# Administration Manual for UrBackup Server 1.3.x

Martin Raiber

February 1, 2014

## Contents

# 1 Introduction

UrBackup is a client/server backup system. This means there exists a server which backups clients. Accordingly UrBackup is divided into a client and server software. The client software currently runs only on Windows while the server software runs on both Linux and Windows.
UrBackup is able to backup files and images. The clients have to define the paths where the files to be backed up are. Images are automatically taken of the system drive (C:).
A lot of effort in UrBackup was made to make setup as easy as possible. If you are happy with the default settings (see section 7) the only thing you need to define on the server side is where backups should be stored. On the clients you only need to say which directories should be backed up. If server and clients are in the same subnet the server will automatically discover the clients and then start backing them up (for details see section 4). This also makes building a decentralized backup strategy very easy, as e.g. one backup server per subnet is responsible for backing up all clients in this subnet. If a computer is moved from one subnet to another this new client is discovered and the server in the new subnet automatically takes over backing it up. If you want to implement

something like this you should also read the section on security (see 3) for details on when a client accepts a server.
The interested administrator should read up on the general UrBackup architecture (section 2), how the backups are stored and performed (section 5), proposals on which file systems are suited (section 9.6) and take a look at some sample a setup with ZFS at section 9.7.1.
Since version 1.0 UrBackup can also handle clients over the Internet (see section 6), and archive backups (see section 9.5).

# 2 Architecture

As already mentioned UrBackup is divided into a server and a client software. The server is responsible for discovering clients, backing them up, deleting backups if the storage is depleted or too many backups are present, generating statistics and managing client settings. The client is relatively dump. It listens to server commands which tell it e.g. that a file list should be build or which file the server wants to download. The server also starts a channel on which the clients can request the server to start a backup or to update the client specific settings.

## 2.1 Server architecture

The server is organized into a core part and an interface. Currently only a webinterface is available. The web interface is accessible via FastCGI (on port 55413) and HTTP (on port 55414). You can use the FastCGI port to make the webinterface accessible via SSL (using e.g. apache web server). For details on that see section 3.2. The server core part consists of several threads with different tasks. One thread discovers new clients, another checks if a client needs to be backed up, while others send pings to clients to see if they are still alive or send them the current backup status. One updates file statistics or deletes old backups. Because there are so many threads UrBackup server profits from modern multi core CPUs (the more cores the better!).

## 2.2 Client architecture

The client is divided into a core process and an interface process. The interface process displays the tray icon and the dialogues and sends settings and commands to the core client process. The core client process listens on port 35622 UDP for UDP broadcast messages from the server and on receiving one sends a message with its name back to the server. As name the Windows computer name is used. It listens on port 35623 TCP for commands from the client interface process and the server and on port 35621 TCP for file requests from the server. The server establishes a permanent connection to each client on its command port with which the clients can request backups or change their settings. The core client process is responsible for building a list of all files in the directories to be backed up. This list is created in the UrBackup client directory as 'urbackup/ data/ filelist.ub'. To speed up the directory list creation directories to be backed up are constantly watched via the Windows Change Journal. The Windows Change Journal can only be used for whole partitions. Thus the first time a directory on a volume is added the UrBackup core client process reads all the directory entries on the new volume into the client database file in 'urbackup/backup_client.db'. After a volume is successfully indexed the database is constantly updated to be in sync with the file system. Thus if large changes in the volume occur the database gets updated more often. This does not have a big performance penalty as only directories are saved in the database. The updating is done every 10 seconds or if a file list is requested. The server downloads the file list from the client and starts the backup by downloading changed or new files from the build in client file server. The image backup is done using only the command port.

# 3 Security

## 3.1 Server webinterface rights management

The server web interface is protected by a pretty standard user system. You can create, manage and delete accounts. Those accounts are only linked loosely to clients by rights management. Be aware that after first installing UrBackup there is no administrator password set and everybody can see all backuped files! If you want to limit access you should immediately go to the account management in the settings and create an administrator account and set its password.
An admin account can do everything including browsing file backups of all clients. The web interface allows one to create a 'limited' account that can only browse backups and view statistics from one client. The more sophisticated rights editor can be used to allow an account to access several clients or to limit some aspects. For example you could setup an account which can do everything except browse backups. Following domains, with which you can limit or expand an accounts rights, are currently available:

| Domain | Description |
|---|---|
| browse_backups | Browse and download files from file backups |
| lastacts | View the last actions (file or image backups) the server did (including backup size and duration) |
| progress | View the progress of currently running file or image backups |
| settings | Allows settings to be changed |
| client_settings | Allows client specific settings to be changed |
| status | Allows the current status to be viewed (last seen, last file backup and last image backup) |
| logs | View the logs which were creating during backups |
| manual_archive | Manually archive file backups |
| stop_backup | Stop backups for client on the server |
| piegraph* | View statistics |
| users* | Get client names |
| general_settings* | Change general settings (like backup storage path) |
| mail_settings | Change the mail server settings |
| usermod* | Create, change and delete users |
| remove_client* | Remove clients and delete all their backups |
| start_backup* | Start backups for a client on the server |

You can set the domains not marked with stars(*) either to one or several client ids (separated by ',') or to 'all' - meaning the account can access all clients. The entries with stars(*) have to be set to 'all' or 'none' and don't allow client ids. In order to be able to view statistics you need to set both 'piegraph' and 'users' to 'all'. There is a special domain 'all' which is a wild card for all domains (this means if you set 'all' to 'all' the account has the right to do everything).

## 3.2 Make webinterface accessible via SSL

The server web interface is accessible via FastCGI (on port 55413 TCP). With this you can connect UrBackup with pretty much every modern web server and thus make the web interface accessible via SSL. This section will describe how to do this with apache and lighttp.

