## CONTENTS

1 Introduction  
1.1 About GNU Taler ......................................................... 1  
1.2 About this manual ..................................................... 1  
1.3 Organizational prerequisites ........................................ 1  
1.4 Architecture overview ............................................... 2  
1.5 Offline keys ............................................................ 3  
1.6 Online signing key security ......................................... 3  

2 Installation  
2.1 Installing from source ................................................ 5  
2.2 Installing the GNU Taler binary packages on Debian .............. 6  
2.3 Installing the GNU Taler binary packages on Trisquel .......... 7  
2.4 Installing the GNU Taler binary packages on Ubuntu ............ 7  

3 Configuration  
3.1 Configuration format ................................................ 9  
3.2 Using taler-config .................................................... 10  
3.3 Keying ................................................................. 10  
3.4 Serving ............................................................... 11  
3.5 Currency ............................................................. 11  
3.6 Database ............................................................. 11  
3.7 Coins (denomination keys) ........................................... 12  
3.8 Sign keys ............................................................. 13  
3.9 Terms of Service ..................................................... 13  
3.10 Bank account ......................................................... 14  
3.11 Auditor configuration ............................................... 15  

4 Deployment  
4.1 Launching an exchange .............................................. 17  
4.2 Keys generation ...................................................... 18  
4.3 Private key storage .................................................. 18  
4.4 Database upgrades ................................................... 19  

5 Testing a deployment .................................................... 21  

6 Diagnostics  
6.1 Internal audits ....................................................... 23  
6.2 Database Scheme .................................................... 23  

7 Benchmarking ........................................................... 25
A  GNU Free Documentation License
A.1  0. PREAMBLE ................................................................. 27
A.2  1. APPLICABILITY AND DEFINITIONS .................................. 27
A.3  2. VERBATIM COPYING ..................................................... 28
A.4  3. COPYING IN QUANTITY .................................................... 29
A.5  4. MODIFICATIONS ............................................................ 29
A.6  5. COMBINING DOCUMENTS ................................................ 30
A.7  6. COLLECTIONS OF DOCUMENTS ........................................ 31
A.8  7. AGGREGATION WITH INDEPENDENT WORKS ....................... 31
A.9  8. TRANSLATION ............................................................... 31
A.10 9. TERMINATION .............................................................. 31
A.11 10. FUTURE REVISIONS OF THIS LICENSE ............................ 32
A.12 11. RELICENSEING ........................................................... 32
A.13  ADDENDUM: How to use this License for your documents ........ 32

Index ................................................................................. 35
This manual is an early draft that still needs significant editing work to become readable.

1.1 About GNU Taler

GNU Taler is an open protocol for an electronic payment system with a free software reference implementation. GNU Taler offers secure, fast and easy payment processing using well understood cryptographic techniques. GNU Taler allows customers to remain anonymous, while ensuring that merchants can be held accountable by governments. Hence, GNU Taler is compatible with anti-money-laundering (AML) and know-your-customer (KYC) regulation, as well as data protection regulation (such as GDPR).

GNU Taler is not yet production-ready, after following this manual you will have a backend that can process payments in “KUDOS”, but not regular currencies. This is not so much because of limitations in the backend, but because we are not aware of a Taler exchange operator offering regular currencies today.

1.2 About this manual

This tutorial targets system administrators who want to install and operate a GNU Taler exchange.

1.3 Organizational prerequisites

Operating a GNU Taler exchange means that you are operating a payment service provider, which means that you will most likely need a bank license and/or follow applicable financial regulation.

GNU Taler payment service providers generally need to ensure high availability and have really good backups (synchronous replication, asynchronous remote replication, off-site backup, 24/7 monitoring, etc.). This manual will not cover these aspects of operating a payment service provider.

We will assume that you can operate a (high-availability, high-assurance) PostgreSQL database. Furthermore, we expect some moderate familiarity with the compilation and installation of free software packages. You need to understand the cryptographic concepts of private and public keys and must be able to protect private keys stored in files on disk.

Note: The Taler merchant backend stores private keys and other sensitive business and customer data in the database. The backend operator SHOULD ensure that backup operations are encrypted and secured from unauthorized access.
1.4 Architecture overview

Taler is a pure payment system, not a new crypto-currency. As such, it operates in a traditional banking context. In particular, this means that in order to receive funds via Taler, the merchant must have a regular bank account, and payments can be executed in ordinary currencies such as USD or EUR. Similarly, the Taler exchange must interact with a bank. The bank of the exchange holds the exchange’s funds in an escrow account.

Note that, given the technical burden (XML-based communications, additional cryptography, and a vast variety of standards) due to interact with banks, the exchange uses an intermediary system to talk to its bank. Such intermediary system abstracts the native banking protocol by exposing the Taler Wire Gateway API; this way, the exchange can conduct its banking operations in a simplified and JSON-based style.

When customers wire money to the escrow account, the bank notifies the exchange about the incoming wire transfers. The exchange then creates a reserve based on the subject of the wire transfer. The wallet which knows the secret key matching the wire transfer subject can then withdraw coins from the reserve, thereby draining it. The liability of the exchange against the reserve is thereby converted into a liability against digital coins issued by the exchange. When the customer later spends the coins at a merchant, and the merchant deposits the coins at the exchange, the exchange first aggregates the amount from multiple deposits from the same merchant and then instructs its bank to make a wire transfer to the merchant, thereby fulfilling its obligation and eliminating the liability. The exchange charges fees for some or all of its operations to cover costs and possibly make a profit.

