# CONTENTS

## 1 Introduction
- 1.1 About GNU Taler ......................................................... 1
- 1.2 About this manual ....................................................... 1
- 1.3 Architecture overview ................................................ 1

## 2 Terminology
- 2.1 Instances ...................................................................... 3
- 2.2 Accounts ....................................................................... 3
- 2.3 Inventory ....................................................................... 3
- 2.4 Orders and Contracts ...................................................... 4
- 2.5 Transfers ....................................................................... 4
- 2.6 Tipping ......................................................................... 4
- 2.7 Reserves ....................................................................... 5

## 3 Installation
- 3.1 Generic instructions for installation from source ..................... 7
- 3.2 Installing the GNU Taler binary packages on Debian .................. 9
- 3.3 Installing the GNU Taler binary packages on Trisquel ................. 10
- 3.4 Installing the GNU Taler binary packages on Ubuntu ................. 10
- 3.5 Installing Taler on Debian GNU/Linux from source ............... 11

## 4 How to configure the merchant’s backend
- 4.1 Configuration format ....................................................... 13
- 4.2 Using taler-config .......................................................... 14
- 4.3 Backend options ............................................................ 15
- 4.4 Sample backend configuration .......................................... 18
- 4.5 Launching the backend .................................................... 19

## 5 Instance setup
- 5.1 KUDOS Accounts ........................................................... 21
- 5.2 IBAN Accounts ............................................................... 21
- 5.3 Setup ........................................................................... 22

## 6 Secure setup
- 6.1 Using UNIX domain sockets ............................................ 23
- 6.2 Reverse proxy configuration ............................................ 23
- 6.3 Access control ............................................................... 24
- 6.4 Status code remapping .................................................... 26

## 7 Customization
- 7.1 Templates ..................................................................... 29
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 manual targets system administrators who want to install a GNU Taler merchant backend.

We expect some moderate familiarity with the compilation and installation of Free Software packages. An understanding of cryptography is not required.

This first chapter of the manual will give a brief overview of the overall Taler architecture, describing the environment in which the Taler backend operates. The second chapter then explains how to install the software, including key dependencies. The third chapter will explain how to configure the backend, including in particular the configuration of the bank account details of the merchant.

The last chapter gives some additional information about advanced topics which will be useful for system administrators but are not necessary for operating a basic backend.

1.3 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. For testing purposes, Taler uses a special currency “KUDOS” and includes its own special bank.

The Taler software stack for a merchant consists of four main components:

- A frontend which interacts with the customer’s browser. The frontend enables the customer to build a shopping cart and place an order. Upon payment, it triggers the respective business logic to satisfy the order. This component is not included with Taler, but rather assumed to exist at the merchant. The Merchant API Tutorial gives an introduction for how to integrate Taler with Web shop frontends.
• A back-office application that enables the shop operators to view customer orders, match them to financial transfers, and possibly approve refunds if an order cannot be satisfied. This component is not included with Taler, but rather assumed to exist at the merchant. The Merchant Backend API provides the API specification that should be reviewed to integrate such a back-office with the Taler backend.

• A Taler-specific payment backend which makes it easy for the frontend to process financial transactions with Taler. This manual primarily describes how to install and configure this backend.

• A DBMS which stores the transaction history for the Taler backend. For now, the GNU Taler reference implementation only supports PostgreSQL, but the code could be easily extended to support another DBMS. Please review the PostgreSQL documentation for details on how to configure the database.

The following image illustrates the various interactions of these key components:

![Diagram of Taler components](image)

Basically, the backend provides the cryptographic protocol support, stores Taler-specific financial information in a DBMS and communicates with the GNU Taler exchange over the Internet. The frontend accesses the backend via a RESTful API. As a result, the frontend never has to directly communicate with the exchange, and also does not deal with sensitive data. In particular, the merchant’s signing keys and bank account information are encapsulated within the Taler backend.

A typical deployment will additionally include a full-blown Web server (like Apache or Nginx). Such a Web server would be responsible for TLS termination and access control to the /private/ and /management/ API endpoints of the merchant backend. Please carefully review the section on Secure setup before deploying a Taler merchant backend to production.
This chapter describes some of the key concepts used throughout the manual.

### 2.1 Instances

The backend allows the user to run multiple *instances* of shops with distinct business entities sharing a single backend. Each instance uses its own bank accounts and key for signing contracts. All major accounting functionality is separate per instance. What is shared is the database, HTTP(S) address and the main Taler configuration (accepted currency, exchanges and auditors).

### 2.2 Accounts

To receive payments, an instance must have configured one or more bank *accounts*. The backend does not have accounts for users, and instances are also not really ‘accounts’. So whenever we use the term *account*, it is about a bank account of a merchant.

### 2.3 Inventory

The Taler backend offers inventory management as an optional function. Inventory is tracked per instance and consists of *products* sold in *units*. Inventory can be finite or infinite (for digital products). Products may include previews (images) to be shown to the user and other meta-data. Inventory management allows the frontend to *lock* products, reserving them for a particular (unpaid) *order*. The backend can keep track of how many units of a product remain in stock and ensure that the number of units sold does not exceed the number of units in stock.

Inventory management is optional, and it is possible for the frontend to include products in orders that are not in the inventory, or to override prices of products in the inventory.
2.4 Orders and Contracts

In Taler, users pay merchants for orders. An order is first created by the merchant, where the merchant specifies the specific terms of the order.

After an order is created, it is claimed by a wallet. Once an order is claimed by a specific wallet, only that wallet will be able to pay for this order, to the exclusion of other wallets even if they see the same order URL. Sharing order URLs is explicitly allowed: if a user shares an order URL with another user, that other user should be given the opportunity to purchase the same product.

To prevent unauthorized wallets from claiming an order, merchants can specify that claims require authorization in the form of a claim token. This is useful in case the order ID is predictable (say because an existing order ID scheme from the merchant frontend is used) and at the same time malicious actors claiming orders is problematic (say because of limited stocks). The use of claim tokens is optional, but if a claim token is used, it must be provided to the wallet as part of the order URI.

Additionally, when stocks are limited, you can configure Taler to set a product lock on items (say, while composing the shopping cart). These locks, as well as the order lock (when the order is complete), can be configured to auto-unlock at certain times.

