

**NAME**

duplicity – Encrypted incremental backup to local or remote storage.

**SYNOPSIS**

For detailed descriptions for each command see chapter **ACTIONS**.

**duplicity [full|incremental] [options] source\_directory target\_url**

**duplicity verify [options] [--compare-data] [--file-to-restore <relpath>] [--time time] source\_url target\_directory**

**duplicity collection-status [options] target\_url**

**duplicity list-current-files [options] [--time time] target\_url**

**duplicity [restore] [options] [--file-to-restore <relpath>] [--time time] source\_url target\_directory**

**duplicity remove-older-than <time> [options] [--force] target\_url**

**duplicity remove-all-but-n-full <count> [options] [--force] target\_url**

**duplicity remove-all-inc-of-but-n-full <count> [options] [--force] target\_url**

**duplicity cleanup [options] [--force] [--extra-clean] target\_url**

**REQUIREMENTS**

Duplicity requires a POSIX-like operating system with a **python** interpreter version 2.4+ installed. It is best used under GNU/Linux.

Some backends also require additional components (probably available as packages for your specific platform):

**boto backend** (S3 Amazon Web Services, Google Cloud Storage)

**boto version 2.0+** - <http://github.com/boto/boto>

**cloudfiles backend (deprecated)** (e.g. Rackspace Open Cloud)

**Cloud Files Python API (deprecated)** -

[http://www.rackspace.com/knowledge\\_center/article/python-api-installation-for-cloud-files](http://www.rackspace.com/knowledge_center/article/python-api-installation-for-cloud-files)

**cfpyrax backend** (Rackspace Cloud)

**Rackspace CloudFiles Pyrax API** - <http://docs.rackspace.com/sdks/guide/content/python.html>

**dpbx backend (Dropbox)**

**Dropbox Python SDK** - <https://www.dropbox.com/developers/reference/sdk>

**ftp backend**

**NcFTP Client** - <http://www.ncftp.com/>

**ftps backend**

**LFTP Client** - <http://lftp.yar.ru/>

**gdocs backend** (Google Docs)

**Google Data APIs Python Client Library** - <http://code.google.com/p/gdata-python-client/>

**gio backend** (Gnome VFS API)

**PyGObject** - <http://live.gnome.org/PyGObject>

**D-Bus (dbus)**- <http://www.freedesktop.org/wiki/Software/dbus>

**rsync backend**

**rsync client binary** - <http://rsync.samba.org/>

**mega backend** ([mega.co.nz](http://mega.co.nz))

**Python library for mega API** - <https://github.com/ckornacker/mega.py>, ubuntu ppa -  
ppa:ckornacker/backup

**Par2 Wrapper Backend**

**par2cmdline** - <http://parchive.sourceforge.net/>

There are two **ssh backends** for scp/sftp/ssh access (also see **A NOTE ON SSH BACKENDS**).

**ssh paramiko backend** (enabled by default)

**paramiko** (SSH2 for python) - <http://pypi.python.org/pypi/paramiko> (downloads);  
<http://github.com/paramiko/paramiko> (project page)

**pycrypto** (Python Cryptography Toolkit) - <http://www.dlitz.net/software/pycrypto/>

**ssh pexpect backend**

**sftp/scp client binaries** OpenSSH - <http://www.openssh.com/>

**swift backend (OpenStack Object Storage)**

**Python swiftclient module** - <https://github.com/openstack/python-swiftclient/>

**Python keystoneclient module** - <https://github.com/openstack/python-keystoneclient/>

**webdav backend**

**certificate authority database file** for ssl certificate verification of HTTPS connections -  
<http://curl.haxx.se/docs/caextract.html>

(also see **A NOTE ON SSL CERTIFICATE VERIFICATION**).

**DESCRIPTION**

Duplicity incrementally backs up files and folders into tar-format volumes encrypted with GnuPG and places them to a remote (or local) storage backend. See chapter **URL FORMAT** for a list of all supported backends and how to address them. Because duplicity uses librsync, incremental backups are space efficient and only record the parts of files that have changed since the last backup. Currently duplicity supports deleted files, full Unix permissions, uid/gid, directories, symbolic links, fifos, etc., but not hard links.

If you are backing up the root directory /, remember to `--exclude /proc`, or else duplicity will probably crash on the weird stuff in there.

**EXAMPLES**

Here is an example of a backup, using sftp to back up /home/me to some\_dir on the other.host machine:

```
duplicity /home/me sftp://uid@other.host/some_dir
```

If the above is run repeatedly, the first will be a full backup, and subsequent ones will be incremental. To force a full backup, use the *full* action:

```
duplicity full /home/me sftp://uid@other.host/some_dir
```

or enforcing a full every other time via `--full-if-older-than <time>`, e.g. a full every month:

```
duplicity --full-if-older-than 1M /home/me sftp://uid@other.host/some_dir
```

Now suppose we accidentally delete /home/me and want to restore it the way it was at the time of last backup:

```
duplicity sftp://uid@other.host/some_dir /home/me
```

Duplicity enters restore mode because the URL comes before the local directory. If we wanted to restore just the file "Mail/article" in /home/me as it was three days ago into /home/me/restored\_file:

```
duplicity -t 3D --file-to-restore Mail/article sftp://uid@other.host/some_dir /home/me/restored_file
```

The following command compares the latest backup with the current files:

```
duplicity verify sftp://uid@other.host/some_dir /home/me
```

Finally, duplicity recognizes several include/exclude options. For instance, the following will backup the root directory, but exclude /mnt, /tmp, and /proc:

```
duplicity --exclude /mnt --exclude /tmp --exclude /proc / file:///usr/local/backup
```

Note that in this case the destination is the local directory /usr/local/backup. The following will backup only the /home and /etc directories under root:

```
duplicity --include /home --include /etc --exclude '*' / file:///usr/local/backup
```

Duplicity can also access a repository via ftp. If a user name is given, the environment variable FTP\_PASSWORD is read to determine the password:

```
FTP_PASSWORD=mypassword duplicity /local/dir ftp://user@other.host/some_dir
```

## ACTIONS

Duplicity knows action commands, which can be finetuned with options.