### 3.2.1 Apache configuration

Either add a symlink to the 'www' UrBackup directory or define it as an alias. For the symlink method you need to go to your SSL webroot and then do e.g.:

```
ln -s /var/lib/urbackup/www urbackup
```

Be sure you have set 'Option +FollowSymLinks' in the webserver configuration on the directory you link into. From now on it is assumed that urbackup should be accessible via

https://hostname/urbackup. Download and install 'libapache2-mod-fastcgi' (this may have another name on other distributions). Add following line to the 'fastcgi.conf':

```
FastCgiExternalServer /var/www/urbackup/x -host 127.0.0.1:55413
```

The path depends of cause on where your web root is and where you want the web interface to be. UrBackup should now be accessible via apache.

#### 3.2.2 Lighttp configuration

Link the urbackup/www directory into the webroot as described in the apache configuration. Add

```
include "conf.d/fastcgi.conf"
```

to your 'lighttp.conf' file. Then add

```
fastcgi.server = (
  "/urbackup/x" =>
  (( "host" => "127.0.0.1",
     "port" => 55413
  ))
)
```

to the 'fastcgi.conf' file.

### 3.3 Client security

UrBackup Client only answers commands if the server or the interface process supply it with credentials. The server credential is saved in '/var/ lib/ urbackup/ server_ident.key'. If it does not exist the server will randomly generate it the first time it runs. The client interface credential is generated in the same way and resides in 'pw.txt' in the UrBackup directory on the client. To give the client core process interface commands you need the contents of 'pw.txt'. The client core process saves the server credentials from which it accepts commands and which it allows to download files in 'server_idents.txt' - one credential per line. You need to remove the preceding '#I' and '#' at the end of the contents of 'server_ident.key' if you want to add a server identity to 'server_idents.txt'. After installation the 'server_idents.txt' does not exist and the client core process accepts(and adds) the first server it sees. After that no other servers with different credentials are accepted and you need to add their credentials manually. This prevents others from accessing files you want to be backed up in public places.
If you want to have several servers to be able to do backups of a client you have two options. Either you manually supply the server credentials to the client (by copying them into 'server_idents.txt') or you give all servers the same credentials by copying the same 'server_ident.key' to all servers.

### 3.4 Transfer security

The transfer of data from client to server is unencrypted on the local network allowing eavesdropping attacks to recover contents of the data that is backed up. With this in mind you should use UrBackup only in trusted local networks.

### 3.5 Internet mode security

The Internet mode uses strong authentication and encryption. The three way handshake is done using a shared key and PBKDF2-HMAC using SHA512 with 20000 iterations. The data is encrypted using AES256 in CFB mode.

# 4 Client discovery in local area networks

UrBackup clients should be discovered automatically given that server and client reside in the same sub-network. The client discovery works as follows:
The UrBackup server broadcasts a UDP message every 50 seconds on all adapters into the local subnet of this adapter. On receiving such a broadcast message the client answers back with its name. Thus it may take up to 50 seconds until a client is recognized as online.
If the client you want to backup is not in the same subnet as the server you can add its IP or host name manually by clicking "show details" in the settings and then adding an "extra client". The server will then additionally send an UDP message directly to that entered IP or resolved host name allowing routers to forward the message across subnet boundaries. Be aware though that all connections are from server to client, e.g. if you use NAT you need to forward the client ports (35622 UDP, 35621 TCP, 35623 TCP) to the client. Currently there is no option to change these ports, so you would be limited to just one client if you have NAT. You should use VPN in this case.

# 5 Backup process

This section will show in detail how a backup is performed.

## 5.1 File backup

- The server detects that the time to the last incremental backup is larger then the interval for incremental backups or the last time to the last full backup is larger then the interval for full backups. Backups can be started on client requests as well.
- The server creates a new directory where it will save the backup. The schema for this directory is YYMMDD-HHMM with YY the year in a format with two decimals. MM the current month. DD the current day. And HHMM the current hour and minute. The directory is created in the backup storage location in a directory which name equals the client name.
- The server requests a file list construction from the client. The client constructs the file list and reports back that it is done.
- The server downloads 'urbackup/data/filelist.ub' from the client. If it is an incremental backup the server compares the new 'filelist.ub' with the last one from the client and calculates the differences.
- The server starts downloading files. If the backup is incremental only new and changed files are downloaded. If the backup is a full one all files are downloaded from the client.
- The server downloads the file into a temporary file. This temporary file is either in the urbackup_tmp_files folder in the backup storage dir, or, if you enabled it in the advanced settings, in the temporary folder. On successfully downloading a file the server calculates its hash and looks if there is another file with the same hash value. If such a file exists they are assumed to be the same and a hard link to the other file is saved and the temporary file deleted. If no such file exists the file is moved to the new backup location. File path and hash value are saved into the server database.
- If the backup is incremental and a file has not changed a hard link to the file in the previous backup is created.
- If the client goes offline during the backup and the backup is incremental the server continues creating hard links to files in the previous backup but does not try to download files again. The files that could not be downloaded are then not saved into the server side file list. If

the backup is a full one and the client goes offline the backup process is interrupted and the partial file list is saved, which includes all files downloaded up to this point.

- If all files were transferred the server updates the 'current' symbolic link in the client backup storage location to point to the new backup. This only happens if the client did not go offline during the backup.