A wallet may pay for a claimed order, at which point the order turns into a (paid) contract. Orders have an expiration date after which the commercial offer expires and any stock of products locked by the order is released, allowing the stock to be sold in other orders.

Once a contract has been paid, the merchant should fulfill the contract. It is possible for the merchant to refund a contract order, for example if the contract cannot be fulfilled after all. Refunds are only possible after the customer paid and before the exchange has wired the payment to the merchant. Once the funds have been wired, refunds are no longer allowed by the Taler exchange. The wire deadline specifies the latest time by which an exchange must wire the funds, while the (earlier) refund deadline specifies the earliest time when an exchange may wire the funds.

Contract information is kept for legal reasons, typically to provide tax records in case of a tax audit. After the legal expiration (by default a decade), contract information is deleted.

2.5 Transfers

The Taler backend can be used to verify that the exchange correctly wired all of the funds to the merchant. However, the backend does not have access to the incoming wire transfers of the merchant’s bank account. Thus, merchants must manually provide the backend with wire transfer data that specifies the wire transfer subject and the amount that was received. Given this information, the backend can detect and report any irregularities that might arise.

2.6 Tipping

Taler does not only allow a Website to be paid, but also to make voluntary, non-contractual payments to visitors, called tips. Such tips could be granted as a reward for filling in surveys or watching advertizemens. For tips, there is no contract, tips are always voluntary actions by the Web site that do not arise from a contractual obligation. Before a Web site can create tips, it must establish a reserve. Once a reserve has been established, the merchant can grant tips, allowing wallets to pick up the tip.
2.7 Reserves

A reserve is a pool of electronic cash at an exchange under the control of a private key. Merchants withdraw coins from a reserve when granting tips. A reserve is established by first generating the required key material in the merchant backend, and then wiring the desired amount of funds to the exchange.

An exchange will automatically close a reserve after a fixed period of time (typically about a month), wiring any remaining funds back to the merchant.
This chapter describes how to install the GNU Taler merchant backend.

3.1 Generic instructions for installation from source

This section provides generic instructions for the merchant backend installation independent of any particular operating system. Operating system specific instructions are provided in the following sections. You should follow the operating system specific instructions if those are available, and only consult the generic instructions if no system-specific instructions are provided for your specific operating system.

3.1.1 Installation of dependencies

The following packages need to be installed before we can compile the backend:

- “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
- libargon2 >= 20171227
- json >= 2.7
- PostgreSQL >= 13, including libpq
- GNU libmicrohttpd >= 0.9.71
- GNUnet >= 0.16 (from source tarball)
- GNU Taler exchange (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. Be careful with GNU libmicrohttpd; here, some distributions only include an older version that will not work.

While you are in the GNU Taler exchange download directory, you might as well also download the tarball for GNU Taler merchant.
GNU Taler components version numbers follow the `MAJOR.MINOR.MICRO` format. The general rule for compatibility is that `MAJOR` and `MINOR` must match. Exceptions to this general rule are documented in the release notes. For example, Taler merchant 0.8.0 is compatible with Taler exchange 0.8.1.

The following sections will provide detailed instructions for installing the `libgnunetutil` and GNU Taler exchange dependencies.

### 3.1.2 Installing GNUnet

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:

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

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.

There is no need to actually run a GNUnet peer to use the Taler merchant backend – all the merchant needs from GNUnet is a number of headers and libraries!

### 3.1.3 Installing the GNU Taler exchange

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

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

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.

There is no need to actually run a Taler exchange to use the Taler merchant backend – all the merchant needs from the Taler exchange is a few headers and libraries!
3.1.4 Installing the GNU Taler merchant backend

GNU Taler merchant has these additional dependencies:

- libqrencode >= 4.0.0

The following steps assume all dependencies are installed.

First, unpack the GNU Taler merchant tarball and change into the resulting directory. Then, use the following commands to build and install the merchant backend:

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

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-exchange=/usr/local and/or --with-gnunet=/usr/local if you installed the exchange and/or GNUnet to /usr/local in the previous steps.

Depending on the prefixes you specified for the installation and the distribution you are using, you may have to edit /etc/ld.so.conf, adding lines for GNUNETPFX/lib/ and EXCHANGEPFX/lib/ and PFX/lib/ (replace the prefixes with the actual paths you used). Afterwards, you should run ldconfig. Without this step, it is possible that the linker may not find the installed libraries and launching the Taler merchant backend would then fail.

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.

3.2 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:

```
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:

```
# 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 merchant backend, you can now simply run:

```
# apt install taler-merchant
```

Note that the package does not complete the integration of the backend with the HTTP reverse proxy (typically with TLS certificates). A configuration fragment for Nginx or Apache will be placed in `/etc/{apache,nginx}/conf-available/taler-merchant.conf`. You must furthermore still configure the instances, and may need to extend the fragment with access control restrictions for non-default instances.

### 3.3 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).

### 3.4 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:

```
deby 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:

```
# 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 merchant backend, you can now simply run:

```
# apt install taler-merchant
```

Note that the package does not complete the integration of the backend with the HTTP reverse proxy (typically with TLS certificates). A configuration fragment for Nginx or Apache will be placed in `/etc/{apache,nginx}/conf-available/taler-merchant.conf`. You must furthermore still configure the instances, and may need to extend the fragment with access control restrictions for non-default instances.
3.5 Installing Taler on Debian GNU/Linux from source

Debian Wheezy is too old and lacks most of the packages required. Debian Jessie, Stretch, and Buster are better, but still lack PostgreSQL 12.

Note: When compiling PostgreSQL 12, make sure to do `make world` to build the `contrib/` modules, and `cd contrib` && `make install` to install them, as well.

