NAME

uscan − scan/watch upstream sources for new releases of software

SYNOPSIS

uscan [*options*] [*path*]

DESCRIPTION

For basic usage, **uscan** is executed without any arguments from the root of the Debianized source tree where you see the *debian/* directory. Then typically the following happens:

- **uscan** reads the first entry in *debian/changelog* to determine the source package name *<spkg>* and the last upstream version.
- **uscan** process the watch lines *debian/watch* from the top to the bottom in a single pass.
  - **uscan** downloads a web page from the specified *URL* in *debian/watch*.
  - **uscan** extracts hrefs pointing to the upstream tarball(s) from the web page using the specified *matching-pattern* in *debian/watch*.
  - **uscan** downloads the upstream tarball with the highest version newer than the last upstream version.
  - **uscan** saves the downloaded tarball to the parent **../** directory: *../<upkg>−<uversion>.tar.gz*
  - **uscan** invokes **mk-origtargz** to create the source tarball: *../<spkg>_<oversion>.orig.tar.gz*
    - For a multiple upstream tarball (MUT) package, the secondary upstream tarball will instead be named *../<spkg>_<oversion>.orig−<component>.tar.gz*.
  - Repeat until all lines in *debian/watch* are processed.
- **uscan** invokes **uupdate** to create the Debianized source tree: *../<spkg>−<oversion>/**

Please note the following.

- For simplicity, the compression method used in examples is **gzip** with **.gz** suffix. Other methods such as **xz**, **bzip2**, and **lzma** with corresponding **xz**, **bz2**, and **lzma** suffixes may also be used.
- The new **version=4** enables handling of multiple upstream tarball (MUT) packages but this is a rare case for Debian packaging. For a single upstream tarball package, there is only one watch line and no *../<spkg>_<oversion>.orig−<component>.tar.gz* .
- **uscan** with the **−−verbose** option produces a human readable report of **uscan**'s execution.
- **uscan** with the **−−debug** option produces a human readable report of **uscan**'s execution including internal variable states.
- **uscan** with the **−−dehs** option produces an upstream package status report in XML format for other programs such as the Debian External Health System.
- The primary objective of **uscan** is to help identify if the latest version upstream tarball is used or not; and to download the latest upstream tarball. The ordering of versions is decided by **dpkg −−compare−versions**.
- **uscan** with the **−−safe** option limits the functionality of **uscan** to its primary objective. Both the repacking of downloaded files and updating of the source tree are skipped to avoid running unsafe scripts. This also changes the default to **−−no−download** and **−−skip−signature**.

FORMAT OF THE WATCH FILE

The current version 4 format of *debian/watch* can be summarized as follows:

- Leading spaces and tabs are dropped.
- Empty lines are dropped.
- A line started by **#** (hash) is a comment line and dropped.

- A single **** (back slash) at the end of a line is dropped and the next line is concatenated after removing leading spaces and tabs. The concatenated line is parsed as a single line. (The existence or non-existence of the space before the tailing single **** is significant.)
- The first non-comment line is:

  version=4

  This is a required line and the recommended version number.

  If you use "**version=3**" instead here, some features may not work as documented here. See ‘‘HISTORY AND UPGRADING’’.
- The following non-comment lines (watch lines) specify the rules for the selection of the candidate upstream tarball URLs and are in one of the following three formats:
  - **opts="** ... **" http://***URL matching-pattern* [*version* [*script*]]
  - **http://***URL matching-pattern* [*version* [*script*]]
  - **opts="** ... **"**

  Here,
  - **opts="** ... **"** specifies the behavior of **uscan**. See ‘‘WATCH FILE OPTIONS’’.
  - **http://***URL* specifies the web page where upstream publishes the link to the latest source archive.
    - **https://***URL* may also be used, as may
    - **ftp://***URL*
    - Some parts of *URL* may be in the regex match pattern surrounded between **(** and **)** such as **/foo/bar−([\.\d]+)/**. (If multiple directories match, the highest version is picked.) Otherwise, the *URL* is taken as verbatim.
  - *matching-pattern* specifies the full string matching pattern for hrefs in the web page. See ‘‘WATCH FILE EXAMPLES’’.
    - All matching parts in **(** and **)** are concatenated with **.** (period) to form the upstream version.
    - If the hrefs do not contain directories, you can combine this with the previous entry. I.e., **http://***URL/matching-pattern* .
  - *version* restricts the upstream tarball which may be downloaded. The newest available version is chosen in each case.
    - **debian** *(default)* requires the downloading upstream tarball to be newer than the version obtained from *debian/changelog*.
    - *version-number* such as **12.5** requires the upstream tarball to be newer than the *version-number*.
    - **same** requires the downloaded version of the secondary tarballs to be exactly the same as the one for the first upstream tarball downloaded. (Useful only for MUT)
    - **previous** restricts the version of the signature file. (Used with pgpmode=previous)
    - **ignore** does not restrict the version of the secondary tarballs. (Maybe useful for MUT)
    - **group** requires the downloading upstream tarball to be newer than the version obtained from *debian/changelog*. Package version is the concatenation of all ‘‘group’’ upstream version.
  - *script* is executed at the end of **uscan** execution with appropriate arguments provided by **uscan** *(default: no action)*.
    - The typical Debian package is a non-native package made from one upstream tarball. Only a single line of the watch line in one of the first two formats is usually used with its *version* set to **debian** and *script* set to **uupdate**.

- A native package should not specify *script*.
- A multiple upstream tarball (MUT) package should specify **uupdate** as *script* in the last watch line and should skip specifying *script* in the rest of the watch lines.

- The last format of the watch line is useful to set the persistent parameters: **user-agent**, **compression**. If this format is used, this must be followed by the *URL* defining watch line(s).
- [and] in the above format are there to mark the optional parts and should not be typed.

There are a few special strings which are substituted by **uscan** to make it easy to write the watch file.

@PACKAGE@
This is substituted with the source package name found in the first line of the *debian/changelog* file.

@ANY_VERSION@
This is substituted by the legal upstream version regex (capturing).

```
[-_]?(\d[\-+\.:\~\da-zA-Z]*)
```

@ARCHIVE_EXT@
This is substituted by the typical archive file extension regex (non-capturing).

```
(?i)\.(?:tar\.xz|tar\.bz2|tar\.gz|zip|tgz|tbz|txz)
```

@SIGNATURE_EXT@
This is substituted by the typical signature file extension regex (non-capturing).

```
(?i)\.(?:tar\.xz|tar\.bz2|tar\.gz|zip|tgz|tbz|txz)\.(?:asc|pgp|gpg|sig|sign)
```

@DEB_EXT@
This is substituted by the typical Debian extension regexp (capturing).

```
[\+~](debian|dfsg|ds|deb)(\.)?(\d+)?$
```

Some file extensions are not included in the above intentionally to avoid false positives. You can still set such file extension patterns manually.

WATCH FILE OPTIONS

uscan reads the watch options specified in **opts="** ... **"** to customize its behavior. Multiple options *option1*, *option2*, *option3*, ... can be set as **opts="***option1***,** *option2***,** *option3***,** ... **"** . The double quotes are necessary if options contain any spaces.

Unless otherwise noted as persistent, most options are valid only within their containing watch line.

The available watch options are:

component=*component*
Set the name of the secondary source tarball as *<spkg>_<oversion>.orig-<component>.tar.gz* for a MUT package.

compression=*method*
Set the compression *method* when the tarball is repacked (persistent).

