and the recommended way is cron.

AJAX

This option is the default option, although it is the least reliable. Every time a user visits the ownCloud page a single background job will be executed. The advantage of this mechanism is, that is does not require access to the system nor registration at a third party service. The disadvantage of this solution compared to the Webcron service is, that it requires regular visits of the page to get triggered.

Webcron

By registering your ownCloud `cron.php` script address at an external webcron service, like e.g. easyCron, you ensure that background jobs will be executed regularly. To use such a service your server need to be reachable via the Internet.

Example

```
URL to call: http[s]://<domain-of-your-server>/owncloud/cron.php
```

Cron

Using the systems cron feature is the preferred way to run regular tasks, because it allows to execute jobs without the limitations which a web server may have.

Example

To run a cron job on a *nix system, e.g. every 15min, under the default webserver user, e.g. **www-data*, you need to set-up the following cron job to call the **cron.php** script. Please check the crontab man page for the exact command syntax.

```
# crontab -u www-data -e
*/15  *  *  *  * php -f /var/www/owncloud/cron.php
```

3.3 3rd-Party Configuration

ownCloud resorts to some 3rd-party PHP components to provide its functionality. These components are part of the software package and are usually shipped in the **/3rdparty** folder.

3.3.1 Parameters

If you want to change the default location of the 3rd-party folder you can use the **3rdpartyroot** parameter to define the absolute file system path to the folder. The **3rdpartyurl** parameter is used to define the http web path to that folder, starting at the ownCloud web root.

```
<?php

"3rdpartyroot" => OC::$SERVERROOT."/3rdparty",
"3rdpartyurl"  => "/3rdparty",
```

3.4 Apps Configuration

After you have installed ownCloud you might realize that it would be nice to provide an additional function on top of the core functionality in your ownCloud installation.

The first step should be to check out the ownCloud apps store. There you will find a lot of ready-to-use apps provided by the ownCloud community.

3.4.1 Parameters

Parameters are set in the `config/config.php` inside the **$CONFIG** array.

Use custom app directories

Use the **apps_paths** array to set the apps folders which should be scanned for available apps and/or where user specific apps should be installed.The key **path** defines the absolute file system path to the app folder. The key **url** defines the http web path to that folder, starting at the ownCloud web root. The key **writable** indicates if a user can install apps in that folder.

Note: If you want to make sure that the default **/apps/** folder only contains apps shipped with ownCloud, you should follow the example and set-up a **/apps2/** folder which will be used to store all apps downloaded by users

---

```
<?php

  "apps_paths" => array (
      0 => array (
              "path"     => OC::$SERVERROOT."/apps",
              "url"      => "/apps",
              "writable" => false,
      ),
      1 => array (
              "path"     => OC::$SERVERROOT."/apps2",
              "url"      => "/apps2",
              "writable" => true,
```

```
    ),
),
```

Use your own appstore

If you want to allow the installation of apps from the apps store you have to set **appstoreenabled** parameter, but this can only be done if at least one of the configured apps directories is writeable.

The **appstoreurl** is used to set the http path to the ownCloud apps store. The appstore server has to use OCS (Open Collaboration Services).

```
<?php

  "appstoreenabled" => true,
  "appstoreurl" => "http://api.apps.owncloud.com/v1",
```

Guard against malicious 3rdparty code

Finally you can enable checks for malicious code fragments of 3rd-party apps by setting the **appcodechecker** parameter.

```
<?php

  "appcodechecker" => false,
```

3.5 Automatic Configuration

If you need to install ownCloud on multiple servers you normally do not want to set-up each instance separately as described in the *Database Configuration*. For this reason the automatic configuration feature has been introduced.

To take advantage of this feature you need to create a configuration file, called `../owncloud/config/autoconfig.php` and set the parameters as required. You can provide all parameters or just part of them - parameters which haven't been provided (if any) will be asked at "Finish setup" screen at first run of ownCloud.

The `../owncloud/config/autoconfig.php` will be automatically removed after the initial configuration has been applied.

3.5.1 Parameters

You need to keep in mind that two parameters are named differently in this configuration file compared to the normal `config.php`.

autoconfig.php	config.php
directory	datadirectory
dbpass	dbpassword

3.5.2 Sample Automatic Configurations

Data Directory

With the configuration below the "Finish setup" screen still will ask for database and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
  "directory"     => "/www/htdocs/owncloud/data",
);
```

SQLite Database

With the configuration below the "Finish setup" screen still will ask for data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "sqlite",
  "dbname"        => "owncloud",
  "dbtableprefix" => "",
);
```

MySQL Database

Keep in mind that the automatic configuration does not unburden you from creating the database user and database in advance, as described in *Database Configuration*.

With the configuration below the "Finish setup" screen still will ask for data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "mysql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
);
```

PostgreSQL Database

Keep in mind that the automatic configuration does not unburden you from creating the database user and database in advance, as described in *Database Configuration*.

With the configuration below the "Finish setup" screen still will ask for data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "pgsql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
);
```

All Parameters

Keep in mind that the automatic configuration does not unburden you from creating the database user and database in advance, as described in *Database Configuration*.

With the configuration below "Finish setup" will be skipped at first ownCloud run since all parameters are already preconfigured.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "mysql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
  "adminlogin"    => "root",
  "adminpass"     => "root-password",
  "directory"     => "/www/htdocs/owncloud/data",
);
```

3.6 Custom Client Configuration

If you want to access your ownCloud, you can choose between the standard Web-GUI and different client sync applications. Download links which point to these applications are shown at the top of the personal menu. The following sync applications are currently available out of the box:

- Desktop sync clients for Windows, Max and Linux OS
- Mobile sync client for Android devices
- Mobile sync client for iOS devices

3.6.1 Parameters

If you want to customize the download links for the sync clients the following parameters need to be modified to fulfil your requirements:

```
<?php

  "customclient_desktop" => "http://owncloud.org/sync-clients/",
  "customclient_android" => "https://play.google.com/store/apps/details?id=com.owncloud.android",
  "customclient_ios"     => "https://itunes.apple.com/us/app/owncloud/id543672169?mt=8",
```

This parameters can be set in the `config/config.php`

3.7 Database Configuration

Owncloud requires a database where administrative data will be held. Four different database types are currently supported, MySQL, MariaDB, SQLite, and PostgreSQL. MySQL or MariaDB are the recommended database engines. By default SQLite is choosen because it is a file based database with the least administrative overhead.

Note: Because SQLite handles multiple users very badly SQLite is only recommended for single user ownCloud installations

3.7.1 Requirements

If you decide to use MySQL, MariaDB, or PostgreSQL you need to install and set-up the database first. These steps will not be covered by this description as they are easy to find elsewhere.

3.7.2 Parameters

MySQL/MariaDB Database

If you decide to use a MySQL or MariaDB database make sure that you have installed and enabled the MySQL extension in PHP and that the **mysql.default_socket** points to the correct socket (if the database runs on same server as ownCloud).

Please note that MariaDB is backwards compatible with MySQL, so all instructions will work for both. You will not need to replace mysql with anything.

The PHP configuration in `/etc/php5/conf.d/mysql.ini` could look like this:

```
# configuration for PHP MySQL module
extension=pdo_mysql.so
extension=mysql.so

[mysql]
mysql.allow_local_infile=On
mysql.allow_persistent=On
mysql.cache_size=2000
mysql.max_persistent=-1
mysql.max_links=-1
mysql.default_port=
mysql.default_socket=/var/lib/mysql/mysql.sock  # debian squeeze: /var/run/mysqld/mysqld.sock
mysql.default_host=
mysql.default_user=
mysql.default_password=
mysql.connect_timeout=60
mysql.trace_mode=Off
```

Now you need to create a database user and the database itself by using the MySQL command line interface. The database tables will be created by ownCloud when you login for the first time.

To start the MySQL command line mode use:

```
mysql -uroot -p
```

Then a **mysql>** or **MariaDB [root]>** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE IF NOT EXISTS owncloud;
GRANT ALL PRIVILEGES ON owncloud.* TO 'username'@'localhost' IDENTIFIED BY 'password';
```

You can quit the prompt by entering:

```
quit
```

In the ownCloud configuration you need to set the hostname on which the database is running and a valid username and password to access it.

```
<?php

  "dbtype"        => "mysql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
```

SQLite Database

If you decide to use a SQLite database make sure that you have installed and enabled the SQLite extension in PHP. The PHP configuration in `/etc/php5/conf.d/sqlite3.ini` could look like this:

```
# configuration for PHP SQLite3 module
extension=pdo_sqlite.so
extension=sqlite3.so
```

It is not necessary to create a database and a database user in advance because this will automatically be done by ownCloud when you login for the first time.

In the ownCloud counfiguration in `config/config.php` you need to set at least the **datadirectory** parameter to the directory where your data and database should be stored. Note that for the PDO SQLite driver this directory must be writable (this is recommended for ownCloud anyway). No authentication is required to access the database therefore most of the default parameters could be taken as is:

```
<?php

  "dbtype"        => "sqlite",
  "dbname"        => "owncloud",
  "dbuser"        => "",
  "dbpassword"    => "",
  "dbhost"        => "",
  "dbtableprefix" => "",
  "datadirectory" => "/www/htdocs/owncloud/data",
```

PostgreSQL Database

If you decide to use a PostgreSQL database make sure that you have installed and enabled the PostgreSQL extension in PHP. The PHP configuration in `/etc/php5/conf.d/pgsql.ini` could look like this:

```
# configuration for PHP PostgreSQL module
extension=pdo_pgsql.so
extension=pgsql.so

[PostgresSQL]
pgsql.allow_persistent = On
pgsql.auto_reset_persistent = Off
pgsql.max_persistent = -1
pgsql.max_links = -1
```

```
pgsql.ignore_notice = 0
pgsql.log_notice = 0
```

Now you need to create a database user and the database itself by using the PostgreSQL command line interface. The database tables will be created by ownCloud when you login for the first time.

To start the postgres command line mode use:

```
psql -hlocalhost -Upostgres
```

Then a **postgres=#** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER username WITH PASSWORD 'password';
CREATE DATABASE owncloud TEMPLATE template0 ENCODING 'UNICODE';
ALTER DATABASE owncloud OWNER TO username;
GRANT ALL PRIVILEGES ON DATABASE owncloud TO username;
```

You can quit the prompt by entering:

```
\q
```

In the ownCloud configuration you need to set the hostname on which the database is running and a valid username (and sometimes a password) to access it. If the database has been installed on the same server as ownCloud a password is very often not required to access the database.

```
<?php

  "dbtype"        => "pgsql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
```

Oracle Database

If you are deploying to an Oracle database make sure that you have installed and enabled the Oracle extension in PHP. The PHP configuration in `/etc/php5/conf.d/oci8.ini` could look like this:

```
# configuration for PHP Oracle extension
extension=oci8.so
```

Make sure that the Oracle environment has been set up for the process trying to use the Oracle extension. For a local Oracle XE installation this can be done by exporting the following environment variables (eg. in `/etc/apache2/envvars` for Apache)

```
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/xe
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
```

Installing and configuring Oracle support for PHP is way out of scope for this document. The official Oracle documentation called The Underground PHP and Oracle Manual should help you through the process.

Creating a database user for ownCloud can be done by using the sqlplus command line interface or the Oracle Application Express web interface. The database tables will be created by ownCloud when you login for the first time.

To start the Oracle command line mode with a DBA account use:

```
sqlplus system AS SYSDBA
```

After entering the password a **SQL>** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER owncloud IDENTIFIED BY password;
ALTER USER owncloud DEFAULT TABLESPACE users
                    TEMPORARY TABLESPACE temp
                    QUOTA unlimited ON users;
GRANT create session
    , create table
    , create procedure
    , create sequence
    , create trigger
    , create view
    , create synonym
    , alter session
   TO owncloud;
```

Note: In Oracle creating a user is the same as creating a database in other RDBMs, so no `CREATE DATABASE` statement is necessary.

You can quit the prompt by entering:

```
exit
```

In the ownCloud configuration you need to set the hostname on which the database is running and a valid username and password to access it. If the database has been installed on the same server as ownCloud to config file could look like this:

```
<?php

  "dbtype"        => "oci",
  "dbname"        => "XE",
  "dbuser"        => "owncloud",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
```

Note: This example assumes you are running an Oracle Express Edition on `localhost`. The `dbname` is the name of the Oracle instance. For Oracle Express Edition it is always `XE`.

3.7.3 Trouble Shooting

How can I find out if my MySQL/PostgreSQL server is reachable?

Use the ping command to check the server availability:

```
ping db.server.dom

PING db.server.dom (ip-address) 56(84) bytes of data.
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=1 ttl=64 time=3.64 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=2 ttl=64 time=0.055 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=3 ttl=64 time=0.062 ms
```

How can I find out if a created user can access a database?