On Debian Stretch and Buster, only GNU libmicrohttpd needs to be compiled from source. To install dependencies on Debian Stretch, run the following commands:

```bash
# apt-get install \
  libbreqrcode-dev \
  libsqlite3-dev \
  libltdl-dev \
  libunistring-dev \
  libodium-dev \
  libargon2-0-dev \
  libcurl4-gnutls-dev \
  libgcrypt20-dev \
  libjansson-dev \
  libpq-dev \
  postgresql-9.6
# wget https://ftpmirror.gnu.org/libmicrohttpd/libmicrohttpd-latest.tar.gz
# wget https://ftpmirror.gnu.org/libmicrohttpd/libmicrohttpd-latest.tar.gz.sig
# gpg -v libmicrohttpd-latest.tar.gz # Should show signed by 939E6BE1E29FC3CC
# tar xf libmicrohttpd-latest.tar.gz
# cd libmicrohttpd-0*
# ./configure
# make install
```

For more recent versions of Debian, you should instead run:

```bash
# apt-get install \
  libbreqrcode-dev \
  libsqlite3-dev \
  libltdl-dev \
  libunistring-dev \
  libodium-dev \
  libargon2-dev \
  libcurl4-gnutls-dev \
  libgcrypt20-dev \
  libjansson-dev \
  libpq-dev \
  postgresql-9.6 \
  libmicrohttpd-dev
```

Note that Stretch requires `libargon2-0-dev`, while later versions of Debian require `libargon2-dev`.

For the rest of the installation, follow the generic installation instructions starting with the installation of `libgnunetutil`. Note that if you used the Debian Stretch instructions above, you need to pass `--with-microhttpd=/usr/local/` to all `configure` invocations.
The installation already provides reasonable defaults for most of the configuration options. However, some must be provided, in particular the database account and bank account that the backend should use. By default, the file $HOME/.config/taler.conf is where the Web shop administrator specifies configuration values that augment or override the defaults. The format of the configuration file is the well-known INI file format. You can edit the file by hand, or use the taler-config commands given as examples.

### 4.1 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:

```
[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,

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

- or by setting them in the environment:

```
$ 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 components. 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.

### 4.2 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 -s ACCOUNT-bank \
  -o WIRE_RESPONSE
$ taler-config -f -s ACCOUNT-bank \
  -o WIRE_RESPONSE
```

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.
4.3 Backend options

The following table describes the options that commonly need to be modified. Here, the notation \[\$section\]/\$option denotes the option \$option under the section \$section in the configuration file.

4.3.1 Service address

The following option sets the transport layer address used by the merchant backend:

\[\text{[MERCHANT]/SERVE} = \text{TCP | UNIX}\]

If given,

- TCP, then we need to set the TCP port in \[\text{[MERCHANT]/PORT}\]
- UNIX, then we need to set the unix domain socket path and mode in \[\text{[MERCHANT]/UNIXPATH}\] and \[\text{[MERCHANT]/UNIXPATH_MODE}\]. The latter takes the usual permission mask given as a number, e.g. 660 for user/group read-write access.

The frontend can then connect to the backend over HTTP using the specified address. If frontend and backend run within the same operating system, the use of a UNIX domain socket is recommended to avoid accidentally exposing the backend to the network.

To run the Taler backend on TCP port 8888, use:

\[
\begin{align*}
\$ & \text{taler-config -s MERCHANT -o SERVE -V TCP} \\
\$ & \text{taler-config -s MERCHANT -o PORT -V 8888}
\end{align*}
\]

**Note:** When using the Debian/Ubuntu packages, these options are already configured in the /etc/taler/conf.d/merchant.conf configuration file.

If you need to change them, you should edit /etc/taler/merchant-overrides.conf, for example by passing \-c /etc/taler/merchant-overrides.conf\ to the taler-config commands above. By default, the Taler merchant package when installed on Debian/Ubuntu will use a UNIX domain socket at /run/taler/merchant-httpd/merchant-http.sock. For the best possible security, it is recommended to leave this in place and configure a reverse proxy (nginx or Apache) as described below.

4.3.2 Currency

Which currency the Web shop deals in, i.e. “EUR” or “USD”, is specified using the option

\[\text{[TALER]/CURRENCY}\]

For testing purposes, the currency MUST match “KUDOS” so that tests will work with the Taler demonstration exchange at https://exchange.demo.taler.net/:

\[
\$ \text{taler-config -s TALER -o CURRENCY -V KUDOS}
\]

**Note:** When using the Debian/Ubuntu packages, these options should be configured in the /etc/taler/taler.conf configuration file (alternatively, you can also edit /etc/taler/merchant-overrides.conf). However, you
must edit the `taler.conf` file manually and **must not** use `taler-config` to do this, as that would inline the include directives and destroy the carefully setup path structure.

### 4.3.3 Database

In principle it is possible for the backend to support different DBMSs. The option `[MERCHANT]/DB` specifies which DBMS is to be used. However, currently only the value `postgres` is supported. This is also the default.

In addition to selecting the DBMS software, the backend requires DBMS-specific options to access the database.

**Note:** When using the Debian/Ubuntu packages, the database should already be configured in the `/etc/taler/secrets/merchant-db.secret.conf` configuration file. The `talermerchant` database is also already configured (unless you answered `no` when asked the question during installation), so you can skip everything in this section.

For the `postgres` backend, you need to provide:

`[MERCHANTDB-postgres]/CONFIG`

This option specifies a `postgres` access path using the format `postgres:///$DBNAME`, where `$DBNAME` is the name of the PostgreSQL database you want to use. Suppose `$USER` is the name of the user who will run the backend process. Then, you need to first run:

```bash
$ sudo -u postgres createuser -d $USER
```

as the PostgreSQL database administrator (usually `postgres`) to grant `$USER` the ability to create new databases. Next, you should as `$USER` run:

```bash
$ createdb $DBNAME
```

to create the backend’s database. Here, `$DBNAME` must match the database name given in the configuration file.

To configure the Taler backend to use this database, run:

```bash
$ taler-config -s MERCHANTDB-postgres -o CONFIG \
   -V postgres:///$DBNAME
```

Now you should create the tables and indices. To do this, run as `$USER`:

```bash
$ taler-merchant-dbinit
```

You can improve your security posture if you now REVOKE the rights to CREATE, DROP or ALTER tables from `$USER`. However, if you do so, please be aware that you may have to temporarily GRANT those rights again when you update the merchant backend. For details on how to REVOKE or GRANT these rights, consult the PostgreSQL documentation.

Commands, like `taler-merchant-dbinit`, that support the `-l LOGFILE` command-line option, send logging output to standard error by default. See `manpages/taler-merchant-dbinit.1` for more information.

**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.
4.3.4 Exchange

To add an exchange to the list of trusted payment service providers, you create a section with a name that starts with “MERCHANT-EXCHANGE-“. In that section, the following options need to be configured:

- The EXCHANGE_BASE_URL option specifies the exchange’s base URL. For example, to use the Taler demonstrator, specify:

  ```
  $ taler-config -s MERCHANT-EXCHANGE-demo \
    -o EXCHANGE_BASE_URL \ 
    -V https://exchange.demo.taler.net/
  ```

- The MASTER_KEY option specifies the exchange’s master public key in base32 encoding. For the Taler demonstrator, use:

  ```
  $ taler-config -s MERCHANT-EXCHANGE-demo \
    -o MASTER_KEY \ 
    -V FH1Y8ZMXCTPQ0YFSZEC0H8C9407JR3YN0MF1706PTG24Q4NEWGV0
  ```

- The CURRENCY option specifies the exchange’s currency. For the Taler demonstrator, use:

  ```
  $ taler-config -s MERCHANT-EXCHANGE-demo \
    -o CURRENCY \ 
    -V KUDOS
  ```

Note that multiple exchanges can be added to the system by using different tokens in place of demo in the examples above. Note that all of the exchanges must use the same currency: If the currency does not match the main currency from the TALER section, the exchange is ignored. If you need to support multiple currencies, you need to configure a backend per currency.

Note: Manually setting up exchanges is only recommended under special circumstances. In general, GNU Taler will include trustworthy auditors (for each currency) in the default configuration, and there is rarely a good reason for trusting an exchange without an accredited auditor.

4.3.5 Auditor

To add an auditor to the list of trusted auditors (which implies that all exchanges audited by this auditor will be trusted!) you create a section with a name that starts with “MERCHANT-AUDITOR-“. In that section, the following options need to be configured:

- The AUDITOR_BASE_URL option specifies the auditor’s base URL. For example, to use the Taler demonstrator’s auditor, specify:

  ```
  $ taler-config -s MERCHANT-AUDITOR-demo \
    -o AUDITOR_BASE_URL \ 
    -V https://exchange.demo.taler.net/
  ```

- The AUDITOR_KEY option specifies the auditor’s public key in base32 encoding. For the Taler demonstrator, use:

  ```
  $ taler-config -s MERCHANT-AUDITOR-demo \
    -o AUDITOR_KEY \ 
    -V DSDASDXAMDAARMNAD53ZA44FHA2QADAMAHHASWDAWN84SDAA11
  ```

- The CURRENCY option specifies the auditor’s currency. For the Taler demonstrator, use:
Note that multiple auditors can be added to the system by using different tokens in place of demo in the examples above. Note that all of the auditors must use the same currency: If the currency does not match the main currency from the TALER section, the auditor is ignored. If you need to support multiple currencies, you need to configure a backend per currency.

Note: Manually adding auditors is only recommended under special circumstances. In general, GNU Taler will include trustworthy auditors (for each currency) in the default configuration, and there is rarely a good reason for adding an auditor that is not coordinating its activities with the Taler developers.

4.4 Sample backend configuration

The following is an example for a complete backend configuration:

```
[TALER]
CURRENCY = KUDOS

[MERCHANT]
SERVE = TCP
PORT = 8888
DATABASE = postgres

[MERCHANTDB-postgres]
CONFIG = postgres:///donations

[merchant-exchange-NAME]
EXCHANGE_BASE_URL = https://exchange.demo.taler.net/
MASTER_KEY = FH1Y8ZM8CFPQ0YFSZECDB8C9407JR3YN0MF1706PTG24Q4NEWGV0
# If currency does not match [TALER] section, the exchange
# will be ignored!
CURRENCY = KUDOS

[merchant-auditor-NAME]
AUDITOR_BASE_URL = https://auditor.demo.taler.net/
AUDITOR_KEY = DSDADSMADAMADMAN53ZAA4AF4AH2QADAMAHHASWDAW8N4SDA11
# If currency does not match [TALER] section, the auditor
# will be ignored!
CURRENCY = KUDOS
```

Given the above configuration, the backend will use a database named donations within PostgreSQL.

The backend will deposit the coins it receives to the exchange at https://exchange.demo.taler.net/, which has the master key FH1Y8ZM8CFPQ0YFSZECDB8C9407JR3YN0MF1706PTG24Q4NEWGV0.

Please note that doc/config.sh will walk you through all configuration steps, showing how to invoke taler-config for each of them.
4.5 Launching the backend

Assuming you have configured everything correctly, you can launch the merchant backend as $USER using

```
$ taler-merchant-httpd
```

To ensure the process runs always in the background and also after rebooting, you should use systemd, cron or some other init system of your operating system to launch the process. Consult the documentation of your operating system for how to start and stop daemons.

**Note:** When using the Debian/Ubuntu packages, the systemd configuration will already exist. You only need to enable and start the service using `systemctl enable taler-merchant-httpd` and `systemctl start taler-merchant-httpd`. Additionally, you should review the `/etc/apache2/sites-available/taler-merchant.conf` or `/etc/nginx/sites-available/taler-merchant` (these files contain additional instructions to follow), symlink it to `sites-enabled/` and restart your HTTP server. After that, you should be able to visit the merchant backend at the respective HTTP(S) endpoint.

If everything worked as expected, the command