Available *method* values are what mk-origtargz supports, so **xz**, **gzip** (alias **gz**), **bzip2** (alias **bz2**), **lzma**, **default**. The default method is currently **xz**. When uscan is launched in a debian source repository which format is "1.0" or undefined, the method switches to **gzip**.

Please note the repacking of the upstream tarballs by **mk-origtargz** happens only if one of the following conditions is satisfied:

- **USCAN_REPACK** is set in the devscript configuration. See "DEVSCRIPT CONFIGURATION VARIABLES".
- **--repack** is set on the commandline. See <COMMANDLINE OPTIONS>.
- **repack** is set in the watch line as **opts="repack,..."**.

- The upstream archive is of **zip** type including **jar**, **xpi**, ...
- **Files-Excluded** or **Files−Excluded−***component* stanzas are set in *debian/copyright* to make **mk-origtargz** invoked from **uscan** remove files from the upstream tarball and repack it. See ''COPYRIGHT FILE EXAMPLES'' and **mk−origtargz** (1).

repack
Force repacking of the upstream tarball using the compression *method*.

repacksuffix=*suffix*
Add *suffix* to the Debian package upstream version only when the source tarball is repackaged. This rule should be used only for a single upstream tarball package.

mode=*mode*
Set the archive download *mode*.

LWP
This mode is the default one which downloads the specified tarball from the archive URL on the web. Automatically internal **mode** value is updated to either **http** or **ftp** by URL.

git This mode accesses the upstream git archive directly with the **git** command and packs the source tree with the specified tag via *matching-pattern* into *spkg-version***.tar.xz**.

If the upstream publishes the released tarball via its web interface, please use it instead of using this mode. This mode is the last resort method.

For git mode, *matching-pattern* specifies the full string matching pattern for tags instead of hrefs. If *matching-pattern* is set to **refs/tags/***tag-matching-pattern*, **uscan** downloads source from the **refs/tags/***matched-tag* of the git repository. The upstream version is extracted from concatenating the matched parts in **(** ... **)** with **.** . See ''WATCH FILE EXAMPLES''.

If *matching-pattern* is set to **HEAD**, **uscan** downloads source from the **HEAD** of the git repository and the pertinent *version* is automatically generated with the date and hash of the **HEAD** of the git repository.

If *matching-pattern* is set to **heads/***branch*, **uscan** downloads source from the named *branch* of the git repository.

The local repository is temporarily created as a bare git repository directory under the destination directory where the downloaded archive is generated. This is normally erased after the **uscan** execution. This local repository is kept if **−−debug** option is used.

If the current directory is a git repository and the searched repository is listed among the registered ''remotes'', then uscan will use it instead of cloning separately. The only local change is that uscan will run a ''fetch'' command to refresh the repository.

pretty=*rule*
Set the upstream version string to an arbitrary format as an optional **opts** argument when the *matching-pattern* is **HEAD** or **heads/***branch* for **git** mode. For the exact syntax, see the **git-log** manpage under **tformat**. The default is **pretty=0.0~git%cd.%h**. No **uversionmangle** rules is applicable for this case.

When **pretty=describe** is used, the upstream version string is the output of the "**git describe −−tags | sed s/−/./g**" command instead. For example, if the commit is the **5**−th after the last tag **v2.17.12** and its short hash is **ged992511**, then the string is **v2.17.12.5.ged992511** . For this case, it is good idea to add **uversionmangle=s/^/0.0~/** or **uversionmangle=s/^v//** to make the upstream version string compatible with Debian. **uversionmangle=s/^v//** may work as well. Please note that in order for **pretty=describe** to function well, upstream need to avoid tagging with random alphabetic tags.

The **pretty=describe** forces to set **gitmode=full** to make a full local clone of the repository automatically.

date=*rule*
Set the date string used by the **pretty** option to an arbitrary format as an optional **opts** argument when the *matching-pattern* is **HEAD** or **heads/***branch* for **git** mode. For the exact syntax, see the **strftime** manpage. The default is **date=%Y%m%d**.

gitmode=*mode*
Set the git clone operation *mode*. The default is **gitmode=shallow**. For some dumb git server, you may need to manually set **gitmode=full** to force full clone operation.

If the current directory is a git repository and the searched repository is listed among the registered ''remotes'', then uscan will use it instead of cloning separately.

pgpmode=*mode*
Set the PGP/GPG signature verification *mode*.

auto
uscan checks possible URLs for the signature file and autogenerates a **pgpsigurlmangle** rule to use it.

default
Use **pgpsigurlmangle=***rules* to generate the candidate upstream signature file URL string from the upstream tarball URL. (default)

If the specified **pgpsigurlmangle** is missing, **uscan** checks possible URLs for the signature file and suggests adding a **pgpsigurlmangle** rule.

mangle
Use **pgpsigurlmangle=***rules* to generate the candidate upstream signature file URL string from the upstream tarball URL.

next
Verify this downloaded tarball file with the signature file specified in the next watch line. The next watch line must be **pgpmode=previous**. Otherwise, no verification occurs.

previous
Verify the downloaded tarball file specified in the previous watch line with this signature file. The previous watch line must be **pgpmode=next**.

self Verify the downloaded file *foo.ext* with its self signature and extract its content tarball file as *foo*.

gittag
Verify tag signature if **mode=git**.

none
No signature available. (No warning.)

searchmode=*mode*
Set the parsing search mode.

html *(default)*: search pattern in ''href'' parameter of <a> HTML tags
plain: search pattern in the full page
This is useful is page content is not HTML but in JSON. Example with npmjs.com:

```
version=4
opts="searchmode=plain" \
 https://registry.npmjs.org/aes-js \
 https://registry.npmjs.org/aes-js/-/aes-js-(\d[\d\.]*)@ARCHIVE_EXT@
```

decompress
Decompress compressed archive before the pgp/gpg signature verification.

bare
Disable all site specific special case code such as URL redirector uses and page content alterations. (persistent)

user−agent=*user-agent-string*
Set the user-agent string used to contact the HTTP(S) server as *user-agent-string*. (persistent)

user-agent option should be specified by itself in the watch line without *URL*, to allow using semicolons and commas in it.

pasv, **passive**
Use PASV mode for the FTP connection.

If PASV mode is required due to the client side network environment, set **uscan** to use PASV mode via ‘‘COMMANDLINE OPTIONS’’ or ‘‘DEVSCRIPT CONFIGURATION VARIABLES’’ instead.

active, **nopasv**
Don’t use PASV mode for the FTP connection.

unzipopt=*options*
Add the extra options to use with the **unzip** command, such as **−a**, **−aa**, and **−b**, when executed by **mk-origtargz**.

dversionmangle=*rules*
Normalize the last upstream version string found in *debian/changelog* to compare it to the available upstream tarball version. Removal of the Debian specific suffix such as **s/@DEB_EXT@//** is usually done here.

You can also use **dversionmangle=auto**, this is exactly the same than **dversionmangle=s/@DEB_EXT@//**

dirversionmangle=*rules*
Normalize the directory path string matching the regex in a set of parentheses of **http://***URL* as the sortable version index string. This is used as the directory path sorting index only.

Substitution such as **s/PRE/˜pre/; s/RC/˜rc/** may help.

pagemangle=*rules*
Normalize the downloaded web page string. (Don’t use this unless this is absolutely needed. Generally, **g** flag is required for these *rules*.)