Auditors are third parties, for example financial regulators, that verify that the exchange operates correctly. The same software is also used to calculate the exchange’s profits, risk and liabilities by the accountants of the exchange.

The Taler software stack for an exchange consists of the following components:

- **HTTP frontend** The HTTP frontend interacts with Taler wallets and merchant backends. It is used to withdraw coins, deposit coins, refresh coins, issue refunds, map wire transfers to Taler transactions, inquire about the exchange’s bank account details, signing keys and fee structure. The binary is the `taler-exchange-httpd`.

- **Crypto-Helpers** The `taler-exchange-secmod-rsa` and `taler-exchange-secmod-eddsa` are two programs that are responsible for managing the exchange’s online signing keys. They must run on the same machine as the `taler-exchange-httpd` as the HTTP frontend communicates with the crypto helpers using UNIX Domain Sockets.

- **Aggregator** The aggregator combines multiple deposits made by the same merchant and (eventually) triggers wire transfers for the aggregate amount. The merchant can control how quickly wire transfers are made. The exchange may charge a fee per wire transfer to discourage excessively frequent transfers. The binary is the `taler-exchange-aggregator`.

- **Closer** The `taler-exchange-closer` tool check that reserves are properly closed. If a customer wires funds to an exchange and then fails to withdraw them, the closer will (eventually) trigger a wire transfer that sends the customer’s funds back to the originating wire account.

- **Transfer** The `taler-exchange-transfer` tool is responsible for actually executing the aggregated wire transfers. It is the only process that needs to have the credentials to execute outgoing wire transfers. The tool uses the Taler Wire Gateway API to execute wire transfers. This API is provided by the Taler Python Bank for stand-alone deployments (like those with KUDOS) and by LibEuFin. LibEuFin is an adapter which maps the Taler Wire REST API to traditional banking protocols like EBICS and FinTS.

- **Wirewatch** The `taler-exchange-wirewatch` tool is responsible for observing incoming wire transfers to the exchange. It needs to have the credentials to obtain a list of incoming wire transfers. The tool also uses the Taler Wire Gateway API to observe such incoming transfers. It is possible that observing incoming and making outgoing wire transfers is done via different bank accounts and/or credentials.

- **Wire adapter** A wire adapter is a component that enables exchange to talk to a bank.

  (1) The `libtalerfakebank` implements a bank with a wire adapter API inside of a testcase.
(2) For the demonstration website (or local currencies), the Python bank provides a bank that directly provides the wire adapter API.

(3) For production, libeufin’s Nexus component implements a wire adapter towards the traditional SEPA banking system with IBAN accounts.

The client-side wire adapter API is implemented in libtalerbank and is used by the transfer to execute wire transfers and for the auditor to query bank transaction histories.

- **DBMS** The exchange requires a DBMS to store the transaction history for the Taler exchange and aggregator, and a (typically separate) DBMS for the Taler auditor. For now, the GNU Taler reference implementation only supports PostgreSQL, but the code could be easily extended to support another DBMS. ... index:: PostgreSQL

- **Auditor** The auditor verifies that the transactions performed by the exchange were done properly. It checks the various signatures, totals up the amounts and alerts the operator to any inconsistencies. It also computes the expected bank balance, revenue and risk exposure of the exchange operator. The main binary is the `taler-auditor`. Aside from the key setup procedures, the most critical setup for deploying an auditor is providing the auditor with an up-to-date copy of the database.

### 1.5 Offline keys

The exchange (and ideally also auditors) uses a long-term offline master signing key that identifies the operator and is used to authenticate critical information, such as the exchange’s bank account and the actual keys the exchange uses online.

Interactions with the offline system are performed using the `taler-exchange-offline` tool. To use the offline system will require exchange operators to copy JSON files from or to the offline system (say using an USB stick). The offline system does not need any significant amount of computing power, a Raspberry-Pi is perfectly sufficient and the form-factor might be good for safe-keeping! (You should keep a copy of the (encrypted) private offline key on more than one physical medium though.)

Exchange operators are strongly advised to secure your private master key and any copies on encrypted, always-offline computers. Again, we assume that you are familiar with good best practices in operational security, including securing key material.

### 1.6 Online signing key security

To provide an additional level of protection for the private *online* signing keys used by the exchange, the actual cryptographic signing operations are performed by two helper processes, the `taler-exchange-secmod-rsa` and the `taler-exchange-secmod-EDDSA`.

The current implementation does not yet support the use of a hardware security module (HSM). If you have such a device with adequate functionality and are interested in Taler supporting it, please contact the developers for HSM integration support.
1.6.1 Functionality

The UNIX domain sockets have mode 0620 (u+rw, g+w). The exchange process MUST be in the same group as the crypto helper processes.

The two helper processes will create the required private keys, and allow anyone with access to the UNIX domain socket to sign arbitrary messages with the keys or to inform them about a key being revoked. The helper processes are also responsible for deleting the private keys if their validity period expires or if they are informed about a key having been revoked.

