Wydawca Manual

version 2.99.90, 12 March 2013

Sergey Poznyakoff.

Published by the Free Software Foundation, 51 Franklin Street, Fifth Floor
Boston, MA 02110-1301, USA

Short Contents

Table of Contents

1 Introduction to Wydawca

Let's begin with a short synopsis. Suppose you run a developer's site, like, e.g. '`gnu.org`'. You have two *distribution URLs*: '`ftp.gnu.org`', which distributes stable versions of the software, and '`alpha.gnu.org`', which distributes alpha and pre-test versions. Now, package maintainers need to have a way of uploading their packages to one of these sites. This is done using the *Automated FTP Upload* method, as described in Section "Automated FTP Uploads" in *Information for maintainers of GNU software*. The following is a short summary of it: there is an FTP *upload site*, which has two *source directories*, each one corresponding to a certain distribution URL. For example,

Source Directory	**Distribution Site**
`/incoming/ftp`	'`ftp.gnu.org`'
`/incoming/alpha`	'`alpha.gnu.org`'

Now, if maintainer of the project '`foo`' wishes to make a release of the stable version `foo-1.0.tar.gz`, he first creates a detached signature `foo-1.0.tar.gz.sig`. Then he creates a special *directive* file, which contains information about where the distributed tarball must be placed, and clear-signs it using his PGP key, thus obtaining the file `foo-1.0.tar.gz.directive.asc`. Finally, he uploads these three files (a *triplet*) to the upload site, storing them into the directory `/incoming/ftp`.

From now on, it is the responsibility of a *release submission daemon* to scan the source directories, gather the triplets, verify them, and to move any files that had successfully passed verification to their distribution sites.

`Wydawca` is such a release submission daemon. It is able to handle any number of '`source/destination`' pairs (called *spools*) in real time, and offers an extensible logging and mail notification mechanism, allowing both package maintainers and site administrators to be immediately notified about any occurring problems.

`Wydawca` supports upload directive versions 1.1[1] and 1.2[2].

The program is written entirely in C, is highly effective and consumes little resources.

[1] See Section "Standalone directives" in *Information for maintainers of GNU software*.

[2] See Section "Standalone directives" in *Information for maintainers of GNU software*.

2 Operation Overview

Usually, `wydawca` is installed on the machine that receives release uploads. It may be run either periodically as a cron-job, or as a standalone daemon. It supposes that both upload and distribution directories are accessible in the local file system hierarchy. If that is not the case (e.g. if upload and distribution sites are handled by different machines), one of them should be mounted using NFS. Future versions will contain special provisions for that case.

A configuration file defines a set of *spools*, i.e. pairs of upload and corresponding distribution directories. In `wydawca` terminology, upload directories are also called *source*, and distribution directories – *destination* directories. The configuration file also supplies all the information necessary to access user and project databases.

When started, `wydawca` scans each source directory and prepares a list of files found there. Then, it compacts this list by looking for *directive files* and re-arranging list members in *triplets*. A *directive file* is a special file that must be supplied with each upload and that contains directive regarding the placement of the uploaded files. A *triplet* is a standard entity, consisting of three files: a clear-signed directive file, a file to be distributed, and a detached signature of the latter. In some special cases, a clear-signed directive file alone is valid. This happens when it contains only *standalone directives*, as described in Section "Standalone directives" in *Information for maintainers of GNU software*.

Each *incomplete* triplet, i.e. a triplet missing one or more necessary files, is then verified by checking if the modification date of its oldest file is older than a predefined amount of time (see Section 4.2 [general], page 12). If so, the triplet is considered *expired*, and all its files are removed. This gives users the possibility to restart interrupted or otherwise broken uploads later.

After completing these preliminary stages, `wydawca` analyzes the directive file and extracts the project name from it. Using this name as a key, it looks up in the *project dictionary* a list of users authorized to make uploads for this project. This list contains user names and their corresponding public PGP keys. `Wydawca` tries to verify the directive file using each PGP key from this list, until a matching key is found, or the list in exhausted. In the latter case, the triplet is rejected. Otherwise, the key and its owner are remembered for the next step.

In this step, the uploaded file and its detached signature are verified. If they do not match the public key obtained in the previous step, the triplet is rejected.

Finally, directives from the directive file are executed. On this stage of the processing, the uploaded files are actually moved to their destination directories, requested symbolic links are created, etc.

2.1 Operation Modes

The program has two operation modes: '`cron mode`' and '`daemon mode`'.

In *cron mode*, `wydawca` runs in foreground and exits when it is done with processing all required spools. By default it processes all configured spools, unless a subset of them is specified in the command line. This is called *cron mode*, because this is the usual way for `wydawca` to be used as a cron job.

In *daemon mode*, `wydawca` detaches itself from the controlling terminal and runs in the background. It watches for the incoming uploads using one or both of the following methods.

On modern GNU/Linux systems `wydawca` uses *inotify* API (see Section "monitoring file system events" in *inotify man page*), which makes it possible to react on each upload immediately after a complete triplet is uploaded and to clean up unfinished or incomplete uploads. This is a preferred mode of operation.

On other systems, the daemon can be configured to listen on a socket for upload notifications. This method can also be used together with inotify, should the need be. This feature uses the TCPMUX protocol[1] and operates as follows:

After establishing connection, the remote party (the *client*) sends the spool tag followed by a CRLF pair. The server scans its configuration for a spool that has the requested ID. If no such spool is found, the server replies with the string '`- Unknown service name`', followed by a CRLF pair and closes the connection.

If a matching spool is found, the server replies with '`+`' acknowledgment, immediately followed by an optional message of explanation, and terminated with a CRLF. Upon receiving this acknowledgment, the client sends the user name of the user who did the upload. The following sample transaction illustrates this:

```
C: stable
S: +OK. URL ftp://ftp.domain.net
C: smith
```

When the user name is received, the server schedules a *job* for processing all triplets submitted by the given user to the given spool.

[1] RFC 1078.

3 How to invoke wydawca.

Wydawca gets all information it needs from its *configuration file* (see Chapter 5 [wydawca.rc], page 39). The default configuration file is *sysconfdir*/wydawca.rc, but if it is located elsewhere, you can specify its new location with the --config-file (-c) command line option.

If you wish to check your configuration file for syntax errors, use --lint (-t) command line option. When given this option, wydawca prints all diagnostics on its standard error and exits with code 0 if the file is OK, or 1 otherwise.

Normally, wydawca attempts to detect automatically whether it is run from an interactive console, and if so it prints its diagnostics on the standard error. Otherwise, the diagnostics is directed to the syslog, using the facility given in the syslog-facility configuration file statement (see Section 4.8 [syslog], page 16). Two options are provided if you wish to disable this autodetection: the option --syslog instructs the program to print all diagnostics via syslog, and the option --stderr (or -e) instructs it to print everything on the standard error.

The operation mode is configured in the configuration file. If the latter configures daemon mode, you can still instruct wydawca to run as a cron job by the --cron command line option. This may be needed, for example, to schedule a daily wydawca run when the main daemon instance is already running.

Usually wydawca attempts to process all the configured spools. You may instruct it to process only a subset of these by using the following options:

--spool=*tag*
-S *tag* Process only spool with the given tag.

--source=*dir*
-s *dir* Process only spool with *dir* as the source directory.

Any number of these options may be supplied, e.g.:

```
$ wydawca --spool=ftp --spool=test --source=/home/ftp/test-upload
```

The --debug (-d) option tells the program to increase its debugging level by 1. The *debugging level* determines amount of information the program reports when it runs. Default level is 0, which means that only errors and other critical conditions are reported. Raising it may be necessary when debugging new configurations. Each -d option raises the level by one, so you can say wydawca -dd to obtain level 2, for example. The maximum debugging level (currently it is 4) prints an impractically big amount of information, and is useful mainly for wydawca developers.

Yet another debugging facility is the --dry-run (-n) option. It instructs wydawca to avoid doing any modifications to the disk contents, and to print a verbose description of any actions it would have taken. It sets the debugging level to 1 and directs the diagnostics output to the standard error, as if --debug --stderr options were given. You can raise debugging level further

by supplying additional `--debug` options. The `--dry-run` option is useful when testing new configurations, for example:

```
$ wydawca -c new.cfg --dry-run
```

In addition, the two usual informational options are available as well: `--help` (`-h`) prints a short usage summary, and `--version` (`-v`) prints program version number.

4 How to Configure wydawca.

Upon startup, `wydawca` reads its settings from the *configuration file* `wydawca.rc`. By default it is located in *$sysconfidr* (i.e., in most cases `/usr/local/etc`, or `/etc`), but an alternative location may be specified using `--config` command line option (see Chapter 3 [starting], page 5).