```
$ wget -O - http://localhost:8888/config
```

should return some basic configuration status data about the service.

Please note that your backend is right now likely globally reachable. You can either:

- Use the `--auth=$TOKEN` command-line option to set an access token to be provided in an `Authorize: Bearer $TOKEN` HTTP header. Note that this can be used at anytime to override access control, but remains only in effect until a first instance is created or an existing instance authentication setting is modified.

- Set the `TALER_MERCHANT_TOKEN` environment variable to `$TOKEN` for the same effect. This method has the advantage of `$TOKEN` not being visible as a command-line interface to other local users on the same machine.

- Set up an instance with an authentication token before some unauthorized person has a chance to access the backend. As the backend is useless without any instance and the chances of remote attackers during the initial configuration is low, this is probably sufficient for most use-cases. Still, keep the first two scenarios in mind in case you ever forget your access token!

Production systems should additionally be configured to bind to a UNIX domain socket and use TLS for improved network privacy, see *Secure setup.*
Chapter 4. How to configure the merchant’s backend
First of all, we recommend the use of the single-page administration application that is served by default at the base URL of the merchant backend. You can use it to perform all steps described in this section (and more!), using a simple Web interface instead of the `wget` commands given below.

The first step for using the backend involves the creation of a default instance. The default instance can also create / delete / configure other instances, similar to the root account on UNIX. When no instance exists and `taler-merchant-httpd` was started without the `--auth` option, then the backend is reachable without any access control (unless you configured some in the reverse proxy).

The following documentation shows how to handle any instance. Thus, if you want to have multiple instances, you may need to perform the steps multiple times, once for each instance.

**Note:** A security concern is that normal API usage leaks instance existence. This means unauthorized users can distinguish between the case where the instance does not exist (HTTP 404) and the case where access is denied (HTTP 403). This is all moot behind a properly configured reverse proxy.

### 5.1 KUDOS Accounts

The main configuration data that must be provided for each instance is the bank account information.

In order to receive payments, the merchant backend needs to communicate bank account details to the exchange.

The bank account information is provided in the form of a `payto://`-URI. See RFC 8905 for the format of `payto://`-URIs.

For first tests, you should sign up for a KUDOS bank account at https://bank.demo.taler.net/. In this case, the `payto://`-URI will be of the form `payto://x-taler-bank/bank.demo.taler.net/$USERNAME` where `$USERNAME` must be replaced with the name of the account that was established at https://bank.demo.taler.net/.

### 5.2 IBAN Accounts

When deploying Taler with the real banking system, you primarily need to change the currency of the configuration from KUDOS to the actual currency (such as EUR, USD, CHF) and provide a `payto://`-URI of your real bank account. In Europe, this will involve knowing your IBAN number. If you have an IBAN, the corresponding `payto://`-URI is simply `payto://iban/$IBAN` where `$IBAN` must be replaced with the actual IBAN number.
5.3 Setup

With the knowledge of the payto://-URI, instances can be configured by POSTing a request to /management/instances. To create a first instance, create a file instance.json with an InstanceConfigurationMessage

```json
{
    "payto_uri": [ "$PAYTO_URI" ],
    "id": "default",
    "name": "example.com",
    "address": { "country": "zz" },
    "auth": { "method": "external" },
    "jurisdiction": { "country": "zz" },
    "default_max_wire_fee": "KUDOS:1",
    "default_wire_fee_amortization": 100,
    "default_max_deposit_fee": "KUDOS:1",
    "default_wire_transfer_delay": { "d_ms": 1209600000 },
    "default_pay_delay": { "d_ms": 1209600000 }
}
```

In the text above, you must replace $PAYTO_URI with your actual payto://-URI. Also, be sure to replace KUDOS with the fiat currency if the setup is for an actual bank. The name field will be shown as the name of your shop. The address field is expected to contain your shop’s physical address. The various defaults specify defaults for transaction fees your shop is willing to cover, how long offers made to the customer are valid, and how long the exchange has before it must wire the funds to your bank account. Those defaults can be modified for individual orders. For details, see the contract terms specification.

You can then create the instance using:

```bash
```

The base URL for the instance will then be http://localhost:8888/instances/default. You can create additional instances by changing the id value to identifies other than default.

Endpoints to modify (reconfigure), permanently disable (while keeping the data) or purge (deleting all associated data) instances exist as well and are documented in the Merchant Backend API documentation.
The Taler backend does not include even the most basic forms of access control or transport layer security. Thus, production setups must deploy the Taler backend behind an HTTP(S) server that acts as a reverse proxy, performs TLS termination and authentication and then forwards requests to the backend.

### 6.1 Using UNIX domain sockets

To ensure that the merchant backend is not exposed directly to the network, you should bind the backend to a UNIX domain socket:

```bash
$ taler-config -s MERCHANT -o SERVE -V UNIX
$ taler-config -s MERCHANT -o UNIXPATH -V /some/path/here.sock
```

Do not use a UNIX domain socket path in “/tmp”: systemd (or other init systems) may give Web servers a private “/tmp” thereby hiding UNIX domain sockets created by other users/processes in “/tmp”.

If UNIX domain sockets are for some reason not possible, you may use a host-based firewall to block access to the TCP port of the merchant backend, but this is not recommended. Relying on NAT or network firewalls for access control is gross negligence.

### 6.2 Reverse proxy configuration

#### 6.2.1 Nginx

For Nginx, a possible basic reverse proxy configuration would be:

```bash
proxy_pass http://unix:/some/path/here.sock;
proxy_redirect off;
proxy_set_header Host $host;
proxy_set_header X-Forwarded-Host "example.com";
proxy_set_header X-Forwarded-Proto "https";
```

Note that the above assumes your domain name is example.com and that you have TLS configured. Leave out the last line if your Nginx reverse proxy does not have HTTPS enabled. Make sure to restart the taler-merchant-httpd process after changing the SERVE configuration.
6.2.2 Apache

In Apache, make sure you have `mod_proxy`, `mod_proxy_http` and `mod_headers` enabled:

```
$ a2enmod proxy
$ a2enmod proxy_http
$ a2enmod headers
```

Then configure your Apache reverse proxy like this (you may change the endpoint):

```
<Location "/">
    ProxyPass "unix:/some/path/here.sock|http://example.com/"
    RequestHeader add "X-Forwarded-Proto" "https"
</Location>
```

Note that the above again assumes your domain name is `example.com` and that you have TLS configured. Note that you must add the `https` header unless your site is not available via TLS.

The above configurations are both incomplete. You must still additionally set up access control!

6.3 Access control

All endpoints with `/private/` in the URL must be restricted to authorized users of the respective instance. Specifically, the HTTP server must be configured to only allow access to `$BASE_URL/private/` and `$BASE_URL/management/` to the authorized users of the default instance, and to `$BASE_URL/instances/$ID/private/` to the authorized users of the instance `$ID`.

How access control is done (TLS client authentication, HTTP basic or digest authentication, etc.) is completely up to the merchant and does not matter to the Taler merchant backend.

Note that all of the other endpoints (without `/private/` or `/management/`) are expected to be fully exposed to the Internet, and wallets may have to interact with those endpoints directly without client authentication.

6.3.1 Nginx

For Nginx, you can implement token-based merchant backend authentication as follows:

```
location ~ /private/ {
    if ($http_authorization !~ "(?i)ApiKey SECURITYTOKEN") {
        return 401;
    }
    proxy_pass ...; # as above
}
location /management/ {
    if ($http_authorization !~ "(?i)ApiKey SECURITYTOKEN") {
        return 401;
    }
    proxy_pass ...; # as above
}
```

Here, `SECURITYTOKEN` should be replaced with the actual shared secret. Note that the `~` ensures that the above matches all endpoints that include the string `/private/`. If you only run a single instance, you could simply specify `/private/` without the `~` to only configure the access policy for the default instance.
If you are running different instances on the same backend, you likely will want to specify different access control tokens for each instance:

```apache
location ~ ^/instances/foo/private/ {
    if ($http_authorization !~ "(?i)ApiKey FOOTOKEN") {
        return 401;
    }
    proxy_pass ...; # as above
}

location ~ ^/instances/bar/private/ {
    if ($http_authorization !~ "(?i)ApiKey BARTOKEN") {
        return 401;
    }
    proxy_pass ...; # as above
}

location /private/ {
    if ($http_authorization !~ "(?i)ApiKey MASTERTOKEN") {
        return 401;
    }
    proxy_pass ...; # as above
}

location /management/ {
    if ($http_authorization !~ "(?i)ApiKey MASTERTOKEN") {
        return 401;
    }
    proxy_pass ...; # as above
}

location ~ /private/ {
    return 401; # access to instances not explicitly configured is forbidden
}
```

### 6.3.2 Apache

For Apache, you should first enable `mod_rewrite`:

```
$ a2enmod rewrite
```

Then, you can restrict to an access control token using:

```
<Location />
RewriteEngine On
RewriteCond "%(HTTP:AUTHORIZATION)" "!=SECURITYTOKEN"
RewriteRule "(.+)/private/" "-" [F]
RewriteRule "/management/" "-" [F]
ProxyPass "unix:/some/path/here.sock|http://example.com/"
</Location>
```

Here, `SECURITYTOKEN` should be replaced with the actual shared secret. Note that the `(.)` ensures that the above matches all endpoints that include the string `/private/`. If you only run a single instance, you could simply specify `/private/` without the `(.)` to only configure the access policy for the default instance.

If you are running different instances on the same backend, you likely will want to specify different access control tokens for each instance:
Please note that these are simply examples of how one could use Nginx or Apache2 for access control. Both HTTP servers support many other forms of authentication, including TLS client certificates, HTTP basic and digest authentication and others, which can all be used (possibly in combination) to restrict access to the internal API to authorized clients.

System administrators are strongly advised to test their access control setup before going into production!

### 6.4 Status code remapping

Normal API usage leaks instance existence information. Distinguishing between 404 (Not found) and 403 (Forbidden) is useful for diagnostics.

For higher security (by leaking less information), you can add the following fragment, which remaps all 404 response codes to 403.

#### 6.4.1 Nginx

```
error_page 404 =403 /empty.gif;
```
6.4.2 Apache

```
cond %{STATUS} =404
set-status 403
```
7.1 Templates

The installation process will install various HTML templates to be served to trigger the wallet interaction. You may change those templates to your own design. The templating language used is Mustach, and the templates are in the share/taler/merchant/templates/ directory.

7.2 Static files

The merchant backend also has the ability to serve small static files under the /static/{FILENAME} endpoint. This is used by the templating logic to load a CSS file, but you can also put other resources such as images or JavaScript.

7.3 Internationalization

Both templates and static files can be internationalized. This is done by having the language of the resource be a part of the filename. For templates the format is {BASENAME}.{LANGUAGE}.must. The language is mandatory for templates, the default language is English (en).

For static files, the format is {BASENAME}.{LANGUAGE}.{EXT} for internationalized files, and {BASENAME}.{EXT} for resources that do not support internationalization. The HTTP client will always request /static/{BASENAME}.{EXT}. If {BASENAME}.{EXT} exists, that resource is returned. Otherwise, an internationalized file based on the language preferences indicated by the browser is returned.

7.4 Limitations

All of the static files must fit into memory and it must be possible for the process to hold open file handles for all of these files. You may want to increase the ulimit of the taler-merchant-httpd process if you have templates for many languages.

The backend determines the MIME type based on the file’s extension. The list of supported extensions is hard-coded and includes common text and image formats.

The current backend only provides a limited set of variables for the Mustach template expansion, and does not make use of scopes and other Mustach features.
This is the general upgrade procedure. Please see the release notes for your specific version to check if a particular release has special upgrade requirements.

Please note that upgrades are ONLY supported for released version of the merchant. Attempting to upgrade from or to a version in Git is not supported and may result in subtle data loss.

To safely upgrade the merchant, you should first stop the existing `taler-merchant-httpd` process, backup your merchant database (see PostgreSQL manual), and then install the latest version of the code.

If you REVOKED database permissions, ensure that the rights to CREATE, DROP, and ALTER tables are GRANTed to `$USER` again. Then, run:

```
$ taler-merchant-dbinit
```

to upgrade the database to the latest schema. After that, you may again REVOKE the database permissions. Finally, restart the HTTP service, either via your systemd or init system, or directly using:

```
$ taler-merchant-httpd
```
Taler can also be used to tip Web site visitors. For example, you may be running an online survey, and you want to reward those people that have dutifully completed the survey. If they have installed a Taler wallet, you can provide them with a tip for their deeds. This section describes how to setup the Taler merchant backend for tipping.

There are three basic steps that must happen to tip a visitor.

### 9.1 Fund the reserve

First, the reserve must be setup in the merchant backend. A reserve is always tied to a particular instance. To create a reserve with 10 KUDOS at instance `default` using the demo exchange, use:

```bash
$ taler-merchant-setup-reserve
  -a KUDOS:10
  -e https://exchange.demo.taler.net/
  -m http://localhost:8888/instances/default
```

The above command assumes that the merchant runs on localhost on port 8888. For more information, including how to transmit authentication information to the backend, see manpages/taler-merchant-setup-reserve.1.

The command will output a `payto://` URI which specifies where to wire the funds and which wire transfer subject to use.

**FIXME:** add full example output.

In our example, the output for the wire transfer subject is:

```bash
QPE24X8PBX3BZ6E7GQ5VAVHV32FWTTCDRR0TRQ183MSS3D2CHNEG
```

You now need to make a wire transfer to the exchange’s bank account using the given wire transfer subject.

Make your wire transfer and (optionally) check at “https://exchange/reserves/QPE24X…” whether your transfer has arrived at the exchange.

Once the funds have arrived, you can start to use the reserve for tipping.

Note that an exchange will typically close a reserve after four weeks, wiring all remaining funds back to the sender’s account. Thus, you should plan to wire funds corresponding to a campaign of about two weeks to the exchange initially. If your campaign runs longer, you should setup another reserve every other week to ensure one is always ready.

33
9.2 Authorize a tip

When your frontend has reached the point where a client is supposed to receive a tip, it needs to first authorize the tip. For this, the frontend must use a POST to /private/reserves/$RESERVE_PUB/authorize-tip. To authorize a tip, the frontend has to provide the following information in the body of the POST request:

- The amount of the tip
- The justification (only used internally for the back-office)
- The URL where the wallet should navigate next after the tip was processed
- The tip-pickup URL (see next section)

In response to this request, the backend will return a tip token, an expiration time and the exchange URL. The expiration time will indicate how long the tip is valid (when the reserve expires). The tip token is an opaque string that contains all the information needed by the wallet to process the tip. The frontend must send this tip token to the browser in a special “402 Payment Required” response inside the X-Taler-Tip header.

The frontend should handle errors returned by the backend, such as misconfigured instances or a lack of remaining funds for tipping.

9.3 Picking up of the tip

The wallet will POST a JSON object to the shop’s /tips/$TIP_ID/pickup handler. The frontend must then forward this request to the backend. The response generated by the backend can then be forwarded directly to the wallet.
10.1 Database Scheme

The merchant database must be initialized using `taler-merchant-dbinit`. This tool creates the tables required by the Taler merchant to operate. The tool also allows you to reset the Taler merchant database, which is useful for test cases but should never be used in production. Finally, `taler-merchant-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 merchant looks as follows:
10.2 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:

```
[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,

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

or by setting them in the environment:

```
$ 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 components. For example, both an exchange and a bank can read values from it.

The deployment repository 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.
10.2.1 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 -s PATHS \
    -o TALER_DATA_HOME
$ taler-config -f -s PATHS \
    -o TALER_DATA_HOME
```

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.
This section describes features that most merchants will not need, or will not need initially.

11.1 Benchmarking

The merchant codebase offers the `taler-merchant-benchmark` tool to populate the database with fake payments. This tool is in charge of starting a merchant, exchange, and bank processes, and provides them all the input to accomplish payments. Note that each component will use its own configuration (as they would do in production).

The main goal of the benchmarking tool is to serve as a starting point (!) for merchants that are interested in developing stress tests to see how far their infrastructure can scale.

The current tool has already a few options, but we expect that to deliver relevant results it will need to be customized to better reflect the workload of a particular merchant. This customization would at this point likely involve writing (C) code. We welcome contributions to make it easier to customize the benchmark and/or to cover more realistic workloads from the start.

11.2 Benchmark setup

The `taler-merchant-benchmark` tool will automatically launch and configure the exchange, (Python) bank and other tools required for the benchmark. However, the configuration file must be provided and have consistent options set. The options that require special care include the exchange’s public key (which must match the private key in the file specified by the configuration), the currency (which must be consistent across the file), the denomination structure (which must enable payments in the range of 100ths of the unit currency (often called cents)). Furthermore, the benchmark will set the Exchange bank account password to be “x”, so the configuration must also specify “x” for the passphrase. Finally, the bank must be configured to allow for substantial debt least the transactions by the benchmark run out of digital cash.

A relatively minimal configuration could look like this:

```
[PATHS]
# Persistent data storage for the benchmark
TALER_TEST_HOME = benchmark_home/

[taler]
# If you change the currency here, you MUST change it
# throughout the file.
CURRENCY = EUR
CURRENCY_ROUND_UNIT = EUR:0.01

[merchant]
```

(continues on next page)
SERVE = tcp
PORT = 8080
DB = postgres

[merchantdb-postgres]
CONFIG = postgres:///talercheck

[exchange]
DB = postgres
SERVE = tcp
PORT = 8081
BASE_URL = http://localhost:8081/
MASTER_PUBLIC_KEY = T1VVFQZZARQ1CMF4BN58EE7SKTW5AV2BS18S87ZEGYS4S29J6DNG

[exchangedb-postgres]
CONFIG = postgres:///talercheck

[auditor]
DB = postgres
SERVE = tcp
PORT = 8083
BASE_URL = http://the.auditor/

[auditordb-postgres]
CONFIG = postgres:///talercheck

[bank]
DATABASE = postgres:///talerbank
SERVE = http
HTTP_PORT = 8082
MAX_DEBT = EUR:5000.0
MAX_DEBT_BANK = EUR:0.0

[merchant-exchange-test]
MASTER_KEY = T1VVFQZZARQ1CMF4BN58EE7SKTW5AV2BS18S87ZEGYS4S29J6DNG
EXCHANGE_BASE_URL = http://localhost:8081/
CURRENCY = EUR

[exchange-account-exchange]
# The account name MUST be 'Exchange'
PAYTO_URI = payto://x-taler-bank=localhost/Exchange
WIRE_RESPONSE = ${TALER_CONFIG_HOME}/exchange/account.json
WIRE_GATEWAY_AUTH_METHOD = basic
USERNAME = Exchange
# The password MUST be 'x'
PASSWORD = x
ENABLE_DEBIT = YES
ENABLE_CREDIT = YES