### 5.2 Image backup

The server detects that the time to the last full image backup is larger then the interval for full image backups, the time to the last incremental backup is larger than the interval for incremental image backups or the client requested an image backup. The server then opens up a connection to the client command service requesting the image of a volume. The client answers by sending an error code or by sending the image. The image is sent sector for sector with each sector preceded by its position on the hard disk. The client only sends sectors used by the file system. If the backup is incremental the client calculates a hash of 256 kbyte chunks and compares it to the previous image backup. If the hash of the chunk has not changed it does not transfer this chunk to the server, otherwise it does. Per default the server writes the image date directly into a VHD file. If enabled in the advanced configuration the server writes the image data to a temprary file first. The temporary files have a maximum size of 1GB. After this size is exceeded the server continues with a new temporary file. The image data is written to a VHD file in parallel and is located in the client directory in the backup storage location. The VHD file's name is 'Image_<Volume>_<YYMMDD_HHMM>.vhd'.<Volume>being the drive letter of the backuped partition and YY the current year, MM the current month, DD the current day in the month and HHMM the hour and minute the image backup was started.

## 6 Internet clients

UrBackup is able to backup clients over the internet, enabling mixed LAN and Internet backups. This can be useful e.g. for mobile devices which are not used in the LAN all the time, but are connected to the Internet. As it causes additional strain on the backup file system this feature is disabled by default. You need to enable and configure it in the settings and restart your server to use it. The minimum you have to configure is the server name or IP on which the backup server will be available on the Internet. As you probably have a Firewall or Router in between backup server and Internet you also need to forward the configured port (default: 55415) to the backup server.
There are two ways to configure the clients illustrated in the two following sections.

### 6.1 Automatically push server configuration to clients

If the client is a mobile device it is easiest to let the server push its name and settings to the client. This will happen automatically. The server will also automatically generate a key for each client and push that one to the client as well. This assumes that the local area network is a secure channel. If a client has been compromised you can still change the key on the server and on the client.

### 6.2 Manually add and configure clients

UrBackup also allows manually adding clients and manually configuring the shared key. To add such a client following steps are necessary:

1. Go to the "Status" screen and select "show details"

2. Under “Internet clients” enter the name of the Laptop/PC you want to add. This must be the real computer name (i.e. the one you see in the advanced system settings, the one you get but running *hostname*) or the computer name configured on the client.
3. After pressing add there will be a new client in the “Status” screen. Go to settings and select that client there.
4. In the Internet settings enter an authentication key for that client. The key acts just like every normal password and should therefore be sufficiently complex. Having a different key for every client makes revoking compromised keys easier, but is not a requirement.
5. On the client go to the settings and enter the same key there in the internet settings. Also enter the public IP or name of your backup server and the port it is reachable at.
6. The server and client should now connect to each other. If it does not work check the client and server logs as described in section 8.2.

## 6.3 File transfer over Internet

If a client is connected via Internet UrBackup automatically uses a bandwidth saving file transfer mode. This mode only transfers changed blocks of files and should therefore conserve bandwidth on files which are not changed completely, such as database files, virtual hard disks etc.. This comes at a cost: UrBackup has to save hashes of every file. Those hashes are saved the folder “.hashes”. They are only saved if the Internet mode is enabled. If the hashes of a file are not present e.g. because Internet mode was just enabled, they are created from the files during the backup and may thus slow down the backup process.

# 7 Server settings

The UrBackup Server allows the administrator to change several settings. There are some global settings which only affect the server and some settings which affect the client and server. For those settings the administrator can set defaults or override the client's settings.

## 7.1 Global Server Settings

The global server settings affect only the server and can be changed by any user with "general_settings" rights.

### 7.1.1 Backup storage path

The backup storage path is where all backup data is saved. To function properly all of this directories' content must lie on the same file system (otherwise hard links cannot be created). How much space is available on this file system for UrBackup determines partly how many backups can be made and when UrBackup starts deleting old backups. Default: "".

### 7.1.2 Do not do image backups

If checked the server will not do image backups at all. Default: Not checked.

### 7.1.3 Do not do file backups

If checked the server does no file backups. Default: Not checked.

### 7.1.4 Automatically shut down server

If you check this UrBackup will try to shut down the server if it has been idle for some time. This also causes too old backups to be deleted when UrBackup is started up instead of in a nightly job. In the Windows server version this works without additional work as the UrBackup server process runs as a SYSTEM user, which can shut down the machine. In Linux UrBackup server runs as a limited user which normally does not have the right to shut down the machine. UrBackup instead creates the file '/var/lib/urbackup/shutdown_now', which you can check for existence in a cron script e.g.:

```
if test -e /var/lib/urbackup/shutdown_now
then
shutdown -h +10
fi
```

Default: Not checked.

### 7.1.5 Download client from update server

If this is checked the server will automatically look for new UrBackup client versions. If there is a new version it will download it from the Internet. The download is protected by a digital signature. Default: Checked.

### 7.1.6 Autoupdate clients

If this is checked the server will send new versions automatically to its clients. The UrBackup client interface will ask the user to install the new client version. If you check silent autoupdate (see Section 7.1.6) it will update in the background. The installer is protected by a digital signature so malfeasance is not possible. Default: Checked.

### 7.1.7 Max number of simultaneous backups

This option limits the number of file and image backups the server will start simultaneously. You can de- or increase this number to balance server load. A large number of simultaneous backups will of course increase the time the server needs for one backup, if many backups are run in parallel. The number of possible simultaneous backups is virtually unlimited. Default: 10.

### 7.1.8 Max number of recently active clients

This option limits the number of clients the server accepts. An active client is a client the server has seen in the last two month. If you have multiple servers in a network you can use this option to balance their load and storage usage. Default: 100.

### 7.1.9 Cleanup time window

UrBackup will do its clean up during this time. This is when old backups and clients are deleted. You can specify the weekday and the hour as intervals. The syntax is the same as for the backup window. Thus please see section 7.3.1 for details on how to specify such time windows. The default value is 1-7/3-4 which means that the clean up will be started on each day (1-Monday - 7-Sunday) between 3 am and 4 am.