The easiet way to test if a database can be accessed is by starting the command line interface:

SQLite:

```
sqlite3 /www/htdocs/owncloud/data/owncloud.db

sqlite> .version
SQLite 3.7.15.1 2012-12-19 20:39:10 6b85b767d0ff7975146156a99ad673f2c1a23318
sqlite> .quit
```

MySQL:

```
mysql -uUSERNAME -p

mysql> SHOW VARIABLES LIKE "version";
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| version       | 5.1.67 |
+---------------+--------+
1 row in set (0.00 sec)
mysql> quit
```

PostgreSQL:

```
psql -Uusername -downcloud

postgres=# SELECT version();
PostgreSQL 8.4.12 on i686-pc-linux-gnu, compiled by GCC gcc (GCC) 4.1.3 20080704 (prerelease), 32-bit
(1 row)
postgres=# \q
```

Oracle:

```
sqlplus username

SQL> select * from v$version;

BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
PL/SQL Release 11.2.0.2.0 - Production
CORE  11.2.0.2.0      Production
TNS for Linux: Version 11.2.0.2.0 - Production
NLSRTL Version 11.2.0.2.0 - Production

SQL> exit
```

Useful SQL commands

Show Database Users:

```
SQLite    : No database user is required.
MySQL     : SELECT User,Host FROM mysql.user;
PostgreSQL: SELECT * FROM pg_user;
Oracle    : SELECT * FROM all_users;
```

Show available Databases:

```
SQLite    : .databases (normally one database per file!)
MySQL     : SHOW DATABASES;
PostgreSQL: \l
Oracle    : SELECT name FROM v$database; (requires DBA privileges)
```

Show ownCloud Tables in Database:

```
SQLite    : .tables
MySQL     : USE owncloud; SHOW TABLES;
PostgreSQL: \c owncloud; \d
Oracle    : SELECT table_name FROM user_tables;
```

Quit Database:

```
SQLite    : .quit
MySQL     : quit
PostgreSQL: \q
Oracle    : quit
```

3.8 Use Server-Side Encryption

ownCloud ships a encryption app, which allows to encrypt all files stored in your ownCloud. Encryption and decryption always happens server-side. This enables the user to continue to use all the other apps to view and edit his data.

The app uses the user's log-in password as encryption-password. This means that by default the user will lose access to his files if he loses his log-in password.

It might be a good idea to make regular backups of all encryption keys. The encryption keys are stored in following folders:

- data/owncloud_private_key (recovery key, if enabled and public share key)
- data/public-keys (public keys from all users)
- data/<user>/files_encryption (users' private keys and all other keys necessary to decrypt the users' files)

3.8.1 Enable File Recovery Feature

The admin can offer the user some kind of protection against password loss. Therefore you have to enable the recovery key in the admin settings and provide a strong recovery key password. The admin settings also enables you to change the recovery key password if you wish. But you should make sure to never lose this password, because that's the only way to recover users' files.

Once the recovery key was enabled every user can choose in his personal settings to enable this feature or not.

3.8.2 Recover User Files

If the recovery feature was enabled the admin will see a additional input field at the top of the user management settings. After entering the recovery-key password the admin can change the user's log-in password which will automatically recover the user's file.

If you use a user back-end which doesn't allow you to change the log-in password directly within ownCloud, e.g. the LDAP back-end, than you can follow the same procedure to recover a user's files. The only difference is that you

need to change the log-in password additionally at your back-end. In this case make sure to use both times the same password.

3.8.3 LDAP and other external user back-ends

if you configure a external user back-end you will be able to change the user's log-in password at the back-end. Since the encryption password must be the same as the user's log-in password this will result in a non-functional encryption system. If the recovery feature was enabled, the administrator will be able to recover the user's files directly over the recovery feature. See the description above. Otherwise the user will be informed that his log-in password and his encryption password no longer matchs after his next log-in. In this case the user will be able to adjust his encryption password in the personal settings by providing both, his old and his new log-in password.

3.9 Knowledge Base Configuration

The usage of ownCloud is more or less self explaining but nevertheless a user might run into a problem where he needs to consult the documentation or knowledge base. To ease access to the ownCloud documentation and knowledge base, a help menu item is shown in the settings menu by default.

3.9.1 Parameters

If you want to disable the ownCloud help menu item you can use the **knowledgebaseenabled** parameter inside the `config/config.php`. The **knowledgebaseurl** parameter is used to set the http path to the ownCloud help page. The server should support OCS.

```
<?php

  "knowledgebaseenabled" => true,
  "knowledgebaseurl"     => "http://api.apps.owncloud.com/v1",
```

Note: Disabling the help menu item might increase the number of support request you have to answer in the future

---

3.10 Language Configuration

In normal cases ownCloud will automatically detect the language of the Web-GUI. If this doesn't work properly or you want to make sure that ownCloud always starts with a given language, you can use the **default_language** parameter.

Please keep in mind, that this will not effect a users language preference, which has been configured under "personal -> language" once he has logged in.

Please check `settings/languageCodes.php` for the list of supported language codes.

3.10.1 Parameters

```
<?php

  "default_language" => "en",
```

This parameters can be set in the `config/config.php`

3.11 Logging Configuration

To get an idea of how the current status of an ownCloud system is or to solve issues log information is a good point to start with. ownCloud allows to configure the way how and which depth of information should be logged.

3.11.1 Parameters

First you need to decide in which way logging should be done. You can choose between the two options **owncloud** and **syslog**. Then you need to configure the log level which directly influences how much information will be logged. You can choose between:

- **0**: DEBUG
- **1**: INFO
- **2**: WARN
- **3**: ERROR

The most detailed information will be written if **0** (DEBUG) is set, the least information will be written if **3** (ERROR) is set. Keep in mind that it might slow down the whole system if a too detailed logging will has been configured. By default the log level is set to **2** (WARN).

This parameters can be set in the `config/config.php`

ownCloud

All log information will be written to a separate log file which can be viewed using the log menu in the admin menu of ownCloud. By default a log file named **owncloud.log** will be created in the directory which has been configured by the **datadirectory** parameter.

The desired date format can optionally be defined using the **logdateformat**. By default the PHP date function parameter "*c*" is used and therefore the date/time is written in the format "*2013-01-10T15:20:25+02:00*". By using the date format in the example the date/time format will be written in the format "*January 10, 2013 15:20:25*".

```
<?php

  "log_type" => "owncloud",
  "logfile" => "owncloud.log",
  "loglevel" => "3",
  "logdateformat" => "F d, Y H:i:s",
```

syslog

All log information will be send to the default syslog deamon of a system.

```
<?php

  "log_type" => "syslog",
  "logfile" => "",
  "loglevel" => "3",
```

3.12 Mail Configuration

ownCloud does not contain a full email program but contains some parameters to allow to send e.g. password reset email to the users. This function relies on the PHPMailer library. To take advantage of this function it needs to be configured properly.

3.12.1 Requirements

Different requirements need to be matched, depending on the environment which you are using and the way how you want to send email. You can choose between **SMTP**, **PHP mail**, **Sendmail** and **qmail**.

