GNU Taler Exchange Manual

Release 0.10.0

GNU Taler team

May 08, 2024

CONTENTS

1 Introduction **3**
- 1.1 About GNU Taler . . . 3
- 1.2 About this manual . . . 3
- 1.3 Organizational prerequisites . . . 3
- 1.4 Architecture overview . . . 4
- 1.5 Key Types . . . 5
- 1.6 Offline keys . . . 5
- 1.7 Online signing key security . . . 6

2 Installation **9**
- 2.1 Before you start . . . 9
- 2.2 Installing from source . . . 9
- 2.3 Installing the GNU Taler binary packages on Debian . . . 11
- 2.4 Installing the GNU Taler binary packages on Trisquel . . . 12
- 2.5 Installing the GNU Taler binary packages on Ubuntu . . . 12
- 2.6 Services, users, groups and file system hierarchy . . . 13

3 Configuration Fundamentals **15**
- 3.1 Configuration format . . . 15

4 Exchange Database Setup **17**

5 Basic Setup: Currency, Denominations and Keys **19**
- 5.1 Coins (denomination keys) . . . 19
- 5.2 Sign keys . . . 20
- 5.3 Setting up the offline signing key . . . 21

6 Wire Gateway Setup **23**
- 6.1 Exchange Bank Account Configuration . . . 23

7 Legal Setup **25**
- 7.1 Legal conditions for using the service . . . 25
- 7.2 Terms of Service . . . 25
- 7.3 Privacy Policy . . . 25
- 7.4 Legal policies directory layout . . . 26
- 7.5 Generating the Legal Terms . . . 26
- 7.6 Adding translations . . . 27
- 7.7 Updating legal documents . . . 27

8 KYC Configuration **29**
- 8.1 Taler KYC Terminology . . . 31

8.2 KYC Configuration Options . . . 31
8.3 OAuth 2.0 specifics . . . 32
8.4 Persona specifics . . . 33
8.5 KYC AID specifics . . . 34

9 Deployment **37**
9.1 Serving . . . 37
9.2 Reverse Proxy Setup . . . 37
9.3 Launching an exchange . . . 38

10 Offline Signing Setup, Key Maintenance and Tear-Down **41**
10.1 Signing the online signing keys . . . 42
10.2 Account signing . . . 42
10.3 Wire fee structure . . . 42
10.4 Auditor configuration . . . 43
10.5 Revocations . . . 43

11 AML Configuration **45**
11.1 AML Officer Setup . . . 45
11.2 AML Triggers . . . 46
11.3 AML Forms . . . 46
11.4 AML Programs . . . 47
11.5 AML Measures . . . 49

12 Setup Linting **51**

13 Testing and Troubleshooting **53**
13.1 taler-config . . . 54
13.2 Using taler-config . . . 54
13.3 Private key storage . . . 55
13.4 Internal audits . . . 55
13.5 Database Scheme . . . 55
13.6 Database upgrades . . . 57

14 Template Customization **59**
14.1 Generic Errors Templates . . . 59
14.2 kycaid-invalid-request . . . 60
14.3 oauth2-authentication-failure . . . 60
14.4 oauth2-authorization-failure . . . 60
14.5 oauth2-authorization-failure-malformed . . . 60
14.6 oauth2-bad-request . . . 61
14.7 oauth2-conversion-failure . . . 61
14.8 oauth2-provider-failure . . . 61
14.9 persona-exchange-unauthorized . . . 61
14.10 persona-load-failure . . . 62
14.11 persona-exchange-unpaid . . . 62
14.12 persona-logic-failure . . . 62
14.13 persona-invalid-response . . . 62
14.14 persona-network-timeout . . . 63
14.15 persona-kyc-failed . . . 63
14.16 persona-provider-failure . . . 63

15 Benchmarking **65**
15.1 Choosing a bank . . . 65
15.2 taler-bank-benchmark . . . 66

15.3 taler-exchange-benchmark . . . 66
15.4 taler-aggregator-benchmark . . . 67

16 FIXMEs **69**

A GNU Free Documentation License **71**
A.1 0. PREAMBLE . . . 71
A.2 1. APPLICABILITY AND DEFINITIONS . . . 71
A.3 2. VERBATIM COPYING . . . 72
A.4 3. COPYING IN QUANTITY . . . 73
A.5 4. MODIFICATIONS . . . 73
A.6 5. COMBINING DOCUMENTS . . . 74
A.7 6. COLLECTIONS OF DOCUMENTS . . . 75
A.8 7. AGGREGATION WITH INDEPENDENT WORKS . . . 75
A.9 8. TRANSLATION . . . 75
A.10 9. TERMINATION . . . 75
A.11 10. FUTURE REVISIONS OF THIS LICENSE . . . 76
A.12 11. RELICENSING . . . 76
A.13 ADDENDUM: How to use this License for your documents . . . 76

Index **79**

Table of Contents

- *Introduction*
- *Installation*
- *Configuration Fundamentals*
- *Exchange Database Setup*
- *Basic Setup: Currency, Denominations and Keys*
- *Wire Gateway Setup*
- *Legal Setup*
- *KYC Configuration*
- *Deployment*
- *Offline Signing Setup, Key Maintenance and Tear-Down*
- *AML Configuration*
- *Setup Linting*
- *Testing and Troubleshooting*
- *Template Customization*
- *Benchmarking*
- *FIXMEs*

CHAPTER

ONE

INTRODUCTION

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).

1.2 About this manual

This manual 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. Exceptions may apply, especially if you are operating a regional currency or a payment system for an event with a closed user group.

GNU Taler payment service providers generally need to ensure high availability and should 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 (sufficiently 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: Taler may store sensitive business and customer data in the database. Any operator SHOULD thus ensure that backup operations are encrypted and secured from unauthorized access.

1.4 Architecture overview

GNU Taler is a pure payment system, not a crypto-currency. As such, it operates in a traditional banking context. In particular, this means that payments can be executed in ordinary currencies such as USD or EUR. Furthermore, a typical merchant in Taler has a regular bank account, and would use it to receive funds via Taler.

Consequently, a typical Taler exchange must interact with a bank. The bank of the exchange holds funds in an account where the balance is basically equivalent to the value of all coins in circulation. (Small mismatches arise whenever customers are about to withdraw coins and have already send the funds into the bank account, or if merchants just deposited coins and are about to receive wire transfers for deposited coins, or due to fees charged by the exchange and the operator not yet having drained the fees from the account.)

The exchange uses an intermediary system to talk to its bank. This shifts the technical burden (XML-based communications, additional cryptography, and a vast variety of standards) for this interaction into another bank-specific subsystem. 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 exchange's bank account, the Taler Wire Gateway API must notify 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`, `taler-exchange-secmod-cs` and `taler-exchange-secmod-eddsa` are three 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. Each wire adapter must implement the Taler Wire Gateway API. Three wire adapters are currently provided:

  (1) The **libtalerfakebank** implements a bank with a wire adapter API inside of a testcase. `taler-fakebank-run` is a stand-alone process using libtalerfakebank. Note that this adapter is only useful for tests, as all transaction data is kept in memory.

  (2) For production, **libeufin**'s `libeufin-nexus` component implements a wire adapter towards the traditional SEPA banking system with IBAN accounts using the EBICS protocol.

  (3) To use GNU Taler with blockchains, the **Depolymerization** component provides a wire gateway API that runs on top of blockchains like Bitcoin and Ethereum.

  The client-side wire adapter API is implemented in **libtalerbank** and is used by `taler-exchange-transfer` to execute wire transfers and by `taler-exchange-wirewatch` and the Taler auditor auditor to query bank transaction histories.

- **DBMS**: The exchange requires a DBMS to stores 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 exchange's database.

1.5 Key Types

The exchange works with four types of keys:

- master key (kept offline, configured manually at merchants and wallets)
- online message signing keys (signs normal messages from the exchange)
- denomination keys (signs digital coins)
- security module keys (signs online message signing 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).

Most of the keys are managed fully automatically or configured as part of the denomination configuration. Some configuration settings must be manually set with regards to the exchange's master key.

1.6 Offline keys

The exchange (and ideally also its auditor(s)) uses a long-term offline master siging 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 their private master key and any copies on encrypted, always-offline computers. Again, this manual assumes that you are familiar with good best practices in operational security, including securing key material.