### 7.1.10 Automatically backup UrBackup database

If checked UrBackup will save a backup of its internal database in a subdirectory called 'urbackup' in the backup storage path. This backup is done daily within the clean up time window.

#### 7.1.11 Total max backup speed for local network

You can limit the total bandwidth usage of the server in the local network with this setting. All connections between server and client are then throttled to remain under the configured speed limit. This is useful if you do not want the backup server to saturate your local network.

#### 7.1.12 Global soft filesystem quota

During cleanups UrBackup will look at the used space of the filesystem the backup folder is on. If the used space is higher than the global soft filesystem quota UrBackup will delete old backups, if possible, till the used space is below the quota. Be aware that not only UrBackup's files count against the quota, but other files as well. A quota that only takes into account UrBackup's files is plannned. You can sepecify the quota via a percentage of total space, or by a size. For example let the size of the Backup device be 1 Tera-byte: If you set the global filesystem quota to "90%", UrBackup will delete old backups as soon as more than about 900 Giga-bytes of the available space is used. You could also directly set the quota to 900 Giga-bytes by setting it to "900G". Other units are possible, e.g. "900000M" or "1T".

### 7.2 Mail settings

#### 7.2.1 Mail server settings

If you want the UrBackup server to send mail reports a mail server should be configured in the 'Mail' settings page. The specific settings and their description are:

| Settings | Description | Example |
|---|---|---|
| Mail server name | Domain name or IP address of mail server | mail.example.com |
| Mail server port | Port of SMTP service. Most of the time 25 or 587 | 587 |
| Mail server username | Username if SMTP server requires one | test@example.com |
| Mail server password | Password for user name if SMTP server requires credentials | password1 |
| Sender E-Mail Address | E-Mail address UrBackup's mail reports will come from | urbackup@example.com |
| Send mails only with SSL/TLS | Only send mails if a secure connection to the mail server can be established (protects password) | |
| Check SSL/TLS certificate | Check if the server certificate is valid and only send mail if it is | |
| Server admin mail address | Address for fatal errors (such as if an emergency cleanup fails or other fatal errors occur) | |

To test whether the entered settings work one can specify an email address to which UrBackup will then send a test mail.

#### 7.2.2 Configure reports

To specify which activities with which errors should be sent via mail you have to go to the 'Logs' page. There on the bottom is a section called 'Reports'. There you can say to which email addresses reports should be sent(e.g. user1@example.com;user2@example.com) and if UrBackup should only send reports about backups that failed/succeeded and contained a log message of a certain level.

If you select the log level 'Info' and 'All' a report about every backup will be sent, because every backup causes at least one info level log message. If you select 'Warning' or 'Error' backups without incidents will not get sent to you.

Every web interface user can configure these values differently. UrBackup only sends reports of client backups to the user supplied address if the user has the 'logs' permission for the specific client. Thus if you want to send reports about one client to a specific email address you have to create a user for this client, login as that user and configure the reporting for that user. The user 'admin' can access logs of all clients and thus also gets reports about all clients.

## 7.3 Client specific settings

| Settings | Description | Default value |
|---|---|---|
| Interval for incremental file backups | The server will start incremental file backups in such intervals. | 5h |
| Interval for full file backups | The server will start full file backups in such intervals. | 30 days |
| Interval for incremental image backups | The server will start incremental image backups in such intervals. | 7 days |
| Interval for full image backups | The server will start full image backups in such intervals. | 30 days |
| Maximal number of incremental file backups | Maximal number of incremental file backups for this client. If the number of incremental file backups exceeds this number the server will start deleting old incremental file backups. | 100 |
| Minimal number of incremental file backups | Minimal number of incremental file backups for this client. If the server ran out of backup storage space the server can delete incremental file backups until this minimal number is reached. If deleting a backup would cause the number of incremental file backups to be lower than this number it aborts with an error message. | 40 |
| Maximal number of full file backups | Maximal number of full file backups for this client. If the number of full file backups exceeds this number the server will start deleting old full file backups. | 10 |
| Minimal number of full file backups | Minimal number of full file backups for this client. If the server ran out of backup storage space the server can delete full file backups until this minimal number is reached. If deleting a backup would cause the number of full file backups to be lower than this number it aborts with an error message. | 2 |
| Maximal number of incremental image backups | Maximal number of incremental image backups for this client. If the number of incremental image backups exceeds this number the server will start deleting old incremental image backups. | 30 |
| Minimal number of incremental image backups | Minimal number of incremental image backups for this client. If the server ran out of backup storage space the server can delete incremental image backups until this minimal number is reached. If deleting a backup would cause the number of incremental image backups to be lower than this number it aborts with an error message. | 4 |