If any errors are encountered in the configuration file, the program reports them on its error output and exits with a non-zero status.

To test the configuration file without starting the server use `--lint` (`-t`) command line option. It causes `wydawca` to check configuration file for syntax errors and other inconsistencies. If no errors were detected, the program exits with code 0. Otherwise, the exit code is 78.

Using this option together with `-d` (`--debug`), causes `wydawca` to produce a dump of the configuration parse tree. Using the `-d` option twice prefixes each statement in the dump with the file location where it appeared.

Before parsing, configuration file is preprocessed using `m4` (see Section 4.1.4 [Preprocessor], page 12). To see the preprocessed configuration without actually parsing it, use `-E` command line option. To avoid preprocessing it, use `--no-preprocessor` option.

The rest of this section describes the configuration file syntax in detail. You can receive a concise summary of all configuration directives any time by running `wydawca --config-help`.

4.1 Configuration file syntax

Wydawca configuration file consists of statements and comments.

There are three classes of lexical tokens: keywords, values, and separators. Blanks, tabs, newlines and comments, collectively called *white space* are ignored except as they serve to separate tokens. Some white space is required to separate otherwise adjacent keywords and values.

4.1.1 Comments

Comments may appear anywhere where white space may appear in the configuration file. There are two kinds of comments: single-line and multi-line comments. *Single-line* comments start with '`#`' or '`//`' and continue to the end of the line:

```
# This is a comment
// This too is a comment
```

Multi-line or *C-style* comments start with the two characters '`/*`' (slash, star) and continue until the first occurrence of '`*/`' (star, slash).

Multi-line comments cannot be nested. However, single-line comments may well appear within multi-line ones.

4.1.2 Pragmatic Comments

Pragmatic comments are similar to usual single-line comments, except that they cause some changes in the way the configuration is parsed. Pragmatic comments begin with a '`#`' sign and end with the next physical newline character. Wydawca version 2.99.90, understands the following pragmatic comments:

`#include <`*file*`>`
`#include` *file*
: Include the contents of the file *file*. If *file* is an absolute file name, both forms are equivalent. Otherwise, the form with angle brackets searches for the file in the *include search path*, while the second one looks for it in the current working directory first, and, if not found there, in the include search path.

  The default include search path is:

  1. *prefix*`/share/wydawca/2.99.90/include`
  2. *prefix*`/share/wydawca/include`

  where *prefix* is the installation prefix.

  New directories can be appended in front of it using `-I` (`--include-directory`) command line option (see Section 4.1.4 [Preprocessor], page 12).

`#include_once <`*file*`>`
`#include_once` *file*
: Same as `#include`, except that, if the *file* has already been included, it will not be included again.

`#line` *num*
`#line` *num* `"`*file*`"`
: This line causes `wydawca` to believe, for purposes of error diagnostics, that the line number of the next source line is given by *num* and the current input file is named by *file*. If the latter is absent, the remembered file name does not change.

`#` *num* `"`*file*`"`
: This is a special form of `#line` statement, understood for compatibility with the C preprocessor.

In fact, these statements provide a rudimentary preprocessing features. For more sophisticated ways to modify configuration before parsing, see Section 4.1.4 [Preprocessor], page 12.

4.1.3 Statements

A *simple statement* consists of a keyword and value separated by any amount of whitespace. Simple statement is terminated with a semicolon ('`;`').

Examples of simple statements:

```
daemon yes;
pidfile /var/run/wydawca.pid;
```

A *keyword* begins with a letter and may contain letters, decimal digits, underscores ('_') and dashes ('-'). Examples of keywords are: '`group`', '`file-sweep-time`'.

A *value* can be one of the following:

number
: A number is a sequence of decimal digits.

boolean
: A boolean value is one of the following: '`yes`', '`true`', '`t`' or '`1`', meaning *true*, and '`no`', '`false`', '`nil`', '`0`' meaning *false*.

unquoted string
: An unquoted string may contain letters, digits, and any of the following characters: '`_`', '`-`', '`.`', '`/`', '`@`', '`*`', '`:`'.

quoted string
: A quoted string is any sequence of characters enclosed in double-quotes ('`"`'). A backslash appearing within a quoted string introduces an *escape sequence*, which is replaced with a single character according to the following rules:

Sequence	Replaced with
\a	Audible bell character (ASCII 7)
\b	Backspace character (ASCII 8)
\f	Form-feed character (ASCII 12)
\n	Newline character (ASCII 10)
\r	Carriage return character (ASCII 13)
\t	Horizontal tabulation character (ASCII 9)
\v	Vertical tabulation character (ASCII 11)
\\	A single backslash ('\')
\"	A double-quote.

Table 4.1: Backslash escapes

In addition, the sequence '`\`*`newline`*' is removed from the string. This allows to split long strings over several physical lines, e.g.:

```
"a long string may be\
 split over several lines"
```

If the character following a backslash is not one of those specified above, the backslash is ignored and a warning is issued.

Two or more adjacent quoted strings are concatenated, which gives another way to split long strings over several lines to improve readability. The following fragment produces the same result as the example above:

```
"a long string may be"
" split over several lines"
```

Depending on the context, the contents of a quoted string may be subject to *meta-variable interpretation*. During this process, any sequence

```
${var}
```

where *var* is the name of a defined meta-variable, is replaced with the value of the variable. This sequence is called *meta-reference*. For example, if the meta-variable '`user`' has the value '`smith`', then the string

```
"where user = '${user}'"
```

becomes

```
"where user = 'smith'"
```

If the name of the variable consists of a single character, the curly braces around it may be omitted. Thus, `${u}` and `$u` are equivalent.

If *var* is not defined, the meta-reference is left unchanged.

The special sequence '`$-`' causes removal of it and any character following it. Most often it is used as a next-to-last character on a line, right before the newline. In this position it causes the removal of the trailing newline, similar to '`dnl`' in `m4`. See [listings], page 37, for a detailed description and examples of '`$-`' use.

To insert a literal '`$`' character in a string that is subject to meta-variable interpretation, duplicate it: '`$$`'.

The exact set of defined meta-variables and their values depend on the context and are discussed in detail below.

Here-document

A *here-document* is a special construct that allows to introduce strings of text containing embedded newlines.

The `<<`*word* construct instructs the parser to read all the following lines up to the line containing only *word*, with possible trailing blanks. Any lines thus read are concatenated together into a single string. For example:

```
<<EOT
A multiline
string
EOT
```

Body of a here-document is interpreted the same way as double-quoted string, unless *word* is preceded by a backslash (e.g. '`<<\EOT`') or enclosed in double-quotes, in which case the text is read as is, without interpretation of escape sequences.

If *word* is prefixed with `-` (a dash), then all leading tab characters are stripped from input lines and the line containing *word*. Furthermore, if `-` is followed by a single space, all leading whitespace is stripped from them. This allows to indent here-documents in a natural fashion. For example:

```
<<- TEXT
    All leading whitespace will be
    ignored when reading these lines.
TEXT
```

It is important that the terminating delimiter be the only token on its line. The only exception to this rule is allowed if a here-document appears as the last element of a statement. In this case a semicolon can be placed on the same line with its terminating delimiter, as in:

```
help-text <<-EOT
        A sample help text.
EOT;
```

list

A *list* is a comma-separated list of values. Lists are enclosed in parentheses. The following example shows a statement whose value is a list of strings:

```
alias (test,null);
```

In any case where a list is appropriate, a single value is allowed without being a member of a list: it is equivalent to a list with a single member. This means that, e.g.

```
alias test;
```

is equivalent to

```
alias (test);
```

time interval specification

The *time interval specification* is a string that defines an interval, much the same way we do this in English: it consists of one or more pairs '`number`'-'`time unit`'. For example, the following are valid interval specifications:

```
"1 hour"
"2 hours 35 seconds"
"1 year 7 months 2 weeks 2 days 11 hours 12 seconds"
```

The pairs can occur in any order, however unusual it may sound to a human ear, e.g. '`2 days 1 year`'. If the '`time unit`' is omitted, seconds are supposed.

A *block statement* introduces a logical group of statements. It consists of a keyword, followed by an optional value, and a sequence of statements enclosed in curly braces, as shown in the example below:

```
spool download {
  source /home/ftp/incoming/ftp;
  destination /home/ftp/pub;
}
```

The closing curly brace may be followed by a semicolon, although this is not required.

4.1.4 Preprocessor

Before parsing its configuration file, `wydawca` preprocesses it. The built-in preprocessor handles only file inclusion and `#line` statements (see Section 4.1.2 [Pragmatic Comments], page 8), while the rest of traditional preprocessing facilities, such as macro expansion, is supported via `m4`, which is used as an external preprocessor.