The actions for backup (full,incr) and restoration (restore) can as well be left out as duplicity detects in what mode it should switch to by the order of target URL and local folder. If the target URL comes before the local folder a restore is in order, is the local folder before target URL then this folder is about to be backed up to the target URL.

If a backup is in order and old signatures can be found duplicity automatically performs an incremental backup.

**Note:** The following explanations explain some but **not** all options that can be used in connection with that action command. Consult the OPTIONS section for more detailed informations.

**full** <folder> <url>

Perform a full backup. A new backup chain is started even if signatures are available for an incremental backup.

**incr** <folder> <url>

If this is requested an incremental backup will be performed. Duplicity will abort if no old signatures can be found.

**verify** [--compare-data] [--time <time>] [--file-to-restore <relpath>] <url> <folder>

Verify compares the backup contents with the source folder. duplicity will exit with a non-zero error level if any files are different. On verbosity level info (4) or higher, a message for each file that has changed will be logged.

The --file-to-restore option restricts verify to that file or folder. The --time option allows to select a backup to verify against. The --compare-data option enables data comparison (see below).

**collection-status** <url>

Summarize the status of the backup repository by printing the chains and sets found, and the number of volumes in each.

**list-current-files** [--time <time>] <url>

Lists the files contained in the most current backup or backup at time. The information will be extracted from the signature files, not the archive data itself. Thus the whole archive does not have to be downloaded, but on the other hand if the archive has been deleted or corrupted, this command will not detect it.

**restore** [--file-to-restore <relpath>] [--time <time>] <url> <target\_folder>

You can restore the full monty or selected folders/files from a specific time. Use the relative path as it is printed by **list-current-files**. Usually not needed as duplicity enters restore mode when it detects that the URL comes before the local folder.

**remove-older-than** <time> [--force] <url>

Delete all backup sets older than the given time. Old backup sets will not be deleted if backup sets newer than *time* depend on them. See the **TIME FORMATS** section for more information. Note, this action cannot be combined with backup or other actions, such as cleanup. Note also that *--force* will be needed to delete the files instead of just listing them.

**remove-all-but-n-full** <count> [--force] <url>

Delete all backups sets that are older than the count:th last full backup (in other words, keep the last *count* full backups and associated incremental sets). *count* must be larger than zero. A value of 1 means that only the single most recent backup chain will be kept. Note that *--force* will be needed to delete the files instead of just listing them.

**remove-all-inc-of-but-n-full** <count> [--force] <url>

Delete incremental sets of all backups sets that are older than the count:th last full backup (in other words, keep only old full backups and not their increments). *count* must be larger than zero. A value of 1 means that only the single most recent backup chain will be kept intact. Note that *--force* will be needed to delete the files instead of just listing them.

**cleanup** [--force] [--extra-clean] <url>

Delete the extraneous duplicity files on the given backend. Non-duplicity files, or files in complete data sets will not be deleted. This should only be necessary after a duplicity session fails or is aborted prematurely. Note that *--force* will be needed to delete the files instead of just listing them.

## OPTIONS

**--allow-source-mismatch**

Do not abort on attempts to use the same archive dir or remote backend to back up different directories. duplicity will tell you if you need this switch.

**--archive-dir** *path*

The archive directory. **NOTE:** This option changed in 0.6.0. The archive directory is now necessary in order to manage persistence for current and future enhancements. As such, this option is now used only to change the location of the archive directory. The archive directory should **not** be deleted, or duplicity will have to recreate it from the remote repository (which may require decrypting the backup contents).

When backing up or restoring, this option specifies that the local archive directory is to be created in *path*. If the archive directory is not specified, the default will be to create the archive directory in `~/.cache/duplicity/`.

The archive directory can be shared between backups to multiple targets, because a subdirectory of the archive dir is used for individual backups (see **--name**).

The combination of archive directory and backup name must be unique in order to separate the data of different backups.

The interaction between the **--archive-dir** and the **--name** options allows for four possible

combinations for the location of the archive dir:

1. neither specified (default)  
*~/.cache/duplicity/hash-of-url*
2. `--archive-dir=/arch`, no `--name`  
*/arch/hash-of-url*
3. no `--archive-dir`, `--name=foo`  
*~/.cache/duplicity/foo*
4. `--archive-dir=/arch`, `--name=foo`  
*/arch/foo*

#### **--asynchronous-upload**

(EXPERIMENTAL) Perform file uploads asynchronously in the background, with respect to volume creation. This means that duplicity can upload a volume while, at the same time, preparing the next volume for upload. The intended end-result is a faster backup, because the local CPU and your bandwidth can be more consistently utilized. Use of this option implies additional need for disk space in the temporary storage location; rather than needing to store only one volume at a time, enough storage space is required to store two volumes.

#### **--cf-backend** *backend*

Allows the explicit selection of a cloudfiles backend. Defaults to **pyrax**. Alternatively you might choose **cloudfiles**.

#### **--compare-data**

Enable data comparison of regular files on action verify. This is disabled by default for performance reasons.

#### **--dry-run**

Calculate what would be done, but do not perform any backend actions

#### **--encrypt-key** *key-id*

When backing up, encrypt to the given public key, instead of using symmetric (traditional) encryption. Can be specified multiple times. The key-id can be given in any of the formats supported by GnuPG; see **gpg(1)**, section "HOW TO SPECIFY A USER ID" for details.