| | | |
|---|---|---|
| Maximal number of full image backups | Maximal number of full image backups for this client. If the number of full image backups exceeds this number the server will start deleting old full image backups. | 5 |
| Minimal number of full image backups | Minimal number of full image backups for this client. If the server ran out of backup storage space the server can delete full image backups until this minimal number is reached. If deleting a backup would cause the number of full image backups to be lower than this number it aborts with an error message. | 2 |
| Delay after system start up | The server will wait for this number of minutes after discovering a new client before starting any backup | 0 min |
| Backup window | The server will only start backing up clients within this window. See section 7.3.1 for details. | 1-7/0-24 |
| Max backup speed for local network | The server will throttle the connections to the client to remain within this speed window. | - |
| Perform auto-updates silently | If this is selected automatic updates will be performed on the client without asking the user | Unchecked |
| Soft client quota | During the nightly cleanup UrBackup will remove backups of this client if there are more backups than the minimal number of file/image backups until this quota is met. The quota can be in percent (e.g. 20%) or absolute (e.g. 1500G, 2000M). | "" |
| Excluded files | Allows you to define which files should be excluded from backups. See section 7.3.2 for details | "" |
| Default directories to backup | Default directories which are backed up. See section 7.3.3 for details | "" |
| Volumes to backup | Specifies of which volumes an image backup is done. Separate different drive letters by a semicolon or comma. E.g. 'C;D' | C |
| Allow client-side changing of the directories to backup | Allow client(s) to change the directories of which a file backup is done | Checked |
| Allow client-side starting of incremental/full file backups | Allow the client(s) to start a file backup | Checked |
| Allow client-side starting of incemental/full image backups | Allow the client(s) to start an image backup | Checked |
| Allow client-side viewing of backup logs | Allow the client(s) to view the logs | Checked |
| Allow client-side pausing of backups | Allow the client(s) to pause backups | Checked |
| Allow client-side changing of settings | If this option is checked the clients can change their client specific settings via the client interface. If you do not check this the server settings always override the clients' settings. | Checked |

### 7.3.1 Backup window

The server will only start backing up clients within the backup windows. The clients can always start backups on their own, even outside the backup windows. If a backup is started it runs till it is finished and does not stop if the backup process does not complete within the backup window. A few examples for the backup window:
1-7/0-24: Allow backups on every day of the week on every hour.
Mon-Sun/0-24: An equivalent notation of the above
Mon-Fri/8:00-9:00, 19:30-20:30;Sat,Sun/0-24: On weekdays backup between 8 and 9 and between 19:30 and 20:30. On Saturday and Sunday the whole time.

As one can see a number can denote a day of the week (1-Monday, 2-Tuesday, 3-Wednesday, 4-Thursday, 5-Friday, 6-Saturday, 7-Sunday). You can also use the abbreviations of the days (Mon, Tues, Wed, Thurs, Fri, Sat, Sun). The times can either consist of only full hours or of hours with minutes. The hours are on the 24 hour clock. You can set multiple days and times per window definition, separated per ",". You can also set multiple window definitions. Separate them with ";".

### 7.3.2 Excluded files

You can exclude files with wild card matching. For example if you want to exclude all MP3s and movie files enter something like this:

```
*.mp3;*.avi;*.mkv;*.mp4;*.mpg;*.mpeg
```

If you want to exclude a directory e.g. Temp you can do it like this:

```
*/Temp/*
```

You can also give the full local name

```
C:\Users\User\AppData\Local\Temp\*
```

or the name you gave the location e.g.

```
C_\Users\User\AppData\Local\Temp
```

Rules are separated by a semicolon (";")

### 7.3.3 Default directories to backup

Enter the different locations separated by a semicolon (";") e.g.

```
C:\Users;C:\Program Files
```

If you want to give the backup locations a different name you can add one with the pipe symbol ("|") e.g:

```
C:\Users|User files;C:\Program Files|Programs
```

gives the "Users" directory the name "User files" and the "Program files" directory the name "Programs".

Those locations are only the default locations. Even if you check "Separate settings for this client" and disable "Allow client to change settings", once the client modified the paths, changes in this field are not used by the client any more.

## 7.4 Internet settings

| Settings | Description | Default value |
|---|---|---|
| Internet server name/IP | The IP or name the clients can reach the server at over the internet | "" |
| Internet server port | The port the server will listen for new clients on | 55415 |
| Do image backups over internet | If checked the server will allow image backups for this client/the clients | Not checked |
| Do full file backups over internet | If checked the server will allow full file backups for this client/the clients | Not checked |
| Max backup speed for internet connection | The maximal backup speed for the Internet client. Setting this correctly can help avoid saturating the Internet connection of a client | - |
| Total max backup speed for internet connection | The total accumulative backup speed for all Internet clients. This can help avoid saturating the server's Internet connection | - |
| Encrypted transfer | If checked all data between server and clients is encrypted | Checked |
| Compressed transfer | If checked all data between server and clients is compressed | Checked |
| Calculate file-hashes on the client | If checked the client calculates hashes for each file before the backups (only hashes of changed files are calculated). The file then does not have to be transfered if another client already transfered the same file | Not checked |

## 7.5 Advanced settings

In this section you will find global server settings which you only have to change for heavy or custom workloads. Most settings will need a server restart to come into effect.

### 7.5.1 Enabling temporary file buffers

Earlier versions of UrBackup always saved incoming data from clients first to temporary files and then copied it to the final destination (if the data is new) – the rationale being, that the final destination may be slow and you want to get the data from the client as fast as possible.
With UrBackup 1.1 this default behaviour was changed to directly copy the data to the final backup storage. The two settings allow you to reenable the old behaviour, e.g., because your backup storage is slow because it is deduplicated. If you reenable it make sure you have at least 1GB of space for each client, and at least as much space as the biggest file you are going to backup times the number of clients, on your temporary storage. You can change the temporary storage directory via the environment variable *TMPDIR* on GNU/Linux and in the server settings on Windows.

### 7.5.2 Transfer modes

UrBackup has different transfer modes for files and images. Those are:

- *raw*. Transfer the data as 'raw' as possible. This is the fastest transfer mode and uses the least amount of CPU cycles on server and client.
- *hashed*. Protects the transferred data from bit errors by hashing the data during the transfer. This uses CPU cycles on the client and the server.
  UrBackup uses TCP/IP to transfer the images and files. TCP/IP implements its own bit error detection mechanism (CRC32). If the network induces a lot of bit errors and if a lot of data is transferred (>2TB), however, the bit error detection mechanism of TCP/IP is not

enough to detect all occuring errors. The 'hashed' transfer mode adds an additional layer of protection to make bit errors less probable.