The detailed description of `m4` facilities lies far beyond the scope of this document. You will find a complete user manual in Section "GNU M4" in *GNU M4 macro processor*. For the rest of this subsection we assume the reader is sufficiently acquainted with `m4` macro processor.

The external preprocessor is invoked with `-s` flag, which instructs it to include line synchronization information in its output. This information is then used by the parser to display meaningful diagnostic. An initial set of macro definitions is supplied by the `pp-setup` file, located in *$prefix*`/share/wydawca/`*version*`/include` directory (where *version* means the version of Wydawca package).

The default `pp-setup` file renames all `m4` built-in macro names so they all start with the prefix '`m4_`'. This is similar to GNU m4 `--prefix-builtin` options, but has an advantage that it works with non-GNU `m4` implementations as well.

Additional control over the preprocessor is provided via the following command line options:

`--define=`*name*`[=`*value*`]`
`-D`*name*`[=`*value*`]`
: Define the preprocessor symbol *name* as having *value*, or empty.

`--include-directory=`*dir*
`-I`*dir*
: Add *dir* to the list of directories searched for preprocessor include files.

`--no-preprocessor`
: Disable preprocessor.

`--preprocessor=`*command*
: Use *command* instead of the default preprocessor.

4.2 General Settings

`foreground` *bool* [Config]
If *bool* is '`yes`', run in foreground. See Chapter 6 [invocation], page 45.

`single-process` *bool* [Config]
Configure single process mode. Normally `wydawca` spawns subprocesses for handling incoming connections and spool jobs. This is disabled if *bool*

is 'yes' (a so-called *single-processs mode*). This mode is designed for debugging purposes. Do not use it in production environments, because it severely impairs performance.

`umask` *value* [Config]
Set the default umask. The *value* argument must be an octal number.

`file-sweep-time` *time* [Config]
Consider triplet expired if its oldest file was created more than *time* seconds ago. See [time interval specification], page 11, for the syntax of *time*.

This parameter may also be set for each spool individually. See Section 4.12 [spool], page 25.

`gpg-homedir` *dir* [Config]
Set default GPG home directory. The keys for signing outgoing messages are looked up in this directory. See Section 4.15.3 [statreports], page 33, and Section 4.15.4 [event notification], page 35.

4.3 Upload Directive Versions

At the time of this writing, FSF has published three versions of the upload directives, numbered 1.0 through 1.2. The version 1.0 is considered obsolete and was withdrawn in 2006. The only difference between versions 1.1 and 1.2 is in handling of files that existed prior to upload. The version 1.1 implied automatic archivation of the existing files and their replacement with the newly uploaded versions. The version 1.2 introduces a new keyword ('replace') for that purpose, which determines its further actions.

For a detailed information about version 1.1, see See Section "Standalone directives" in *Information for maintainers of GNU software*.

The version 1.2 and its differences from 1.1 are discussed in See Section "Standalone directives" in *Information for maintainers of GNU software*.

By default, `wydawca` supports both versions. The supported range of versions can be abridged using the following configuration statements:

`min-version` *vn* [Config]
Sets minimal allowed directive file version. The *vn* argument must have the form '*major.minor*' and can not be less than '1.1'.

`max-version` *vn* [Config]
Sets maximal allowed directive file version.

For example, the following statements configure `wydawca` to accept only directive files of version 1.2:

```
min-version 1.2;
max-version 1.2;
```

4.4 User Privileges

`Wydawca` refuses to run with the root privileges. You should configure its user privileges by using `user` and, optionally, `group` statements in its configuration file:

`user` *name* [Config]
Run with UID and GID of the user *name*.

`group` *list* [Config]
Retain the supplementary groups from the *list*. The latter must contain group names. For example:

```
group (nogroup, ftp);
```

4.5 Daemon Configuration

Statements in this section configure the daemon mode.

`daemon` *bool* [Config]
Enable daemon mode.

`inotify` *bool* [Config]
Enables or disables the *inotify* watcher. By default, inotify is always enabled on GNU/Linux systems (unless disabled at the configure time). It can also be configured for each spool individually (See Section 4.12 [spool], page 25. See [inotify], page 4, for a detailed description of this feature.

`listen` *url* [Config]
Define a socket to listen on. Allowed values for *url* are:

inet://*ip*:*port*
: Listen on IPv4[1]. address *ip*. *Ip* may be given either in a dotted quad notation or as a symbolic host name. *Port* is either a decimal port name, or a service name from `/etc/services`.

local://*file*
file://*file*
unix://*file*
: Listen on the UNIX socket file *file*, which is either an absolute or relative file name.

`all-spools` *name* [Config]
Declare a special service name, which will be treated as a request to process all spools.

`wakeup-interval` *time* [Config]
Specifies the wake-up interval for the daemon. If no connections are requested during *time*, the server will wake up and sweep all the configured

[1] Support for IPv6 will be added in future versions.

spools. It is useful for periodical removal of expired triplets. See also `file-sweep-time` statement (see Section 4.2 [general], page 12).

See [time interval specification], page 11, for the syntax of *time*.

`pidfile` *file* [Config]

Store master process PID in *file*. Default pidfile location is *localstatedir*`/run/wydawca.pid`.

4.6 TCP Wrappers

Access to the socket specified in `listen` statement is controlled by the `tcp-wrapper` block statement:

`tcp-wrapper` *{ ... }* [Config]

```
tcp-wrapper {
  enable arg:boolean;
  daemon name:string;
  allow-table file:string;
  deny-table file:string;
  allow-syslog-priority prio:string;
  deny-syslog-priority prio:string;
}
```

This statement is available only if `wydawca` was compiled with support for TCP wrappers.

`enable` *bool* [Config: tcp-wrapper]

Enable or disable the use of TCP wrappers.

`daemon` *name* [Config: tcp-wrapper]

Set the *daemon name*. It is the name before the colon in the access control file, that marks the line controlling access to `wydawca`. The default is '`wydawca`'.

`allow-table` *file* [Config: tcp-wrapper]

File name of the positive access control file. By default `/etc/hosts.allow`.

`deny-table` *file* [Config: tcp-wrapper]

File name of the negative access control file. By default `/etc/hosts.deny`.

`allow-syslog-priority` *prio* [Config: tcp-wrapper]

Log allowed accesses via the given `syslog` priority.

`deny-syslog-priority` *prio* [Config: tcp-wrapper]

Log denied accesses via the given `syslog` priority.

Allowed values for *prio* in the '`allow-syslog-priority`' and '`deny-syslog-priority`' statements are: '`emerg`', '`alert`', '`crit`', '`err`', '`warning`', '`notice`', '`info`', and '`debug`'.

4.7 Locking Configuration

To avoid a possibility of two `wydawca` instances handling the same triplet, `wydawca` *locks* the spool before processing it. This is done by creating a *lock file*. The parameters of the locking subsystem are configured via the `locking` statement:

`locking` *{ ... }* [Config]

```
locking {
  enable arg:boolean;
  directory dir:string;
  expire-time time:interval;
  timeout time:interval;
}
```

`enable` *bool* [Config: locking]
Enable or disable locking. By default it is enabled.

`directory` *dir* [Config: locking]
Sets directory for lock files. Make sure *dir* is writable for the user (or group) `wydawca` runs at (see Section 4.4 [user privileges], page 14.
The default directory is *`localstatedir`*`/lock/wydawca`.

`expire-time` *time* [Config: locking]
Sets expiration interval for lock files. An existing lock file older than *time* is considered stale and removed.
See [time interval specification], page 11, for the syntax of *time*.

`timeout` *time* [Config: locking]
Timeout for acquiring locks. If a lock file cannot be acquired during this time, `wydawca` reports error and exits.
See [time interval specification], page 11, for the syntax of *time*.

4.8 Syslog Configuration Directives

Unless told otherwise, `wydawca` uses `syslog` to print its diagnostic messages. By default, the program uses the '`local1`' facility. The `syslog` statement allows to change that:

`syslog` *{ ... }* [Config]

```
syslog {
  facility local1;
  tag wydawca;
  print-priority yes;
}
```

`facility` *name* [Config: syslog]
Configures the syslog facility to use. Allowed values are: '`auth`', '`authpriv`', '`cron`', '`daemon`', '`ftp`', '`local0`' through '`local7`', and '`mail`'. These names are case-insensitive and may be optionally prefixed with '`log_`' (case-insensitive as well).

`tag` *string* [Config: syslog]
This statement sets the *syslog tag*, a string identifying each message issued by the program. By default, the name of the program ('`wydawca`') is used.

`print-priority` *bool* [Config: syslog]
In addition to priority segregation, provided by `syslog`, you can instruct `wydawca` to prefix each syslog message with its priority. To do so, set:

```
print-priority yes;
```

4.9 SQL Databases

Several statements in configuration file may need to access an SQL database. `Wydawca` is able to use any number of databases simultaneously, the only restriction being that they must be `MySQL` databases (this restriction will be removed in future releases).