1.6.2 Security goals

From a security point of view, the helpers are designed to only make it harder for an attacker who took control of the HTTP daemon’s account to extract the private keys, limiting the attackers ability to creating signatures to the duration of their control of that account.

Note: In the future, the helper processes should additionally provide a mechanism to track the total number of signatures they have made for the various keys.

1.6.3 Setup

The helper processes should be run under a user ID that is separate from that of the user running the main taler-exchange-httpd service. For security, it is important that helpers run under a different user ID than the main HTTP frontend, in fact ideally each helper should run under its own user ID. The taler-exchange-httpd service’s will securely communicate with the helpers using UNIX domain sockets. To enable access to the keys, the service’s user must be in the group of the helper processes (and no other users should be in that group).

1.6.4 Configuration

The helpers and the HTTP service need both access to the same configuration information. Having divergent configurations may result in run-time failures. It is recommended that the configuration file (-c option) is simply shared between all of the different processes, even though they run as different system users. The configuration does not contain any sensitive information.
Before installing a Taler exchange, please make sure that your system does not have swap space enabled. Swap space is a security risk that Taler does not try to mitigate against.

Please install the following packages before proceeding with the exchange compilation.

- Python3 module jinja2
- “Sphinx RTD Theme” Python package aka python3-sphinx-rtd-theme on Debian-based systems (for GNUnet documentation support, can be omitted if GNUnet is configured with --disable-documentation)
- libsqlite3 >= 3.16.2
- GNU libunistring >= 0.9.3
- libcurl >= 7.26 (or libgnurl >= 7.26)
- libqrencode >= 4.0.0
- GNU libgcrypt >= 1.6
- libsodium >= 1.0
- libargon2 >= 20171227
- libjansson >= 2.7
- PostgreSQL >= 13, including libpq
- GNU libmicrohttpd >= 0.9.71
- GNUnet >= 0.16 (from source tarball)
- GNU Taler exchange (from download directory, see release announcement)

Except for the last two, these are available in most GNU/Linux distributions and should just be installed using the respective package manager.

2.1 Installing from source

The following instructions will show how to install libgnunetutil and the GNU Taler exchange from source.

Before you install GNUnet, you must download and install the dependencies mentioned in the previous section, otherwise the build may succeed, but could fail to export some of the tooling required by GNU Taler.

To install GNUnet, unpack the tarball and change into the resulting directory, then proceed as follows:
If you did not specify a prefix, GNUnet will install to /usr/local, which requires you to run the last step as root. The ldconfig command (also run as root) makes the shared object libraries (.so files) visible to the various installed programs.

After installing GNUnet, unpack the GNU Taler exchange tarball, change into the resulting directory, and proceed as follows:

```bash
$ ./configure [--prefix=EXCHANGEPFX] \n    [--with-gnunet=GNUNETPFX]
$ make
$ make install
# Each dependency can be fetched from non standard locations via 
# the '--with-<LIBNAME>' option. See './configure --help'.
```

If you did not specify a prefix, the exchange will install to /usr/local, which requires you to run the last step as root. You have to specify --with-gnunet=/usr/local if you installed GNUnet to /usr/local in the previous step.

Please note that unlike most packages, if you want to run the make check command, you should run it only after having done make install. The latter ensures that necessary binaries are copied to the right place.

Gratuitous editorial note by TTN: I think this is a quirk that we should fix in the long-term as such weirdness might hide other build issues. However, this is probably a minority viewpoint.

In any case, if make check fails, please consider filing a bug report with the Taler bug tracker.

### 2.2 Installing the GNU Taler binary packages on Debian

To install the GNU Taler Debian packages, first ensure that you have the right Debian distribution. At this time, the packages are built for Bullseye.

You need to add a file to import the GNU Taler packages. Typically, this is done by adding a file /etc/apt/sources.list.d/taler.list that looks like this:

```bash
deb https://deb.taler.net/apt/debian bullseye main
```

Next, you must import the Taler Systems SA public package signing key into your keyring and update the package lists:

```bash
# wget -O - https://taler.net/taler-systems.gpg.key | apt-key add -
# apt update
```

**Note:** You may want to verify the correctness of the Taler Systems key out-of-band.

Now your system is ready to install the official GNU Taler binary packages using apt.

To install the Taler exchange, you can now simply run:

```bash
$ ./configure [--prefix=GNUNETPFX]
$ make
$ make install
```

The ldconfig command (also run as root) makes the shared object libraries (.so files) visible to the various installed programs.
# apt install -t sid taler-exchange

Note that the package does not perform any configuration work except for setting up the various users and the systemd service scripts. You still must configure at least the database, HTTP reverse proxy (typically with TLS certificates), denomination and fee structure, bank account, auditor(s), offline signing and the terms of service.

Sample configuration files for the HTTP reverse proxy can be found in /etc/taler-exchange/.

## 2.3 Installing the GNU Taler binary packages on Trisquel

To install the GNU Taler Trisquel packages, first ensure that you have the right Trisquel distribution. Packages are currently available for Trisquel GNU/Linux 10.0. Simply follow the same instructions provided for Ubuntu 20.04 LTS (Focal Fossa).