- *Block differences - hashed.* Only available for file backups (as it is automatically done for images). Blocks of the transferred files are compared using CRC32 and MD5 hash functions. Only blocks which have changed are send over the network. In cases where only some blocks of a file change, this reduces the amount of transferred data. It also causes more messages to be sent between server and client and uses CPU cycles, which is why it is only enabled for Internet clients per default.

### 7.5.3 File hash collection

During full file backups or for new files in incremental backup a database entry, which maps the files hash to its storage path is created. This entry alllows succeeding same files to be linked to the file encountered first, without storing it twice. To speed up this process, updates to the database are batched, i.e., file entries are first stored in a temporary table, and later moved over to the real database. As the temporary table is only visible to the thread currently running for one client, the longer this copy process is delayed the more it becomes possible that another client does not link to an already existing file, because it cannot find it in the database. With the file hash collection parameters you can influence when the copying from the temporary table to the shared database happens:

- *File hash collection amount.* After $x$ amount of file entries are in the temporary table the contents of the temporary table are copied to the shared (final) table.
- *File hash collection timeout.* After not having created any file entries in the temporary table for $x$ milliseconds the contents of the temporary table are copied to the shared (final) table.

### 7.5.4 Database cache size

UrBackup is using a per thread database cache. With the database cache size parameters you can influence the size of the database caches of some of the threads.

- *File hash collection database cachesize.* The size of the database cache for the thread doing the file lookups and linking. Increase this cache size if you have slow file backups (stalled at 100% and slowly decreasing queue). Make sure you have enough RAM, as UrBackup will use this amount of Megabytes times the number of simultaneous file backups.
- *Update stats database cachesize.* Size of the database cache for the thread which updates the statistics (i.e., which client uses how much space). There will only ever be one such thread and it will not be running while other backups are running, so you can set this to a relatively high value.

### 7.5.5 File entry cache database

Optionally UrBackup can use a cache database for file entries storing the file hash to files mapping used to hard link to files, if they already exist on the backup storage. Enabling the file entry cache may be advantageous if there are a lot of files on some clients or if you do full backups often. The file entry cache is enabled by selecting the file entry cache type. The cache is created when the server is restarted. Cache creation may take some time.

- *Cache database type for file entries.* By selecting something other then "None" the file entry cache is enabled. SQLite probably gives better performance when the underlying storage is slow and if the file entries do not fit into memory. LMDB should only be used on 64bit systems. When in doubt select "SQLite".

- *Cache database size for file entries.* Only relevant if LMDB was selected. This is the maximum database size of the LMDB database. LMDB creates a memory mapped file of this size, so UrBackup uses this much virtual memory. On Windows LMDB also creates a sparse file of this size (which can confuse backup programs).
- *Suspend index limit.* When the file entry cache is enabled, file entries are put into a temporary table and copied to the main table during a statistics update. If there are more file entries in the temporary table then the suspend index limit, indexes on the main table are destroyed, the file entries moved over from the temporary table and then rebuild. This may be faster for a large number of file entries then moving the file entries with indexes enabled.

### 7.6 End-to-end verification of all file backups

This is a setting for debugging purposes or for the paranoid. If end-to-end verification is enabled UrBackup clients >=1.3 will create file hashes for every file for every file backup reading every file that is to be backed up. At the end of the backup process the hashes of the files stored on the server are compared to the hashes calculated on the client. If hashes differ the backup fails and an email is sent to the server admin.

## 8 Miscellaneous

### 8.1 Manually update UrBackup clients

You should test UrBackup clients before using them on the clients. This means UrBackup should not automatically download the newest client version from the Internet and install it. This means disabling the autoupdate described in Section 7.1.6. You can still centrally update the client from the server if you disabled autoupdate. Go to `http://update1.urbackup.org` and download all files to */var/urbackup* on Linux and *C:\\Program Files\UrBackupServer\urbackup* per default on Windows. UrBackup will then push the new version to the clients once they reconnect. If you checked silent autoupdates, the new version will be installed silently on the clients, otherwise there will be a popup asking the user to install the new version.

### 8.2 Logging

UrBackup generally logs all backup related things into several log facilities. Each log message has a certain severity, namely *error*, *warning*, *info* or *debug*. Each log output can be filtered by this severity, such that e.g. only errors are shown. Both server and client have separate logs. During a backup process the UrBackup server tries to log everything which belongs to a certain backup in a client specific logs and at the end sends this log to the client. Those are the logs you see on the client interface. The same logs can also be viewed via the web interface in the "Logs" section. One can also send them per mail as described in subsection 7.2.2.
Everything which cannot be accredited to a certain client or which would cause too much log traffic is logged in a general log file. On Linux this is */var/log/urbackup.log* on Windows *C:\\Progam files\UrBackupServer\urbackup.log* for the server per default. The client has as defaults */var/log/urbackup_client.log* and *C:\\Progam files\UrBackup\debug.log*. Per default those files only contain log messages with severity *warning* or higher. In Windows there is a *args.txt* in the same directory as the log file. Change *warn* here to *debug*, *info* or *error* to get a different set of log messages. You need to restart the server for this change to come into effect. On Linux this depends on the distribution. On Debian one changes the setting in */etc/default/urbackup_srv*.

### 8.3 Used network ports

The Server uses following default ports:

| Port | Usage |
|---|---|
| 55413 | FastCGI for web interface |
| 55414 | HTTP web interface |
| 55415 | Internet clients |
| 35623 | UDP broadcasts (sending) |

The Client uses following default ports:

| Port | Usage |
|---|---|
| 35622 | UDP broadcasts (receiving) |
| 35621 | Sending files during file backups |
| 35623 | Commands and image backups |

# 9 Storage

The UrBackup server storage system is designed in a way that it is able to save as much backups as possible and thus uses up as much space on the storage partition as possible. With that in mind it is best practice to use a separate file system for the backup storage or to set a quota for the 'urbackup' user. Some filesystems behave badly if they are next to fully occupied (fragmentation and bad performance). With such filesystems you should always limit the quota UrBackup can use up to say 95% of all the available space.