A database is defined using `sql` block statement:

`sql` *id* { ... } [Config]

```
sql id {
  config-file file;
  config-group group;
  host hostname;
  database dbname;
  user username;
  password string;
  ssl-ca string;
}
```

Here, *id* is a string uniquely identifying this database. It is used by other configuration statements (e.g. by dictionaries, see the next section) to refer to this database.

`config-file` *name* [Config: sql]
Set the name of the SQL configuration file to read.

`config-group` *name* [Config: sql]
Set the name of the group in the SQL configuration file, from where to read configuration options.

The statements above allow to keep all security-sensitive information, such as SQL username and password, in an external configuration file and thus to relax permission requirements for `wydawca.rc`. The exact format of such external configuration file depends on the flavor of SQL DBMS in use. As of version 2.99.90 `wydawca` supports only '`MySQL`', so the configuration file is what is called *option file* in '`MySQL`' parlance (see Section "option-files" in *MySQL Manual*).

For example, suppose your `wydawca.rc` contains the following:

```
sql default {
  config-file /etc/wydawca.mysql;
  config-group wydawca;
```

```
}
```

Then, the `/etc/wydawca.mysql` would contain the actual parameters for accessing the database, e.g.:

```
[wydawca]
socket = /var/db/mysql.sock
database = savane
user = savane
pass = guessme
```

Another way to specify database credentials is by using the statements described below. If you prefer this way, you will have to tighten the permissions of `wydawca.rc` so that no third person could see the SQL password. The recommended permissions are '`0600`'.

`host` *hostname*[*:port-or-socket*] [Config: sql]
Set the hostname or IP address of the host running the database. Optional *port-or-socket* specifies port number (for TCP connections) or socket name (for UNIX sockets) to use. In the latter case, the *hostname* and the colon may be omitted. If, however, it is present, it must be '`localhost`'.

`database` *name* [Config: sql]
Specifies the database name.

`user` *name* [Config: sql]
Sets the database user name.

`password` *string* [Config: sql]
Sets the password for accessing the database.

`ssl-ca` *file* [Config: sql]
Sets the pathname to the certificate authority file, if you wish to use a secure connection to the server via SSL.

An example `sql` statement follows:

```
sql default {
  host project.database.com:3306;
  database savane;
  user root;
  password guessme;
}
```

It is possible to combine both methods, e.g.:

```
sql default {
  config-file /etc/wydawca.sql;
  host project.database.com:3306;
  database savane;
}
```

Then, `wydawca` will attempt to obtain the missing information (username and password, in this case) from the `/etc/wydawca.sql` file.

4.10 Dictionaries

A *dictionary* defines the ways to retrieve user information necessary to verify the submission. This information can be, for example, the user's PGP key or his permissions on a project.

A dictionary is defined in configuration file using the following syntax:

`dictionary` *{ ... }* [Config]

```
dictionary dict-id {
  type type;
  query string;
  params (param1,param2,...);
}
```

The `dictionary` statement can appear either in the global scope of the configuration file, or inside a `spool` statement (see Section 4.12 [spool], page 25). Global definitions affect all directory pairs in the configuration file, and ones inside a `directory` statement override them for that particular spool.

There are two dictionaries, identified by the value of *dict-id* tag:

project-owner
: Keeps email addresses and real names of administrators (or *owners*) of a project. It may return any number of rows, each one consisting of two columns: an email address and a user name, in this order.

project-uploader
: Keeps system user names, real names, emails and GPG keys of the users that are allowed to make uploads for the project.

The sub-statements of `dictionary` are:

`type` *name* [Config: dictionary]

Defines the type of this dictionary. *Name* is one of the following:

builtin
: The data are supplied in the configuration file.

sql
: Retrieve data from an SQL database. Currently only `MySQL` is supported.

external
: Retrieve data using an external program. This dictionary type is reserved for future use.

See below for a detailed description of these dictionary types.

`query` *string* [Config: dictionary]

Sets the query used for retrieving the data. The *string* is subject to meta-variable interpretation (see [meta-interpretation], page 10). The following meta-variables are defined:

`p`
`project` The system name of the project for which the triplet is submitted. It is defined as the value of directive `directory`, or, in case this value contains slashes, the shortest initial prefix of that value, not containing slashes.

`spool` The name of the distribution spool where this upload originates (see Section 4.12 [spool], page 25).

`url` The URL of the spool, as set in the `url` statement of the `spool` block (see Section 4.12 [spool], page 25).

`dir` Directory (relative to the project distribution root) where the files are going to be uploaded.

`dest-dir` Spool destination directory (see Section 4.12 [spool], page 25).

`source-dir`
Spool source directory (see Section 4.12 [spool], page 25).

`u`
`user`
`user:name`
The system name of the user that submitted the triplet. This is defined only for '`project-owner`' dictionaries.

`comment` The value of the '`comment`' field from the directive file.

`params` (*param1*, *param2*, ...) [Config: dictionary]
Supplies additional parameters.

4.10.1 SQL Dictionary

Dictionaries of '`sql`' type retrieve information from an SQL database (as of version 2.99.90, only '`MySQL`' databases are supported).

The `query` statement supplies the SQL query to execute. Normally, it should be a `SELECT` query.

The `params` statement must supply a single parameter – the identifier of one of the preceding `sql` blocks (see Section 4.9 [sql], page 17), which determines database name and user credentials needed to access it.

The following sub-nodes contain sample definitions for the `sql` dictionaries. They are based on the database structure used in `Savane` system.

4.10.1.1 Project-owner: an SQL Implementation

Retrieve email addresses and real names of administrators (or *owners*) of a project. It may return any number of rows, each one consisting of two columns: an email address and a user name, in this order.

```
dictionary project-owner {
  type sql;
  params (default);
```

```
  query   "SELECT user.email, user.realname "
          "FROM user,user_group,groups "
          "WHERE user_group.user_id=user.user_id "
          "AND user_group.group_id=groups.group_id "
          "AND user_group.admin_flags = 'A' "
          "AND groups.unix_group_name = '${project}'";
}
```

4.10.1.2 Project-uploader: an SQL Implementation

This dictionary assumes that the 'user' table has a special column, 'upload_flags', whose value is 'Y' for those users who can do uploads for this project:

```
dictionary project-uploader {
  type sql;
  params (default);
  query   "SELECT user.email, user.realname "
          "FROM user,user_group,groups "
          "WHERE user_group.user_id=user.user_id "
          "AND user_group.group_id=groups.group_id "
          "AND user_group.upload_flags = 'Y' "
          "AND groups.unix_group_name = '${project}'";
}
```

4.10.2 Built-in Dictionary

Builtin dictionaries are small dictionaries that keep all data in their `params` list. They are designed mainly for testing purposes.

Look ups in builtin dictionaries are performed as follows: The `query` value is expanded (see [query], page 19). The resulting value is used as a *key* for lookup in `params` list. The list scanned as follows:

1. INIT

   Let *i* be the index of the current element in `params`. Set *i* to 0.

2. GETEL

   Get the *i*th element.

3.

   If it begins with a slash, interpret it as *comparison type indicator*. Its possible values are:

   /exact — Exact comparison. The key must be exactly equivalent to the dictionary field.

   /fnmatch — Dictionary field is treated as an *fnmatch globbing pattern*. See Section "globbing pattern" in *glob man page*.

   /regex — Dictionary field is treated as a regular expression. Unless configured otherwise by flags (see below), POSIX extended regular expressions are used (see Section "Extended regular expressions" in *GNU sed*).

If that word ends with a comma, the characters following it are *flags*, defining the type of matching. Allowed flags are:

Flag	Meaning
i	Ignore case
b	Use basic regular expressions

For example, the string '`/exact,i`' specifies case-insensitive exact comparison, the string '`/regex,bi`' specifies case-insensitive basic regular expression matching, etc.

Go to step '`INCR`'.

4. COMP

   Compare the element with the key, using currently selected comparison method.

5.

   If the element matches the key, add elements `i+1` through `i+n` to the result set. The value for *n* is selected as follows:

Dictionary	*n*
project-owner	2
project-uploader	4

6.

   Set `i = i + n`

7. INCR

   Set `i = i + 1`.

8. LOOP

   If *i* is greater than the number of elements in `param`, then stop. Otherwise, go to step '`GETEL`'.

For example, the following defines the '`project-owner`' dictionary, containing data for projects '`foo`' and '`bar`':

```
dictionary project-owner {
  type builtin;
  query "${project}";
  params ("/exact",
          "foo", "foo-owner@domain.net", "Foo Admin",
          "bar", "smith@other.net", "John Smith");
}
```

4.10.3 External Dictionary

As of version 2.99.90 this dictionary is not yet implemented.