This is handy if you wish to access Amazon AWS or Subversion repositories in which <a href=‘‘...’’> is not used.

uversionmangle=*rules*
Normalize the candidate upstream version strings extracted from hrefs in the source of the web page. This is used as the version sorting index when selecting the latest upstream version.

Substitution such as **s/PRE/˜pre/; s/RC/˜rc/** may help.

versionmangle=*rules*
Syntactic shorthand for **uversionmangle=***rules***, dversionmangle=***rules*

hrefdecode=percent−encoding
Convert the selected upstream tarball href string from the percent-encoded hexadecimal string to the decoded normal URL string for obfuscated web sites. Only **percent-encoding** is available and it is decoded with **s/%([A−Fa−f\d]{2})/chr hex $1/eg**.

downloadurlmangle=*rules*
Convert the selected upstream tarball href string into the accessible URL for obfuscated web sites. This is run after **hrefdecode**.

filenamemangle=*rules*
Generate the upstream tarball filename from the selected href string if *matching-pattern* can extract the latest upstream version <*uversion*> from the selected href string. Otherwise, generate the upstream tarball filename from its full URL string and set the missing <*uversion*> from the generated upstream tarball filename.

Without this option, the default upstream tarball filename is generated by taking the last component of

the URL and removing everything after any '?' or '#'.

pgpsigurlmangle=*rules*
Generate the candidate upstream signature file URL string from the upstream tarball URL.

oversionmangle=*rules*
Generate the version string *<oversion>* of the source tarball *<spkg>_<oversion>.orig.tar.gz* from *<uversion>*. This should be used to add a suffix such as **+dfsg1** to a MUT package.

Here, the mangling rules apply the *rules* to the pertinent string. Multiple rules can be specified in a mangling rule string by making a concatenated string of each mangling *rule* separated by **;** (semicolon).

Each mangling *rule* cannot contain **;** (semicolon), **,** (comma), or **"** (double quote).

Each mangling *rule* behaves as if a Perl command "*$string* **=~** *rule*" is executed. There are some notable details.

- *rule* may only use the **s**, **tr**, and **y** operations.

  s/*regex***/***replacement***/***options*
  Regex pattern match and replace the target string. Only the **g**, **i** and **x** flags are available. Use the **$1** syntax for back references (No **\1** syntax). Code execution is not allowed (i.e. no **(?{})** or **(??{})** constructs).

  y/*source***/***dest***/** or **tr/***source***/***dest***/**
  Transliterate the characters in the target string.

EXAMPLE OF EXECUTION

uscan reads the first entry in *debian/changelog* to determine the source package name and the last upstream version.

For example, if the first entry of *debian/changelog* is:

- *bar* (**3:2.03+dfsg1−4**) unstable; urgency=low

then, the source package name is *bar* and the last Debian package version is **3:2.03+dfsg1−4**.

The last upstream version is normalized to **2.03+dfsg1** by removing the epoch and the Debian revision.

If the **dversionmangle** rule exists, the last upstream version is further normalized by applying this rule to it. For example, if the last upstream version is **2.03+dfsg1** indicating the source tarball is repackaged, the suffix **+dfsg1** is removed by the string substitution **s/\+dfsg\d*$//** to make the (dversionmangled) last upstream version **2.03** and it is compared to the candidate upstream tarball versions such as **2.03**, **2.04**, ... found in the remote site. Thus, set this rule as:

- **opts="dversionmangle=s/\+dfsg\d*$//"**

uscan downloads a web page from **http://***URL* specified in *debian/watch*.

- If the directory name part of *URL* has no parentheses, **(** and **)**, it is taken as verbatim.
- If the directory name part of *URL* has parentheses, **(** and **)**, then **uscan** recursively searches all possible directories to find a page for the newest version. If the **dirversionmangle** rule exists, the generated sorting index is used to find the newest version. If a specific version is specified for the download, the matching version string has priority over the newest version.

For example, this **http://***URL* may be specified as:

- **http://www.example.org/([\d\.]+)/**

Please note the trailing **/** in the above to make **([\d\.]+)** as the directory.

If the **pagemangle** rule exists, the whole downloaded web page as a string is normalized by applying this rule to it. This is very powerful tool and needs to be used with caution. If other mangling rules can be used to address your objective, do not use this rule.

The downloaded web page is scanned for hrefs defined in the **<a href="** ... **">** tag to locate the candidate upstream tarball hrefs. These candidate upstream tarball hrefs are matched by the Perl regex pattern

matching-pattern such as **DL−(?:[\d\.]+?)/foo−(.+)\.tar\.gz** to narrow down the candidates. This pattern match needs to be anchored at the beginning and the end. For example, candidate hrefs may be:

- **DL−2.02/foo−2.02.tar.gz**
- **DL−2.03/foo−2.03.tar.gz**
- **DL−2.04/foo−2.04.tar.gz**

Here the matching string of **(.+)** in *matching-pattern* is considered as the candidate upstream version. If there are multiple matching strings of capturing patterns in *matching-pattern*, they are all concatenated with **.** (period) to form the candidate upstream version. Make sure to use the non-capturing regex such as **(?:[\d\.]+?)** instead for the variable text matching part unrelated to the version.

Then, the candidate upstream versions are:

- **2.02**
- **2.03**
- **2.04**

The downloaded tarball filename is basically set to the same as the filename in the remote URL of the selected href.

If the **uversionmangle** rule exists, the candidate upstream versions are normalized by applying this rule to them. (This rule may be useful if the upstream version scheme doesn't sort correctly to identify the newest version.)

The upstream tarball href corresponding to the newest (uversionmangled) candidate upstream version newer than the (dversionmangled) last upstream version is selected.

If multiple upstream tarball hrefs corresponding to a single version with different extensions exist, the highest compression one is chosen. (Priority: **tar.xz > tar.lzma > tar.bz2 > tar.gz**.)

If the selected upstream tarball href is the relative URL, it is converted to the absolute URL using the base URL of the web page. If the **<base href="** ... **">** tag exists in the web page, the selected upstream tarball href is converted to the absolute URL using the specified base URL in the base tag, instead.

If the **downloadurlmangle** rule exists, the selected upstream tarball href is normalized by applying this rule to it. (This is useful for some sites with the obfuscated download URL.)

If the **filenamemangle** rule exists, the downloaded tarball filename is generated by applying this rule to the selected href if *matching-pattern* can extract the latest upstream version *<uversion>* from the selected href string. Otherwise, generate the upstream tarball filename from its full URL string and set the missing *<uversion>* from the generated upstream tarball filename.

Without the **filenamemangle** rule, the default upstream tarball filename is generated by taking the last component of the URL and removing everything after any '?' or '#'.

uscan downloads the selected upstream tarball to the parent **../** directory. For example, the downloaded file may be:

- *../foo−2.04.tar.gz*

Let's call this downloaded version **2.04** in the above example generically as *<uversion>* in the following.

If the **pgpsigurlmangle** rule exists, the upstream signature file URL is generated by applying this rule to the (downloadurlmangled) selected upstream tarball href and the signature file is tried to be downloaded from it.

If the **pgpsigurlmangle** rule doesn't exist, **uscan** warns user if the matching upstream signature file is available from the same URL with their filename being suffixed by the 5 common suffix **asc**, **gpg**, **pgp**, **sig** and **sign**. (You can avoid this warning by setting **pgpmode=none**.)