3.12.2 Parameters

All parameters need to be set in `config/config.php`

SMTP

If you want to send email using a local or remote SMTP server it is necessary to enter the name or ip address of the server, optionally followed by a colon separated port number, e.g. **:425**. If this value is not given the default port 25/tcp will be used unless you will change that by modifying the **mail_smtpport** parameter. Multiple server can be entered separated by semicolon:

```
<?php

  "mail_smtpmode"     => "smtp",
  "mail_smtphost"     => "smtp-1.server.dom;smtp-2.server.dom:425",
  "mail_smtpport"     => 25,
```

or

```
<?php

  "mail_smtpmode"     => "smtp",
  "mail_smtphost"     => "smtp.server.dom",
  "mail_smtpport"     => 425,
```

If a malware or SPAM scanner is running on the SMTP server it might be necessary that you increase the SMTP timeout to e.g. 30s:

```
<?php

  "mail_smtptimeout"  => 30,
```

If the SMTP server accepts unsecure connections, the default setting can be used:

```
<?php

"mail_smtpsecure"   => '',
```

If the SMTP server only accepts secure connections you can choose between the following two variants:

SSL

A secure connection will be initiated using the outdated SMTPS protocol which uses the port 465/tcp:

```
<?php

  "mail_smtphost"     => "smtp.server.dom:465",
  "mail_smtpsecure"   => 'ssl',
```

TLS

A secure connection will be initiated using the STARTTLS protocol which uses the default port 25/tcp:

```
<?php

  "mail_smtphost"     => "smtp.server.dom",
  "mail_smtpsecure"   => 'tls',
```

And finally it is necessary to configure if the SMTP server requires authentication, if not, the default values can be taken as it.

```
<?php

  "mail_smtpauth"     => false,
  "mail_smtpname"     => "",
  "mail_smtppassword" => "",
```

If SMTP authentication is required you have to set the required username and password and can optionally choose between the authentication types **LOGIN** (default) or **PLAIN**.

```
<?php

  "mail_smtpauth"     => true,
  "mail_smtpauthtype" => "LOGIN",
  "mail_smtpname"     => "username",
  "mail_smtppassword" => "password",
```

PHP mail

If you want to use PHP mail it is necessary to have an installed and working email system on your server. Which program in detail is used to send email is defined by the configuration settings in the **php.ini** file. (On *nix systems this will most likely be Sendmail.) ownCloud should be able to send email out of the box.

```
<?php

  "mail_smtpmode"     => "php",
  "mail_smtphost"     => "127.0.0.1",
  "mail_smtpport"     => 25,
  "mail_smtptimeout"  => 10,
  "mail_smtpsecure"   => "",
  "mail_smtpauth"     => false,
  "mail_smtpauthtype" => "LOGIN",
  "mail_smtpname"     => "",
  "mail_smtppassword" => "",
```

Sendmail

If you want to use the well known Sendmail program to send email, it is necessary to have an installed and working email system on your *nix server. The sendmail binary (**/usr/sbin/sendmail**) is ususally part of that system. ownCloud should be able to send email out of the box.

```
<?php

  "mail_smtpmode"     => "sendmail",
  "mail_smtphost"     => "127.0.0.1",
  "mail_smtpport"     => 25,
  "mail_smtptimeout"  => 10,
  "mail_smtpsecure"   => "",
  "mail_smtpauth"     => false,
  "mail_smtpauthtype" => "LOGIN",
  "mail_smtpname"     => "",
  "mail_smtppassword" => "",
```

qmail

If you want to use the qmail program to send email, it is necessary to have an installed and working qmail email system on your server. The sendmail binary (**/var/qmail/bin/sendmail**) will then be used to send email. ownCloud should be able to send email out of the box.

```
<?php

  "mail_smtpmode"     => "qmail",
  "mail_smtphost"     => "127.0.0.1",
  "mail_smtpport"     => 25,
  "mail_smtptimeout"  => 10,
  "mail_smtpsecure"   => "",
  "mail_smtpauth"     => false,
  "mail_smtpauthtype" => "LOGIN",
  "mail_smtpname"     => "",
  "mail_smtppassword" => "",
```

3.12.3 Send a Test Email

The only way to test your email configuration is, to force a login failure, because a function to send a test email has not be implemented yet.

First make sure that you are using a full qualified domain and not an ip address in the ownCloud URL, like:

```
http://my-owncloud-server.domain.dom/owncloud/
```

The password reset function fetches the domain name from that URL to build the email sender address, e.g.:

```
john@domain.dom
```

Next you need to enter your login and an *invalid* password. As soon as you press the login button the login mask reappears and a **I've forgotten my password** link will be shown above the login field. Click on that link, re-enter your login and press the **Reset password** button - that's all.

3.12.4 Trouble shooting

My web domain is different from my mail domain?

The default domain name used for the sender address is the hostname where your ownCloud installation is served. If you have a different mail domain name you can override this behavior by setting the following configuration parameter:

```
<?php

  "mail_domain" => "example.com",
```

Now every mail send by ownCloud e.g. password reset email, will have the domain part of the sender address look like:

```
no-reply@example.com
```

How can I find out if a SMTP server is reachable?

Use the ping command to check the server availability:

```
ping smtp.server.dom

PING smtp.server.dom (ip-address) 56(84) bytes of data.
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=1 ttl=64 time=3.64 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=2 ttl=64 time=0.055 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=3 ttl=64 time=0.062 ms
```

How can I find out if the SMTP server is listening on a specific tcp port?

A SMTP server is usually listening on port **25/tcp** (smtp) and/or in rare circumstances is also listening on the outdated port **465/tcp** (smtps). You can use the telnet command to check if a port is available:

```
telnet smtp.domain.dom 25

Trying 192.168.1.10...
Connected to smtp.domain.dom.
Escape character is '^]'.
220 smtp.domain.dom ESMTP Exim 4.80.1 Tue, 22 Jan 2013 22:28:14 +0100
```

How can I find out if a SMTP server supports the outdated SMTPS protocol?

A good indication that a SMTP server supports the SMTPS protocol is that it is listening on port **465/tcp**. How this can be checked has been described previously.

How can I find out if a SMTP server supports the TLS protocol?

A SMTP server usually announces the availability of STARTTLS right after a connection has been established. This can easily been checked with the telnet command. You need to enter the marked lines to get the information displayed:

```
telnet smtp.domain.dom 25
```

```
Trying 192.168.1.10...
Connected to smtp.domain.dom.
Escape character is '^]'.
220 smtp.domain.dom ESMTP Exim 4.80.1 Tue, 22 Jan 2013 22:39:55 +0100
EHLO your-server.local.lan                                    # <<< enter this command
250-smtp.domain.dom Hello your-server.local.lan [ip-address]
250-SIZE 52428800
250-8BITMIME
250-PIPELINING
250-AUTH PLAIN LOGIN CRAM-MD5
250-STARTTLS                                                  # <<< STARTTLS is supported!
250 HELP
QUIT                                                          # <<< enter this command
221 smtp.domain.dom closing connection
Connection closed by foreign host.
```

How can I find out which authentication types/methods a SMTP server supports?