4.11 Archivation

There may be cases when project maintainers need to overwrite existing distributed files with another ones, having the same names. (Note, however,

that this practice is not encouraged). In that case, wydawca needs to first *archive* the already existing file, and then put the new one in its place. Moreover, the directive file format allows maintainers to explicitly require archivation of their existing files.

Wydawca supports two basic archivation methods: to a tar file, and to a separate directory. The method to be used is configured using archive statement. This statement can appear either in the global scope, in which case it affects all spools, or within a spool block (see Section 4.12 [spool], page 25), where it affects only the given spool.

archive *type* [Config]

```
archive type {
  name file-or-dir;
  backup type;
}
```

The *type* argument specifies the archivation type:

none	Disable archivation.
tar	Add to a tar archive.
directory	Store file in a separate directory.

name *file-or-dir* [Config: archive]

Specify the name of the tar archive (if type 'tar' is used) or destination directory (if type 'directroy' is used).

If the archivation type tar is used, the name statement sets the full name of the tar archive to use, e.g.:

```
archive tar {
  name /var/spool/uploads/archive.tar;
}
```

The file being archived is appended to the archive using tar -r (see Section "Appending Files to an Archive" in *GNU tar: an archiver tool*). Any archived instance can subsequently be retrieved using GNU tar --occurrence option (see Section "Multiple Files with the Same Name" in *GNU tar: an archiver tool*).

tar-program *name* [Config]

By default, wydawca will search for tar binary in your search path. If you wish to use a particular binary, you may specify its full file name using tar-program statement.

The 'directory' archivation type means that archive copies will be stored in a directory specified by the name statement. If it begins with a slash (i.e. represents an absolute file name), an exact copy of the distribution directory hierarchy will be created under it. For example, given this configuration:

```
archive directory {
  name /var/backups/gnu;
}
```

all files from `/home/ftp/gnu/tar` will be archived in `/var/backups/gnu/tar`, and files from `/home/ftp/gnu/tar/old` will be archived in `/var/backups/gnu/tar/old`, etc.

If the directory name does not begin with a slash, it will be created under the corresponding distribution directory. For example, the following archivation settings:

```
archive directory {
  name .archive;
}
```

mean that files from `/home/ftp/gnu/tar` will be archived in the directory `/home/ftp/gnu/tar/.archive`, files from `/home/ftp/gnu/tar/old` — in `/home/ftp/gnu/tar/.archive/old`, etc.

`backup` *type* [Config: archive]

When using the '`directory`' archivation type, it may happen that the archive file with the same name as the one about to be created already exists. This statement specifies how to handle the existing copy, in other words, how to *backup* it. The *type* argument corresponds to the '`version-control`' Emacs variable. The following table describes its possible values:

'`t`'
'`numbered`'
: Always make numbered backups.

'`nil`'
'`existing`'
: Make numbered backups of files that already have them, and simple backups of the others.

'`never`'
'`simple`'
: Always make simple backups.

If no backup method is given, '`existing`' is assumed

Signature files (i.e. the ones ending with '`.sig`') are usually located in the same directory as the files they sign. To enforce this rule, `wydawca` implements *implicit signature archivation* facility. It works as follows. When archivation of *file* is requested by `archive:` ***`file`*** statement in the directive file (see Section "FTP Upload Directive File - v1.1" in *Information For Maintainers of GNU Software*), `wydawca` also checks if the file named ***`file`***`.sig` exists. If so, it is archived along with ***`file`***.

`archive-signatures` *bool* [Config]

If implicit signature archivation is not needed, use the `archive-signatures` statement to disable it, e.g.:

```
archive-signatures no;
```

4.12 Distribution Spool

A *distribution spool* defines the location of the source directory and the corresponding distribution (or *destination*) directory. It may also set archivation type, various dictionaries and notifications for that directory, thus overriding the global settings.

The `spool` block statement defines a distribution spool:

`spool` *tag { ... }* [Config]

```
spool tag {
  url url;
  alias (aliases);
  inotify bool;
  source dir;
  destination dir;
  file-sweep-time interval;
  dictionary { ... }
  archive { ... }
  notify-event { ... }
}
```

The *tag* argument defines a unique identifier for this spool. It will be used in log messages, timers (see [spool-timers], page 34) and in meta-variable interpretation (see [meta-interpretation], page 10).

`alias` *list* [Config: spool]
Defines a list of *aliases*, i.e. alternative tag names for this spool.

`inotify` *bool* [Config: spool]
Enables or disables the *inotify* watcher for this spool. By default, inotify is always enabled on GNU/Linux systems (unless disabled at the configure time). See [inotify], page 4, for a detailed description of this feature.

`url` *string* [Config: spool]
Defines download URL, associated with this spool. Its value may be used as '`${url}`' meta-variable in mail notifications.

`source` *dir* [Config: spool]
Specifies the location of the source directory.

`destination` *dir* [Config: spool]
Specifies the type and location of the destination directory. The *dir* argument must be either an absolute name of a directory on the local file system, or a special URL. `Wydawca` version 2.99.90 supports two destination URL schemes:

file://*dir-name*
dir://*dir-name*
: Equivalent to *dir-name* alone. Defines a destination directory located on the local file system.

null: Defines a *null upload spool*. Null spools implement all tests described in Chapter 2 [overview], page 3, but do not do any actual copying. The uploaded files are simply removed after checks are over. Null spools are useful mainly for diagnostic purposes.

The following statements, if present, override the corresponding global definitions for this spool.

`archive` *{ ... }* [Config: spool]
Configure spool-specific archivation. See Section 4.11 [archivation], page 22, for its description.

`dictionary` *tag { ... }* [Config: spool]
Configure spool-specific dictionary. See Section 4.10 [dictionaries], page 19, for a detailed discussion of this statement.

`file-sweep-time` *time* [Config: spool]
Set expiration time for triplets in this spool. A triplet is considered expired if its oldest file was created more than *time* seconds ago. This statement overrides the global '`file-sweep-time`' setting (see Section 4.2 [general], page 12).

`notify-event` *{ ... }* [Config: spool]
Configure spool-specific event notification. See Section 4.15 [notification], page 30, for a detailed discussion of this statement.

The `source` and `destination` statements are mandatory.

For example, the following definition says that valid uploads to `/home/ftp/incoming/ftp` should be transferred to `/home/ftp/gnu`:

```
spool ftp {
 url ftp://ftp.gnu.org.ua;
 source /home/ftp/incoming/ftp;
 destination /home/ftp/gnu;
}
```

This spool defines no particular archivation type, dictionary or notifications, so it will inherit these settings from the global configuration.

The following example shows the same spool, that additionally sets its own archivation method:

```
spool ftp {
 url ftp://ftp.gnu.org.ua;
 source /home/ftp/incoming/ftp;
 destination /home/ftp/gnu;
 archive directory {
   name .archive;
   backup numbered;
 }
}
```

4.13 Distribution Verification

After the submission has been verified, `wydawca` may also run an additional check to verify whether the main file (normally, a tarball) is OK to be distributed. To set up such *distribution verification*, add the following statement either in the global scope, or within a '`spool`' declaration:

`check-script` *text* [Config]
`check-script` *text* [Config:spool]
Define the distribution verification script. The *text* must be a valid `sh` program. It is executed without arguments, in a temporary directory which contains a copy of the main distribution file. The script can refer to the following environment variables:

`WYDAWCA_SPOOL` [Check Environment]
Spool tag.

`WYDAWCA_SOURCE` [Check Environment]
Spool source directory, as set by the `source` statement (see Section 4.12 [tag], page 25).

`WYDAWCA_DEST` [Check Environment]
Spool destination directory (see Section 4.12 [destination], page 25).

`WYDAWCA_URL` [Check Environment]
Spool URL (see Section 4.12 [url], page 25).

`WYDAWCA_TRIPLET_BASE` [Check Environment]
Base name of the triplet.

`WYDAWCA_DIST_FILE` [Check Environment]
File name of the main distribution file.

Apart from these, the script inherits `wydawca` environment.

The submission is accepted only if the script returns 0. Otherwise, it is rejected and the '`check-failure`' event (see Section 4.15.4 [event notification], page 35) is generated.

In case of non-zero return, the script may return additional diagnostics on the standard output. This diagnostics will be available for use in notification messages via the '`check:diagn`' meta-variable.

Additionally, the actual return code of the script, in decimal, is available in the '`check:result`' meta-variable. If the script terminates on a signal, the value of this variable is '`SIG+`*n*', where *n* is the signal number.

If both global and spool '`check-script`'s are defined, `wydawca` executes both scripts as if they were connected by a logical '`&&`', i.e. per-spool script is executed only if the global one returned success ('`0`'). The submission is accepted only if both scripts returned '`0`'.