## 9.1 Nightly backup deletion

UrBackup automatically deletes old file and image backups between 3am and 5am. Backups are deleted when a client has more incremental/full file/image backups then the configured maximum number of incremental/full file/image backups. Backups are deleted until the number of backups is within these limits again.
If the administrator has turned automatic shut-down on, this clean up process is started on server start up instead (as the server is most likely off during the night). Deleting backups and the succeeding updating of statistics can have a huge impact on system performance.

## 9.2 Emergency cleanup

If the server runs out of storage space during a backup it deletes backups until enough space is available again. Images are favoured over file backups and the oldest backups are deleted first. Backups are only deleted if there are at least the configured minimal number of incremental/full file/image backups other file/image backups in storage for the client owning the backup. If no such backup is found UrBackup cancels the current backup with a fatal error. Administrators should monitor storage space and add storage or configure the minimal number of incremental/full file/image backups to be lower if such an error occurs.

## 9.3 Cleanup for servers with file backups with lots of files

UrBackup's database is in a mode which enables high concurrency. Since the cleanup procedure can sometimes be bottlenecked by the database it may be advisable to switch the database into a mode which allows less concurrency but is fast for some operations for the cleanup procedure. This is not possible while UrBackup is running, so you should tweak the backup window such that you can be sure there are no backups running at some point. Then you can stop the server run the cleanup separately by calling

```
start_urbackup_server --cleanup x
```

on GNU/Linux or on Windows:

```
cleanup.bat x
```

Where $x$ is the percent of space to free on the backup storage or the number of Bytes/ Megabytes/ Gigabytes e.g. "20G" or "10%". If it should only delete old backups use "0%".

## 9.4 Cleaning the storage folder of files not known by UrBackup

Sometimes, e.g., by using a database backup, there are backups in the storage directory which UrBackup does not know about, i.e., there are no entries for those backups in the database. In does cases the command

```
start_urbackup_server --remove_unknown
```

on GNU/Linux or on Windows:

```
remove_unknown.bat
```

removes files and folders in the urbackup storage directory which are not present in the UrBackup database.

## 9.5 Archiving

UrBackup has the ability to automatically archive file backups. Archived file backups cannot be deleted by the nightly or emergency clean up – only when they are not archived any more. You can setup archival under Settings->Archival for all or specific clients. When an archival is due and the the server is currently in a archival window (See 9.5.1) the last file backup of the selected type will be archived for the selected amount of time. After that time it will be automatically not archived any more. You can see the archived backups in the "Backups" section. If a backup is archived for only a limited amount of time there will be a time symbol next to the check mark. Hovering over that time symbol will tell you how long that file backup will remain to be archived.

### 9.5.1 Archival window

The archival window allows you to archive backups at very specific times. The format is very similar to *crontab*. The fields are the same except that there are no minutes:

| Field | Allowed values | Remark |
|---|---|---|
| Hour | 0-23 | |
| Day of month | 1-31 | |
| Month | 1-12 | No names allowed |
| Day of week | 0-7 | 0 and 7 are Sunday |

To archive a file backup on the first Friday of every month we would then set "Archive every" to something like 27 days. After entering the time we want the backups archived for we would then add

```
*;*;*;5
```

as window (hour;day of month;month;day of week). To archive a backup every Friday we would set "Archive every" to a value greater than one day but less than 7 days. This works because both conditions have to apply: The time since the last backup archival must be greater than "Archive every" and the server must be currently in the archive window.
Other examples are easier. To archive a backup on the first of every month the window would be

```
*;1;*;*
```

and "Archive every" something like 2-27 days.
One can add several values for every field by separating them via a comma such that

```
*;*;*;3,5
```

and "Archive every" one day would archive a backup on Wednesday and Friday. Other advanced features found in *crontab* are not present.

## 9.6 Suitable Filesystems

Because UrBackup has the option to save all incoming data to temporary files first (see Section 7.5.1) and then copies them to the final location in parallel backup performance will still be good even if the backup storage space is slow. This means you can use a fully featured file system with compression and de-duplication without that much performance penalty. At the worst the server writes away an image backup over the night (having already saved the image's contents into temporary files during the day). This section will show which filesystems are suited for UrBackup.

### 9.6.1 Ext4/XFS

Ext4 and XFS, are both available in Linux and can handle big files, which is needed for storing image backups. They do not have compression or de duplication though. Compression can be achieved by using a fuse file system on top of them such as fusecompress. There are some block-level de-duplication fuse layers as well, but I would advise against them as they do not seem very stable. You will have to use the kernel user/group level quota support to limit the UrBackup storage usage.

### 9.6.2 NTFS

NTFS is pretty much the only option you have if you run the UrBackup server under Windows. It supports large files and compression as well as hard links and as such is even more suited for UrBackup than the standard Linux filesystems XFS and Ext4.

### 9.6.3 btrfs

Btrfs is a pretty new Linux file system and as such it is probably not suited for production use yet. It supports compression and supports block-level deduplication. UrBackup has a special snapshotting backup mode which is much faster with btrfs. See 9.7.2 for details.