A SMTP server usually announces the available authentication types/methods right after a connection has been established. This can easily been checked with the telnet command. You need to enter the marked lines to get the information displayed:

```
telnet smtp.domain.dom 25

Trying 192.168.1.10...
Connected to smtp.domain.dom.
Escape character is '^]'.
220 smtp.domain.dom ESMTP Exim 4.80.1 Tue, 22 Jan 2013 22:39:55 +0100
EHLO your-server.local.lan                                    # <<< enter this command
250-smtp.domain.dom Hello your-server.local.lan [ip-address]
250-SIZE 52428800
250-8BITMIME
250-PIPELINING
250-AUTH PLAIN LOGIN CRAM-MD5                                 # <<< available Authentication
250-STARTTLS
250 HELP
QUIT                                                          # <<< enter this command
221 smtp.domain.dom closing connection
Connection closed by foreign host.
```

Enable Debug Mode

If you are still not able to send email it might be useful to activate further debug messages by setting the following parameter. Right after you have pressed the **Reset password** button, as described before, a lot of **SMTP -> get_lines(): ...** messages will be written on the screen.

```
<?php

  "mail_smtpdebug" => true;
```

3.13 Maintenance Mode Configuration

If you want to prevent users to login to ownCloud before you start doing some maintenance work, you need to set the value of the **maintenance** parameter to *true*. Please keep in mind that users who are already logged-in are kicked out of ownCloud instantly.

3.13.1 Parameters

```
<?php

  "maintenance" => false,
```

This parameters can be set in the `config/config.php`

3.14 Reverse Proxy Configuration

The automatic hostname, protocol or webroot detection of ownCloud can fail in certain reverse proxy situations. This configuration allows to manually override the automatic detection.

3.14.1 Parameters

If ownCloud fails to automatically detected the hostname, protocol or webroot you can use the **overwrite** parameters inside the `config/config.php`. The **overwritehost** parameter is used to set the hostname of the proxy. You can also specify a port. The **overwriteprotocol** parameter is used to set the protocol of the proxy. You can choose between the two options **http** and **https**. The **overwritewebroot** parameter is used to set the absolute web path of the proxy to the ownCloud folder. When you want to keep the automatic detection of one of the three parameters you can leave the value empty or don't set it. The **overwritecondaddr** parameter is used to overwrite the values dependent on the remote address. The value must be a **regular expression** of the IP addresses of the proxy. This is useful when you use a reverse SSL proxy only for https access and you want to use the automatic detection for http access.

3.14.2 Example

Multiple Domains Reverse SSL Proxy

If you want to access your ownCloud installation **http://domain.tld/owncloud** via a multiple domains reverse SSL proxy **https://ssl-proxy.tld/domain.tld/owncloud** with the IP address **10.0.0.1** you can set the following parameters inside the `config/config.php`.

```
<?php
$CONFIG = array (
  "overwritehost"     => "ssl-proxy.tld",
  "overwriteprotocol" => "https",
  "overwritewebroot"  => "/domain.tld/owncloud",
  "overwritecondaddr" => "^10\.0\.0\.1$",
);
```

Note: If you want to use the SSL proxy during installation you have to create the `config/config.php` otherwise you have to extend to existing **$CONFIG** array.

3.15 Uploading big files > 512MB (as set by default)

It's usefull to know limiting factors, that make it impossible to exceed the values given by the ownCloud-system:

3.15.1 Not outnumberable upload limits:

- < 2GB on 32Bit OS-architecture
- < 2GB with Server Version 4.5 or older
- < 2GB with IE6 - IE8
- < 4GB with IE9 - IE10

3.15.2 Other recommendable preconditions:

- Make sure, that the latest version of php (at least 5.4.9) is installed
- Disable user quota. This means: set the user quota of the account, you are currently logged in, to "unlimited". This is important, because you possibly could not watch otherwise, whether the desired changes take effect.

3.16 Enabling uploading big files

Note: The order of the following steps is important! If you swap steps or substeps described below, the settings may fail.

Go to the admin section in the ownCloud-WebUI and do the following:

- Under "File handling" set the Maximum upload size to the desired value (e.g. 16GB)
- Klick the "save"-Button

Open the php.ini - file

- Under Debian or Suse and their derivates this file lies at /etc/php5/apache2/php.ini
- On Windows, you can find this file within C:/Program Files (x86)/PHP/PHP.ini

Do the following:

- Set the following three parameters inside th php.ini to the same value as choosen inside the admin-section one step before:
- upload_max_filesize = 16G (e.g., to stay consistent with the example value above)
- post_max_size = 16G (e.g., to stay consistent with the example value above)
- output_buffering = 16384 (e.g., to stay consistent with the example value above)

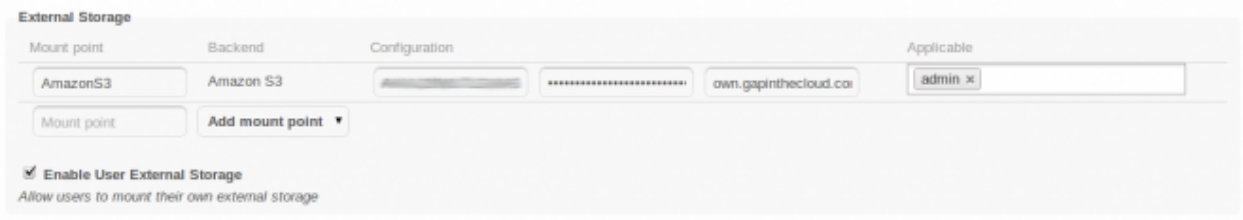
whereas the "output_buffering" has to be given in MegaBytes but as a plain figure (without size-units as 'M' or 'G')

These client configurations have been prooven by test up to filesizes of 16 GigaBytes:

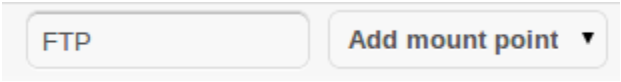
- Linux 32 Bit: Ubuntu, Firefox => 16GB
- Windows 8 64 Bit: Google Chrome => 8GB

3.17 Custom Mount Configuration Web-GUI

Since ownCloud 5.0 it is possible to mount external storage providers into ownCloud's virtual file system. To add an external storage backend to your ownCloud head to *Settings* -> *Admin* or *Personal*. As administrator you can mount external storage for any group or user. Users are also allowed to mount external storage for themselves if this setting has been enabled by the administrator.