Since the script usually contains several lines, the '`config-script`' value is usually supplied using a here-document construct (see [here-document], page 10).

The following example illustrates the use of '`config-script`' to catch possible security holes in the distributed `Makefile.in` files[2]

```
  check-script <<EOT
case ${WYDAWCA_DIST_FILE} in
*.tar|*.tar.*)
  if tar -xOf ${WYDAWCA_DIST_FILE} --occurrence=1 \
      --wildcards --no-wildcards-match-slash '*/Makefile.in' | \
      grep -q 'perm -777'; then
    fmt <<_EOF_
The top-level Makefile.in in ${WYDAWCA_DIST_FILE} changes mode of
all the directories below the build tree to 777 before creating
the tarball. This constitutes a security hole (see CVE-2009-4029[1],
for more details).

Please, rebuild the package using a newer Automake (at least v. 1.11.1)
and resubmit.
_EOF_
    cat <<_EOF_
--
[1] http://article.gmane.org/gmane.comp.sysutils.autotools.announce/131
_EOF_
    exit 1
  fi
  ;;
*)
  ;;
esac

exit 0
EOT;
```

4.14 Statistics

At the end of the run, `wydawca` prints a detailed statistics of its execution on the diagnostic channel '`info`'. The statistics is printed only if at least one of its items is not zero. The following example illustrates what you might get if you configured full statistics output:

```
errors: 0
warnings: 2
bad signatures: 0
access violation attempts: 0
complete triplets: 6
incomplete triplets: 2
bad triplets: 0
expired triplets: 0
```

[2] See http://article.gmane.org/gmane.comp.sysutils.autotools.announce/131.

```
triplet successes: 6
files uploaded: 12
files archived: 2
symlinks created: 0
symlinks removed: 0
```

Each item in this statistics is configurable, and a unique configuration keyword is associated with it. The statistics items and their corresponding keywords are described in the table below:

`errors` Any error that occurred during the run.

`warnings` Any warning condition occurred during the run.

`bad-signatures`
: A PGP signature not matches the public key for the user that issued it.

`access-violations`
: A user is attempting to upload files for some project, but it is not authorized to do so.

`complete-triplets`
: A complete triplet is registered.

`incomplete-triplets`
: An incomplete triplet is registered, i.e. such that misses one or more of its files. Notice, that a directive file alone is counted as a complete triplet, provided that its signature verifies correctly and that it does not contain `file` directive.

`bad-triplets`
: A triplet contains files owned by different users.

`expired_triplets`
: A triplet has expired.

`triplet_success`
: A triplet is processed successfully

`uploads` An upload is processed successfully. An upload is defined as a move of a file and its detached signature from the source to the destination directory.

`archives` An archivation is performed

`symlinks` A symlink is created.

`rmsymlinks`
: A symlink is removed.

`statistics` *list* [Config]

The amount of information included in the statistics summary is configured using the `statistics` statement. This statement takes a list of

arguments, each one being one of the keywords, described above. For example, the following statement causes only the information about errors and warnings to be printed:

```
statistics (errors, warnings);
```

The output produced looks like:

```
errors: 0
warnings: 2
```

A special keyword '`none`' can be used to suppress this output altogether (which is the default), as in

```
statistics none;
```

Another special keyword is '`all`'. It enables full statistics output. This keyword may also be followed by any number of statistics keywords, which are in this case *excluded* from the summary. For example, to output all statistics, except errors and warnings one would set:

```
statistics (all, errors, warnings);
```

4.15 Mail Notification

While running, `wydawca` keeps track of certain events occurring, such as, for example, broken PGP signatures or file uploads attempted by unauthorized users. The utility can notify, via email, project administrators about any of those events that concern their projects. Additionally, the system administrator can choose to obtain via email the execution statistics, described in the previous section.

The sender email address for these notifications is set using the `from-address` statement.

`from-address` *address* [Config]

Set sender address for outgoing mails. E.g.:

```
from-address ftp-uploads@gnu.org.ua;
```

It is not strictly necessary to specify the sender address. In the absence of `from-address` statement, the sender email will be constructed from the name of the user `wydawca` runs as (see Section 4.4 [user privileges], page 14) and the full domain name of the machine it runs at.

`admin-address` *email* [Config]

Sets the admin email address or addresses. The statistics notifications and any notifications configured to be sent to admins will be forwarded to this address. The *email* argument is either a RFC 822 email address, or a list of such addresses. For example, the following statement configures a single admin address:

```
admin-address root@gnu.org.ua;
```

The example below illustrates how to configure multiple addresses:

```
admin-address "root@gnu.org.ua,ftp-adm@gnu.org.ua";
```

Yet another way to configure them is:

```
admin-address (root@gnu.org.ua, ftp-adm@gnu.org.ua);
```

4.15.1 Mailer

To send messages, `wydawca` uses a special logical entity called a *mailer*. It is set in the configuration file using `mailer` keyword.

`mailer` *url* [Config]
Set mailer URL.

A mailer URL consists of a scheme specification, followed by '`://`' separator and additional data. The URLs supported by Wydawca version 2.99.90 are described in the table below. As usual, square brackets indicate optional parts:

smtp://*host*[:*port*]
Use an SMTP server on *host* to relay messages. The *host* part is either an IP address in dotted-quad notation or as a symbolic host name. In the latter case, DNS system is be used to resolve it. Optional *port* specifies port number or symbolic name (as defined in `/etc/services`). It defaults to 25. For example:

```
mailer smtp://remote.server.net:24;
```

sendmail://*progname*
Use sendmail-compatible program *progname*. *Sendmail-compatible* means that the program must be able to read an RFC-822 message from its standard input and must support the following command line options:

`-oi` Do not treat '`.`' as message terminator.

`-f` *addr* Use *addr* as the sender address.

`-t` Get recipient addresses from the message.

Example:

```
mailer sendmail:///usr/sbin/exim;
```

sendmail: This is a special form of the '`sendmail`' mailer. It uses the `sendmail` binary from the `_PATH_SENDMAIL` macro in your `/usr/include/paths.h`. It is the default mailer.

prog://*progname*?*query*
A *prog* mailer. This is a generalization of '`sendmail`' mailer that allows to use arbitrary external programs as mailers.

The full file name of the program is given in *progname* part. The *query* part is a list of arguments, separated by '`&`' signs. Arguments may contain the following macro-substitutions:

'`${sender}`'
Expands to the sender email address.

‘`${rcpt}`’ Expands to the recipient email addresses.

The program *progname* must read an RFC-822 message from its standard input.

An example of ‘`prog`’ mailer definition:

```
mailer "prog:///bin/nullmail?localhost&-F${sender}&${rcpt}
```

When sending a mail, `wydawca` will invoke:

```
/bin/nullmail localhost -Fsender rcpt
```

where *sender* means the sender address, and *rcpt* stands for the recipient email address.

‘`| prog args..`’ Equivalent to the ‘`prog`’ mailer, described above, but written in a more natural fashion. In this notation, the example definition above becomes:

```
mailer "|/bin/nullmail localhost -F${sender} ${rcpt}"
```

4.15.2 Message Templates

Each notification message is build from a message template, by expanding meta-variables (see [meta-interpretation], page 10) within it. The message text may be specified either in place within the configuration directive it belongs to (see Section 4.15 [notification], page 30), or defined by `define-message` statement.

`define-message` *name text* [Config]
Define message *name* to be *text*. This message can be referred to from other configuration statements by `@name` notation.

The message text must be formatted as a valid RFC-822 message, i.e. it must consist of two parts, message headers and body, separated by a single empty line. Therefore *text* is usually a *here-document* construct (see [here-document], page 10). For example:

```
define-message my-message <<EOT
From: Wydawca
Subject: test

This is a test message.
EOT;
```

If you do not wish to supply any headers (which is unlikely, because a mail should at least have a `Subject` header), simply begin the message text with an empty line, like this:

```
define-message my-message <<EOT

This is a test message.
EOT;
```

4.15.3 Statistic Reports

`mail-statistics` *{ ... }* [Config]
The `mail-statistics` statement configures the statistic reports sent to the system administrator.

```
mail-statistics {
  message text-or-id;
  statistics item-list;
  gpg-sign key;
}
```

`message` *text-or-id* [Config: mail-statistics]
Define the message text. The argument is either the message text template, or a reference to a template previously defined by a `define-message` (see Section 4.15.2 [templates], page 32). The reference syntax is:

```
message @name;
```

where *name* is the message name as used in `define-message`.