#### **--encrypt-secret-keyring** *filename*

This option can only be used with **--encrypt-key**, and changes the path to the secret keyring for the encrypt key to *filename*. This keyring is not used when creating a backup. If not specified, the default secret keyring is used which is usually located at `.gnupg/secring.gpg`

#### **--encrypt-sign-key** *key-id*

Convenience parameter. Same as **--encrypt-key** *key-id* **--sign-key** *key-id*.

#### **--exclude** *shell\_pattern*

Exclude the file or files matched by *shell\_pattern*. If a directory is matched, then files under that directory will also be matched. See the **FILE SELECTION** section for more information.

**--exclude-device-files**

Exclude all device files. This can be useful for security/permissions reasons or if rdiff-backup is not handling device files correctly.

**--exclude-filelist** *filename*

Excludes the files listed in *filename*. See the **FILE SELECTION** section for more information.

**--exclude-filelist-stdin**

Like **--exclude-filelist**, but the list of files will be read from standard input. See the **FILE SELECTION** section for more information.

**--exclude-globbing-filelist** *filename*

Like **--exclude-filelist** but each line of the filelist will be interpreted according to the same rules as **--include** and **--exclude**.

**--exclude-if-present** *filename*

Exclude directories if *filename* is present. This option needs to come before any other include or exclude options.

**--exclude-other-filesystems**

Exclude files on file systems (identified by device number) other than the file system the root of the source directory is on.

**--exclude-regexp** *regexp*

Exclude files matching the given *regexp*. Unlike the **--exclude** option, this option does not match files in a directory it matches. See the **FILE SELECTION** section for more information.

**--extra-clean**

When cleaning up, be more aggressive about saving space. For example, this may delete signature files for old backup chains. See the **cleanup** argument for more information.

**--file-prefix, --file-prefix-manifest, --file-prefix-archive, --file-prefix-signature**

Adds a prefix to all files, manifest files, archive files, and/or signature files.

The same set of prefixes must be passed in on backup and restore.

If both global and type-specific prefixes are set, global prefix will go before type-specific prefixes.

See also **A NOTE ON FILENAME PREFIXES**

**--file-to-restore** *path*

This option may be given in restore mode, causing only *path* to be restored instead of the entire contents of the backup archive. *path* should be given relative to the root of the directory backed up.

**--full-if-older-than** *time*

Perform a full backup if an incremental backup is requested, but the latest full backup in the collection is older than the given *time*. See the **TIME FORMATS** section for more information.

- force** Proceed even if data loss might result. Duplicity will let the user know when this option is required.
- ftp-passive**  
Use passive (PASV) data connections. The default is to use passive, but to fallback to regular if the passive connection fails or times out.
- ftp-regular**  
Use regular (PORT) data connections.
- gio** Use the GIO backend and interpret any URLs as GIO would.
- hidden-encrypt-key** *key-id*  
Same as **--encrypt-key**, but it hides user's key id from encrypted file. It uses the gpg's **--hidden-recipient** command to obfuscate the owner of the backup. On restore, gpg will automatically try all available secret keys in order to decrypt the backup. See gpg(1) for more details.
- ignore-errors**  
Try to ignore certain errors if they happen. This option is only intended to allow the restoration of a backup in the face of certain problems that would otherwise cause the backup to fail. It is not ever recommended to use this option unless you have a situation where you are trying to restore from backup and it is failing because of an issue which you want duplicity to ignore. Even then, depending on the issue, this option may not have an effect.
- Please note that while ignored errors will be logged, there will be no summary at the end of the operation to tell you what was ignored, if anything. If this is used for emergency restoration of data, it is recommended that you run the backup in such a way that you can revisit the backup log (look for lines containing the string `IGNORED_ERROR`).
- If you ever have to use this option for reasons that are not understood or understood but not your own responsibility, please contact duplicity maintainers. The need to use this option under production circumstances would normally be considered a bug.
- imap-mailbox** *option*  
Allows you to specify a different mailbox. The default is "INBOX". Other languages may require a different mailbox than the default.
- gpg-options** *options*  
Allows you to pass options to gpg encryption. The *options* list should be of the form "opt1=parm1 opt2=parm2" where the string is quoted and the only spaces allowed are between options.
- include** *shell\_pattern*  
Similar to **--exclude** but include matched files instead. Unlike **--exclude**, this option will also match parent directories of matched files (although not necessarily their contents). See the **FILE SELECTION** section for more information.
- include-filelist** *filename*  
Like **--exclude-filelist**, but include the listed files instead. See the **FILE SELECTION** section for more information.

**--include-filelist-stdin**

Like **--include-filelist**, but read the list of included files from standard input.

**--include-globbing-filelist** *filename*

Like **--include-filelist** but each line of the filelist will be interpreted according to the same rules as **--include** and **--exclude**.

**--include-regexp** *regexp*

Include files matching the regular expression *regexp*. Only files explicitly matched by *regexp* will be included by this option. See the **FILE SELECTION** section for more information.

**--log-fd** *number*

Write specially-formatted versions of output messages to the specified file descriptor. The format used is designed to be easily consumable by other programs.

**--log-file** *filename*

Write specially-formatted versions of output messages to the specified file. The format used is designed to be easily consumable by other programs.

**--name** *symbolicname*

Set the symbolic name of the backup being operated on. The intent is to use a separate name for each logically distinct backup. For example, someone may use "home\_daily\_s3" for the daily backup of a home directory to Amazon S3. The structure of the name is up to the user, it is only important that the names be distinct. The symbolic name is currently only used to affect the expansion of **--archive-dir**, but may be used for additional features in the future. Users running more than one distinct backup are encouraged to use this option.

If not specified, the default value is a hash of the backend URL.

**--no-encryption**

Do not use GnuPG to encrypt files on remote system. Instead just write gzipped volumes.