## 2.4 Installing the GNU Taler binary packages on Ubuntu

To install the GNU Taler Ubuntu packages, first ensure that you have the right Ubuntu distribution. At this time, the packages are built for Ubuntu 22.04 LTS (Jammy Jellyfish).

A typical /etc/apt/sources.list.d/taler.list file for this setup would look like this:

```bash
deb https://deb.taler.net/apt/ubuntu/ jammy main
```

The last line is crucial, as it adds the GNU Taler packages.

Next, you must import the Taler Systems SA public package signing key into your keyring and update the package lists:

```bash
# wget -O /etc/apt/trusted.gpg.d/taler-systems.asc \
   https://taler.net/taler-systems.gpg.key
# apt update

Note: You may want to verify the correctness of the Taler Systems key out-of-band.
```

Now your system is ready to install the official GNU Taler binary packages using apt.

To install the Taler exchange, you can now simply run:

```bash
# apt install -t focal-fossa taler-exchange

Note that the package does not perform any configuration work except for setting up the various users and the systemd service scripts. You still must configure at least the database, HTTP reverse proxy (typically with TLS certificates), denomination and fee structure, bank account, auditor(s), offline signing and the terms of service.

Sample configuration files for the HTTP reverse proxy can be found in /etc/taler-exchange/.
```
This chapter provides an overview of the exchange configuration. Or at least eventually will do so, for now it is a somewhat wild description of some of the options.

### 3.1 Configuration format

In Taler realm, any component obeys to the same pattern to get configuration values. According to this pattern, once the component has been installed, the installation deploys default values in `${prefix}/share/taler/config.d/` in .conf files. In order to override these defaults, the user can write a custom .conf file and either pass it to the component at execution time, or name it `taler.conf` and place it under `$HOME/.config/`.

A config file is a text file containing sections, and each section contains its values. The right format follows:

```plaintext
[section1]
value1 = string
value2 = 23

[section2]
value21 = string
value22 = /path22
```

Throughout any configuration file, it is possible to use `$`-prefixed variables, like `$VAR`, especially when they represent filesystem paths. It is also possible to provide defaults values for those variables that are unset, by using the following syntax: `{VAR:-default}`. However, there are two ways a user can set `$`-prefixable variables:

- by defining them under a `[paths]` section, see example below,

  ```plaintext
  [paths]
  TALER_DEPLOYMENT_SHARED = ${HOME}/shared-data
  ...
  [section-x]
  path-x = ${TALER_DEPLOYMENT_SHARED}/x
  ```

- or by setting them in the environment:

  ```bash
  $ export VAR=/x
  ```

The configuration loader will give precedence to variables set under `[path]`, though.

The utility `taler-config`, which gets installed along with the exchange, serves to get and set configuration values without directly editing the .conf. The option `-f` is particularly useful to resolve pathnames, when they use several levels of `$`-expanded variables. See `taler-config --help`. 
Note that, in this stage of development, the file $HOME/.config/taler.conf can contain sections for all the component. For example, both an exchange and a bank can read values from it.

The repository git://taler.net/deployment contains examples of configuration file used in our demos. See under deployment/config.

Note

Expectably, some components will not work just by using default values, as their work is often interdependent. For example, a merchant needs to know an exchange URL, or a database name.

3.2 Using taler-config

The tool taler-config can be used to extract or manipulate configuration values; however, the configuration use the well-known INI file format and can also be edited by hand.

Run

```
$ taler-config -s $SECTION
```

to list all of the configuration values in section $SECTION.

Run

```
$ taler-config -s $section -o $option
```

to extract the respective configuration value for option $option in section $section.

Finally, to change a setting, run

```
$ taler-config -s $section -o $option -V $value
```

to set the respective configuration value to $value. Note that you have to manually restart the Taler backend after you change the configuration to make the new configuration go into effect.

Some default options will use $-variables, such as $DATADIR within their value. To expand the $DATADIR or other $-variables in the configuration, pass the -f option to taler-config. For example, compare:

```
$ taler-config --section exchange-offline --option MASTER_PRIV_FILE
$ taler-config -f --section exchange-offline --option MASTER_PRIV_FILE
```

While the configuration file is typically located at $HOME/.config/taler.conf, an alternative location can be specified to taler-merchant-httpd and taler-config using the -c option.

3.3 Keying

The exchange works with four types of keys:

- master key (kept offline)
  
  To create a master key, use:

  `$ taler-exchange-offline setup`

- sign keys (sign normal messages from the exchange)

- denomination keys (signs electronic coins, see section Coins)
• security module keys (signs sign keys and denomination keys)

Additionally, the exchange is sometimes concerned with the auditor’s public key (to verify messages signed by auditors approved by the exchange operator) and the merchant’s public key (to verify refunds are authorized by the merchant).

Key options include:
• [exchange-offline/MASTER_PRIV_FILE]: Path to the exchange’s master private file. Only needs to be provided on the offline system where the taler-exchange-offline command is used.
• [exchange/MASTER_PUBLIC_KEY]: Must specify the exchange’s master public key. Needed for the exchange to verify information signed by the offline system.

3.4 Serving

The exchange can serve HTTP over both TCP and UNIX domain socket.