`statistics` *item-list* [Config: mail-statistics]
The argument is a list of statistics keywords as described in Section 4.14 [statistics], page 28. A report will be sent only if statistics counters for at least one of the requested categories are not zero. For example, the following statement requires sending notifications only if there occurred any errors or access violation attempts, or any bad signature was uploaded:

```
statistics (errors, access-violations, bad-signatures);
```

`gpg-sign` *key* [Config: mail-statistics]
If this statement is present, the message will be signed using the supplied GPG *key*. The key is looked up in the GPG home directory (see [gpg-homedir], page 13).

The statistics message is sent to addresses configured by `admin-address` statement (see Section 4.15 [notification], page 30).

The meta-variables available for use in statistics reports are:

Variable	**Replaced with**
date	Current date and time in the current locale.
stat:errors	Number of errors detected.
stat:warnings	Number of warnings reported.
stat:bad_signatures	Number of bad signatures detected.
stat:access_violations	Number of access violation attempts.
stat:complete_triplets	Number of complete triplets processed.
stat:incomplete_triplets	Number of incomplete triplets left in the source directory.
stat:bad_triplets	Number of bad triplets seen.
stat:expired_triplets	Number of expired triplets.

stat:triplet_success	Number of successfully processed triplets.
stat:uploads	Number of successful uploads.
stat:archives	Number of archivations performed.
stat:symlinks	Number of symbolic links created.
stat:rmsymlinks	Number of symbolic links removed.
stat:check-failures	Number of verification failures (see Section 4.13 [verification], page 27).

The following special variables, called *timers*, are replaced with the real or CPU time (in seconds) used while processing a certain task:

timer:wydawca:real	Real time spent in `wydawca` main code.
timer:wydawca:system	System CPU time spent in `wydawca` main code.
timer:wydawca:user	User CPU time spent in `wydawca` main code.
timer:*tag*:real	Real time spent processing the *tag* spool.
timer:*tag*:system	System CPU time spent processing the *tag* spool.
timer:*tag*:user	User CPU time spent processing the *tag* spool.

An example definition of the admin notification template follows:

```
mail-statistics {
  statistics (errors,warnings,bad_signatures,
              access_violations);
  message <<EOT
Subject: Wydawca stats

This is to notify you that my run on ${date}
caused the following results:

errors ............................. ${stat:errors}
warning ............................ ${stat:warnings}
bad signatures ..................... ${stat:bad_signatures}
access violation attempts .......... ${stat:access_violations}

Timings:
Real ............................... ${timer:wydawca:real} \
(${timer:releases:real} + \
${timer:alpha:real} + \
${timer:test:real})
System ............................. ${timer:wydawca:system} \
(${timer:releases:system} + \
${timer:alpha:system} + \
${timer:test:system})
User ............................... ${timer:wydawca:user} \
(${timer:releases:user} + \
${timer:alpha:user} + \
${timer:test:user})

Regards,
Wydawca
EOT;
}
```

4.15.4 Event Notification

A number of *events* are tracked during the execution. Any of them can be used to trigger an email notification of any party concerned: the system administrator, project administrators, the user that initiated the upload, or any other recipients, specified by their email addresses. These notifications are configured using the `notify-event` statement:

`notify-event` *{ ... }* [Config]

```
notify-event {
  event ev-id;
  recipient who;
  message text-or-id;
  gpg-key key;
}
```

`event` *ev-id* [Config: notify-event]

Send notification when the event *ev-id* occurs. The following table describes the available *ev-id*s:

success
: Successful upload.

bad-ownership
: An unauthorized user attempted to upload files for their project.

bad-directive-signature
: The directive signature does not match the public key of the uploader.

bad-detached-signature
: The detached signature does not match the public key of the uploader.

check-failure
: Distribution verification failed. See Section 4.13 [verification], page 27, for a detailed description.

`recipient` *who* [Config: notify-event]

Determines who should receive the notification. The following values for *who* are allowed:

`read message`
: Read recipients from the '`To`', '`Cc`' and '`Bcc`' headers of the message.

`admin`
: The system administrator, as defined in `admin-address` statement (see Section 4.15 [notification], page 30).

`owner`
: Administrators of the project for which the files where uploaded. Their addresses are retrieved from the '`project-owner`' dictionary (see Section 4.10 [dictionaries], page 19).

`user` User name of the user who uploaded files.

`message` *text-or-id* [Config: notify-event]
Define the message text. The argument is either the message text template, or a reference to a template previously defined by a `define-message` (see Section 4.15.2 [templates], page 32).

`gpg-sign` *key* [Config: notify-event]
If this statement is present, the message will be signed using the supplied GPG *key*. The key is looked up in the GPG home directory (see [gpg-homedir], page 13).

For example, the following two statements instruct `wydawca` to email notifications about any `bad-directive-signature` event to project administrators and to the user who did the upload, using two different templates:

```
notify-event {
  event bad-directive-signature;
  recipient user;
  message @usermsg;
}

notify-event {
  event bad-directive-signature;
  recipient owner;
  message @ownermsg;
}
```

The following macro-variables may be used in templates for these notifications:

Variable	**Replaced with**
project	Project system name.
url	URL of the distribution site.
spool	Name of the spool (see Section 4.12 [spool], page 25).
dir	Directory (relative to the project distribution root) where the files where uploaded.
dest-dir	Value of the `destination` keyword.
source-dir	Value of the `source` keyword.
triplet:dist	File name of the main distribution file.
triplet:sig	File name of the detached signature file.
triplet:dir	File name of the directive file.
triplet:ls:full	A full listing of the uploaded triplet[3].
triplet:upload	Listing of the uploaded files (see below).

[3] It is equivalent to:
```
${triplet:ls:dist}
${triplet:ls:sig}
${triplet:ls:dir}
```

triplet:ls:dist	Listing of the main distribution file (see below).
triplet:ls:sig	Listing of the detached signature file (see below).
triplet:ls:dir	Listing of the directive file (see below).
user	System name of the user who uploaded the triplet.
user:name	System name of the user who uploaded the triplet.
user:real-name	Real name of the user who uploaded the triplet.
user:email	Email of the user who uploaded the triplet.
email:admin	Full[4]. email address of the systems administrator, as set by the '`admin-address`' (see Section 4.15 [admin-address], page 30).
email:owner	Full email address of the project administrator (*owner*).
email:user	Full email address of the user who did the upload. Not to be confused with '`user:email`'.
check:result	Code returned by external checker, in decimal. See Section 4.13 [check-result], page 27, for a detailed description.
check:diagn	Diagnostics text returned by external checker. See Section 4.13 [verification], page 27, for a detailed description.

The following *timers* (see [spool-timers], page 34) are defined:

Variable	**Replaced with**
timer:wydawca:real	Real time spent in `wydawca` main code.
timer:wydawca:system	System CPU time spent in `wydawca` main code.
timer:wydawca:user	User CPU time spent in `wydawca` main code.
timer:triplet:real	Real time spent processing this triplet.
timer:triplet:system	System CPU time spent processing this triplet.
timer:triplet:user	User CPU time spent processing this triplet.
timer:spool:real	Real time spent while processing this spool.
timer:spool:system	System CPU time spent while processing this spool.
timer:spool:user	User CPU time spent while processing this spool.

Listings referred to in the table above, are similar to those produced by the `ls` command, and include information on file permissions, ownership, size and modification date. For example, here is a possible `${triplet:ls:full}` listing:

```
-rw-r--r-- gray users 2707278 2007-09-06 22:14:35 tar-1.18.tar.gz
-rw-r--r-- gray users     189 2007-09-06 22:14:35 tar-1.18.tar.gz.sig
-rw-r--r-- gray user       62 2007-09-06 22:14:35 tar-1.18.tar.gz.directive.asc
```

[4] *Full* here means an email address with eventual personal part

The following example shows how to configure success notification for the user. Notice the use of the '`$-`' after '`${triplet:ls:upload}`': it removes the newline character after it and thus allows for more natural indentation of the next line. Without it, the expanded message would have contained two newlines after the full listing: one produced by '`${triplet:ls:upload}`' and the second one taken verbatim from the message template.

```
notify-event {
  event success;
  recipient user;
  message <<EOT
Subject: Upload of ${project} successful

Upload of ${project} to ${url}/${dir} finished successfully.
Files uploaded:

${triplet:ls:upload}$-

Resource usage: ${timer:triplet:real}/${timer:wydawca:real}r \
${timer:triplet:user}/${timer:wydawca:user}u \
${timer:triplet:system}/${timer:wydawca:system}s

Regards,
Wydawca
The Project Submission Robot
EOT;
}
```