If the signature file is downloaded, the downloaded upstream tarball is checked for its authenticity against the downloaded signature file using the armored keyring *debian/upstream/signing−key.asc* (see "KEYRING FILE EXAMPLES"). If its signature is not valid, or not made by one of the listed keys, **uscan** will report an

error.

If the **oversionmangle** rule exists, the source tarball version *oversion* is generated from the downloaded upstream version *uversion* by applying this rule. This rule is useful to add suffix such as **+dfsg1** to the version of all the source packages of the MUT package for which the repacksuffix mechanism doesn't work.

uscan invokes **mk-origtargz** to create the source tarball properly named for the source package with **.orig.** (or **.orig−<component>.** for the secondary tarballs) in its filename.

case A: packaging of the upstream tarball as is
: **mk-origtargz** creates a symlink *../bar_<oversion>.orig.tar.gz* linked to the downloaded local upstream tarball. Here, *bar* is the source package name found in *debian/changelog*. The generated symlink may be:

  - *../bar_2.04.orig.tar.gz −> foo−2.04.tar.gz* (as is)

  Usually, there is no need to set up **opts="dversionmangle=** ... **"** for this case.

case B: packaging of the upstream tarball after removing non-DFSG files
: **mk-origtargz** checks the filename glob of the **Files-Excluded** stanza in the first section of *debian/copyright*, removes matching files to create a repacked upstream tarball. Normally, the repacked upstream tarball is renamed with *suffix* to *../bar_<oversion><suffix>.orig.tar.gz* using the **repacksuffix** option for the single upstream package. Here *<oversion>* is updated to be *<oversion><suffix>*.

  The removal of files is required if files are not DFSG-compliant. For such case, **+dfsg1** is used as *suffix*.

  So the combined options are set as **opts="dversionmangle=s/\+dfsg\d*$// ,repacksuffix=+dfsg1"**, instead.

  For example, the repacked upstream tarball may be:

  - *../bar_2.04+dfsg1.orig.tar.gz* (repackaged)

uscan normally invokes "**uupdate −−find −−upstream−version** *oversion* " for the version=4 watch file.

Please note that **−−find** option is used here since **mk-origtargz** has been invoked to make ***.orig.tar.gz** file already. **uscan** picks *bar* from *debian/changelog*.

It creates the new upstream source tree under the *../bar−<oversion>* directory and Debianize it leveraging the last package contents.

WATCH FILE EXAMPLES

When writing the watch file, you should rely on the latest upstream source announcement web page. You should not try to second guess the upstream archive structure if possible. Here are the typical *debian/watch* files.

Please note that executing **uscan** with **−v** or **−vv** reveals what exactly happens internally.

The existence and non-existence of a space the before tailing **** (back slash) are significant.

Some undocumented shorter configuration strings are used in the below EXAMPLES to help you with typing. These are intentional ones. **uscan** is written to accept such common sense abbreviations but don't push the limit.

HTTP site (basic)

Here is an example for the basic single upstream tarball.

```
version=4
http://example.com/~user/release/foo.html \
    files/foo-([\d\.]+)\.tar\.gz debian uupdate
```

Or using the special strings:

```
version=4
http://example.com/~user/release/@PACKAGE@.html \
    files/@PACKAGE@@ANY_VERSION@@ARCHIVE_EXT@ debian uupdate
```

For the upstream source package **foo−2.0.tar.gz**, this watch file downloads and creates the Debian **orig.tar** file **foo_2.0.orig.tar.gz**.

HTTP site (pgpsigurlmangle)

Here is an example for the basic single upstream tarball with the matching signature file in the same file path.

```
version=4
opts="pgpsigurlmangle=s%$%.asc%" http://example.com/release/@PACKAGE@.html \
    files/@PACKAGE@@ANY_VERSION@@ARCHIVE_EXT@ debian uupdate
```

For the upstream source package **foo−2.0.tar.gz** and the upstream signature file **foo−2.0.tar.gz.asc**, this watch file downloads these files, verifies the authenticity using the keyring *debian/upstream/signing−key.asc* and creates the Debian **orig.tar** file **foo_2.0.orig.tar.gz**.

Here is another example for the basic single upstream tarball with the matching signature file on decompressed tarball in the same file path.

```
version=4
opts="pgpsigurlmangle=s%@ARCHIVE_EXT@$%.asc%,decompress" \
    http://example.com/release/@PACKAGE@.html \
    files/@PACKAGE@@ANY_VERSION@@ARCHIVE_EXT@ debian uupdate
```

For the upstream source package **foo−2.0.tar.gz** and the upstream signature file **foo−2.0.tar.asc**, this watch file downloads these files, verifies the authenticity using the keyring *debian/upstream/signing−key.asc* and creates the Debian **orig.tar** file **foo_2.0.orig.tar.gz**.

HTTP site (pgpmode=next/previous)

Here is an example for the basic single upstream tarball with the matching signature file in the unrelated file path.

```
version=4
opts="pgpmode=next" http://example.com/release/@PACKAGE@.html \
    files/(?:\d+)/@PACKAGE@@ANY_VERSION@@ARCHIVE_EXT@ debian
opts="pgpmode=previous" http://example.com/release/@PACKAGE@.html \
    files/(?:\d+)/@PACKAGE@@ANY_VERSION@@SIGNATURE_EXT@ previous uupdate
```

(?:\d+) part can be any random value. The tarball file can have **53**, while the signature file can have **33**.

([\d\.]+) part for the signature file has a strict requirement to match that for the upstream tarball specified in the previous line by having **previous** as *version* in the watch line.

HTTP site (flexible)

Here is an example for the maximum flexibility of upstream tarball and signature file extensions.

```
version=4
opts="pgpmode=next" http://example.com/DL/ \
    files/(?:\d+)/@PACKAGE@@ANY_VERSION@@ARCHIVE_EXT@ debian
opts="pgpmode=previous" http://example.com/DL/ \
    files/(?:\d+)/@PACKAGE@@ANY_VERSION@@SIGNATURE_EXT@ \
    previous uupdate
```

HTTP site (basic MUT)

Here is an example for the basic multiple upstream tarballs.

```
version=4
opts="pgpsigurlmangle=s%$%.sig%" \
    http://example.com/release/foo.html \
    files/foo-([\d\.]+)\.tar\.gz debian
opts="pgpsigurlmangle=s%$%.sig%, component=bar" \
    http://example.com/release/foo.html \
    files/foobar-([\d\.]+)\.tar\.gz same
opts="pgpsigurlmangle=s%$%.sig%, component=baz" \
    http://example.com/release/foo.html \
    files/foobaz-([\d\.]+)\.tar\.gz same uupdate
```

For the main upstream source package **foo−2.0.tar.gz** and the secondary upstream source packages **foobar−2.0.tar.gz** and **foobaz−2.0.tar.gz** which install under *bar/* and *baz/*, this watch file downloads and creates the Debian **orig.tar** file **foo_2.0.orig.tar.gz**, **foo_2.0.orig−bar.tar.gz** and **foo_2.0.orig−baz.tar.gz**. Also, these upstream tarballs are verified by their signature files.

HTTP site (recursive directory scanning)

Here is an example with the recursive directory scanning for the upstream tarball and its signature files released in a directory named after their version.

```
version=4
opts="pgpsigurlmangle=s%$%.sig%, dirversionmangle=s/-PRE/~pre/;s/-RC/~rc/" \
    http://tmrc.mit.edu/mirror/twisted/Twisted/([\d+\.]+)/ \
    Twisted-([\d\.]+)\.tar\.xz debian uupdate
```

Here, the web site should be accessible at the following URL:

```
http://tmrc.mit.edu/mirror/twisted/Twisted/
```

Here, **dirversionmangle** option is used to normalize the sorting order of the directory names.