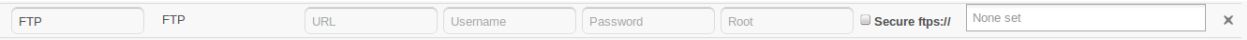


At first the mount point has to be entered, this is the directory in ownCloud's virtual file system, that the storage will be mounted to. Then the storage backend has to be selected from the list of supported backends. As of writing ownCloud currently supports the following storage backends:



- Local file system (mount local storage that is outside ownCloud's data directory)
- FTP (or FTPS)
- SFTP
- SMB
- WebDAV
- Amazon S3
- Dropbox
- Google Drive
- OpenStack Swift

Please keep in mind, that users are not allowed to mount local file storage for security purposes.



Once a backend has been selected, more configuration fields will appear. The displayed configuration fields may vary depending on the selected storage backend. For example, the FTP storage backend needs the following configuration details to be entered:

- **host**: the hostname of the ftp server
- **user**: the username used to login to the ftp server
- **password**: the password to login to the ftp server
- **secure**: whether to use ftps:// (FTP over TLS) to connect to the ftp server instead of ftp:// (optional, defaults to false)
- **root**: the name of the folder inside the ftp server to mount (optional, defaults to '/')

3.17.1 Dropbox

Mounting a Dropbox account requires that you create an app with Dropbox and then provide the app key and secret to the external storage configuration user interface. Go to My apps at Dropbox and create an app. Select *Full Dropbox* access level. Copy the app key and app secret and paste them into the corresponding fields for the Dropbox storage.

Click the *Grant access* button and you will be redirected to a Dropbox website to give ownCloud permission to access your account.

3.17.2 Google Drive

Mounting a Google Drive account requires that you create an API project in the Google APIs Console. Select *Services* and enable both *Drive API* and *Drive SDK*. Next select *API Access* and click *Create an OAuth 2.0 client ID*. Fill out the Branding Information as you see fit and click Next. In Client Id Settings select *Web application* for the Application type. Next to *Your site or hostname* click more options) and add the following to the Authorized Redirect URIs:

- http://yourowncloud/index.php/settings/personal
- http://yourowncloud/index.php/settings/admin

Copy the client id and client secret and paste them into the corresponding fields for the Google Drive Storage.

Click the *Grant access* button and you will be redirected to a Google website to give ownCloud permission to access your account.

3.18 Custom Mount Configuration

Since ownCloud 4.0 it is possible to configure the filesystem to mount external storage providers into ownCloud's virtual file system. You can configure these file systems by creating and editing `data/mount.json`. This file contains all settings in JSON (JavaScript Object Notation) format. At the moment two different types of entries exist:

- **Group mounts:** each entry configures a mount for each user in group.
- **User mounts:** each entry configures a mount for a single user or for all users.

For each type, there is a JSON array with the user/group name as key, and an array of configuration entries as value. Each entry consist of the class name of the storage backend and an array of backend specific options and will be replaced by the user login. The template **$user** can be used in the mount point or backend options. As of writing the following storage backends are available for use:

- Local file system
- FTP (or FTPS)
- SFTP
- SMB
- WebDAV
- Amazon S3
- Dropbox
- Google Drive
- OpenStack Swift

Please keep in mind that some formatting has been applied and carriage returns have been added for better readability. In the `data/mount.json` all values need to be concatenated and written in a row without these modifications!

It is recommended to use the *Web-GUI* in the administrator panel to add, remove or modify mount options to prevent any problems!

3.18.1 Example

```
{"group":{
    "admin":{
        "\/$user\/files\/Admin_Stuff":{
            "class":"\\OC\\Files\\Storage\\Local",
            "options":{ ... }
            }
        }
    }
 "user":{
    "all":{
        "\/$user\/files\/Pictures":{
            "class":"\\OC\\Files\\Storage\\DAV",
            "options":{ ... }
            }
        }
    "someuser":{
        "\/someuser\/files\/Music":{
            "class":"\\OC\\Files\\Storage\\FTP",
            "options":{ ... }
            }
        }
    }
}
```

3.18.2 Backends

Local Filesystem

The local filesystem backend mounts a folder on the server into the virtual filesystem, the class to be used is **\OC\Files\Storage\Local** and takes the following options:

- **datadir** : the path to the local directory to be mounted

Example

```
{ "class":"\\OC\\Files\\Storage\\Local",
  "options":{ "datadir":"\/mnt\/additional_storage" }
}
```

Note: You must ensure that the web server has sufficient permissions on the folder.

FTP (or FTPS)

The FTP backend mounts a folder on a remote FTP server into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\FTP** and takes the following options:

- **host**: the hostname of the ftp server
- **user**: the username used to login on the ftp server
- **password**: the password to login on the ftp server
- **secure**: whether to use ftps:// (FTP over TLS) to connect to the ftp server instead of ftp:// (optional, defaults to false)
- **root**: the folder inside the ftp server to mount (optional, defaults to '/')

Example

```
{    "class":"\\OC\\Files\\Storage\\FTP",
     "options":{
         "host":"ftp.myhost.com",
         "user":"johndoe",
         "password":"secret",
         "root":"\/Videos",
         "secure":"false"
     }
}
```

Note: PHP needs to be build with FTP support for this backend to work.

SFTP

The SFTP backend mounts a folder on a remote SSH server into the virtual filesystem and is part of the 'External storage support' app. The class to be used is **\OC\Files\Storage\SFTP** and takes the following options:

- **host**: the hostname of the SSH server
- **user**: the username used to login to the SSH server
- **password**: the password to login on the SSH server
- **root**: the folder inside the SSH server to mount (optional, defaults to '/')

Example

```
{    "class":"\\OC\\Files\\Storage\\SFTP",
     "options":{
         "host":"ssh.myhost.com",
         "user":"johndoe",
         "password":"secret",
         "root":"\/Books"
     }
}
```

Note: PHP needs to be build with SFTP support for this backend to work.

SMB

The SMB backend mounts a folder on a remote Samba server, a NAS appliance or a Windows machine into the virtual file system. It is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\SMB** and takes the following options:

- **host**: the host name of the samba server
- **user**: the user name used to login on the samba server
- **password**: the password to login on the samba server
- **share**: the share on the samba server to mount
- **root**: the folder inside the samba share to mount (optional, defaults to '/')

Note: The SMB backend requires **smbclient** to be installed on the server.