[fees-x-taler-bank]
WIRE-FEE-2020 = EUR:0.01
WIRE-FEE-2021 = EUR:0.01
WIRE-FEE-2022 = EUR:0.01
WIRE-FEE-2023 = EUR:0.01
WIRE-FEE-2024 = EUR:0.01
WIRE-FEE-2025 = EUR:0.01
WIRE-FEE-2026 = EUR:0.01
WIRE-FEE-2027 = EUR:0.01
CLOSING-FEE-2020 = EUR:0.01
CLOSING-FEE-2021 = EUR:0.01
CLOSING-FEE-2022 = EUR:0.01
CLOSING-FEE-2023 = EUR:0.01
CLOSING-FEE-2024 = EUR:0.01
CLOSING-FEE-2025 = EUR:0.01
CLOSING-FEE-2026 = EUR:0.01
CLOSING-FEE-2027 = EUR:0.01

[coin_eur_ct_1]
value = EUR:0.01
duration_withdraw = 7 days
duration_spend = 2 years
duration_legal = 3 years
fee_withdraw = EUR:0.00
fee_deposit = EUR:0.00
fee_refresh = EUR:0.01
fee_refund = EUR:0.01
rsa_keysize = 1024

[coin_eur_ct_10]
value = EUR:0.10
duration_withdraw = 7 days
duration_spend = 2 years
duration_legal = 3 years
fee_withdraw = EUR:0.01
fee_deposit = EUR:0.01
fee_refresh = EUR:0.03
fee_refund = EUR:0.01
rsa_keysize = 1024

[coin_eur_1]
value = EUR:1
duration_withdraw = 7 days
duration_spend = 2 years
duration_legal = 3 years
fee_withdraw = EUR:0.01
fee_deposit = EUR:0.01
fee_refresh = EUR:0.03
fee_refund = EUR:0.01
rsa_keysize = 1024

[coin_eur_5]
value = EUR:5
duration_withdraw = 7 days
Note that the public key must match the exchange’s private key and that the PostgreSQL database must exist before launching the benchmark. You also will need to ensure that the Exchange’s details are set up. For details, see the Exchange Operator Manual.

### 11.3 Running the benchmark command

The tool takes all of the values it needs from the command line, with one of them being mandatory:

- `--exchange-account=SECTION` Specifies which configuration section specifies the bank account for the exchange that should be used for the benchmark. For the example configuration above, the SECTION value provided must be `exchange-account-exchange`.

The tool comes with two operation modes: *ordinary*, and *corner*. The first just executes normal payments, meaning that it uses the default instance and make sure that all payments get aggregated. The second gives the chance to leave some payments unaggregated, and also to use merchant instances other than the default (which is, actually, the one used by default by the tool).

Note: the ability of driving the aggregation policy is useful for testing the back-office facility.

Any subcommand is also equipped with the canonical `--help` option, so feel free to issue the following command in order to explore all the possibilities. For example:

```bash
$ taler-merchant-benchmark corner --help
```

will show all the options offered by the *corner* mode. Among the most interesting, there are:

- `--two-coins=TC` This option instructs the tool to perform *TC* many payments that use two coins, because normally only one coin is spent per payment.

- `--unaggregated-number=UN` This option instructs the tool to perform *UN* (one coin) payments that will be left unaggregated.

As for the *ordinary* subcommand, it is worth explaining the following options:

- `--payments-number=PN` Instructs the tool to perform *PN* payments.

- `--tracks-number=TN` Instructs the tool to perform *TN* tracking operations. Note that the total amount of operations will be two times *TN*, since “one” tracking operation accounts for `track/transaction` and `track/transfer`. This command should only be used to see if the operation ends without problems, as no actual measurement of performance is provided (despite of the ‘benchmark’ word used in the tool’s name).
12.1 Installing Taler using Docker

This section provides instructions for the merchant backend installation using ‘Docker’.

For security reasons, we run Docker against a VirtualBox instance, so the `docker` command should connect to a `docker-machine` instance that uses the VirtualBox driver.

Therefore, the needed tools are: “docker“, “docker-machine“ and “docker-compose“. Please refer to Docker’s official\(^1\) documentation in order to get those components installed, as that is not in this manual’s scope.

Before starting to build the merchant’s image, make sure a “docker-machine“ instance is up and running.

Because all of the Docker source file are kept in our “deployment” repository, we start by checking out the git://
git.taler.net/deployment codebase:

```
$ git clone git://git.taler.net/deployment
```

Now we actually build the merchant’s image. From the same directory as above:

```
$ cd deployment/docker/merchant/
$ docker-compose build
```

If everything worked as expected, the merchant is ready to be launched. From the same directory as the previous step:

```
# Recall: the docker-machine should be up and running.
$ docker-compose up
```

You should see some live logging from all the involved containers. At this stage of development, you should also ignore some (harmless) error message from postgresql about already existing roles and databases.

To test if everything worked as expected, it suffices to issue a simple request to the merchant, for example:

```
$ wget -O - http://$(docker-machine ip)/
# A greeting message should be returned by the merchant.
```

\(^1\) https://docs.docker.com/
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, 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 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.
• 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 partiesâ€”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.

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”.

48 Appendix A. GNU Free Documentation License
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:
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 Affero General Public License, to permit their use in free
software.
INDEX

A
account, 3

B
back-office, 1
backend, 1, 9, 19
Bullseye, 11
Buster, 11

C
claim, 4
close, 5
configuration, 18, 37
contract, 4
crypto-currency, 1
currency, 15

D
DBMS, 1, 15
Debian, 11

E
exchange, 8, 15

F
frontend, 1

G
GNUnet, 8
grant, 4

I
instance, 3, 15, 19
inventory, 3

J
Jessie, 11

K
KUDOS, 1, 15

L
legal expiration, 4
lock, 3, 4

O
order, 3, 4

P
pay, 4
pick up, 4
port, 15
PostgreSQL, 1, 15
product, 3

R
refund, 4
reserve, 5, 33
RESTful, 2

S
security, 23
Stretch, 11

T
taler.conf, 13
taler-config, 13, 38
taler-merchant-httpd, 19
TCP, 15
tip, 4
tipping, 33
TLS, 23
transfer, 4

U
unit, 3
UNIX domain socket, 15

W
Wheezy, 11
wire deadline, 4
wire format, 15
wire transfer, 4