HTTP site (alternative shorthand)

For the bare HTTP site where you can directly see archive filenames, the normal watch file:

```
version=4
opts="pgpsigurlmangle=s%$%.sig%" \
    http://www.cpan.org/modules/by-module/Text/ \
    Text-CSV_XS-(.+)\.tar\.gz \
    debian uupdate
```

can be rewritten in an alternative shorthand form only with a single string covering URL and filename:

```
version=4
opts="pgpsigurlmangle=s%$%.sig%" \
    http://www.cpan.org/modules/by-module/Text/Text-CSV_XS-(.+)\.tar\.gz \
    debian uupdate
```

In version=4, initial white spaces are dropped. Thus, this alternative shorthand form can also be written as:

```
version=4
opts="pgpsigurlmangle=s%$%.sig%" \
    http://www.cpan.org/modules/by-module/Text/\
    Text-CSV_XS-(.+)\.tar\.gz \
    debian uupdate
```

Please note the subtle difference of a space before the tailing **** between the first and the last examples.

HTTP site (funny version)

For a site which has funny version numbers, the parenthesized groups will be joined with **.** (period) to make a sanitized version number.

```
version=4
http://www.site.com/pub/foobar/foobar_v(\d+)_(\d+)\.tar\.gz \
debian uupdate
```

HTTP site (DFSG)

The upstream part of the Debian version number can be mangled to indicate the source package was repackaged to clean up non-DFSG files:

```
version=4
opts="dversionmangle=s/\+dfsg\d*$//,repacksuffix=+dfsg1" \
http://some.site.org/some/path/foobar-(.+)\.tar\.gz debian uupdate
```

See "COPYRIGHT FILE EXAMPLES".

HTTP site (filenamemangle)

The upstream tarball filename is found by taking the last component of the URL and removing everything after any '?' or '#'.

If this does not fit to you, use **filenamemangle**. For example, *<A href="http://foo.bar.org/dl/?path=&dl=foo-0.1.1.tar.gz">* could be handled as:

```
version=4
opts=filenamemangle=s/.*=(.*)/$1/ \
http://foo.bar.org/dl/\?path=&dl=foo-(.+)\.tar\.gz \
debian uupdate
```

<A href="http://foo.bar.org/dl/?path=&dl_version=0.1.1"> could be handled as:

```
version=4
opts=filenamemangle=s/.*=(.*)/foo-$1\.tar\.gz/ \
http://foo.bar.org/dl/\?path=&dl_version=(.+) \
debian uupdate
```

If the href string has no version using <I>matching-pattern</I>, the version can be obtained from the full URL using **filenamemangle**.

```
version=4
opts=filenamemangle=s&.*/dl/(.*)/foo\.tar\.gz&foo-$1\.tar\.gz& \
http://foo.bar.org/dl/([\.\d]+)/ foo.tar.gz \
debian uupdate
```

HTTP site (downloadurlmangle)

The option **downloadurlmangle** can be used to mangle the URL of the file to download. This can only be used with **http://** URLs. This may be necessary if the link given on the web page needs to be transformed in some way into one which will work automatically, for example:

```
version=4
opts=downloadurlmangle=s/prdownload/download/ \
http://developer.berlios.de/project/showfiles.php?group_id=2051 \
http://prdownload.berlios.de/softdevice/vdr-softdevice-(.+).tgz \
debian uupdate
```

HTTP site (oversionmangle, MUT)

The option **oversionmangle** can be used to mangle the version of the source tarball (**.orig.tar.gz** and **.orig-bar.tar.gz**). For example, **+dfsg1** can be added to the upstream version as:

```
version=4
opts=oversionmangle=s/(.*)/$1+dfsg1/ \
http://example.com/~user/release/foo.html \
files/foo-([\d\.]*).tar.gz debian
opts="component=bar" \
http://example.com/~user/release/foo.html \
files/bar-([\d\.]*).tar.gz same uupdate
```

See "COPYRIGHT FILE EXAMPLES".

HTTP site (pagemangle)

The option **pagemangle** can be used to mangle the downloaded web page before applying other rules. The non-standard web page without proper **<a href="** << ... >> **">** entries can be converted. For example, if *foo.html* uses **<a bogus="** ... **">**, this can be converted to the standard page format with:

```
version=4
opts=pagemangle="s/<a\s+bogus=/<a href=/g" \
http://example.com/release/foo.html \
files/@PACKAGE@@ANY_VERSION@@ARCHIVE_EXT@ debian uupdate
```

Please note the use of **g** here to replace all occurrences.

If *foo.html* uses **<Key>** ... **</Key>**, this can be converted to the standard page format with:

```
version=4
opts="pagemangle=s%<Key>([^<]*)</Key>%<Key><a href="$1">$1</a></Key>%g" \
http://example.com/release/foo.html \
(?:.*)/@PACKAGE@@ANY_VERSION@@ARCHIVE_EXT@ debian uupdate
```

FTP site (basic):

```
version=4
ftp://ftp.tex.ac.uk/tex-archive/web/c_cpp/cweb/cweb-(.+)\.tar\.gz \
debian uupdate
```

FTP site (regex special characters):

```
version=4
ftp://ftp.worldforge.org/pub/worldforge/libs/\
Atlas-C++/transitional/Atlas-C\+\+-(.+)\.tar\.gz debian uupdate
```

Please note that this URL is connected to be ... **libs/Atlas-C++/** ... . For **++**, the first one in the directory path is verbatim while the one in the filename is escaped by ****.

FTP site (funny version)

This is another way of handling site with funny version numbers, this time using mangling. (Note that multiple groups will be concatenated before mangling is performed, and that mangling will only be performed on the basename version number, not any path version numbers.)

```
version=4
opts="uversionmangle=s/^/0.0./" \
ftp://ftp.ibiblio.org/pub/Linux/ALPHA/wine/\
development/Wine-(.+)\.tar\.gz debian uupdate
```

sf.net

For SourceForge based projects, qa.debian.org runs a redirector which allows a simpler form of URL. The format below will automatically be rewritten to use the redirector with the watch file:

```
version=4
https://sf.net/<project>/ <tar-name>-(.+)\.tar\.gz debian uupdate
```

For **audacity**, set the watch file as:

```
version=4
https://sf.net/audacity/ audacity-minsrc-(.+)\.tar\.gz debian uupdate
```

Please note, you can still use normal functionalities of **uscan** to set up a watch file for this site without using the redirector.

```
version=4
opts="uversionmangle=s/-pre/~pre/, \
      filenamemangle=s%(?:.*)audacity-minsrc-(.+)\.tar\.xz/download%\
                       audacity-$1.tar.xz%" \
      http://sourceforge.net/projects/audacity/files/audacity/(\d[\d\.]+)/ \
      (?:.*)audacity-minsrc-([\d\.]+)\.tar\.xz/download debian uupdate
```

Here, **%** is used as the separator instead of the standard **/**.