Example

```
{    "class":"\\OC\\Files\\Storage\\SMB",
     "options":{
         "host":"myhost.com",
         "user":"johndoe",
         "password":"secret",
         "share":"\/test",
         "root":"\/Pictures"
     }
}
```

WebDAV

The WebDAV backend mounts a folder on a remote WebDAV server into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\DAV** and takes the following options:

- **host**: the hostname of the webdav server.
- **user**: the username used to login on the webdav server
- **password**: the password to login on the webdav server
- **secure**: whether to use https:// to connect to the webdav server instead of http:// (optional, defaults to false)
- **root**: the folder inside the webdav server to mount (optional, defaults to '/')

Example

```
{    "class":"\\OC\\Files\\Storage\\DAV",
     "options":{
         "host":"myhost.com\/webdav.php",
         "user":"johndoe",
```

```
        "password":"secret",
        "secure":"true"
    }
}
```

Amazon S3

The Amazon S3 backend mounts a bucket in the Amazon cloud into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\AmazonS3** and takes the following options:

- **key**: the key to login to the Amazon cloud
- **secret**: the secret to login to the Amazon cloud
- **bucket**: the bucket in the Amazon cloud to mount

Example

```
{   "class":"\\OC\\Files\\Storage\\AmazonS3",
    "options":{
        "key":"key",
        "secret":"secret",
        "bucket":"bucket"
    }
}
```

Dropbox

The Dropbox backend mounts a dropbox in the Dropbox cloud into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\Dropbox** and takes the following options:

- **configured**: whether the drive has been configured or not (true or false)
- **app_key**: the app key to login to your Dropbox
- **app_secret**: the app secret to login to your Dropbox
- **token**: the OAuth token to login to your Dropbox
- **token_secret**: the OAuth secret to login to your Dropbox

Example

```
{   "class":"\\OC\\Files\\Storage\\Dropbox",
    "options":{
        "configured":"#configured",
        "app_key":"key",
        "app_secret":"secret",
        "token":"#token",
        "token_secret":"#token_secret"
    }
}
```

Google Drive

The Google Drive backend mounts a share in the Google cloud into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\Google** and is done via an OAuth2.0 request. That means that the App must be registered through the Google APIs Console. The result of the registration process is a set of values (incl. client_id, client_secret). It takes the following options:

- **configured**: whether the drive has been configured or not (true or false)
- **client_id**: the client id to login to the Google drive
- **client_secret**: the client secret to login to the Google drive
- **token**: a compound value including access and refresh tokens

Example

```
{    "class":"\\OC\\Files\\Storage\\Google",
     "options":{
          "configured":"#configured",
          "client_id":"#client_id",
          "client_secret":"#client_secret",
          "token":"#token"
     }
}
```

OpenStack Swift

The Swift backend mounts a container on an OpenStack Object Storage server into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\SWIFT** and takes the following options:

- **host**: the hostname of the authentication server for the swift storage.
- **user**: the username used to login on the swift server
- **token**: the authentication token to login on the swift server
- **secure**: whether to use ftps:// to connect to the swift server instead of ftp:// (optional, defaults to false)
- **root**: the container inside the swift server to mount (optional, defaults to '/')

Example

```
{    "class":"\\OC\\Files\\Storage\\SWIFT",
     "options":{
          "host":"swift.myhost.com\/auth",
          "user":"johndoe",
          "token":"secret",
          "root":"\/Videos",
          "secure":"true"
     }
}
```

3.19 Custom User Backend Configuration

Starting with ownCloud 4.5 is possible to configure additional user backends in ownCloud's configuration `config/config.php` using the following syntax:

```
<?php

"user_backends" => array (
    0 => array (
            "class"     => ...,
            "arguments" => array (
                               0 => ...
                               ),
    ),
),
```

Currently the "External user support" (user_external) app provides the following user backends:

3.19.1 IMAP

Provides authentication against IMAP servers

- **Class:** OC_User_IMAP
- **Arguments:** a mailbox string as defined in the PHP documention
- **Example:**

```
<?php

"user_backends" => array (
    0 => array (
            "class"     => "OC_User_IMAP",
            "arguments" => array (
                               0 => '{imap.gmail.com:993/imap/ssl}'
                               ),
    ),
),
```

3.19.2 SMB

Provides authentication against Samba servers

- **Class:** OC_User_SMB
- **Arguments:** the samba server to authenticate against
- **Example:**

```
<?php

"user_backends" => array (
    0 => array (
            "class"     => "OC_User_SMB",
            "arguments" => array (
                               0 => 'localhost'
                               ),
```

```
    ),
),
```

FTP

Provides authentication against FTP servers

- **Class:** OC_User_FTP
- **Arguments:** the FTP server to authenticate against
- **Example:**

```
<?php

"user_backends" => array (
    0 => array (
            "class"     => "OC_User_FTP",
            "arguments" => array (
                                0 => 'localhost'
                                ),
    ),
),
```

3.20 Serving static files via web server

Since ownCloud 5 it is possible to let web servers handle static file serving. This should generally improve performance (web servers are optimized for this) and in some cases permits controlled file serving (i.e. pause and resume downloads).

Note: This feature can currently only be activated for local files, i.e. files inside the **data/** directory and local mounts. Controlled file serving **does not work for generated zip files**. This is due to how temporary files are created.

3.20.1 Apache2 (X-Sendfile)

It is possible to let Apache handle static file serving via mod_xsendfile.

Installation

On Debian and Ubuntu systems use:

```
apt-get install libapache2-mod-xsendfile
```

Configuration

Configuration of mod_xsendfile for ownCloud depends on its version. For versions below 0.10 (Debian squeeze ships with 0.9)

```
<Directory /var/www/owncloud>
    ...
    SetEnv MOD_X_SENDFILE_ENABLED 1
    XSendFile On
    XSendFileAllowAbove On
</Directory>
```

For versions >=0.10 (e.g. Ubuntu 12.10)

```
<Directory /var/www/owncloud>
    ...
    SetEnv MOD_X_SENDFILE_ENABLED 1
    XSendFile On
    XSendFilePath /tmp/oc-noclean
    XSendFilePath /home/valerio
</Directory>
```

- **SetEnv MOD_X_SENDFILE_ENABLED**: tells ownCloud scripts that they should add the X-Sendfile header when serving files
- **XSendFile**: enables web server handling of X-Sendfile headers (and therefore file serving) for the specified Directory
- **XSendFileAllowAbove (<0.10)**: enables file serving through web server on path outside the specified Directory. This is needed for PHP temporary directory where zip files are created and for configured local mounts which may reside outside data directory
- **XSendFilePath (>=0.10)**: a white list of paths that the web server is allowed to serve outside of the specified Directory. At least PHP temporary directory concatenated with *oc-noclean* must be configured. Temporary zip files will be created inside this directory when using mod_xsendfile. Other paths which correspond to local mounts should be configured here aswell. For a more in-dept documentation of this directive refer to mod_xsendfile website linked above

3.20.2 LigHTTPd (X-Sendfile2)

LigHTTPd uses similar headers to Apache2, apart from the fact that it does not handle partial downloads in the same way Apache2 does. For this reason, a different method is used for LigHTTPd.