**--no-print-statistics**

By default duplicity will print statistics about the current session after a successful backup. This switch disables that behavior.

**--null-separator**

Use nulls (`\0`) instead of newlines (`\n`) as line separators, which may help when dealing with filenames containing newlines. This affects the expected format of the files specified by the `--{include|exclude}-filelist[-stdin]` switches as well as the format of the directory statistics file.

**--numeric-owner**

On restore always use the numeric uid/gid from the archive and not the archived user/group names, which is the default behaviour. Recommended for restoring from live cds which might have the users with identical names but different uids/gids.

**--num-retries** *number*

Number of retries to make on errors before giving up.



**--old-filenames**

Use the old filename format (incompatible with Windows/Samba) rather than the new filename format.

**--par2-redundancy *percent***

Adjust the level of redundancy in *percent* for Par2 recovery files (default 10%)

**--progress**

When selected, duplicity will output the current upload progress and estimated upload time. To annotate changes, it will perform a first dry-run before a full or incremental, and then runs the real operation estimating the real upload progress.

**--progress\_rate *number***

Sets the update rate at which duplicity will output the upload progress messages (requires **--progress** option). Default is to prompt the status each 3 seconds.

**--rename *<original path>* *<new path>***

Treats the path *orig* in the backup as if it were the path *new*. Can be passed multiple times. An example:

```
duplicity restore --rename Documents/metal Music/metal sftp://uid@other.host/some_dir
/home/me
```

**--rsync-options *options***

Allows you to pass options to the rsync backend. The *options* list should be of the form "opt1=parm1 opt2=parm2" where the option string is quoted and the only spaces allowed are between options. The option string will be passed verbatim to rsync, after any internally generated option designating the remote port to use. Here is a possibly useful example:

```
duplicity --rsync-options="--partial-dir=.rsync-partial" /home/me rsync://uid@other.host/some_dir
```

**--s3-european-buckets**

When using the Amazon S3 backend, create buckets in Europe instead of the default (requires **--s3-use-new-style**). Also see the **EUROPEAN S3 BUCKETS** section.

**--s3-unencrypted-connection**

Don't use SSL for connections to S3.

This may be much faster, at some cost to confidentiality.

With this option, anyone who can observe traffic between your computer and S3 will be able to tell: that you are using Duplicity, the name of the bucket, your AWS Access Key ID, the increment dates and the amount of data in each increment.

This option affects only the connection, not the GPG encryption of the backup increment files. Unless that is disabled, an observer will not be able to see the file names or contents.

**--s3-use-new-style**

When operating on Amazon S3 buckets, use new-style subdomain bucket addressing. This is now the preferred method to access Amazon S3, but is not backwards compatible if your bucket name contains upper-case characters or other characters that are not valid in a hostname.

**--s3-use-rrs**

Store volumes using Reduced Redundancy Storage when uploading to Amazon S3. This will lower the cost of storage but also lower the durability of stored volumes to 99.99% instead the 99.999999999% durability offered by Standard Storage on S3.

**--s3-use-multiprocessing**

Allow multipart volume uploads to S3 through multiprocessing. This option requires Python 2.6 and can be used to make uploads to S3 more efficient. If enabled, files duplicity uploads to S3 will be split into chunks and uploaded in parallel. Useful if you want to saturate your bandwidth or if large files are failing during upload.

**--s3-multipart-chunk-size**

Chunk size (in MB) used for S3 multipart uploads. Make this smaller than **--volsize** to maximize the use of your bandwidth. For example, a chunk size of 10MB with a volsize of 30MB will result in 3 chunks per volume upload.

**--s3-multipart-max-procs**

Specify the maximum number of processes to spawn when performing a multipart upload to S3. By default, this will choose the number of processors detected on your system (e.g. 4 for a 4-core system). You can adjust this number as required to ensure you don't overload your system while maximizing the use of your bandwidth.

**--s3\_multipart\_max\_timeout**

You can control the maximum time (in seconds) a multipart upload can spend on uploading a single chunk to S3. This may be useful if you find your system hanging on multipart uploads or if you'd like to control the time variance when uploading to S3 to ensure you kill connections to slow S3 endpoints.

**--scp-command *command***

(only ssh pexpect backend with **--use-scp** enabled) The *command* will be used instead of "scp" to send or receive files. To list and delete existing files, the sftp command is used. See also **A NOTE ON SSH BACKENDS** section **SSH pexpect backend**.

**--sftp-command *command***

(only ssh pexpect backend) The *command* will be used instead of "sftp". See also **A NOTE ON SSH BACKENDS** section **SSH pexpect backend**.

**--short-filenames**

If this option is specified, the names of the files duplicity writes will be shorter (about 30 chars) but less understandable. This may be useful when backing up to MacOS or another OS or FS that doesn't support long filenames.

**--sign-key *key-id***

This option can be used when backing up, restoring or verifying. When backing up, all backup files will be signed with keyid *key*. When restoring, duplicity will signal an error if any remote file is not signed with the given key-id. The key-id can be given in any of the formats supported by GnuPG; see **gpg(1)**, section "HOW TO SPECIFY A USER ID" for details. Should be specified only once because currently only **one** signing key is supported. Last entry overrides all other entries.

See also **A NOTE ON SYMMETRIC ENCRYPTION AND SIGNING**

**--ssh-askpass**

Tells the ssh backend to prompt the user for the remote system password, if it was not defined in target url and no FTP\_PASSWORD env var is set. This password is also used for passphrase-protected ssh keys.

**--ssh-backend** *backend*

Allows the explicit selection of a ssh backend. Defaults to **paramiko**. Alternatively you might choose **pexpect**.

See also **A NOTE ON SSH BACKENDS**.