1.7 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 three helper processes, `taler-exchange-secmod-rsa`, `taler-exchange-secmod-cs` and `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.

1.7.1 Functionality

The UNIX domain sockets of the *secmod* helpers have mode 0620 (u+rw, g+w). The exchange process MUST thus be in the same group as the crypto helper processes to enable access to the keys. No other users should be in that group!

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.

1.7.2 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.

1.7.3 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. To get any security benefit from this, 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.

1.7.4 Configuration

The helpers and the exchange 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.

CHAPTER

TWO

INSTALLATION

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.

We recommend the setup of offline signing keys to be done on a second machine that does not have Internet access.

In this guide's shell-session fragments, the command prompt shows two pieces of information:

- Who is performing the command (`$user` vs `root`, and ending character `$` vs `#`).
- Host where the command is supposed to be executed (`exchange-offline` vs `exchange-online`). It is possible to do the entire setup on one machine, but we do not recommend this for security reasons.

2.1 Before you start

To deploy this with a real bank, you need:

- IBAN of the bank account to use
- BIC of the bank
- EBICS host, user and partner IDs

Information to write down during the installation:

- LibEuFin Nexus superuser password
- Taler facade base URL
- exchange Nexus username and password

2.2 Installing from source

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

The package sources can be find in our download directory.

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 1.3.0 should be compatible with Taler exchange 1.4.x as the MAJOR version matches. A MAJOR version of 0 indicates experimental development, and you are expected to always run all of the *latest* releases together (no compatibility guarantees).

First, 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 (Taler merchant only)
- GNU libgcrypt >= 1.6 (1.10 or later highly recommended)
- libsodium >= 1.0
- libargon2 >= 20171227
- libjansson >= 2.7
- PostgreSQL >= 15, including libpq
- GNU libmicrohttpd >= 0.9.71
- GNUnet >= 0.20 (from source tarball)
- Python3 with `jinja2`

If you are on Debian stable or later, the following command may help you install these dependencies:

```
# apt-get install \
  libqrencode-dev \
  libsqlite3-dev \
  libltdl-dev \
  libunistring-dev \
  libsodium-dev \
  libargon2-dev \
  libcurl4-gnutls-dev \
  libgcrypt20-dev \
  libjansson-dev \
  libpq-dev \
  libmicrohttpd-dev \
  python3-jinja2 \
  postgresql-15
```

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.

On Ubuntu, you also need to install pkg-config, for example:

```
$ apt-get install pkg-config
```

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.

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.

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

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!

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

```
$ ./configure [--prefix=EXCHANGEPFX] \
              [--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.

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.

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

2.3 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 Debian bookworm.

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 [signed-by=/etc/apt/keyrings/taler-systems.gpg] https://deb.taler.net/apt/debian␣
↪bookworm main
```

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

```
# wget -O /etc/apt/keyrings/taler-systems.gpg \
    https://taler.net/taler-systems.gpg
# apt update
```

Note: You may want to verify the correctness of the Taler Systems SA 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:

```
[root@exchange-online]# apt install 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.

On the offline system, you should run at least:

```
[root@exchange-offline]# apt install taler-exchange-offline
```

2.4 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.

2.5 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 Lunar and Ubuntu Jammy. Make sure to have `universe` in your `/etc/apt/sources.list` (after `main`) as we depend on some packages from Ubuntu `universe`.

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

```
deb [signed-by=/etc/apt/keyrings/taler-systems.gpg] https://deb.taler.net/apt/ubuntu/␣
↪lunar taler-lunar
```

For Ubuntu Mantic use this instead:

```
deb [signed-by=/etc/apt/keyrings/taler-systems.gpg] https://deb.taler.net/apt/ubuntu/␣
↪mantic taler-mantic
```

For Ubuntu Noble use this instead:

```
deb [signed-by=/etc/apt/keyrings/taler-systems.gpg] https://deb.taler.net/apt/ubuntu/␣
↪noble taler-noble
```

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

```
# wget -O /etc/apt/keyrings/taler-systems.gpg \
    https://taler.net/taler-systems.gpg
# 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:

```
[root@exchange-online]# apt install 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.

On the offline system, you should run at least:

```
[root@exchange-offline]# apt install taler-exchange-offline
```

2.6 Services, users, groups and file system hierarchy

The *taler-exchange* package will create several system users to compartmentalize different parts of the system:

- `taler-exchange-httpd`: runs the HTTP daemon with the core business logic.
- `taler-exchange-secmod-rsa`: manages the RSA private online signing keys.
- `taler-exchange-secmod-cs`: manages the CS private online signing keys.
- `taler-exchange-secmod-eddsa`: manages the EdDSA private online signing keys.
- `taler-exchange-closer`: closes idle reserves by triggering wire transfers that refund the originator.
- `taler-exchange-aggregator`: aggregates deposits into larger wire transfer requests.
- `taler-exchange-transfer`: performs wire transfers with the bank (via LibEuFin/Nexus).
- `taler-exchange-wirewatch`: checks for incoming wire transfers with the bank (via LibEuFin/Nexus).
- `postgres`: runs the PostgreSQL database (from *postgresql* package).
- `www-data`: runs the frontend HTTPS service with the TLS keys (from *nginx* package).

Note: The *taler-merchant* package additionally creates a `taler-merchant-httpd` user to run the HTTP daemon with the merchant business logic.

The exchange setup uses the following system groups:

- `taler-exchange-db`: group for all Taler users with direct database access, specifically taler-exchange-httpd, taler-exchange-wirewatch, taler-exchange-closer and taler-exchange-aggregator.
- `taler-exchange-secmod`: group for processes with access to online signing keys; this group must have four users: taler-exchange-secmod-rsa, taler-exchange-secmod-cs, taler-exchange-secmod-eddsa and taler-exchange-httpd.
- `taler-exchange-offline`: group for the access to the offline private key (only used on the offline host and not used on the online system).

The package will deploy systemd service files in `/usr/lib/systemd/system/` for the various components:

- `taler-exchange-aggregator.service`: service that schedules wire transfers which combine multiple deposits to the same merchant.
- `taler-exchange-closer.service`: service that watches for reserves that have been abandoned and schedules wire transfers to send the money back to the originator.
- `taler-exchange-httpd.service`: main Taler exchange logic with the public REST API.
- `taler-exchange-httpd.socket`: systemd socket activation for the Taler exchange HTTP daemon.
- `taler-exchange-secmod-eddsa.service`: software security module for making EdDSA signatures.
- `taler-exchange-secmod-rsa.service`: software security module for making RSA signatures.
- `taler-exchange-secmod-cs.service`: software security module for making CS signatures.
- `taler-exchange-transfer.service`: service that triggers outgoing wire transfers (pays merchants).

- `taler-exchange-wirewatch.service`: service that watches for incoming wire transfers (first step of withdraw).
- `taler-exchange.target`: Main target for the Taler exchange to be operational.

The deployment creates the following key locations in the system:

- `/etc/taler/`: configuration files.
- `/run/taler/`: contains the UNIX domain sockets for inter-process communication (IPC).
- `/var/lib/taler/`: serves as the $HOME for all Taler users and contains sub-directories with the private keys; which keys are stored here depends on the host:
  - online system: exchange-secmod-eddsa, exchange-secmod-cs and exchange-secmod-rsa keys.
  - offline system: exchange-offline keys.

CHAPTER

THREE

CONFIGURATION FUNDAMENTALS

This chapter provides fundamental details about the exchange configuration.

The configuration for all Taler components uses a single configuration file as entry point: `/etc/taler/taler.conf`.

System defaults are automatically loaded from files in `/usr/share/taler/config.d`. These default files should never be modified.

The default configuration `taler.conf` configuration file also includes all configuration files in `/etc/taler/conf.d`. The settings from files in `conf.d` are only relevant to particular components of Taler, while `taler.conf` contains settings that affect all components.

The directory `/etc/taler/secrets` contains configuration file snippets with values that should only be readable to certain users. They are included with the `@inline-secret@` directive and should end with `.secret.conf`.

To view the entire configuration annotated with the source of each configuration option, you can use the `taler-config` helper:

```
[root@exchange-online]# taler-config --diagnostics
< ... annotated, full configuration ... >
```

Warning: While `taler-config` also supports rewriting configuration files, we strongly recommend to edit configuration files manually, as `taler-config` does not preserve comments and, by default, rewrites `/etc/taler/taler.conf`.

3.1 Configuration format

All GNU Taler components are designed to possibly share the same configuration files. When installing a GNU Taler component, the installation deploys default values in configuration files located at ${prefix}/share/taler/config.d/ where ${prefix} is the installation prefix. Different components must be installed to the same prefix.

In order to override these defaults, the user can write a custom configuration file and either pass it to the component at execution time using the *-c* option, or name it taler.conf and place it under $HOME/.config/ which is where components will look by default. Note that the systemd service files pass `-c /etc/taler/taler.conf`, thus making `/etc/taler/taler.conf` the primary location for the configuration.

A config file is a text file containing sections, and each section contains maps options to their values. Configuration files follow basically the INI syntax:

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

(continues on next page)

(continued from previous page)

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

Comments start with a hash (#). Throughout the configuration, it is possible to use $-substitution for options relating to names of files or directories. It is also possible to provide defaults values for those variables that are unset, by using the following syntax: `${VAR:-default}`. There are two ways a user can set the value of $-prefixable variables:

(1) by defining them under a `[paths]` section:

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

(2) or by setting them in the environment:

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

The configuration loader will give precedence to variables set under `[path]` over environment variables.

The utility `taler-config`, which gets installed along with the exchange, can be used get and set configuration values without directly editing the configuration file. The option `-f` is particularly useful to resolve pathnames, when they use several levels of $-expanded variables. See `taler-config --help`.

The repository `git://git.taler.net/deployment` contains example code for generating configuration files under `deployment/netzbon/`.

CHAPTER

FOUR

EXCHANGE DATABASE SETUP

The access credentials for the exchange's database are configured in `/etc/taler/secrets/exchange-db.secret.conf`. Currently, only PostgreSQL is supported as a database backend.

The following users must have access to the exchange database:

- taler-exchange-httpd
- taler-exchange-wire
- taler-exchange-aggregator
- taler-exchange-closer

These users are all in the taler-exchange-db group, and the `exchange-db.secret.conf` should be only readable by users in this group.

Note: The **taler-exchange-dbconfig** tool can be used to automate the database setup. When using the Debian/Ubuntu packages, the users should already have been created, so you can just run the tool without any arguments and should have a working database configuration. The rest of this section only explains what the **taler-exchange-dbconfig** shell script fully automates.

To create a database for the Taler exchange on the local system, run:

```
[root@exchange-online]# su - postgres
[postgres@exchange-online]# createuser taler-exchange-httpd
[postgres@exchange-online]# createuser taler-exchange-wire
[postgres@exchange-online]# createuser taler-exchange-aggregator
[postgres@exchange-online]# createuser taler-exchange-closer
[postgres@exchange-online]# createdb -O taler-exchange-httpd taler-exchange
[postgres@exchange-online]# exit
```

This will create a `taler-exchange` database owned by the `taler-exchange-httpd` user. We will use that user later to perform database maintenance operations.

Assuming the above database setup, the database credentials to configure in the configuration file would simply be:

Listing 1: /etc/taler/secrets/exchange-db.secret.conf

```
[exchange]
DB = postgres

[exchangedb-postgres]
CONFIG=postgres:///taler-exchange
```

If the database is run on a different host, please follow the instructions from the PostgreSQL manual for configuring remote access.

After configuring the database credentials, the exchange database needs to be initialized with the following command:

```
[root@exchange-online]# sudo -u taler-exchange-httpd taler-exchange-dbinit

..note::

  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
  :doc:`manpages/taler-exchange-dbinit.1`.
```

Finally we need to grant the other accounts limited access:

```
[root@exchange-online]# sudo -u taler-exchange-httpd bash
[taler-exchange-httpd@exchange-online]# echo 'GRANT SELECT,INSERT,UPDATE ON ALL TABLES␣
↪IN SCHEMA exchange TO "taler-exchange-aggregator";' \
  | psql taler-exchange
[taler-exchange-httpd@exchange-online]# echo 'GRANT SELECT,INSERT,UPDATE ON ALL TABLES␣
↪IN SCHEMA exchange TO "taler-exchange-closer";' \
  | psql taler-exchange
[taler-exchange-httpd@exchange-online]# echo 'GRANT SELECT,INSERT,UPDATE ON ALL TABLES␣
↪IN SCHEMA exchange TO "taler-exchange-wire";' \
  | psql taler-exchange
[taler-exchange-httpd@exchange-online]# echo 'GRANT USAGE ON ALL SEQUENCES IN SCHEMA␣
↪exchange TO "taler-exchange-aggregator";' \
  | psql taler-exchange
[taler-exchange-httpd@exchange-online]# echo 'GRANT USAGE ON ALL SEQUENCES IN SCHEMA␣
↪exchange TO "taler-exchange-closer";' \
  | psql taler-exchange
[taler-exchange-httpd@exchange-online]# echo 'GRANT USAGE ON ALL SEQUENCES IN SCHEMA␣
↪exchange TO "taler-exchange-wire";' \
  | psql taler-exchange
[taler-exchange-httpd@exchange-online]# exit
```

Note: The above instructions for changing database permissions only work *after* having initialized the database with `taler-exchange-dbinit`, as the tables need to exist before permissions can be granted on them. The `taler-exchange-dbinit` tool cannot setup these permissions, as it does not know which users will be used for which processes.

CHAPTER

FIVE

BASIC SETUP: CURRENCY, DENOMINATIONS AND KEYS

A Taler exchange only supports a single currency. The currency and the smallest currency unit supported by the bank system must be specified in `/etc/taler/taler.conf`.

Listing 1: /etc/taler/taler.conf

```
[taler]
CURRENCY = EUR
CURRENCY_ROUND_UNIT = EUR:0.01

# ... rest of file ...
```

Warning:

When editing `/etc/taler/taler.conf`, take care to not accidentally remove the `@inline-matching@` directive to include the configuration files in `conf.d`.

5.1 Coins (denomination keys)

Next, the electronic cash denominations that the exchange offers must be specified.

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.
- **AGE_RESTRICTED: Set to YES to make this a denomination with support**
  for age restrictions. See age restriction extension below for details. This option is optional and defaults to NO.

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!

The taler-wallet-cli has a helper command that generates a reasonable denomination structure.

```
[root@exchange-online]# taler-wallet-cli deployment gen-coin-config \
                          --min-amount EUR:0.01 \
                          --max-amount EUR:100 \
                          > /etc/taler/conf.d/exchange-coins.conf
```

You can manually review and edit the generated configuration file. The main change that is possibly required is updating the various fees. Note that you MUST NOT edit a coin configuration section after the initial setup. If you must change the values, you must instead create a new section with a different unique name (still with the coin- prefix) and comment out or remove the existing section. Do take care to not introduce the name of the disabled section again in the future.

5.2 Sign keys

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

- [taler-exchange-secmod-eddsa/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 DURATION of a few weeks to a few months for sign keys.
- [taler-exchange-secmod-eddsa/OVERLAP_DURATION]: What is the overlap of the timespan for sign keys? We recommend a few minutes or hours. Must be smaller than DURATION.
- [taler-exchange-secmod-eddsa/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 `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.

5.3 Setting up the offline signing key

Before launching an exchange, the offline signing (master) key must be generated and set in the configuration. The offline signing keys of the exchange should be stored on a different machine. The responsibilities of this offline signing machine are:

- Generation of the exchange's offline master signing key.
- Secure storage of the exchange's offline master signing key.
- Generation of certificates (signed with the offline master signing key) that will be imported by the exchange.
- Revocation of keys when the online system was compromised or is being terminated

Configuration file options related to the master key are:

- **`[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. The default value is usually fine and does not require adjustment.
- **`[exchange/MASTER_PUBLIC_KEY]`: Must specify the exchange's master public**
  key. Needed for the exchange to verify information signed by the offline system. This value must almost always be set explicitly by hand.

```
[root@exchange-offline]# taler-exchange-offline setup
< ... prints the exchange master public key >
```

The public key printed as the output of this command must be put into the configuration of the online machine:

Listing 2: /etc/taler/conf.d/exchange-business.conf

```
[exchange]
MASTER_PUBLIC_KEY = YE6Q6TR1ED...