The following options are to be configured in the section [exchange]:
• SERVE: Must be set to tcp to serve HTTP over TCP, or unix to serve HTTP over a UNIX domain socket.
• PORT: Set to the TCP port to listen on if SERVE is tcp.
• UNIXPATH: Set to the UNIX domain socket path to listen on if SERVE is unix.
• UNIXPATH_MODE: Number giving the mode with the access permission mask for the UNIXPATH (i.e. 660 = rw-rw-r--).

3.5 Currency

The exchange supports only one currency. This data is set under the respective option CURRENCY in section [taler].

3.6 Database

The option DB in section [exchange] gets the database backend’s name the exchange is going to use. So far, only db = postgres is supported. After choosing the backend, it is mandatory to supply the connection string (namely, the database name). This is possible in two ways:
• via an environment variable: TALER_EXCHANGEDB_POSTGRES_CONFIG.
• via configuration option CONFIG, under section [exchange-$BACKEND]. For example, the demo exchange is configured as follows:

```
[exchange]
... 
DB = postgres 
...

[exchange-db-postgres]
CONFIG = postgres:///talerdemo
```

Given this database configuration, the database can be initialized using:

```
$ taler-exchange-dbinit
```
Note that to run this command, the user must have CREATE TABLE, CREATE INDEX, ALTER TABLE and (in the future possibly even) DROP TABLE permissions. Those permissions are only required for this step (which may have to be repeated when upgrading a deployment). Afterwards, during normal operation, permissions to CREATE or ALTER tables are not required by any of the Taler exchange processes and thus should not be granted. For more information, see manpages/taler-exchange-dbinit.1.

Commands, like taler-exchange-dbinit, that support the -l LOGFILE command-line option, send logging output to standard error by default.

### 3.7 Coins (denomination keys)

Sections specifying denomination (coin) information start with coin_. By convention, the name continues with $CURRENCY_[$SUBUNIT]_VALUE_$REVISION, i.e. [coin_eur_ct_10_0] for a 10 cent piece. However, only the coin_ prefix is mandatory. Once configured, these configuration values must not change. The $REVISION part of the section name should be incremented if any of the coin attributes in the section changes. Each coin_ section must then have the following options:

- **VALUE**: How much is the coin worth, the format is CURRENCY:VALUE.FRACTION. For example, a 10 cent piece is “EUR:0.10”.
- **DURATION_WITHDRAW**: How long can a coin of this type be withdrawn? This limits the losses incurred by the exchange when a denomination key is compromised.
- **DURATION_SPEND**: How long is a coin of the given type valid? Smaller values result in lower storage costs for the exchange.
- **DURATION_LEGAL**: How long is the coin of the given type legal?
- **FEE_WITHDRAW**: What does it cost to withdraw this coin? Specified using the same format as value.
- **FEE_DEPOSIT**: What does it cost to deposit this coin? Specified using the same format as value.
- **FEE_REFRESH**: What does it cost to refresh this coin? Specified using the same format as value.
- **FEE_REFUND**: What does it cost to refund this coin? Specified using the same format as value.
- **CIPHER**: Which cipher to use for this coin? Must be either RSA or CS.
- **RSA_KEYSIZE**: How many bits should the RSA modulus (product of the two primes) have for this type of coin.

See manpages/taler.conf.5 for information on duration values (i.e. DURATION_WITHDRAW and DURATION_SPEND above, and OVERLAP_DURATION and DURATION below). Additionally, there are two global configuration options of note:

- [taler-exchange-secmod-rsa/OVERLAP_DURATION]: What is the overlap of the withdrawal timespan for denomination keys? The value given here must be smaller than any of the DURATION_WITHDRAW values for any of the coins.
- [taler-exchange-secmod-rsa/LOOKAHEAD_SIGN]: For how far into the future should denomination keys be pre-generated? This allows the exchange and auditor operators to download, offline-sign, and upload denomination key signatures for denomination keys that will be used in the future by the exchange.

**Note**: We recommend setting the LOOKAHEAD_SIGN value to at least one year and then to perform the offline-signing procedure at least once every 6 months to ensure that there is sufficient time for wallets to learn the new keys and to avoid unavailability in case this critical maintenance procedure is delayed.
Note: It is crucial that the configuration provided in these sections is identical (!) for the exchange and the crypto helpers. We recommend pointing both users to the same configuration file!

3.8 Sign keys

There are three global configuration options of note for sign keys:

- \([\text{taler-exchange-secmod-eddsa}/\text{DURATION}]\): How long are sign keys used to sign messages? After this time interval expires, a fresh sign key will be used (key rotation). We recommend using a \text{DURATION} of a few weeks to a few months for sign keys.

- \([\text{taler-exchange-secmod-eddsa}/\text{OVERLAP\_DURATION}]\): What is the overlap of the timespan for sign keys? We recommend a few minutes or hours. Must be smaller than \text{DURATION}.

- \([\text{taler-exchange-secmod-eddsa}/\text{LOOKAHEAD\_SIGN}]\): For how far into the future should sign keys be pre-generated? This allows the exchange and auditor operators to download, offline-sign, and upload sign key signatures for sign keys that will be used in the future by the exchange.