github.com

For GitHub based projects, you can use the tags or releases page. The archive URL uses only the version as the filename. You can rename the downloaded upstream tarball from into the standard *<project>-<version>.tar.gz* using **filenamemangle**:

```
version=4
opts="filenamemangle=s%(?:.*?)?v?(\d[\d.]*)\.tar\.gz%<project>-$1.tar.gz%" \
    https://github.com/<user>/<project>/tags \
    (?:.*?/)?v?(\d[\d.]*)\.tar\.gz debian uupdate
```

PyPI

For PyPI based projects, pypi.debian.net runs a redirector which allows a simpler form of URL. The format below will automatically be rewritten to use the redirector with the watch file:

```
version=4
https://pypi.python.org/packages/source/<initial>/<project>/ \
    <tar-name>-(.+)\.tar\.gz debian uupdate
```

For **cfn-sphere**, set the watch file as:

```
version=4
https://pypi.python.org/packages/source/c/cfn-sphere/ \
    cfn-sphere-([\d\.]+).tar.gz debian uupdate
```

Please note, you can still use normal functionalities of **uscan** to set up a watch file for this site without using the redirector.

```
version=4
opts="pgpmode=none" \
    https://pypi.python.org/pypi/cfn-sphere/ \
    https://pypi.python.org/packages/.*/.*/.*/\
    cfn-sphere-([\d\.]+).tar.gz#.* debian uupdate
```

code.google.com

Sites which used to be hosted on the Google Code service should have migrated to elsewhere (github?). Please look for the newer upstream site if available.

npmjs.org (node modules)

npmjs.org modules are published in JSON files. Here is a way to read them:

```
version=4
opts="searchmode=plain" \
 https://registry.npmjs.org/aes-js \
 https://registry.npmjs.org/aes-js/-/aes-js-(\d[\d\.]*)@ARCHIVE_EXT@
```

grouped package

Some node modules are split into multiple little upstream package. Here is a way to group them:

```
version=4
opts="searchmode=plain,pgpmode=none" \
  https://registry.npmjs.org/mongodb \
  https://registry.npmjs.org/mongodb/-/mongodb-(\d[\d\.]*)@ARCHIVE_EXT@ group
opts="searchmode=plain,pgpmode=none,component=bson" \
  https://registry.npmjs.org/bson \
  https://registry.npmjs.org/bson/-/bson-(\d[\d\.]*)@ARCHIVE_EXT@ group
opts="searchmode=plain,pgpmode=none,component=mongodb-core" \
  https://registry.npmjs.org/mongodb-core \
  https://registry.npmjs.org/mongodb-core/-/mongodb-core-(\d[\d\.]*)@ARCHIVE_EX
opts="searchmode=plain,pgpmode=none,component=requireoptional" \
  https://registry.npmjs.org/require_optional \
  https://registry.npmjs.org/require_optional/-/require_optional-(\d[\d\.]*)@AR
```

Package version is then the concatenation of upstream versions separated by "+~".

direct access to the git repository (tags)

If the upstream only publishes its code via the git repository and its code has no web interface to obtain the release tarball, you can use **uscan** with the tags of the git repository to track and package the new upstream release.

```
version=4
opts="mode=git, gitmode=full, pgpmode=none" \
http://git.ao2.it/tweeper.git \
refs/tags/v([\d\.]+) debian uupdate
```

Please note "**git ls-remote**" is used to obtain references for tags.

If a tag **v20.5** is the newest tag, the above example downloads *spkg*−**20.5.tar.xz** after making a full clone of the git repository which is needed for dumb git server.

If tags are signed, set **pgpmode=gittag** to verify them.

direct access to the git repository (HEAD)

If the upstream only publishes its code via the git repository and its code has no web interface nor the tags to obtain the released tarball, you can use **uscan** with the HEAD of the git repository to track and package the new upstream release with an automatically generated version string.

```
version=4
opts="mode=git, pgpmode=none" \
https://github.com/Debian/dh-make-golang \
HEAD debian uupdate
```

Please note that a local shallow copy of the git repository is made with "**git clone −−bare −−depth=1** ..." normally in the target directory. **uscan** generates the new upstream version with "**git log −−date=format:%Y%m%d −−pretty=0.0~git%cd.%h**" on this local copy of repository as its default behavior.

The generation of the upstream version string may the adjusted to your taste by adding **pretty** and **date** options to the **opts** arguments.

COPYRIGHT FILE EXAMPLES

Here is an example for the *debian/copyright* file which initiates automatic repackaging of the upstream tarball into *<spkg>_<oversion>.orig.tar.gz* (In *debian/copyright*, the **Files-Excluded** and **Files−Excluded−***component* stanzas are a part of the first paragraph and there is a blank line before the following paragraphs which contain **Files** and other stanzas.):

```
Format: http://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
Files-Excluded: exclude-this
 exclude-dir
 */exclude-dir
 .*
 */js/jquery.js

 Files: *
 Copyright: ...
 ...
```

Here is another example for the *debian/copyright* file which initiates automatic repackaging of the multiple upstream tarballs into *<spkg>_<oversion>.orig.tar.gz* and *<spkg>_<oversion>.orig-bar.tar.gz*:

```
Format: http://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
Files-Excluded: exclude-this
 exclude-dir
 */exclude-dir
 .*
 */js/jquery.js
Files-Excluded-bar: exclude-this
 exclude-dir
 */exclude-dir
 .*
 */js/jquery.js

 Files: *
 Copyright: ...
 ...
```

See **mk-origtargz** (1).

KEYRING FILE EXAMPLES

Let's assume that the upstream "**uscan test key (no secret) <none@debian.org>**" signs its package with a secret OpenPGP key and publishes the corresponding public OpenPGP key. This public OpenPGP key can be identified in 3 ways using the hexadecimal form.

- The fingerprint as the 20 byte data calculated from the public OpenPGP key. E. g., '**CF21 8F0E 7EAB F584 B7E2 0402 C77E 2D68 7254 3FAF**'
- The long keyid as the last 8 byte data of the fingerprint. E. g., '**C77E2D6872543FAF**'
- The short keyid is the last 4 byte data of the fingerprint. E. g., '**72543FAF**'

Considering the existence of the collision attack on the short keyid, the use of the long keyid is recommended for receiving keys from the public key servers. You must verify the downloaded OpenPGP key using its full fingerprint value which you know is the trusted one.

The armored keyring file *debian/upstream/signing-key.asc* can be created by using the **gpg** (or **gpg2**) command as follows.

```
$ gpg --recv-keys "C77E2D6872543FAF"
...
$ gpg --finger "C77E2D6872543FAF"
pub   4096R/72543FAF 2015-09-02
      Key fingerprint = CF21 8F0E 7EAB F584 B7E2  0402 C77E 2D68 7254 3FAF
uid                  uscan test key (no secret) <none@debian.org>
sub   4096R/52C6ED39 2015-09-02
$ cd path/to/<upkg>-<uversion>
$ mkdir -p debian/upstream
$ gpg --export --export-options export-minimal --armor \
      'CF21 8F0E 7EAB F584 B7E2  0402 C77E 2D68 7254 3FAF' \
      >debian/upstream/signing-key.asc
```

The binary keyring files, *debian/upstream/signing−key.pgp* and *debian/upstream−signing−key.pgp*, are still supported but deprecated.

If a group of developers sign the package, you need to list fingerprints of all of them in the argument for **gpg −−export ...** to make the keyring to contain all OpenPGP keys of them.