# ... rest of file ...
```

CHAPTER

SIX

WIRE GATEWAY SETUP

The Taler Wire Gateway is an API that connects the Taler exchange to the underlying core banking system. There are several implementations of wire gateways:

- Project deploymerization implements a wire gateway on top of Bitcoin or Ethereum
- The libeufin-bank provides a wire gateway interface on top of a regional currency bank
- The **taler-fakebank-run** command is an in-memory bank simulator with a wire gateway interface for testing

Before continuing, you need to decide which wire gateway you want to use, and install and configure it on your system. Afterwards, you need to have two key pieces of information from that setup:

- The username and password to access the exchange's account in the system.
- The `payto://` URI of that account (see RFC 8905).

6.1 Exchange Bank Account Configuration

An exchange must be configured with the right settings to access its bank account via a Taler wire gateway. An exchange can be configured to use multiple bank accounts by using multiple wire gateways. Typically only one wire gateway is used.

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

- The `payto://` URI of the bank account, which uniquely idenfies 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 2nd bank account a 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 needs 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.

Each Taler wire gateway is configured in a configuration section that follows the pattern `exchange-account-$id`, where `$id` is an internal identifier for the bank account accessed by the exchange. The basic information for an account should be put in `/etc/taler/conf.d/exchange-business.conf`. The secret credentials to access the Taler Wire Gateway API should be put into a corresponding `exchange-accountcredentials-$id` section in `/etc/taler/secrets/exchange-accountcredentials.conf`. The latter file should be only readable for the `taler-exchange-wire` user. Only the `taler-exchange-wirewatch` and `taler-exchange-transfer` services should run as the `taler-exchange-wire` user. Other exchange processes do not need to have access to the account credentials.

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

Listing 1: /etc/taler/conf.d/exchange-business.conf

```
[exchange-account-1]
# Account identifier in the form of an RFC-8905 payto:// URI.
# For SEPA, looks like payto://iban/$IBAN?receiver-name=$NAME
# Make sure to URL-encode spaces in $NAME!
#
# With x-taler-bank (for Fakebank)
# PAYTO_URI = "payto://x-taler-bank/bank.demo.taler.net/Exchange?receiver-name=exop"
#
# Example using IBAN (for use with LibEuFin)
PAYTO_URI = "payto://iban/CH9300762011623852957?receiver=name=exop"

# 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

@inline-secret@ exchange-accountcredentials-1 ../secrets/exchange-accountcredentials.
↪secret.conf
```

Listing 2: /etc/taler/secrets/exchange-accountcredentials.secret.conf

```
[exchange-accountcredentials-1]

# LibEuFin expects basic auth.
WIRE_GATEWAY_AUTH_METHOD = basic

# Username and password to access the Taler wire gateway.
USERNAME = ...
PASSWORD = ...

# Base URL of the Taler wire gateway.
WIRE_GATEWAY_URL = ...
```

Such a wire gateway configuration can be tested with the following commands:

```
[root@exchange-online]# taler-exchange-wire-gateway-client \
  --section exchange-accountcredentials-1 --debit-history
[root@exchange-online]# taler-exchange-wire-gateway-client \
  --section exchange-accountcredentials-1 --credit-history
```

On success, you will see some of your account's transaction history (or an empty history), while on failure you should see an error message.

CHAPTER

SEVEN

LEGAL SETUP

This chapter describes how to setup certain legal aspects of a GNU Taler exchange. Users that just want to set up an exchange as an experiment without legal requirements can safely skip these steps.

7.1 Legal conditions for using the service

The service has well-known API endpoints to return its legal conditions to the user in various languages and various formats. This section describes how to setup and configure the legal conditions.

7.2 Terms of Service

The service has an endpoint "/terms" to return the terms of service (in legal language) of the service operator. Client software show these terms of service to the user when the user is first interacting with the service. Terms of service are optional for experimental deployments, if none are configured, the service 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 configuration file for the service:

- `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 `TERMS_ETAG`, you MUST also provide the respective files in `TERMS_DIR` (see below).
- `TERMS_DIR`: The directory that contains the terms of service. The files in the directory must be readable to the service process.

7.3 Privacy Policy

The service has an endpoint "/pp" to return the terms privacy policy (in legal language) of the service operator. Clients should show the privacy policy to the user when the user explicitly asks for it, but it should not be shown by default. Privacy policies are optional for experimental deployments, if none are configured, the service will return a simple statement saying that there is no privacy policy available.

To configure the privacy policy response, there are two options in the configuration file for the service:

- `PRIVACY_ETAG`: The current "Etag" to return for the privacy policy. This value must be changed whenever the privacy policy is updated. A common value to use would be a version number. Note that if you change the `PRIVACY_ETAG`, you MUST also provide the respective files in `PRIVACY_DIR` (see below).
- `PRIVACY_DIR`: The directory that contains the privacy policy. The files in the directory must be readable to the service process.

7.4 Legal policies directory layout

The `TERMS_DIR` and `PRIVACY_DIR` directory structures must follow a particular layout. You may use the same directory for both the terms of service and the privacy policy, as long as you use different ETAGs. Inside of the directory, there should be sub-directories 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 `TERMS_ETAG` or `PRIVACY_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 service, and includes ".epub", ".html", ".md", ".pdf" and ".txt" files. If other files are present, the service may show a warning on startup.

7.4.1 Example

A sample file structure for a `TERMS_ETAG` of "tos-v0" would be:

- TERMS_DIR/en/tos-v0.txt
- TERMS_DIR/en/tos-v0.html
- TERMS_DIR/en/tos-v0.pdf
- TERMS_DIR/en/tos-v0.epub
- TERMS_DIR/en/tos-v0.md
- TERMS_DIR/de/tos-v0.txt
- TERMS_DIR/de/tos-v0.html
- TERMS_DIR/de/tos-v0.pdf
- TERMS_DIR/de/tos-v0.epub
- TERMS_DIR/de/tos-v0.md

If the user requests an HTML format with language preferences "fr" followed by "en", the service would return `TERMS_DIR/en/tos-v0.html` lacking a version in French.

7.5 Generating the Legal Terms

The `taler-terms-generator` script can be used to generate directories with terms of service and privacy policies in multiple languages and all required data formats from a single source file in `.rst` format and GNU gettext translations in `.po` format.

To use the tool, you need to first write your legal conditions in English in reStructuredText (rst). You should find a templates in `$PREFIX/share/terms/*.rst` where `$PREFIX` is the location where you installed the service to. Whenever you make substantive changes to the legal terms, you must use a fresh filename and change the respective `ETAG`. The resulting file must be called `$ETAG.rst` and the first line of the file should be the title of the document.

Once you have written the `$ETAG.rst` file in English, you can generate the first set of outputs:

```
$ taler-terms-generator -i $ETAG
```

Afterwards, you should find the terms in various formats for all configured languages (initially only English) in `$PREFIX/share/terms/`. The generator has a few options which are documented in its man page.

7.6 Adding translations

Translations must be available in subdirectories `locale/$LANGUAGE/LC_MESSAGES/$ETAG.po`. To start translating, you first need to add a new language:

```
$ taler-terms-generator -i $ETAG -l $LANGUAGE
```

Here, `$LANGUAGE` should be a two-letter language code like `de` or `fr`. The command will generate a file `locale/$LANGUAGE/LC_MESSAGES/$ETAG.po` which contains each English sentence or paragraph in the original document and an initially empty translation. Translators should update the `.po` file. Afterwards, simply re-run