Installation

X-Sendfile and X-Sendfile2 are supported by default in LigHTTPd and no additional operation should be needed to install it.

Configuration

Your server configuration should include the following statements:

```
fastcgi.server          = ( ".php" => ((
   ...
   "allow-x-send-file" => "enable",
   "bin-environment" => (
      "MOD_X_SENDFILE2_ENABLED" => "1",
   ),
)))
```

- **allow-x-send-file**: enables LigHTTPd to use X-Sendfile and X-Sendfile2 headers to serve files

- **bin-environment**: is used to parse MOD_X_SENDFILE2_ENABLED to the ownCloud backend, to make it use the X-Sendfile and X-Sendfile2 headers in it's response

3.20.3 Nginx (X-Accel-Redirect)

Nginx supports handling of static files differently from Apache. Documentation can be found in the Nginx Wiki section Mod X-Sendfile and section X-Accell. The header used by Nginx is X-Accel-Redirect.

Installation

X-Accel-Redirect is supported by default in Nginx and no additional operation should be needed to install it.

Configuration

Configuration is similar to Apache:

```
location ~ \.php$ {
    ...
    fastcgi_param MOD_X_ACCEL_REDIRECT_ENABLED on;
}

location ~ ^/home/valerio/(owncloud/)?data {
    internal;
    root /;
}

location ~ ^/tmp/oc-noclean/.+$ {
    internal;
    root /;
}
```

- **fastcgi_param MOD_X_ACCEL_REDIRECT_ENABLED:** tells ownCloud scripts that they should add the X-Accel-Redirect header when serving files
- **internal location:** each directory that contains local user data should correspond to an internal location. In the example uses the following directories:
  - **/home/valerio/owncloud/data**: ownCloud data directory
  - **/home/valerio/data**: a local mount
  - **/tmp/oc-noclean**: PHP temporary directory concatenated with *oc-noclean*. Temporary zip files will be created inside this directory when using X-Accel-Redirect

3.20.4 How to check if it's working?

You are still able to download stuff via the web interface and single, local file downloads can be paused and resumed.

CHAPTER

FOUR

MAINTENANCE

4.1 Migrating ownCloud Installations

To migrate an ownCloud install there are three things you need to retain:

1. The config folder
2. The data folder
3. The database (found in the data folder for sqlite installs)

To restore an ownCloud instance:

1. Extract ownCloud to your webserver
2. Copy over your config folder
3. Copy over your data folder
4. Import your database
5. Update config.php of any changes to your database connection

4.2 Updating ownCloud

4.2.1 Update

Updating means updating ownCloud to the latest *point release*, e.g. ownCloud 4.0.6 → 4.0.7. To update an ownCloud installation manually, follow those steps:

Note: If you have installed ownCloud from a repository, your package management should take care of it.

---

1. Make a backup.
2. Unpack the release tarball in the owncloud directory, i.e. copy all new files into the ownCloud installation.
3. Make sure that the file permissions are correct.
4. After the next page request the update procedures will run.

Assuming your ownCloud installation is at **./owncloud/** and you want to update to the latest version, you could do the following:

Use rsync in archive mode (this leaves file owner, permissions, and time stamps untouched) to recursively copy all content from **./owncloud/** to a backup directory which contains the current date:

```
rsync -a owncloud/ owncloud_bkp`date +"%Y%m%d"`/
```

Download the latest version to the working directory:

```
wget http://download.owncloud.org/community/owncloud-latest.tar.bz2
```

Extract content of archive to **./owncloud_latest/**:

```
mkdir owncloud_latest; tar -C owncloud_latest -xjf owncloud-latest.tar.bz2
```

Use rsync to recursivly copy extracted files (new) to ownCloud installation (old) using modification times of the new files, but preserving owner and permissions of the old files:

> **Warning:** You should not use this [–inplace] option to update files that are being accessed by others *(from rysnc man page)*

```
rsync --inplace -rtv owncloud_latest/owncloud/ owncloud/
```

Clean up:

```
rm -rf owncloud-latest.tar.bz2 owncloud_latest/
```

4.2.2 Upgrade

Upgrade is to bring an ownCloud instance to a new *major release*, e.g. ownCloud 4.0.7 → 4.5.0. Always do backups anyway.

To upgrade ownCloud, follow those steps:

1. Make sure that you ran the latest point release of the major ownCloud version, e.g. 4.0.7 in the 4.0 series. If not, update to that version first (see above).
2. **Make a backup of the ownCloud folder and the database**
3. Deactivate all third party applications.
4. Delete everything from your ownCloud installation directory, except data and config.
5. Unpack the release tarball in the owncloud directory (or copy the files thereto).
6. Make sure that the file permissions are correct.
7. With the next page request the update procedures will run.
8. If you had 3rd party applications, check if they provide versions compatible with the new release.

If so, install and enable them, update procedures will run if needed. 9. If you installed ownCloud from a repository, your package management should take care of it. Probably you will need to look for compatible third party applications yourself. Always do backups anyway.

4.3 Backing Up ownCloud

To backup an ownCloud installation there are three main things you need to retain:

1. The config folder

2. The data folder
3. The database (found in the data folder for sqlite installs)

To restore an ownCloud instance:

1. Extract ownCloud to your webserver
2. Copy over your config folder with your backed up config folder
3. Copy over your data folder with your backed up data folder
4. Import your database
5. Update config.php of any changes to your database connection

4.3.1 Backup Folders

Simply copy your config and data folder(or even your whole ownCloud install and data folder) to a place outside of your ownCloud environment.

4.3.2 Backup Database

MySQL

MySQL is the recommended database engine. To backup MySQL:

```
mysqldump –lock-tables -u [username] -p[password] > owncloud.sql
```

SQLite

```
sqlite3 owncloud.db .dump > owncloud.bak
```

PostgreSQL

```
pg_dump owncloud > owncloud.bak
```

CHAPTER

FIVE

ISSUES

If you think you have found a bug in ownCloud, please:

- Search for a solution
- Double check your configuration

If you can't find a solution, please file an issue:

- If the issue is with the ownCloud server, report it to the GitHub core repository
- If the issue is with the ownCloud client, report it to the GitHub mirall repository
- If the issue with with an ownCloud app, report it to where that app is developed
  - If the app is listed here report it to the correct repository
  - If the app is listed here report it to the apps repository

Please note that the mailing list should not be used for bug reports, as it is hard to track them there.

CHAPTER

SIX

INDICES AND TABLES

- *genindex*