Note: We recommend setting the \text{LOOKAHEAD\_SIGN} value to at least one year and then to perform the offline-signing procedure at least once every 6 months to ensure that there is sufficient time for wallets to learn the new keys and to avoid unavailability in case this critical maintenance procedure is delayed.

3.9 Terms of Service

The exchange has an endpoint “/terms” to return the terms of service (in legal language) of the exchange operator. The wallet will show those terms of service to the user when the user is first withdrawing coins. Terms of service are optional for experimental deployments, if none are configured, the exchange will return a simple statement saying that there are no terms of service available.

To configure the terms of service response, there are two options in the \text{[exchange]} section:

- \text{TERMS\_ETAG}: The current “Etag” to return for the terms of service. This value must be changed whenever the terms of service are updated. A common value to use would be a version number. Note that if you change the \text{TERMS\_ETAG}, you MUST also provide the respective files in \text{TERMS\_DIR} (see below).

- \text{TERMS\_DIR}: The directory that contains the terms of service. The files in the directory must be readable to the exchange process.

The \text{TERMS\_DIR} directory structure must follow a particular layout. First, inside of \text{TERMS\_DIR}, there should be subdirectories using two-letter language codes like “en”, “de”, or “jp”. Each of these directories would then hold translations of the current terms of service into the respective language. Empty directories are permitted in case translations are not available.

Then, inside each language directory, files with the name of the value set as the \text{TERMS\_ETAG} must be provided. The extension of each of the files should be typical for the respective mime type. The set of supported mime types is currently hard-coded in the exchange, and includes HTML, PDF and TXT files. If other files are present, the exchange may show a warning on startup.
3.9.1 Example

A sample file structure for a TERMS_ETAG of “v1” would be:

- TERMS_DIR/en/v1.txt
- TERMS_DIR/en/v1.html
- TERMS_DIR/en/v1.pdf
- TERMS_DIR/de/v1.txt
- TERMS_DIR/de/v1.html
- TERMS_DIR/de/v1.pdf
- TERMS_DIR/fr/v1.pdf

If the user requests an HTML format with language preferences “fr” followed by “en”, the exchange would return TERMS_DIR/en/v1.html lacking an HTML version in French.

3.10 Bank account

To configure a bank account in Taler, we need to furnish two pieces of information:

- The payto:// URI of the bank account, which uniquely identifies the account. Examples for such URIs include payto://iban/CH9300762011623852957 for a bank account with an IBAN or payto://x-taler-bank/localhost:8080/2 for the second bank account at the Taler bank demonstrator running at localhost on port 8080. The first part of the URI following payto:// (“iban” or “x-taler-bank”) is called the wire method.

- The taler-exchange-wirewatch and taler-exchange-transfer tools need to be provided resources for authentication to the respective banking service. The format in which the authentication information is currently a username and password for HTTP basic authentication.

You can configure multiple accounts for an exchange by creating sections starting with “exchange-account-” for the section name. You can ENABLE for each account whether it should be used, and for what (incoming or outgoing wire transfers):

```plaintext
[exchange-account-1]
# With x-taler-bank (say for PyBank)
PAYTO_URI = "payto://x-taler-bank/bank.demo.taler.net/Exchange"

# Example using IBAN (for use with LibEuFin)
# PAYTO_URI = "payto://iban/CH9300762011623852957"

# URL for talking to the bank wire the wire API.
WIRE_GATEWAY_URL = https://bank.demo.taler.net/taler-wire-gateway/Exchange

# Use for exchange-aggregator (outgoing transfers)
ENABLE_DEBIT = YES
# Use for exchange-wirewatch (and listed in /wire)
ENABLE_CREDIT = YES

# Authentication options for exchange bank account go here.
# (Next sections have examples of authentication mechanisms)
WIRE_GATEWAY_AUTH_METHOD = basic
```

(continues on next page)
The command line tool `taler-exchange-offline` must be used to sign the `payto://` URI in a way suitable to convince wallets that this is the correct address to wire funds to. For example, the utility may be invoked as follows to enable a wire account:

```
$ taler-exchange-offline enable-account payto://iban/CH9300762011623852957
```

The resulting JSON output must be uploaded to the exchange using `taler-exchange-offline upload`. For details, see manpages/taler-exchange-offline.1.

### 3.10.1 Wire fee structure

For each wire method (“iban” or “x-taler-bank”) the exchange must know about applicable wire fees. This is also done using the `taler-exchange-offline` tool:

```
$ taler-exchange-offline wire-fee 2040 iban EUR:0.05 EUR:0.10 EUR:0.15
```

The above sets the wire fees for wire transfers involving iban accounts (in Euros) in the year 2040 to 5 cents (wire fee) and 10 cents (closing fee). The tool only supports setting fees that applies for the entire calendar year.

We recommend provisioning an exchange with wire fees at least for the next two years. Note that once the fees have been set for a year, they cannot be changed (basically, by signing the fees the exchange makes a legally binding offer to the customers).

**Note:** Provisioning future wire fees, like provisioning future denomination and signing keys, are key regular maintenance procedures for every exchange operator. We recommend setting automated reminders for this maintenance activity!