```
$ taler-terms-generator -i $ETAG
```

to make the current translation(s) available to the service.

Note: You must restart the service whenever adding or updating legal documents or their translations.

7.7 Updating legal documents

When making minor changes without legal implications, edit the `.rst` file, then re-run the step to add a new language for each existing translation to produce an updated `.po` file. Translate the sentences that have changed and finally run the generator (without `-l`) on the ETAG (`-i $ETAG`) to create the final files.

When making major changes with legal implications, you should first rename (or copy) the existing `.rst` file and the associated translation files to a new unique name. Afterwards, make the major changes, update the `.po` files, complete the translations and re-create the final files. Finally, do not forget to update the `ETAG` configuration option to the new name and to restart the service.

CHAPTER

EIGHT

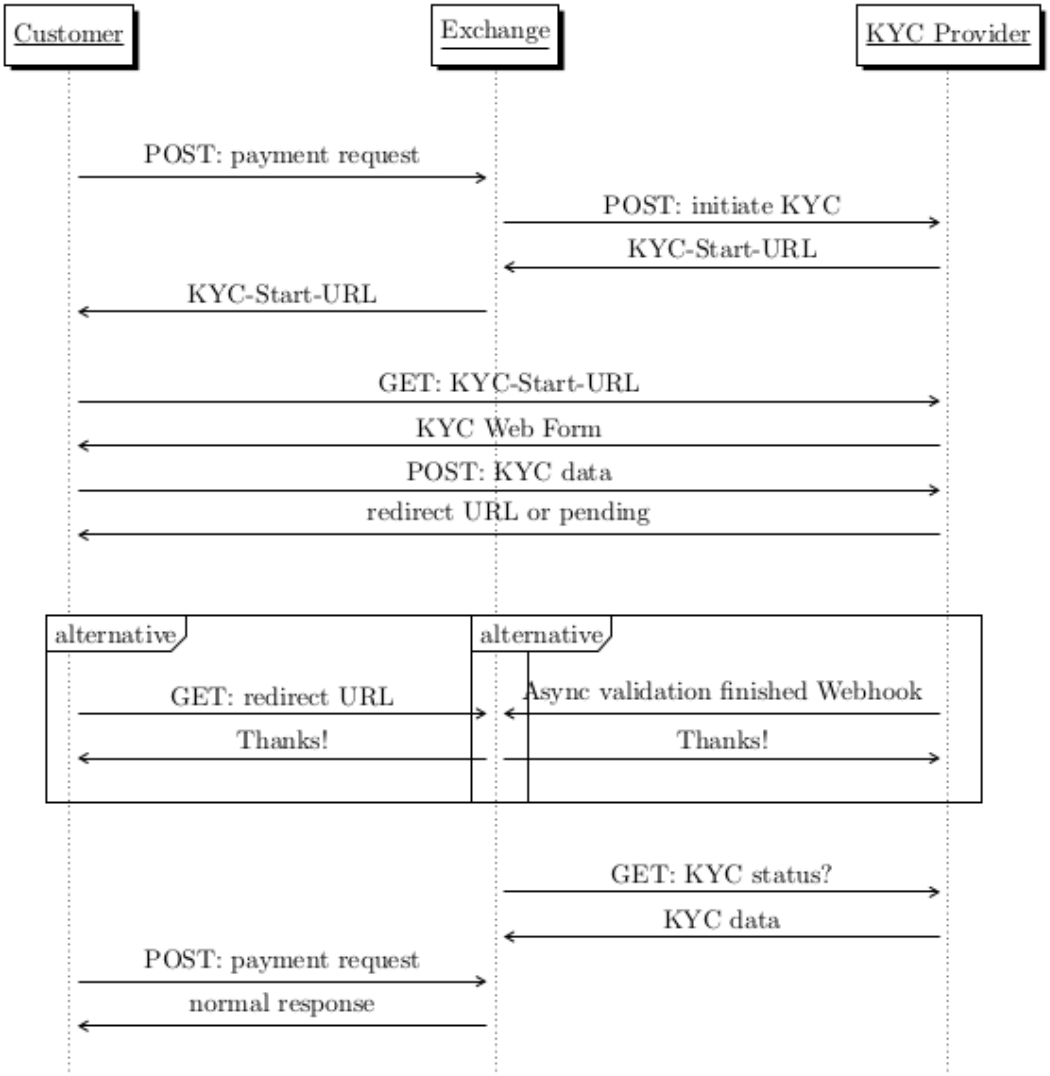
KYC CONFIGURATION

To legally operate, Taler exchange operators may have to comply with KYC regulation that requires financial institutions to identify parties involved in transactions at certain points.

Taler permits an exchange to require KYC data under the following circumstances:

- Customer withdraws money over a threshold
- Wallet receives (via refunds) money resulting in a balance over a threshold
- Wallet receives money via P2P payments over a threshold
- Merchant receives money over a threshold
- Reserve is "opened" for invoicing (**planned feature**)

Any of the above requests can trigger the KYC process, which can be illustrated as follows:



At the end of the KYC process, the wallet re-tries the original request, and assuming KYC was successful, the request should then succeed.

8.1 Taler KYC Terminology

- **Check**: A check establishes a particular attribute of a user, such as their name based on an ID document and lifeness, mailing address, phone number, taxpayer identity, etc.
- **Type of operation**: The operation type determines which Taler-specific operation has triggered the KYC requirement. We support four types of operation: withdraw (by customer), deposit (by merchant), P2P receive (by wallet) and (high) wallet balance.
- **Condition**: A condition specifies when KYC is required. Conditions include the *type of operation*, a threshold amount (e.g. above EUR:1000) and possibly a time period (e.g. over the last month).
- **Cost**: Metric for the business expense for a KYC check at a certain *provider*. Not in any currency, costs are simply relative and non-negative values. Costs are considered when multiple choices are allowed by the *configuration*.
- **Expiration**: KYC legitimizations may be outdated. Expiration rules determine when *checks* have to be performed again.
- **Legitimization rules**: The legitimization rules determine under which *conditions* which *checks* must be performend and the *expiration* time period for the *checks*.
- **Logic**: Logic refers to a specific bit of code (realized as an exchange plugin) that enables the interaction with a specific *provider*. Logic typically requires configuration for access control (such as an authorization token) and possibly the endpoint of the specific *provider* implementing the respective API.
- **Provider**: A provider performs a specific set of *checks* at a certain *cost*. Interaction with a provider is performed by provider-specific *logic*.

8.1.1 Configuration of possible KYC/AML checks

The configuration specifies a set of possible KYC checks offered by external providers. The names of the configuration sections must being with `kyc-check-` followed by an arbitrary `$CHECK_NAME`.

The list of possible FORM names is fixed in the SPA for a particular exchange release.

The outcome of *any* check should always be uploaded encrypted into the `kyc_attributes` table. It MUST include an `expiration_time`.

8.2 KYC Configuration Options

The KYC configuration determines the *legitimization rules*, and specifies which providers offer which *checks* at what *cost*.

The configuration specifies a set of providers, one per configuration section. The names of the configuration sections must being with `kyc-proider-` followed by an arbitrary `$PROVIDER_ID`:

Listing 1: /etc/taler/conf.d/exchange-kyc-providers.conf

```
[kyc-provider-$PROVIDER_ID]
# How expensive is it to use this provider?
# Used to pick the cheapest provider possible.
COST = 42
# Which plugin is responsible for this provider?
# Choices include "oauth2", "kycaid" and "persona".
LOGIC = oauth2
```

(continues on next page)

(continued from previous page)

```
# Which type of user does this provider handle?
# Either INDIVIDUAL or BUSINESS.
USER_TYPE = INDIVIDUAL
# Which checks does this provider provide?
# List of strings, no specific semantics.
PROVIDED_CHECKS = SMS GOVID PHOTO
# Plus additional logic-specific options, e.g.:
AUTHORIZATION_TOKEN = superdupersecret
FORM_ID = business_legi_form
# How long is the check considered valid?
EXPIRATION = 3650d
```

The configuration also must specify a set of legitimization requirements, again one per configuration section:

Listing 2: /etc/taler/conf.d/exchange-kyc-rules.conf

```
[kyc-legitimization-$RULE_NAME]
# Operation that triggers this legitimization.
# Must be one of WITHDRAW, DEPOSIT, P2P-RECEIVE
# or WALLET-BALANCE.
OPERATION_TYPE = WITHDRAW
# Required checks to be performed.
# List of strings, must individually match the
# strings in one or more provider's PROVIDED_CHECKS.
REQUIRED_CHECKS = SMS GOVID
# Threshold amount above which the legitimization is
# triggered.  The total must be exceeded in the given
# timeframe.
THRESHOLD = KUDOS:100
# Timeframe over which the amount to be compared to
# the  THRESHOLD is calculated.  Can be 'forever'.
# Ignored for WALLET-BALANCE.
TIMEFRAME = 30d
```

8.3 OAuth 2.0 specifics

In terms of configuration, the OAuth 2.0 logic requires the respective client credentials to be configured apriori to enable access to the legitimization service. The OAuth 2.0 configuration options are:

Listing 3: /etc/taler/conf.d/exchange-oauth2.conf