**--ssh-options** *options*

Allows you to pass options to the ssh backend. The *options* list should be of the form "-oOpt1=parm1 -oOpt2=parm2" where the option string is quoted and the only spaces allowed are between options. The option string will be passed verbatim to both scp and sftp, whose command line syntax differs slightly hence the options should therefore be given in the long option format described in **ssh\_config(5)**, like in this example:

```
duplicity --ssh-options="-oProtocol=2 -oIdentityFile=/my/backup/id" /home/me
scp://uid@other.host/some_dir
```

**NOTE:** *ssh paramiko backend* currently supports only the **-oIdentityFile** setting.

**--ssl-cacert-file** *file*

(**only webdav backend**) Provide a cacert file for ssl certificate verification.

See also **A NOTE ON SSL CERTIFICATE VERIFICATION**.

**--ssl-no-check-certificate**

(**only webdav backend**) Disable ssl certificate verification.

See also **A NOTE ON SSL CERTIFICATE VERIFICATION**.

**--tempdir** *directory*

Use this existing directory for duplicity temporary files instead of the system default, which is usually the /tmp directory. This option supersedes any environment variable.

See also **ENVIRONMENT VARIABLES**.

**-ttime, --time** *time*, **--restore-time** *time*

Specify the time from which to restore or list files.

**--time-separator** *char*

Use *char* as the time separator in filenames instead of colon (":").

**--timeout** *seconds*

Use *seconds* as the socket timeout value if duplicity begins to timeout during network operations. The default is 30 seconds.

**--use-agent**

If this option is specified, then **--use-agent** is passed to the GnuPG encryption process and it will try to connect to **gpg-agent** before it asks for a passphrase for **--encrypt-key** or **--sign-key** if needed.

**Note:** GnuPG 2 and newer ignore this option and will always use a running **gpg-agent** if no passphrase was delivered.

**--use-scp**

If this option is specified, then the ssh backend will use the scp protocol rather than sftp for backend operations.

See also **A NOTE ON SSH BACKENDS**.

**--verbosity *level*, -v*level***

Specify output verbosity level (log level). Named levels and corresponding values are 0 Error, 2 Warning, 4 Notice (default), 8 Info, 9 Debug (noisiest).

*level* may also be

**a character:** e, w, n, i, d

**a word:** error, warning, notice, info, debug

The options -v4, -vn and -vnotice are functionally equivalent, as are the mixed/upper-case versions -vN, -vNotice and -vNOTICE.

**--version**

Print duplicity's version and quit.

**--volsize *number***

Change the volume size to *number* Mb. Default is 25Mb.

**ENVIRONMENT VARIABLES****TMPDIR, TEMP, TMP**

In decreasing order of importance, specifies the directory to use for temporary files (inherited from Python's tempfile module). Eventually the option **--tempdir** supercedes any of these.

**FTP\_PASSWORD**

Supported by most backends which are password capable. More secure than setting it in the backend url (which might be readable in the operating systems process listing to other users on the same machine).

**PASSPHRASE**

This passphrase is passed to GnuPG. If this is not set, the user will be prompted for the passphrase.

**SIGN\_PASSPHRASE**

The passphrase to be used for **--sign-key**. If omitted **and** sign key is also one of the keys to encrypt against **PASSPHRASE** will be reused instead. Otherwise, if passphrase is needed but not set the user will be prompted for it.

**URL FORMAT**

Duplicity uses the URL format (as standard as possible) to define data locations. The generic format for a URL is:

```
scheme://[user[:password]@]host[:port]/[/]path
```

It is not recommended to expose the password on the command line since it could be revealed to anyone with permissions to do process listings, it is permitted however. Consider setting the environment variable **FTP\_PASSWORD** instead, which is used by most, if not all backends, regardless of it's name.

In protocols that support it, the path may be preceded by a single slash, '/path', to represent a relative path to the target home directory, or preceded by a double slash, '//path', to represent an absolute filesystem path.

Formats of each of the URL schemes follow:

**Rackspace Cloud Files**

```
cf+http://container_name
```

See also **A NOTE ON CLOUD FILES ACCESS**

### Dropbox

dpbx:///some\_dir

Make sure to read **A NOTE ON DROPBOX ACCESS** first!

file://[relative/absolute]/local/path

ftp[s]://user[:password]@other.host[:port]/some\_dir

gdocs://user[:password]@other.host/some\_dir

### Google Cloud Storage

gs://bucket[/prefix]

hsi://user[:password]@other.host/some\_dir

imap[s]://user[:password]@host.com[/from\_address\_prefix]

See also **A NOTE ON IMAP**

mega://user[:password]@mega.co.nz/some\_dir

### Par2 Wrapper Backend

par2+scheme://[user[:password]@]host[:port]/[/]path

See also **A NOTE ON PAR2 WRAPPER BACKEND**

### using rsync daemon

rsync://user[:password]@host.com[:port]::[/]module/some\_dir

### using rsync over ssh (only key auth)

rsync://user@host.com[:port]/[relative/absolute]\_path

s3://host/bucket\_name[/prefix]

s3+http://bucket\_name[/prefix]

See also **A NOTE ON EUROPEAN S3 BUCKETS**

scp://.. or ssh://.. are synonymous with

sftp://user[:password]@other.host[:port]/[/]some\_dir

See also **--ssh-backend**, **--ssh-askpass**, **--use-scp**, **--ssh-options** and **A NOTE ON SSH BACKENDS**.

swift://container\_name

See also **A NOTE ON SWIFT (OPENSTACK OBJECT STORAGE) ACCESS**

tahoe://alias/directory

webdav[s]://user[:password]@other.host[:port]/some\_dir

## TIME FORMATS

duplicity uses time strings in two places. Firstly, many of the files duplicity creates will have the time in their filenames in the w3 datetime format as described in a w3 note at <http://www.w3.org/TR/NOTE-datetime>. Basically they look like "2001-07-15T04:09:38-07:00", which means what it looks like. The "-07:00" section means the time zone is 7 hours behind UTC.