Sometimes you may wonder who made a signature file. You can get the public keyid used to create the detached signature file *foo−2.0.tar.gz.asc* by running **gpg** as:

```
$ gpg -vv foo-2.0.tar.gz.asc
gpg: armor: BEGIN PGP SIGNATURE
gpg: armor header: Version: GnuPG v1
:signature packet: algo 1, keyid C77E2D6872543FAF
      version 4, created 1445177469, md5len 0, sigclass 0x00
      digest algo 2, begin of digest 7a c7
      hashed subpkt 2 len 4 (sig created 2015-10-18)
      subpkt 16 len 8 (issuer key ID C77E2D6872543FAF)
      data: [4091 bits]
gpg: assuming signed data in `foo-2.0.tar.gz'
gpg: Signature made Sun 18 Oct 2015 11:11:09 PM JST using RSA key ID 72543FAF
...
```

COMMANDLINE OPTIONS

For the basic usage, **uscan** does not require to set these options.

−−conffile, **−−conf−file**
: Add or replace default configuration files (`/etc/devscripts.conf` and `~/.devscripts`). This can only be used as the first option given on the command-line.

  replace:

  ```
  uscan --conf-file test.conf --verbose
  ```

  add:

  ```
  uscan --conf-file +test.conf --verbose
  ```

  If one **−−conf−file** has no +, default configuration files are ignored.

−−no−conf, **−−noconf**
: Don't read any configuration files. This can only be used as the first option given on the command-line.

−−no−verbose
: Don't report verbose information. (default)

−−verbose, **−v**
: Report verbose information.

−−debug, **−vv**
Report verbose information including the downloaded web pages as processed to STDERR for debugging.

−−dehs
Send DEHS style output (XML-type) to STDOUT, while send all other uscan output to STDERR.

−−no−dehs
Use only traditional uscan output format. (default)

−−download, **−d**
Download the new upstream release. (default)

−−force−download, **−dd**
Download the new upstream release even if up-to-date. (may not overwrite the local file)

−−overwrite−download, **−ddd**
Download the new upstream release even if up-to-date. (may overwrite the local file)

−−no−download, **−−nodownload**
Don't download and report information.

Previously downloaded tarballs may be used.

Change default to **−−skip−signature**.

−−signature
Download signature. (default)

−−no−signature
Don't download signature but verify if already downloaded.

−−skip−signature
Don't bother download signature nor verifying signature.

−−safe, **−−report**
Avoid running unsafe scripts by skipping both the repacking of the downloaded package and the updating of the new source tree.

Change default to **−−no−download** and **−−skip−signature**.

When the objective of running **uscan** is to gather the upstream package status under the security conscious environment, please make sure to use this option.

−−report−status
This is equivalent of setting "**−−verbose −−safe**".

−−download−version *version*
Specify the *version* which the upstream release must match in order to be considered, rather than using the release with the highest version. (a best effort feature)

−−download−debversion *version*
Specify the Debian package version to download the corresponding upstream release version. The **dversionmangle** and **uversionmangle** rules are considered. (a best effort feature)

−−download−current−version
Download the currently packaged version. (a best effort feature)

−−check−dirname−level *N*
See the below section ‘‘Directory name checking’’ for an explanation of this option.

−−check−dirname−regex *regex*
See the below section ‘‘Directory name checking’’ for an explanation of this option.

−−destdir *path* Normally, **uscan** changes its internal current directory to the package's source directory where the *debian/* is located. Then the destination directory for the downloaded tarball and other files is set to the parent directory *../* from this internal current directory.

This default destination directory can be overridden by setting **−−destdir** option to a particular *path*. If this *path* is a relative path, the destination directory is determined in relative to the internal current directory of **uscan** execution. If this *path* is a absolute path, the destination directory is set to *path* irrespective of the internal current directory of **uscan** execution.

The above is true not only for the sinple **uscan** run in the single source tree but also for the advanced scanning **uscan** run with subdirectories holding multiple source trees.

One exception is when **−−watchfile** and **−−package** are used together. For this case, the internal current directory of **uscan** execution and the default destination directory are set to the current directory . where **uscan** is started. The default destination directory can be overridden by setting **−−destdir** option as well.

−−package *package*
Specify the name of the package to check for rather than examining *debian/changelog*; this requires the **−−upstream−version** (unless a version is specified in the *watch* file) and **−−watchfile** options as well. Furthermore, no directory scanning will be done and nothing will be downloaded. This option automatically sets **−−no−download** and **−−skip−signature**; and probably most useful in conjunction with the DEHS system (and **−−dehs**).

−−upstream−version *upstream-version*
Specify the current upstream version rather than examine *debian/watch* or *debian/changelog* to determine it. This is ignored if a directory scan is being performed and more than one *debian/watch* file is found.

−−watchfile *watchfile*
Specify the *watchfile* rather than perform a directory scan to determine it. If this option is used without **−−package**, then **uscan** must be called from within the Debian package source tree (so that *debian/changelog* can be found simply by stepping up through the tree).

One exception is when **−−watchfile** and **−−package** are used together. **uscan** can be called from anywhare and the internal current directory of **uscan** execution and the default destination directory are set to the current directory . where **uscan** is started.

See more in the **−−destdir** explanation.

−−bare
Disable all site specific special case codes to perform URL redirections and page content alterations.

−−no−exclusion
Don't automatically exclude files mentioned in *debian/copyright* field **Files-Excluded**.

−−pasv
Force PASV mode for FTP connections.

−−no−pasv
Don't use PASV mode for FTP connections.

−−no−symlink
Don't rename nor repack upstream tarball.

−−timeout *N*
Set timeout to *N* seconds (default 20 seconds).

−−user−agent, **−−useragent**
Override the default user agent header.

−−help
Give brief usage information.

−−version
Display version information.

uscan also accepts following options and passes them to **mk-origtargz**:

−−symlink
Make **orig.tar.gz** (with the appropriate extension) symlink to the downloaded files. (This is the default behavior.)

−−copy
Instead of symlinking as described above, copy the downloaded files.

−−rename
Instead of symlinking as described above, rename the downloaded files.

−−repack
After having downloaded an lzma tar, xz tar, bzip tar, gz tar, zip, jar, xpi archive, repack it to the specified compression (see **−−compression**).

The unzip package must be installed in order to repack zip and jar archives, the mozilla-devscripts package must be installed to repack xpi archives, and the xz-utils package must be installed to repack lzma or xz tar archives.

−−compression [**gzip** | **bzip2** | **lzma** | **xz**]
In the case where the upstream sources are repacked (either because **−−repack** option is given or *debian/copyright* contains the field **Files-Excluded**), it is possible to control the compression method via the parameter. The default is **gzip** for normal tarballs, and **xz** for tarballs generated directly from the git repository.

−−copyright−file *copyright-file*
Exclude files mentioned in **Files-Excluded** in the given *copyright-file*. This is useful when running **uscan** not within a source package directory.

DEVSCRIPT CONFIGURATION VARIABLES

For the basic usage, **uscan** does not require to set these configuration variables.

The two configuration files */etc/devscripts.conf* and *~/.devscripts* are sourced by a shell in that order to set configuration variables. These may be overridden by command line options. Environment variable settings are ignored for this purpose. If the first command line option given is **−−noconf**, then these files will not be read. The currently recognized variables are:

USCAN_DOWNLOAD
Download or report only:

no: equivalent to **−−no−download**, newer upstream files will not be downloaded.
yes: equivalent to **−−download**, newer upstream files will be downloaded. This is the default behavior.
See also **−−force−download** and **−−overwrite−download**.