```
[kyc-provider-example-oauth2]
LOGIC = oauth2
# (generic options omitted)
# How long is the KYC check valid?
KYC_OAUTH2_VALIDITY = forever

# URL to which we redirect the user for the login process
KYC_OAUTH2_AUTHORIZE_URL = "http://kyc.example.com/authorize"
# URL where we POST the user's authentication information
KYC_OAUTH2_TOKEN_URL = "http://kyc.example.com/token"
```

(continues on next page)

(continued from previous page)

```
# URL of the user info access point.
KYC_OAUTH2_INFO_URL = "http://kyc.example.com/info"

# Where does the client get redirected upon completion?
KYC_OAUTH2_POST_URL = "http://example.com/thank-you"

# For authentication to the OAuth2.0 service
KYC_OAUTH2_CLIENT_ID = testcase
KYC_OAUTH2_CLIENT_SECRET = password

# Mustach template that converts OAuth2.0 data about the user
# into GNU Taler standardized attribute data.
KYC_OAUTH2_CONVERTER_HELPER = taler-exchange-kyc-oauth2-challenger.sh
```

The converter helper is expected to be customized to the selected OAuth2.0 service: different services may return different details about the user or business, hence there cannot be a universal converter for all purposes. The default shell script uses the `jq` tool to convert the JSON returned by the service into the KYC attributes (also in JSON) expected by the exchange. The script will need to be adjusted based on the attributes collected by the specific backend.

The Challenger service for address validation supports OAuth2.0, but does not have a static AUTHORIZE_URL. Instead, the AUTHORIZE_URL must be enabled by the client using a special authenticated request to the Challenger's `/setup` endpoint. The exchange supports this by appending `#setup` to the AUTHORIZE_URL (note that fragments are illegal in OAuth2.0 URLs). Be careful to quote the URL, as # is otherwise interpreted as the beginning of a comment by the configuration file syntax.

Listing 4: /etc/taler/conf.d/exchange-challenger-oauth2.conf

```
[kyc-provider-challenger-oauth2]
LOGIC = oauth2
KYC_OAUTH2_AUTHORIZE_URL = "http://challenger.example.com/authorize/#setup"
KYC_OAUTH2_TOKEN_URL = "http://challenger.example.com/token"
KYC_OAUTH2_INFO_URL = "http://challenger.example.com/info"
```

When using OAuth 2.0, the *CLIENT REDIRECT URI* must be set to the `/kyc-proof/$PROVIDER_SECTION` endpoint. For example, given the configuration above and an exchange running on the host `exchange.example.com`, the redirect URI would be `https://exchange.example.com/kyc-proof/kyc-provider-challenger-oauth2/`.

8.4 Persona specifics

We use the hosted flow. The Persona endpoints return a `request-id`, which we log for diagnosis.

Persona should be configured to use the `/kyc-webhook/` endpoint of the exchange to notify the exchange about the completion of KYC processes. The webhook is authenticated using a shared secret, which should be in the configuration. To use the Persona webhook, you must set the webhook URL in the Persona service to `$EXCHANGE_BASE_URL/kyc-webhook/$SECTION_NAME/` where `$SECTION_NAME` is the name of the configuration section. You should also extract the authentication token for the webhook and put it into the configuration as shown above.

Listing 5: /etc/taler/conf.d/exchange-persona.conf

```
[kyclogic-persona]
# Webhook authorization token. Global for all uses
# of the persona provider!
```

(continues on next page)

(continued from previous page)

```
WEBHOOK_AUTH_TOKEN = wbhsec_698b5a19-c790-47f6-b396-deb572ec82f9

[kyc-provider-example-persona]
LOGIC = persona
# (generic options omitted)

# How long is the KYC check valid?
KYC_PERSONA_VALIDITY = 365d

# Which subdomain is used for our API?
KYC_PERSONA_SUBDOMAIN = taler

# Authentication token to use.
KYC_PERSONA_AUTH_TOKEN = persona_sandbox_42XXXX

# Form to use.
KYC_PERSONA_TEMPLATE_ID = itempl_Uj6Xxxxx

# Where do we redirect to after KYC finished successfully.
KYC_PERSONA_POST_URL = "https://taler.net/kyc-done"

# Salt to give to requests for idempotency.
# Optional.
# KYC_PERSONA_SALT = salt

# Helper to convert JSON with KYC data returned by Persona into GNU Taler
# internal format. Should probably always be set to some variant of
# "taler-exchange-kyc-persona-converter.sh".
KYC_PERSONA_CONVERTER_HELPER = "taler-exchange-kyc-persona-converter.sh"
```

The converter helper is expected to be customized to the selected template: different templates may return different details about the user or business, hence there cannot be a universal converter for all purposes. The default shell script uses the `jq` tool to convert the JSON returned by Persona into the KYC attributes (also in JSON) expected by the exchange. The script will need to be adjusted based on the attributes collected by the specific template.

8.5 KYC AID specifics

We use the hosted flow.

KYCAID must be configured to use the `/kyc-webhook/$SECTION_NAME/` endpoint of the exchange to notify the exchange about the completion of KYC processes.

Listing 6: /etc/taler/conf.d/exchange-kycaid.conf

```
[kyc-provider-example-kycaid]
LOGIC = kycaid
# (generic options omitted)

# How long is the KYC check valid?
KYC_KYCAID_VALIDITY = 365d
```

(continues on next page)

(continued from previous page)

```
# Authentication token to use.
KYC_KYCAID_AUTH_TOKEN = XXX

# Form to use.
KYC_KYCAID_FORM_ID = XXX

# URL to go to after the process is complete.
KYC_KYCAID_POST_URL = "https://taler.net/kyc-done"

# Script to convert the KYCAID data into the Taler format.
KYC_KYCAID_CONVERTER_HELPER = taler-exchange-kyc-kycaid-converter.sh
```

The converter helper is expected to be customized to the selected template: different templates may return different details about the user or business, hence there cannot be a universal converter for all purposes. The default shell script uses the `jq` tool to convert the JSON returned by Persona into the KYC attributes (also in JSON) expected by the exchange. The script will need to be adjusted based on the attributes collected by the specific template.

CHAPTER

NINE

DEPLOYMENT

This chapter describes how to deploy the exchange once the basic installation and configuration are completed.

9.1 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---`). Make sure to set it in such a way that your reverse proxy has permissions to access the UNIX domain socket. The default (660) assumes that the reverse proxy is a member of the group under which the exchange HTTP server is running.

9.2 Reverse Proxy Setup

By default, the `taler-exchange-httpd` service listens for HTTP connections on a UNIX domain socket. To make the service publicly available, a reverse proxy such as nginx should be used. We strongly recommend to configure nginx to use TLS.

The public URL that the exchange will be served under should be put in `/etc/taler/conf.d/exchange-business.conf` configuration file.

Listing 1: /etc/taler/conf.d/exchange-business.conf

```
[exchange]
BASE_URL = https://example.com/

# ... rest of file ...
```

The `taler-exchange` package ships with a sample configuration that can be enabled in nginx:

```
[root@exchange-online]# vim /etc/nginx/sites-available/taler-exchange
< ... customize configuration ... >
[root@exchange-online]# ln -s /etc/nginx/sites-available/taler-exchange \
```

(continues on next page)

(continued from previous page)

```
                                /etc/nginx/sites-enabled/taler-exchange
[root@exchange-online]# systemctl reload nginx
```

Note that the reverse proxy must set a HTTP `X-Forwarded-Host` header to refer to the hostname used by nginx and a HTTP `X-Forwarded-Proto` header to inform the exchange whether the external protocol was `http` or `https`. Thus, depending on your setup, you will likely have to edit those parts of the provided `taler-exchange` configuration file.

With this last step, we are finally ready to launch the main exchange process.

9.3 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-cs` (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 (`secmod`) 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.)

Given proper packaging, all of the above are realized via a simple systemd target. This enables the various processes of an exchange service to be started using a simple command:

```
[root@exchange-online]# systemctl start taler-exchange.target
```

Note: At this point, the exchange service is not yet fully operational.

To check whether the exchange is running correctly under the advertised base URL, run:

```
[root@exchange-online]# export BASE_URL=$(taler-config -s exchange -o base_url)
[root@exchange-online]# wget ${BASE_URL}management/keys
```

The request might take some time to complete on slow machines, because a lot of key material will be generated.

CHAPTER

TEN

OFFLINE SIGNING SETUP, KEY MAINTENANCE AND TEAR-DOWN

The exchange HTTP service must be running before you can complete the following offline signing procedure. Note that when an exchange is running without offline keys its not fully operational. To make the exchange HTTP service fully operational, the following steps involving the offline signing machine must be completed:

1. The public keys of various online keys used by the exchange service are exported via a management HTTP API.
2. The offline signing system validates this request and signs it. Additionally, the offline signing system signs policy messages to configure the exchange's bank accounts and associated fees.
3. The messages generated by the offline signing system are uploaded via the management API of the exchange HTTP service.

A typical minimal setup would look something like this:

```
[anybody@exchange-online]# taler-exchange-offline \
  download > sig-request.json

[root@exchange-offline]# taler-exchange-offline \
  sign < sig-request.json > sig-response.json
[root@exchange-offline]# taler-exchange-offline \
  enable-account payto://iban/$IBAN?receiver-name=$NAME > acct-response.json
[root@exchange-offline]# taler-exchange-offline \
  wire-fee now iban EUR:0 EUR:0 > fee-response.json
[root@exchange-offline]# taler-exchange-offline \
  global-fee now EUR:0 EUR:0 EUR:0 4w 6y 4 > global-response.json

[anybody@exchange-online]# taler-exchange-offline upload < sig-response.json
[anybody@exchange-online]# taler-exchange-offline upload < acct-response.json
[anybody@exchange-online]# taler-exchange-offline upload < fee-response.json
[anybody@exchange-online]# taler-exchange-offline upload < global-response.json
```

The following sections will discuss these steps in more depth.

10.1 Signing the online signing keys

To sign the online signing keys, first 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.

The `download sign upload` sequence in the commands above has to be done periodically, as it signs the various online signing keys of the exchange which periodically expire.

10.2 Account signing

The `enable-account` step is important to 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. Note that for each bank account, additional options **must** be set in the configuration file to tell the exchange how to access the bank account. The offline tool *only* configures the externally visible portions of the setup. The chapter on bank account configuration has further details.

taler-exchange-offline accepts additional options to configure the use of the account. For example, additional options can be used to add currency conversion or to restrict interactions to bank accounts from certain countries:

```
$ taler-exchange-offline \
    enable-account payto://iban/CH9300762011623852957
      conversion-url https://conversion.example.com/
```

For details on optional `enable-account` arguments, see manpages/taler-exchange-offline.1.

10.3 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
```

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!

10.4 Auditor configuration

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 an auditor to affirm that they are auditing this exchange. Before an auditor can do this, the exchange service 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 (using `taler-auditor-offline setup`) 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. For more information on how to setup and operate an auditor, see manpages/taler-auditor-offline.1 and taler-auditor-manual.

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 using `taler-exchange-offline upload`.

10.5 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-denominatin $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.

CHAPTER

ELEVEN

AML CONFIGURATION

The AML configuration steps are used to add or remove keys of exchange operator staff that are responsible for anti-money laundering (AML) compliance. These AML officers are shown suspicious transactions and are granted access to the KYC data of an exchange. They can then investigate the transaction and decide on freezing or permitting the transfer. They may also request additional KYC data from the consumer and can change the threshold amount above which a further AML review is triggered.

11.1 AML Officer Setup

To begin the AML setup, AML staff should launch the GNU Taler exchange AML SPA Web interface. (FIXME-Sebastian: how?). The SPA will generate a public-private key pair and store it in the local storage of the browser. The public key will be displayed and must be securely transmitted to the offline system for approval. Using the offline system, one can then configure which staff has access to the AML operations:

```
[root@exchange-offline]# taler-exchange-offline \
   aml-enable $PUBLIC_KEY "Legal Name" rw > aml.json
[root@exchange-online]# taler-exchange-offline \
   upload < aml.json
```

The above commands would add an AML officer with the given "Legal Name" with read-write (rw) access to the AML officer database. Using "ro" instead of "rw" would grant read-only access to the data, leaving out the ability to actually make AML decisions. Once AML access has been granted, the AML officer can use the SPA to review cases and (with "rw" access) take AML decisions.

Access rights can be revoked at any time using:

```
[root@exchange-offline]# taler-exchange-offline \
   aml-disable $PUBLIC_KEY "Legal Name" > aml-off.json
[root@exchange-online]# taler-exchange-offline \
   upload < aml-off.json
```

11.2 AML Triggers

AML decision processes are automatically triggered under certain configurable conditions. The primary condition that *must* be configured is the `AML_THRESHOLD`:

Listing 1: /etc/taler/conf.d/exchange-business.conf

```
[exchange]
# Accounts or wallets with monthly transaction volumes above this threshold
# are considered suspicious and are automatically flagged for AML review
# and put on hold until an AML officer has reached a decision.
AML_THRESHOLD = "EUR:1000000"
```

Additionally, certain KYC attributes (such as the user being a politically exposed person) may lead to an account being flagged for AML review. The specific logic is configured by providing the exchange with an external helper program that makes the decision given the KYC attributes:

Listing 2: /etc/taler/conf.d/exchange-business.conf

```
[exchange]
# Specifies a program to run on KYC attribute data to decide
# whether we should immediately flag an account for AML review.
KYC_AML_TRIGGER = taler-exchange-kyc-aml-pep-trigger.sh
```

The given program will be given the KYC attributes in JSON format on standard input, and must return 0 to continue without AML and non-zero to flag the account for manual review. To disable this trigger, simply leave the option to its default value of '[/usr/bin/]true'. To flag all new users for manual review, simply set the program to '[/usr/bin/]false'.

11.3 AML Forms

AML forms are defined by the DD 54 dynamic forms. The shipped implementation with of the exchange is installed in

```
${INSTALL_PREFIX}/share/taler/exchange/spa/forms.js
```

The variable `form` contains the list of all form available. For every entry in the list the next properties are expected to be present:

`label`: used in the UI as the name of the form

`id`: identification name, this will be saved in the exchange database along with the values to correctly render the form again. It should simple, short and without any character outside numbers, letters and underscore.

`version`: when editing a form, instead of just replacing fields it will be better to create a new form with the same id and new version. That way old forms in the database will used old definition of the form. It should be a number.

`impl` : a function that returns the design and behavior of form. See DD 54 dynamic forms.

> **Attention:** do not remove a form the list if it has been used. Otherwise you won't be able to see the information save in the exchange database.

To add a new one you can simply copy and paste one element, and edit it.

It is much easier to download `@gnu-taler/aml-backoffice-ui` source from `https://git.taler.net/wallet-core.git/`, compile and copy the file from the `dist/prod`.

11.4 AML Programs

AML programs are helper programs that can:

- Generate a list of *required* context field names for the helper (introspection!) using the "–required-context" command-line switch. The output should use the same syntax as the REQUIRES clause of `[kyc-check-]` configuration sections, except that new lines MUST be used to separate fields instead of ";".
- Generate a list of *required* attribute names for the helper (introspection!) using the "–required-attributes" command-line switch. The output should use the same list of names as the ATTRIBUTES in the `[kyc-provider-]` configuration section (but may also include FORM field names).
- Process an input JSON object of type *AmlProgramInput* into a JSON object of type *AmlOutcome*. This is the default behavior if no command-line switches are provided.

```
interface AmlProgramInput {

  // JSON object that was provided as
  // part of the *measure*.  This JSON object is
  // provided under "context" in the main JSON object
  // input to the AML program.  This "context" should
  // satify both the REQUIRES clause of the respective
  // check and the output of "--requires" from the
  // AML program's command-line option.
  context?: Object;

  // JSON object that captures the
  // output of a [kyc-provider-] or (HTML) FORM.
  // The keys in the JSON object will be the attribute
  // names and the values must be strings representing
  // the data. In the case of file uploads, the data
  // MUST be base64-encoded.
  attributes: Object;

  // JSON array with the results of historic
  // AML desisions about the account.
  aml_history: AmlDecisionDetail[];

  // JSON array with the results of historic
  // KYC data about the account.
  kyc_history: KycDetail[];

}
```

```
interface AmlOutcome {

  // Should the client's account be investigated
  // by AML staff?
  // Defaults to false.
  to_investigate?: boolean;

  // Free-form properties about the account.
  // Can be used to store properties such as PEP,
  // risk category, type of business, hits on
```

(continues on next page)

(continued from previous page)

```
  // sanctions lists, etc.
  properties?: AccountProperties;

  // Types of events to add to the KYC events table.
  // (for statistics).
  events?: string[];

  // KYC rules to apply.  Note that this
  // overrides *all* of the default rules
  // until the expiration_time and specifies
  // the successor measure to apply after the
  // expiration time.
  new_rules: LegitimizationRuleSet;

}
```

If the AML program fails (exits with a failure code or does not provide well-formed JSON output) the AML/KYC process continues with the FALLBACK measure. This should usually be one that asks AML staff to contact the systems administrator.