### 9.6.4 ZFS

ZFS is a file system originating from Solaris. It is available as a fuse module for Linux (zfs-fuse) and as a kernel module (ZFSOnLinux). ZFSOnLinux as of 0.6.2 still has memory issues with UrBackup/rsync style workloads, so you should carefully evaluate if it is stable enough. There were licensing issues which prevented prior porting of ZFS to Linux. If you want the most performance and stability an option would be using a BSD or Debian/kFreeBSD. The ZFS in the BSD kernels is stable. The upstream Solaris ZFS has been available for some time and as such should be very stable as well. ZFS has some pretty neat features like compression, block-level de-duplication, snapshots and build in raid support that make it well suited for backup storage. How to build a UrBackup server with ZFS is described in detail in section 9.7.1.

## 9.7 Storage setup proposals

In this section a sample storage setup with ZFS is shown which allows off-site backups via Internet or via tape like manual off-site storage and a storage setup using the Linux file system btrfs using the btrfs snapshot mechanism to speed up file backup creation and destruction and to save the file backups more efficiently.

### 9.7.1 Mirrored storage with ZFS

Note: It is assumed that UrBackup runs on a Unix system such as Linux or BSD. An example would be Debian/Linux or Debian/kFreeBSD with the kFreeBSD kernel being preferred, because of its better ZFS performance. We will use all ZFS features such as compression, de-duplication and snapshots. It is assumed that the server has two hard drives (sdb,sdc) dedicated to backups and a hot swappable hard drive slot (sdd). It is assumed there is a caching device to speed up de-duplication as well in /dev/sde. Even a fast usb stick can speed up de-duplication because it has better random access performance then normal hard disks. Use SSDs for best performance.

First setup the server such that the temporary directory (/tmp) is on a sufficiently large performant file system. If you have a raid setup you could set /tmp to be on a striped device. We will now create a backup storage file system in /media/BACKUP.
Create a ZFS-pool 'backup' from the two hard drives. The two are mirrored. Put a hard drive of the same size into the hot swappable hard drive slot. We will mirror it as well:

```
zpool create backup /dev/sdb /dev/sdc /dev/sdd cache /dev/sde -m /media/BACKUP
```

Enable de-duplication and compression. You do not need to set a quota as de-duplication fragments everything anyway (that's why we need the caching device).

```
zfs set dedup=on backup
zfs set compression=on backup
```

Now we want to implement a grandfather, father, son or similar backup scheme where we can put hard disks in a fireproof safe. So each time we want to have an off-site backup we remove the hot swappable device and plug in a new one. Then we either run

```
zpool replace backup /dev/sdd /dev/sdd
```

or

```
zpool scrub
```

You can see the progress of the re-silvering/scrub with 'zpool status'. Once it is done you are ready to take another hard disk somewhere.

Now we want to save the backups on a server on another location. First we create the ZFS backup pool on this other location.
Then we transfer the full file system (otherserver is the host name of the other server):

```
zfs snapshot backup@last
zfs send backup@last | ssh -l root otherserver zfs recv backup@last
```

Once this is done we can sync the two filesystems incrementally:

```
zfs snapshot backup@now
ssh -l root otherserver zfs rollback -r backup@last
zfs send -i backup@last backup@now | ssh -l root otherserver zfs recv backup@now
zfs destroy backup@last
zfs rename backup@last backup@now
ssh -l root otherserver zfs destory backup@last
ssh -l root otherserver zfs rename backup@last backup@now
```

You can also save these full and incremental zfs streams into files on the other server and not directly into a ZFS file system.

### 9.7.2 Btrfs

Btrfs is an advanced file system for Linux capable of creating copy on write snapshots of sub-volumes. Currently, as of Linux kernel 3.12, btrfs is still declared unstable. This is not just a lablel, during testing users of UrBackup ran into performance problems or were unable to delete files. It is advised that you think twice before using btrfs as storage backend, even though it does have considerable advantages compared to other file systems. For UrBackup to be able to use the snapshotting mechanism the Linux kernel must be at least 3.6.

If UrBackup detects a btrfs filesystem it uses a special snaphotting file backup mode. It saves every file backup of every client in a separate btrfs sub-volume. When creating a incremental file backup UrBackup then creates a snapshot of the last file backup and removes, adds and changes only the files required to update the snapshot. This is much faster than the normal method, where UrBackup links (hard link) every file in the new incremental file backups to the file in the last one. It also uses less metadata (information about files, i.e., directory entries). If a new/changed file is detected as the same as a file of another client or the same as in another backup, UrBackup uses cross device reflinks to save the data in this file only once on the file system. Using btrfs also allows UrBackup to backup files changed between incremental backups in a way that only changed data in the file is stored. This greatly decreases the storage amount needed for backups, especially for large database files (such as e.g. the Outlook archive file). The ZFS deduplication in the previous section (9.7.1) saves even more storage, but comes at a much greater cost in form of a massive decrease of read and write performance.

In order to create and remove btrfs snapshots UrBackup installs a setuid executable *urbackup_snapshot_helper*. UrBackup also uses this tool to test if cross-device reflinks are possible. Only if UrBackup can create cross-device reflinks and is able to create and destroy btrfs snapshots, is the btrfs mode enabled. *urbackup_snapshot_helper* needs to be told separately where the UrBackup backup folder is. This path is read from */etc/urbackup/backupfolder*. Thus, if */media/backup/urbackup* is the folder where UrBackup is saving the paths, following commands would properly create this file:

```
mkdir /etc/urbackup
echo "/etc/urbackup/backupfolder" > /etc/urbackup/backupfolder
```

You can then test if UrBackup will use the btrfs features via

```
urbackup_snapshot_helper test
echo $?
```

If *urbackup_snapshot_helper* returns 0 UrBackup will use the btrfs features after a server restart. If not, you need to check if the kernel is new enough and that the backup folder is on a btrfs volume.

You should then be able to enjoy much faster incremental file backups which use less storage space.