### 3.11 Auditor configuration

The exchange must be informed about any auditor that is expected to provision it with auditor signatures. This is also done using the `taler-exchange-offline` tool on the offline system. First, the auditor must be configured and provide the exchange operator with its public key and the URL of it’s REST API. The exchange operator also needs a human-readable name that may be shown to users to identify the auditor. Given this information, the exchange operator can enable the auditor:

```
$ taler-exchange-offline enable-auditor $PUB_KEY $REST_URL "$AUDITOR_NAME" > auditor.json
```

As before, the `auditor.json` file must then be copied from the offline system to a system connected to the exchange and there uploaded to the exchange.
This chapter describes how to deploy the exchange once it has been configured.

### 4.1 Launching an exchange

A running exchange requires starting the following processes:

- `taler-exchange-secmod-rsa` (as special user, sharing group with the HTTPD)
- `taler-exchange-secmod- eddsa` (as special user, sharing group with the HTTPD)
- `taler-exchange-httpd` (needs database access)
- `taler-exchange-aggregator` (only needs database access)
- `taler-exchange-closer` (only needs database access)
- `taler-exchange-wirewatch` (needs bank account read credentials and database access)
- `taler-exchange-transfer` (needs credentials to initiate outgoing wire transfers and database access)

The crypto helpers must be started before the `taler-exchange-httpd` and they should use the same configuration file.

For the most secure deployment, we recommend using separate users for each of these processes to minimize information disclosures should any of them be compromised. The helpers do not need access to the PostgreSQL database (and thus also should not have it).

The processes that require access to the bank account need to have a configuration file with the respective credentials in it. We recommend using a separate configuration at least for `taler-exchange-transfer` which is the only process that needs to know the credentials to execute outgoing wire transfers.

All of these processes should also be started via a hypervisor like `systemd` or `gnunet-arm` that automatically re-starts them should they have terminated unexpectedly. If the bank is down (say for maintenance), it is possible to halt the `taler-exchange-wirewatch` and/or `taler-exchange-transfer` processes (to avoid them making requests to the bank API that can only fail) without impacting other operations of the exchange. Naturally, incoming wire transfers will only be observed once `taler-exchange-wirewatch` is resumed, and merchants may complain if the disabled `taler-exchange-transfer` process causes payment deadlines to be missed.

**Note:** The `taler-exchange-httpd` does not ship with HTTPS enabled by default. In production, it should be run behind an HTTPS reverse proxy that performs TLS termination on the same system. Thus, it would typically be configured to listen on a UNIX domain socket. The `/management` and `/auditors` APIs do technically not have to be exposed on the Internet (only to the administrators running `taler-exchange-offline`) and should be blocked by the
reverse proxy for requests originating from outside of the bank. (However, this is not a strong security assumption: in principle having these endpoints available should do no harm. However, it increases the attack surface.)

### 4.2 Keys generation

Once the configuration is properly set up, all the keys can be signed using the offline key on the offline system by the tool `taler-exchange-offline`. To do this, one must first start the crypto helpers and the `taler-exchange-httpd` process (the tools for wire transfers may also be started, but do not have to run yet).

Next, the `future` key material should be downloaded using:

```
$ taler-exchange-offline download > future-keys.json
```

Afterwards, `future-keys.json` contains data about denomination and online signing keys that the exchange operator needs to sign with the offline tool. The file should be copied to the offline system. There, the operator should run:

```
$ taler-exchange-offline show < future-keys.json
```

and verify that the output contains the fee structure and key lifetimes they expect to see. They should also note the public keys being shown and communicate those to the *auditors* over a secure channel. Once they are convinced the file is acceptable, they should run:

```
$ taler-exchange-offline sign < future-keys.json > offline-sigs.json
```

The `offline-sigs.json` file must then be copied to an online system that is able to again communicate with the exchange. On that system, run:

```
$ taler-exchange-offline upload < offline-sigs.json
```

to provision the signatures to the exchange. At this point, the exchange will be able to use those keys, but wallets and merchants may not yet trust them! Thus, the next step is for the auditor to affirm that they are auditing this exchange. Details about this are described in taler-auditor-manual.

The simplistic (without using offline keys for the auditor) way to do this would be:

```
$ taler-auditor-offline download sign upload
```

For more information, see manpages/taler-auditor-offline.1.

### 4.3 Private key storage

Keeping the private keys the helpers create secret is paramount. If the private keys are lost, it is easy to provision fresh keys (with the help of the auditor). Thus, we recommend that the private keys of the crypto helpers are *not* backed up: in the rare event of a disk failure, they can be regenerated. However, we do recommend using RAID (1+1 or 1+1+1) for all disks of the system.
4.4 Database upgrades

Currently, there is no way to upgrade the database between Taler versions. The exchange database can be re-initialized using:

```
$ taler-exchange-dbinit -r
```

However, running this command will result in all data in the database being lost, which may result in significant financial liabilities as the exchange can then not detect double-spending. Hence this operation must not be performed in a production system.

4.4.1 Revocations

When an exchange goes out of business or detects that the private key of a denomination key pair has been compromised, it may revoke some or all of its denomination keys. At this point, the hashes of the revoked keys must be returned as part of the /keys response under “recoup”. Wallets detect this, and then return unspent coins of the respective denomination key using the /recoup API.