Secondly, the **-t**, **--time**, and **--restore-time** options take a time string, which can be given in any of several formats:

1. the string "now" (refers to the current time)
2. a sequences of digits, like "123456890" (indicating the time in seconds after the epoch)
3. A string like "2002-01-25T07:00:00+02:00" in datetime format
4. An interval, which is a number followed by one of the characters s, m, h, D, W, M, or Y (indicating seconds, minutes, hours, days, weeks, months, or years respectively), or a series of such pairs. In this case the string refers to the time that preceded the current time by the length of the interval. For instance, "1h78m" indicates the time that was one hour and 78 minutes ago. The

calendar here is unsophisticated: a month is always 30 days, a year is always 365 days, and a day is always 86400 seconds.

5. A date format of the form YYYY/MM/DD, YYYY-MM-DD, MM/DD/YYYY, or MM-DD-YYYY, which indicates midnight on the day in question, relative to the current time zone settings. For instance, "2002/3/5", "03-05-2002", and "2002-3-05" all mean March 5th, 2002.

## FILE SELECTION

duplicity accepts the same file selection options **rdiff-backup** does, including `--exclude`, `--exclude-filelist`, `--exclude-filelist-stdin`, etc.

When duplicity is run, it searches through the given source directory and backs up all the files specified by the file selection system. The file selection system comprises a number of file selection conditions, which are set using one of the following command line options:

```
--exclude
--exclude-device-files
--exclude-filelist
--exclude-filelist-stdin
--exclude-globbing-filelist
--exclude-regexp
--include
--include-filelist
--include-filelist-stdin
--include-globbing-filelist
--include-regexp
```

Each file selection condition either matches or doesn't match a given file. A given file is excluded by the file selection system exactly when the first matching file selection condition specifies that the file be excluded; otherwise the file is included.

For instance,

```
duplicity --include /usr --exclude /usr /usr scp://user@host/backup
```

is exactly the same as

```
duplicity /usr scp://user@host/backup
```

because the include and exclude directives match exactly the same files, and the **--include** comes first, giving it precedence. Similarly,

```
duplicity --include /usr/local/bin --exclude /usr/local /usr scp://user@host/backup
```

would backup the `/usr/local/bin` directory (and its contents), but not `/usr/local/doc`.

The **include**, **exclude**, **include-globbing-filelist**, and **exclude-globbing-filelist** options accept some *extended shell globbing patterns*. These patterns can contain `*`, `**`, `?`, and `[...]` (character ranges). As in a normal shell, `*` can be expanded to any string of characters not containing `/`, `?` expands to any character except `/`, and `[...]` expands to a single character of those characters specified (ranges are acceptable). The new special pattern, `**`, expands to any string of characters whether or not it contains `/`. Furthermore, if the pattern starts with "ignorecase:" (case insensitive), then this prefix will be removed and any character in the string can be replaced with an upper- or lowercase version of itself.

Remember that you may need to quote these characters when typing them into a shell, so the shell does not interpret the globbing patterns before duplicity sees them.

The **--exclude** pattern option matches a file if:

1. *pattern* can be expanded into the file's filename, or
2. the file is inside a directory matched by the option.

Conversely, the **--include** pattern matches a file if:

1. *pattern* can be expanded into the file's filename, or
2. the file is inside a directory matched by the option, or
3. the file is a directory which contains a file matched by the option.

For example,

```
--exclude /usr/local
```

matches e.g. `/usr/local`, `/usr/local/lib`, and `/usr/local/lib/netcape`. It is the same as `--exclude /usr/local --exclude '/usr/local/**'`.

On the other hand

```
--include /usr/local
```

specifies that `/usr`, `/usr/local`, `/usr/local/lib`, and `/usr/local/lib/netcape` (but not `/usr/doc`) all be backed up. Thus you don't have to worry about including parent directories to make sure that included subdirectories have somewhere to go.

Finally,

```
--include ignorecase:'/usr/[a-z0-9]foo/**.py'
```

would match a file like `/usr/5fOO/hello/there/world.py`. If it did match anything, it would also match `/usr`. If there is no existing file that the given pattern can be expanded into, the option will not match `/usr` alone.

The **--include-filelist**, **--exclude-filelist**, **--include-filelist-stdin**, and **--exclude-filelist-stdin** options also introduce file selection conditions. They direct duplicity to read in a file, each line of which is a file specification, and to include or exclude the matching files. Lines are separated by newlines or nulls, depending on whether the `--null-separator` switch was given. Each line in a filelist is interpreted similarly to the way *extended shell patterns* are, with a few exceptions:

1. Globbing patterns like `*`, `**`, `?`, and `[...]` are not expanded.
2. Include patterns do not match files in a directory that is included. So `/usr/local` in an include file will not match `/usr/local/doc`.
3. Lines starting with `" + "` are interpreted as include directives, even if found in a filelist referenced by **--exclude-filelist**. Similarly, lines starting with `" - "` exclude files even if they are found within an include filelist.

For example, if file `"list.txt"` contains the lines:

```
/usr/local
- /usr/local/doc
/usr/local/bin
+ /var
- /var
```

then **--include-filelist list.txt** would include `/usr`, `/usr/local`, and `/usr/local/bin`. It would exclude `/usr/local/doc`, `/usr/local/doc/python`, etc. It neither excludes nor includes `/usr/local/man`, leaving the fate of this directory to the next specification condition. Finally, it is undefined what happens with `/var`. A single file list should not contain conflicting file specifications.