USCAN_SAFE
If this is set to **yes**, then **uscan** avoids running unsafe scripts by skipping both the repacking of the downloaded package and the updating of the new source tree; this is equivalent to the **−−safe** options; this also sets the default to **−−no−download** and **−−skip−signature**.

USCAN_PASV
If this is set to yes or no, this will force FTP connections to use PASV mode or not to, respectively. If this is set to default, then **Net::FTP(3)** makes the choice (primarily based on the **FTP_PASSIVE** environment variable).

USCAN_TIMEOUT
If set to a number *N*, then set the timeout to *N* seconds. This is equivalent to the **−−timeout** option.

USCAN_SYMLINK
If this is set to no, then a *pkg_version***.orig.tar.{gz|bz2|lzma|xz}** symlink will not be made (equivalent to the **−−no−symlink** option). If it is set to **yes** or **symlink**, then the symlinks will be made. If it is set to **rename**, then the files are renamed (equivalent to the **−−rename** option).

USCAN_DEHS_OUTPUT
If this is set to **yes**, then DEHS-style output will be used. This is equivalent to the **−−dehs** option.

USCAN_VERBOSE
If this is set to **yes**, then verbose output will be given. This is equivalent to the **−−verbose** option.

USCAN_USER_AGENT
If set, the specified user agent string will be used in place of the default. This is equivalent to the **−−user−agent** option.

USCAN_DESTDIR
If set, the downloaded files will be placed in this directory. This is equivalent to the **−−destdir** option.

USCAN_REPACK
If this is set to yes, then after having downloaded a bzip tar, lzma tar, xz tar, or zip archive, uscan will repack it to the specified compression (see **−−compression**). This is equivalent to the **−−repack** option.

USCAN_EXCLUSION
If this is set to no, files mentioned in the field **Files-Excluded** of *debian/copyright* will be ignored and no exclusion of files will be tried. This is equivalent to the **−−no−exclusion** option.

EXIT STATUS

The exit status gives some indication of whether a newer version was found or not; one is advised to read the output to determine exactly what happened and whether there were any warnings to be noted.

0 Either **−−help** or **−−version** was used, or for some *watch* file which was examined, a newer upstream version was located.

1 No newer upstream versions were located for any of the *watch* files examined.

ADVANCED FEATURES

uscan has many other enhanced features which are skipped in the above section for the simplicity. Let's check their highlights.

uscan actually scans not just the current directory but all its subdirectories looking for *debian/watch* to process them all. See ''Directory name checking''.

uscan can be executed with *path* as its argument to change the starting directory of search from the current directory to *path* .

If you are not sure what exactly is happening behind the scene, please enable the **−−verbose** option. If this is not enough, enable the **−−debug** option too see all the internal activities.

See ''COMMANDLINE OPTIONS'' and ''DEVSCRIPT CONFIGURATION VARIABLES'' for other variations.

Custom script

The optional *script* parameter in *debian/watch* means to execute *script* with options after processing this line if specified.

See ''HISTORY AND UPGRADING'' for how **uscan** invokes the custom *script*.

For compatibility with other tools such as **git-buildpackage**, it may not be wise to create custom scripts with random behavior. In general, **uupdate** is the best choice for the non-native package and custom scripts, if created, should behave as if **uupdate**. For possible use case, see <http://bugs.debian.org/748474> as an example.

URL diversion

Some popular web sites changed their web page structure causing maintenance problems to the watch file. There are some redirection services created to ease maintenance of the watch file. Currently, **uscan** makes automatic diversion of URL requests to the following URLs to cope with this situation.

- <http://sf.net>
- <http://pypi.python.org>

Directory name checking

Similarly to several other scripts in the **devscripts** package, **uscan** explores the requested directory trees looking for *debian/changelog* and *debian/watch* files. As a safeguard against stray files causing potential problems, and in order to promote efficiency, it will examine the name of the parent directory once it finds the *debian/changelog* file, and check that the directory name corresponds to the package name. It will only attempt to download newer versions of the package and then perform any requested action if the directory name matches the package name. Precisely how it does this is controlled by two configuration file variables **DEVSCRIPTS_CHECK_DIRNAME_LEVEL** and **DEVSCRIPTS_CHECK_DIRNAME_REGEX**, and their corresponding command-line options **−−check−dirname−level** and **−−check−dirname−regex**.

DEVSCRIPTS_CHECK_DIRNAME_LEVEL can take the following values:

0 Never check the directory name.

1 Only check the directory name if we have had to change directory in our search for *debian/changelog*, that is, the directory containing *debian/changelog* is not the directory from which **uscan** was invoked. This is the default behavior.

2 Always check the directory name.

The directory name is checked by testing whether the current directory name (as determined by **pwd** (1)) matches the regex given by the configuration file option **DEVSCRIPTS_CHECK_DIRNAME_REGEX** or by the command line option **−−check−dirname−regex** *regex*. Here regex is a Perl regex (see **perlre** (3perl)), which will be anchored at the beginning and the end. If regex contains a **/**, then it must match the full directory path. If not, then it must match the full directory name. If regex contains the string *package*, this will be replaced by the source package name, as determined from the *debian/changelog*. The default value for the regex is: *package***(−.+)?**, thus matching directory names such as *package* and *package−version*.

HISTORY AND UPGRADING

This section briefly describes the backwards-incompatible *watch* file features which have been added in each *watch* file version, and the first version of the **devscripts** package which understood them.

Pre-version 2

The *watch* file syntax was significantly different in those days. Don't use it. If you are upgrading from a pre-version 2 *watch* file, you are advised to read this manpage and to start from scratch.

Version 2

devscripts version 2.6.90: The first incarnation of the current style of *watch* files.

Version 3

devscripts version 2.8.12: Introduced the following: correct handling of regex special characters in the path part, directory/path pattern matching, version number in several parts, version number mangling. Later versions have also introduced URL mangling.

If you are upgrading from version 2, the key incompatibility is if you have multiple groups in the pattern part; whereas only the first one would be used in version 2, they will all be used in version 3. To avoid this behavior, change the non-version-number groups to be **(?:** ... **)** instead of a plain **(** ... **)** group.

- **uscan** invokes the custom *script* as "*script* **−−upstream−version** *version* **../***spkg_version***.orig.tar.gz**".
- **uscan** invokes the standard **uupdate** as "**uupdate −−no−symlink −−upstream−version** *version* **../***spkg_version***.orig.tar.gz**".

Version 4

devscripts version 2.15.10: The first incarnation of *watch* files supporting multiple upstream tarballs.

The syntax of the watch file is relaxed to allow more spaces for readability.

If you have a custom script in place of **uupdate**, you may also encounter problems updating from Version 3.

- **uscan** invokes the custom *script* as "*script* **−−upstream−version** *version*".
- **uscan** invokes the standard **uupdate** as "**uupdate −−find −−upstream−version** *version*".

Restriction for **−−dehs** is lifted by redirecting other output to STDERR when it is activated.

SEE ALSO

dpkg (1), **mk−origtargz** (1), **perlre** (1), **uupdate** (1), **devscripts.conf** (5)

AUTHOR

The original version of uscan was written by Christoph Lameter <clameter@debian.org>. Significant improvements, changes and bugfixes were made by Julian Gilbey <jdg@debian.org>. HTTP support was added by Piotr Roszatycki <dexter@debian.org>. The program was rewritten in Perl by Julian Gilbey. Xavier Guimard converted it in object-oriented Perl using Moo.