5 Wydawca configuration file.

This chapter summarizes the configuration statements. For each statement, a reference to its detailed description is provided.

```
# Enable daemon mode.
# See Section 4.5 [daemon], page 14.
daemon arg:boolean;

# Start in foreground even in daemon mode.
# See Section 4.2 [general], page 12.
foreground arg:boolean;

# Do not spawn subprocesses.
# See Section 4.2 [general], page 12.
single-process arg:boolean;

# Set wake-up interval.
# See Section 4.2 [general], page 12.
wakeup-interval time:string;

# Set pid file name.
# See Section 4.2 [general], page 12.
pidfile file:string;

# Run with UID and GID of this user.
# See Section 4.4 [user privileges], page 14.
user name:string;

# Retain these supplementary groups:
# See Section 4.4 [user privileges], page 14.
group arg:list of string;

# Configure locking
# See Section 4.7 [locking], page 16.
locking {
  # Enable or disable locking.
  enable arg:boolean;

  # Set directory for lock files.
  directory dir:string;

  # Define lock expiration interval.
  expire-time time:interval;

  # Locking timeout.
  timeout time:interval;
}

# Listen on this address.
# See Section 4.5 [daemon], page 14.
listen socket:sock-addr;
```

```
# Configure TCP wrappers.
# See Section 4.6 [tcp-wrapper], page 15.
tcp-wrapper {
  # Enable TCP wrapper access control. Default is 'yes'.
  enable arg:boolean;

  # Set daemon name for TCP wrapper lookups. Default is program name.
  daemon name:string;

  # Use file for positive client address access control.
  # (default: /etc/hosts.allow).
  allow-table file:string;

  # Use file for negative client address access control.
  # (default: /etc/hosts.deny).
  deny-table file:string;

  # Log host allows at this syslog priority.
  allow-syslog-priority prio:string;

  # Log host denies at this syslog priority.
  deny-syslog-priority prio:string;
}

# Set mailer URL.
# See Section 4.15.1 [mailer], page 31.
mailer url:string;

# Set admin email address.
# See Section 4.15 [notification], page 30.
admin-address email:string;

# Set sender email address.
# See Section 4.15 [notification], page 30.
from-address email:string;

# Define file sweep time.
# See Section 4.2 [general], page 12.
file-sweep-time time:interval;

# Set tar invocation command line.
# See Section 4.11 [archivation], page 22.
tar-program prog:string;

# Set umask.
# See Section 4.2 [general], page 12.
umask mask:octal;

# Control implicit signature archivation.
# See Section 4.11 [archivation], page 22.
archive-signatures arg:boolean;
```

```
# Print these stats at the end of each run.
# See Section 4.14 [statistics], page 28.
statistics items:string;

# Service names that request scanning all spools.
# See Section 4.5 [daemon], page 14.
all-spools arg:list of string;

# GPG home directory.
# See [gpg-homedir], page 13.
gpg-homedir arg:string;

# Define SQL database.
# See Section 4.9 [sql], page 17.
sql id:string {
  # Set the name of the configuration file to read.
  config-file name:string;
  # Set the name of the configuration file group to use.
  config-group name:string;

  # Set SQL server hostname or IP address.
  host host:string;

  # Set database name.
  database dbname:string;

  # Set SQL user name.
  user name:string;

  # Set SQL user password.
  password arg:string;

  # File name of the Certificate Authority (CA) certificate.
  ssl-ca file:string;
}

# Configure syslog logging.
# See Section 4.8 [syslog], page 16.
syslog {
  # Set syslog facility.
  facility name:string;

  # Tag syslog messages with this string.
  tag string:string;

  # Prefix each message with its priority.
  print-priority arg:boolean;
}

# Define message text.
# See Section 4.15.2 [templates], page 32.
define-message ident:string text:string;
```

```
# Set up archivation.
# See Section 4.11 [archivation], page 22.
archive type:string {
  # Name of archive file or directory.
  name file-or-dir:string;

  # Define backup type.
  # See [backup-methods], page 24.
  backup type:string;
}

# Send statistics.
# See Section 4.15.3 [statreports], page 33.
mail-statistics {
  # Message text.
  message text:string;

  # Send mail if one or more of these items are set.
  statistics items:string;

  # Sign message with this key.
  gpg-sign key:string;
}

# Configure notification.
# See Section 4.15 [notification], page 30.
notify-event {
  # Event on which to notify.
  event ev-id:string;

  # Notify this recipient.
  recipient who:string;

  # Text of the notification or identifier of
  # a defined message template.
  message text-or-id:string;

  # Sign message with this key.
  gpg-sign key:string;
}

# Define data dictionary.
# See Section 4.10 [dictionaries], page 19.
dictionary ident:string {
  # Dictionary type.
  type type:string;

  # Query template.
  query string:string;

  # Set dictionary parameters.
```

```
  params arg:list of string;
}

# Define distribution spool.
# See Section 4.12 [spool], page 25.
spool tag:string {
  # URL corresponding to this spool.
  url arg:string;

  # Aliases.
  alias arg:list of string;

  # Source directory.
  source dir:string;

  # Destination directory.
  destination dir:string;

  # Define file sweep time.
  file-sweep-time time:interval;

  # Define data dictionary.
  # See above.
  dictionary ident:string { ... }

  # Set up archivation.
  archive type:string { ... }

  # Configure notification.
  notify-event { ... }
}
```

6 `Wydawca` invocation summary.

This chapter presents a short reference of all `wydawca` command line options, in alphabetical order.

`--config-file=`*file*
`-c` *file*
: Use *file* instead of the default configuration file.
See [config-file], page 5.

`--cron`
: Run in cron mode. See Chapter 3 [cron], page 5.
See [stderr], page 5.

`--debug`
`-d`
: Increase debugging level by 1.
See [debug], page 5.

`--define=`*name*`[=`*value*`]`
`-D`*name*`[=`*value*`]`
: Define the preprocessor symbol *name* as having *value*, or empty.
See Section 4.1.4 [Preprocessor], page 12.

`--dump-grammar-trace`
: Dump configuration grammar traces. This is useful for debugging `wydawca` configuration file parser.

`--dump-lex-trace`
: Dump lexical analyzer traces. This is useful for debugging `wydawca` configuration file parser.

`--dry-run`
`-n`
: *Dry-run mode*: do nothing, print almost everything. This option implies `--debug --stderr`.
See [dry-run], page 5.

`--stderr`
`-e`
: Log to the standard error.
See [stderr], page 5.

`--spool=`*tag*
`-S` *tag*
: Process only spool with the given tag. See [spool selection], page 5.

`--source=`*dir*
`-s` *dir*
: Process only spool with *dir* as the source directory. See [spool selection], page 5.

`--syslog`
: Log all diagnostics to syslog.
See [stderr], page 5.

`--force`
: Force start-up, even if if the PID file already exists.

`--foreground`
: Remain in the foreground. This is mostly for debugging `wydawca`.

`--include-directory=`*`dir`*
`-I` *`dir`*
: Add *dir* to include search path.

  See Section 4.1.2 [Pragmatic Comments], page 8. See Section 4.1.4 [Preprocessor], page 12.

`--lint`
`-t`
: Parse configuration file, report any errors on the standard error and exit with code 0, if the syntax is OK, and with code 1 otherwise.

  See [lint], page 5.

`--no-preprocessor`
: Disable preprocessor. see Section 4.1.4 [Preprocessor], page 12.

`--preprocessor=`*`command`*
: Use *command* instead of the default preprocessor. see Section 4.1.4 [Preprocessor], page 12.

`--single-process`
: Serialize job invocations by not forking subprocesses for each job. *Do not use this option in production environment.*

`--source=`*`name`*
`-s`*`name`*
: Process only the spool with the given source name. This option may be given multiple times, to select several spools by their source names.

`--spool=`*`tag`*
`-S`*`tag`*
: Process only spool with the given tag. This option may be given multiple times, to select several spools by their tag names.

`--help`
`-h`
: Print a concise usage summary and exit.

`--version`
`-v`
: Print the program version and exit.

7 How to Report a Bug

Email bug reports to bug-wydawca@gnu.org.ua.

As the purpose of bug reporting is to improve software, please be sure to include a detailed information when reporting a bug. The minimum information needed is:

- Program version you use (see the output of `wydawca --version`.
- A description of the bug.
- Conditions under which the bug appears.
- It is often helpful to send the contents of `config.log` file along with your bug report. This file is created after running `./configure` in `wydawca` source root directory.

Appendix A GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within

that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque

copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If

there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire

aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (C) *year your name*.
> Permission is granted to copy, distribute and/or modify this document
> under the terms of the GNU Free Documentation License, Version 1.2
> or any later version published by the Free Software Foundation;
> with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
> A copy of the license is included in the section entitled ‘‘GNU
> Free Documentation License’’.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

> with the Invariant Sections being *list their titles*, with
> the Front-Cover Texts being *list*, and with the Back-Cover Texts
> being *list*.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Concept Index

This is a general index of all issues discussed in this manual

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W