The **--include-globbing-filelist** and **--exclude-globbing-filelist** options also specify filelists, but each line in the filelist will be interpreted as a globbing pattern the way **--include** and **--exclude** options are interpreted (although `" + "` and `" - "` prefixing is still allowed). For instance, if the file `"globbing-list.txt"` contains the lines:

```
dir/foo
+ dir/bar
- **
```

Then **--include-globbing-filelist globbing-list.txt** would be exactly the same as specifying **--include dir/foo --include dir/bar --exclude \*\*** on the command line.

Finally, the **--include-regexp** and **--exclude-regexp** options allow files to be included and excluded if their filenames match a python regular expression. Regular expression syntax is too complicated to explain here, but is covered in Python's library reference. Unlike the **--include** and **--exclude** options, the regular expression options don't match files containing or contained in matched files. So for instance

```
--include '[0-9]{7}(?!foo)'
```

matches any files whose full pathnames contain 7 consecutive digits which aren't followed by 'foo'. However, it wouldn't match /home even if /home/ben/1234567 existed.

### A NOTE ON CLOUD FILES ACCESS

Pyrax is Rackspace's next-generation Cloud management API, including Cloud Files access. The cfpyrax backend requires the pyrax library to be installed on the system. See **REQUIREMENTS** above.

Cloudfiles is Rackspace's now deprecated implementation of OpenStack Object Storage protocol. Users wishing to use Duplicity with Rackspace Cloud Files should migrate to the new Pyrax plugin to ensure support.

The backend requires python-cloudfiles to be installed on the system. See **REQUIREMENTS** above.

It uses three environment variables for authentication: **CLOUDFILES\_USERNAME** (required), **CLOUDFILES\_APIKEY** (required), **CLOUDFILES\_AUTHURL** (optional)

If **CLOUDFILES\_AUTHURL** is unspecified it will default to the value provided by python-cloudfiles, which points to rackspace, hence this value *must* be set in order to use other cloud files providers.

### A NOTE ON DROPBOX ACCESS

1. "some\_dir" must already exist in the Dropbox Application folder for this application, like "Apps/Duplicity/some\_dir".
2. The first run of the backend must be ininteractive! It will print the URL that you need to open in the browser to obtain OAuth token for the application. The token will be saved in the file \$HOME/.dropbox.token\_store.txt and used in the future runs.
3. When using Dropbox for storage, be aware that all files, including the ones in the Apps folder, will be synced to all connected computers. You may prefer to use a separate Dropbox account specially for the backups, and not connect any computers to that account.

### A NOTE ON EUROPEAN S3 BUCKETS

Amazon S3 provides the ability to choose the location of a bucket upon its creation. The purpose is to enable the user to choose a location which is better located network topologically relative to the user, because it may allow for faster data transfers.

duplicity will create a new bucket the first time a bucket access is attempted. At this point, the bucket will be created in Europe if **--s3-european-buckets** was given. For reasons having to do with how the Amazon S3 service works, this also requires the use of the **--s3-use-new-style** option. This option turns on subdomain based bucket addressing in S3. The details are beyond the scope of this man page, but it is important to know that your bucket must not contain upper case letters or any other characters that are not valid parts of a hostname. Consequently, for reasons of backwards compatibility, use of subdomain based



bucket addressing is not enabled by default.

Note that you will need to use **--s3-use-new-style** for all operations on European buckets; not just upon initial creation.

You only need to use **--s3-european-buckets** upon initial creation, but you may use it at all times for consistency.

Further note that when creating a new European bucket, it can take a while before the bucket is fully accessible. At the time of this writing it is unclear to what extent this is an expected feature of Amazon S3, but in practice you may experience timeouts, socket errors or HTTP errors when trying to upload files to your newly created bucket. Give it a few minutes and the bucket should function normally.

## A NOTE ON GOOGLE CLOUD STORAGE

Support for Google Cloud Storage relies on its Interoperable Access, which must be enabled for your account. Once enabled, you can generate Interoperable Storage Access Keys and pass them to duplicity via the **GS\_ACCESS\_KEY\_ID** and **GS\_SECRET\_ACCESS\_KEY** environment variables. Alternatively, you can run **gsutil config -a** to have the Google Cloud Storage utility populate the **~/.boto** configuration file.

Enable Interoperable Access: <https://code.google.com/apis/console#:storage>

Create Access Keys: <https://code.google.com/apis/console#:storage:legacy>

## A NOTE ON IMAP

An IMAP account can be used as a target for the upload. The userid may be specified and the password will be requested.

The **from\_address\_prefix** may be specified (and probably should be). The text will be used as the "From" address in the IMAP server. Then on a restore (or list) command the **from\_address\_prefix** will distinguish between different backups.

## A NOTE ON PAR2 WRAPPER BACKEND

Par2 Wrapper Backend can be used in combination with all other backends to create recovery files. Just add **par2+** before a regular scheme (e.g. *par2+ftp://user@host/dir* or *par2+s3+http://bucket\_name*). This will create par2 recovery files for each archive and upload them all to the wrapped backend.

Before restoring, archives will be verified. Corrupt archives will be repaired on the fly if there are enough recovery blocks available.

Use **--par2-redundancy percent** to adjust the size (and redundancy) of recovery files in *percent*.

## A NOTE ON SSH BACKENDS

The *ssh backends* support *sftp* and *scp/ssh* transport protocols. This is a known user-confusing issue as these are fundamentally different. If you plan to access your backend via one of those please inform yourself about the requirements for a server to support *sftp* or *scp/ssh* access. To make it even more confusing the user can choose between two ssh backends via **--ssh-backend** option.

Both support **--use-scp**, **--ssh-askpass** and **--ssh-options**. Only the **pexpect** backend allows to define **--scp-command** and **--sftp-command**.

**SSH paramiko backend** (selected by default) is a complete reimplement of ssh protocols natively in python. Advantages are speed and maintainability. Minor disadvantage is that extra packages are needed as listed in **REQUIREMENTS** above. In *sftp* (default) mode all operations are done via the according sftp commands. In *scp* mode (*--use-scp*) though scp access is used for put/get operations but listing is done via ssh remote shell.