To revoke a denomination key, you need to know the hash of the denomination public key, $HDP. The $HDP value is usually included in the security report that is generated when a compromise is detected). Given this value, the key revocation can be approved on the offline system:

```
$ taler-exchange-offline revoke-denomination $HDP > revocation.json
```

The resulting revocation.json must be copied to a system connected to the exchange and uploaded to the exchange using the upload subcommand of taler-exchange-offline.

**Note:** Denomination key revocations should only happen under highly unusual (“emergency”) conditions and not in normal operation.
We recommend testing whether an exchange deployment is functional by using the Taler wallet command line interface. The tool can be used to withdraw and deposit electronic cash via the exchange without having to deploy and operate a separate merchant backend and storefront. For more information, see taler-wallet-cli-manual.
This chapter includes various sections on specific topics that might be helpful to understand how the exchange operates. The information may also be helpful for diagnostics.

### 6.1 Internal audits

While an exchange should use an external auditor to attest to regulators that it is operating correctly, an exchange operator can also use the auditor’s logic to perform internal checks. For this, an exchange operator can generally follow the auditor guide. However, instead of using `taler-auditor-sync`, an internal audit can and likely should be performed either directly against the production exchange database or against a synchronous copy created using standard database replication techniques. After all, the exchange operator runs this for diagnostics and can generally trust its own database to maintain the database invariants.

Running the auditor against the original production database (without using `taler-auditor-sync`) enables the auditing logic to perform a few additional checks that can detect inconsistencies. These checks are enabled by passing the `-i` option to the `taler-auditor` command. As always, the resulting report should be read carefully to see if there are any problems with the setup.

Reports are generally created incrementally, with `taler-auditor` reporting only incidents and balance changes that were not covered in previous reports. While it is possible to reset the auditor database and to restart the audit from the very beginning, this is generally not recommended as this may be too expensive.

### 6.2 Database Scheme

The exchange database must be initialized using `taler-exchange-dbinit`. This tool creates the tables required by the Taler exchange to operate. The tool also allows you to reset the Taler exchange database, which is useful for test cases but should never be used in production. Finally, `taler-exchange-dbinit` has a function to garbage collect a database, allowing administrators to purge records that are no longer required.

The database scheme used by the exchange looks as follows:
This chapter describes how to run the Taler exchange benchmark. The benchmark can be used to measure the performance of the exchange by running a (possibly large) number of simulated clients against one Taler deployment with a bank, exchange and auditor. For the bank, both a “fakebank” (-f) and a “Pythonbank” deployment are currently supported. The `taler-exchange-benchmark` program can launch all required services and clients, or only launch the parallel clients (-m), for example for distributed testing over a network.

For each parallel (-p) client, a number of reserves (-r) is first established by transferring money from a “user” account (42) to the Exchange’s account with the respective reserve public key as wire subject. Next, the client will withdraw a number of coins (-n) from the reserve and deposit them. Additionally, a fraction (-R) of the dirty coins will then be subject to refreshing. For some deposits, the auditor will receive deposit confirmations.

Operations that are not covered today include closing reserves, refunds and recoups.

The existing `benchmark.conf` file in `src/benchmark/` can be used as a starting point for a configuration to run the benchmark. The existing configuration file only requires that the `talercheck` database already exists and will launch all required services locally as needed.

You can run a first simple benchmark using:

```
$ createdb talercheck # if it does not yet exist
$ taler-exchange-dbinit -c benchmark.conf
$ taler-exchange-httpd -c benchmark.conf &
$ HTTPD_PID=$!
$ taler-exchange-offline -c benchmark.conf \
  download sign \
  enable-account payto://iban/CH9300762011623852957 \
  wire-fee iban EUR:0 EUR:0 EUR:0 \
  global-fee EUR:0 EUR:0 EUR:0 EUR:0 4w 4w 6y 4 \
  upload
$ kill -TERM $HTTPD_PID
$ taler-exchange-benchmark -c benchmark.conf -p 4 -r 1 -n 10
```

This will run 4 parallel clients withdrawing 10 coins from 1 reserve and then depositing those coins. The default refresh probability is 10 percent. Note that the tiny run should only take a few seconds, most of it will be spent in the setup of the original key material. For meaningful runs, all three values should likely be increased.

The output of `taler-exchange-benchmark` will include for each parallel client the total time spent in each of the major operations, possible repetitions (i.e. if the operation failed the first time), total execution time (operating system and user space) and other details.
Naturally, additional instrumentation (including using features of the PostgreSQL database itself) may help discover performance issues.
GNU FREE DOCUMENTATION LICENSE

Version 1.3, 3 November 2008
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

A.1 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, eithercommercially 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 Affero 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.

A.2 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.

The “publisher” means any person or entity that distributes copies of the Document to the public.

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.

### A.3 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.
A.4 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.

A.5 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.
A.6 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”.

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.
A.7 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.

A.8 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 in 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.

A.9 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.

A.10 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.
Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

A.11 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 https://www.gnu.org/licenses/.

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. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

A.12 11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

A.13 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 1.3 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 â€¦ Texts.” line with this:
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 Affero General Public License, to permit their use in free software.
INDEX

F
fee, 15

M
maintenance, 12, 15

W
wire fee, 15