**SSH pexpect backend** is the legacy ssh backend using the command line ssh binaries via pexpect. Older versions used *scp* for get and put operations and *sftp* for list and delete operations. The current version uses *sftp* for all four supported operations, unless the *--use-scp* option is used to revert to old behavior.

**Why use sftp instead of scp?** The change to sftp was made in order to allow the remote system to chroot

the backup, thus providing better security and because it does not suffer from shell quoting issues like scp. Scp also does not support any kind of file listing, so sftp or ssh access will always be needed in addition for this backend mode to work properly. Sftp does not have these limitations but needs an sftp service running on the backend server, which is sometimes not an option.

### A NOTE ON SSL CERTIFICATE VERIFICATION

Certificate verification as implemented right now [01.2013] only in the webdav backend needs a file based database of certification authority certificates (cacert file). It has to be a **PEM** formatted text file as currently provided by the **CURL** project. See

<http://curl.haxx.se/docs/caextract.html>

After creating/retrieving a valid cacert file you should copy it to either

~/duplicity/cacert.pem

~/duplicity\_cacert.pem

/etc/duplicity/cacert.pem

Duplicity searches it there in the same order and will fail if it can't find it. You can however specify the option **--ssl-cacert-file** *<file>* to point duplicity to a copy in a different location.

Finally there is the **--ssl-no-check-certificate** option to disable certificate verification altogether, in case some ssl library is missing or verification is not wanted. Use it with care, as even with self signed servers manually providing the private ca certificate is definitely the safer option.

### A NOTE ON SWIFT (OPENSTACK OBJECT STORAGE) ACCESS

Swift is the OpenStack Object Storage service.

The backend requires python-swiftclient to be installed on the system. python-keystoneclient is also needed to use OpenStack's Keystone Identity service. See **REQUIREMENTS** above.

It uses four environment variables for authentication: **SWIFT\_USERNAME** (required), **SWIFT\_PASSWORD** (required), **SWIFT\_AUTHURL** (required), **SWIFT\_TENANTNAME** (optional, the tenant can be included in the username)

If the user was previously authenticated, the following environment variables can be used instead: **SWIFT\_PREAUTHURL** (required), **SWIFT\_PREAUTHTOKEN** (required)

If **SWIFT\_AUTHVERSION** is unspecified, it will default to version 1.

### A NOTE ON SYMMETRIC ENCRYPTION AND SIGNING

Signing and symmetrically encrypt at the same time with the gpg binary on the command line, as used within duplicity, is a specifically challenging issue. Tests showed that the following combinations proved working.

1. Setup gpg-agent properly. Use the option **--use-agent** and enter both passphrases (symmetric and sign key) in the gpg-agent's dialog.
2. Use a **PASSPHRASE** for symmetric encryption of your choice but the signing key has an **empty** passphrase.
3. The used **PASSPHRASE** for symmetric encryption and the passphrase of the signing key are identical.

### A NOTE ON FILENAME PREFIXES

Filename prefixes can be used in conjunction with S3 lifecycle rules to transition archive files to Glacier, while keeping metadata (signature and manifest files) on S3.

Duplicity does not require access to archive files except when restoring from backup.

## KNOWN ISSUES / BUGS

Hard links currently unsupported (they will be treated as non-linked regular files).

Bad signatures will be treated as empty instead of logging appropriate error message.

## OPERATION AND DATA FORMATS

This section describes duplicity's basic operation and the format of its data files. It should not necessary to read this section to use duplicity.

The files used by duplicity to store backup data are tarfiles in GNU tar format. They can be produced independently by **rdiffdir**(1). For incremental backups, new files are saved normally in the tarfile. But when a file changes, instead of storing a complete copy of the file, only a diff is stored, as generated by **rdiff**(1). If a file is deleted, a 0 length file is stored in the tar. It is possible to restore a duplicity archive "manually" by using **tar** and then **cp**, **rdiff**, and **rm** as necessary. These duplicity archives have the extension **difftar**.

Both full and incremental backup sets have the same format. In effect, a full backup set is an incremental one generated from an empty signature (see below). The files in full backup sets will start with **duplicity-full** while the incremental sets start with **duplicity-inc**. When restoring, duplicity applies patches in order, so deleting, for instance, a full backup set may make related incremental backup sets unusable.

In order to determine which files have been deleted, and to calculate diffs for changed files, duplicity needs to process information about previous sessions. It stores this information in the form of tarfiles where each entry's data contains the signature (as produced by **rdiff**) of the file instead of the file's contents. These signature sets have the extension **sigtar**.

Signature files are not required to restore a backup set, but without an up-to-date signature, duplicity cannot append an incremental backup to an existing archive.

To save bandwidth, duplicity generates full signature sets and incremental signature sets. A full signature set is generated for each full backup, and an incremental one for each incremental backup. These start with **duplicity-full-signatures** and **duplicity-new-signatures** respectively. These signatures will be stored both locally and remotely. The remote signatures will be encrypted if encryption is enabled. The local signatures will not be encrypted and stored in the archive dir (see **--archive-dir**).

## AUTHOR

**Original Author** - Ben Escoto <bescoto@stanford.edu>

**Current Maintainer** - Kenneth Loafman <kenneth@loafman.com>

### Continuous Contributors

Edgar Soldin, Mike Terry

Most backends were contributed individually. Information about their authorship may be found in the according file's header.

Also we'd like to thank everybody posting issue to the mailing list or on launchpad, sending in patches or contributing otherwise. Duplicity wouldn't be as stable and useful if it weren't for you.

## SEE ALSO

**rdiffdir**(1), **python**(1), **rdiff**(1), **rdiff-backup**(1).