AML programs are listed in the configuration file, one program per section:

```
[aml-program-$PROG_NAME]

# Program to run.
COMMAND = taler-helper-aml-pep

# Human-readable description of what this
# AML helper program will do. Used to show
# to the AML staff.
DESCRIPTION = "check if the customer is a PEP"

# True if this AML program is enabled (and thus can be
# used in measures and exposed to AML staff).
# Optional, default is NO.
ENABLED = YES

# **original** measure to take if COMMAND fails
# Usually points to a measure that asks AML staff
# to contact the systems administrator. The fallback measure
# context always includes the reasons for the
# failure.
FALLBACK = MEASURE_NAME
```

11.5 AML Measures

The exchange configuration specifies a set of **original** *measures* one per configuration section:

```
[kyc-measure-$MEASURE_NAME]

# Possible check for this measure.  Optional.
# If not given, PROGRAM should be run immediately
# (on an empty set of attributes).
CHECK_NAME = IB_FORM

# Context for the check. The context can be
# just an empty JSON object if there is none.
CONTEXT = {"choices":["individual","business"]}

# Program to run on the context and check data to
# determine the outcome and next measure.
PROGRAM = taler-aml-program
```

If no `CHECK_NAME` is provided at all, the AML `PROGRAM` is to be run immediately. This is useful if no client-interaction is required to arrive at a decision.

Note: The list of *measures* is not complete: AML staff may freely define new measures dynamically, usually by selecting checks, an AML program, and providing context.

CHAPTER

TWELVE

SETUP LINTING

The `taler-wallet-cli` package comes with an experimental tool that runs various checks on the current GNU Taler exchange deployment:

```
[root@exchange-online]# apt install taler-wallet-cli
[root@exchange-online]# taler-wallet-cli deployment lint-exchange
```

You can optionally pass the `--debug` option to get more verbose output, and `--continue` to continue with further checks even though a previous one has failed.

CHAPTER

THIRTEEN

TESTING AND TROUBLESHOOTING

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.

The following shell session illustrates how the wallet can be used to withdraw electronic cash from the exchange and subsequently spend it. For these steps, a merchant backend is not required, as the wallet acts as a merchant.

```
# This will now output a payto URI that money needs to be sent to in order to allow␣
↪withdrawal
# of taler coins.
$ taler-wallet-cli advanced withdraw-manually --exchange $EXCHANGE_URL --amount EUR:10.50
```

Show the status of the manual withdrawal operation.

```
$ taler-wallet-cli transactions
```

At this point, a bank transfer to the exchange's bank account needs to be made with the correct subject / remittance information as instructed by the wallet after the first step. With the above configuration, it should take about 5 minutes after the wire transfer for the incoming transfer to be observed by the Nexus.

Run the following command to check whether the exchange received an incoming bank transfer:

```
[root@exchange-online]# taler-exchange-wire-gateway-client \
   --section exchange-accountcredentials-1 --credit-history
```

Once the transfer has been made, try completing the withdrawal using:

```
$ taler-wallet-cli run-pending
```

Afterwards, check the status of transactions and show the current wallet balance:

```
$ taler-wallet-cli transactions
$ taler-wallet-cli balance
```

Now, we can directly deposit coins via the exchange into a target account. (Usually, a payment is made via a merchant. The wallet provides this functionality for testing.)

```
$ taler-wallet-cli deposit create EUR:5 \
  payto://iban/$IBAN?receiver-name=Name
$ taler-wallet-cli run-pending
```

Check if this transaction was successful (from the perspective of the wallet):

```
$ taler-wallet-cli transactions
```

If the transaction failed, fix any open issue(s) with the exchange and run the "run-pending" command.

The wallet can also track if the exchange wired the money to the merchant account. The "deposit group id" can be found in the output of the transactions list.

```
$ taler-wallet-cli deposit track $DEPOSIT_GROUP_ID
```

You can also check using the exchange-tools whether the exchange sent the an outgoing transfer:

```
[root@exchange-online]# taler-exchange-wire-gateway-client \
  --section exchange-accountcredentials-1 --debit-history
```

After enough time has passed, the money should arrive at the specified IBAN.

For more information on the taler-wallet-cli tool, see taler-wallet.

13.1 taler-config

13.2 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 is generally better edited by hand to preserve comments and structure. Thus, `taler-config` should primarily be used to inspect or understand a configuration that is in place, and not to update it!

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 and clobber your entire configuration file structure, inlining all values and removing all comments, run

```
$ taler-config -s $SECTION -o $OPTION -V $VALUE
```

to set the respective configuration value to `$VALUE`. Note that you have to manually restart affected Taler components 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 any GNU Taler component using the `-c` option.

13.3 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.

13.4 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.

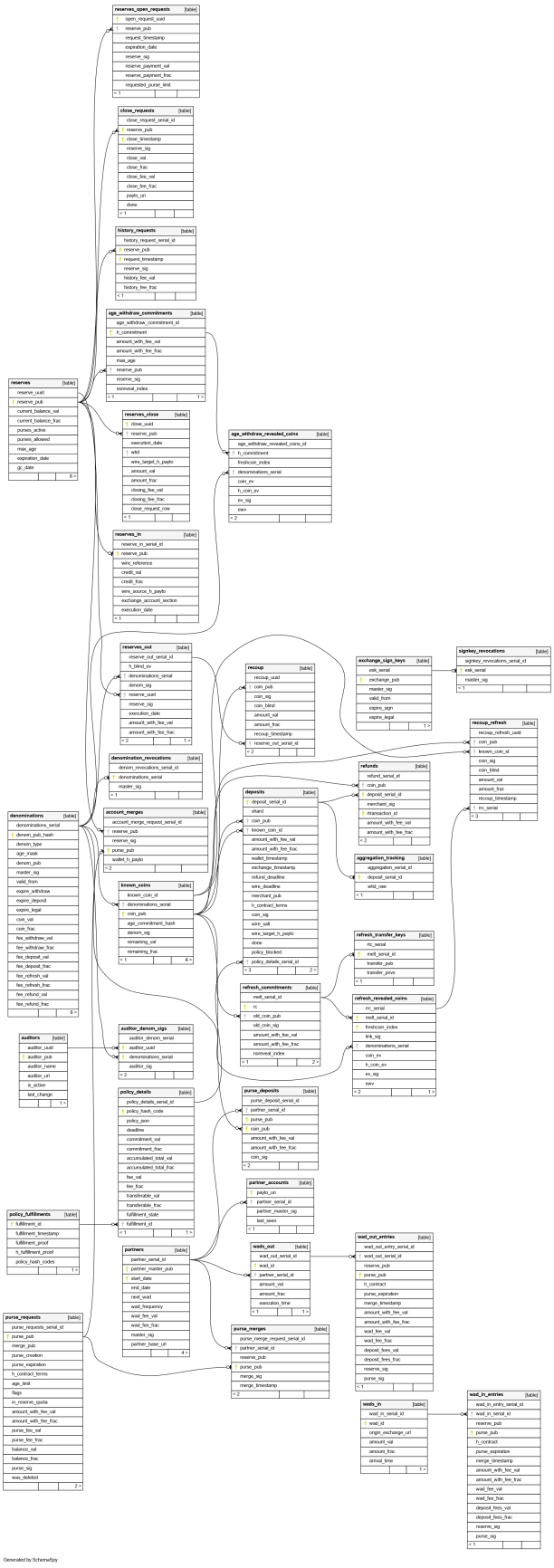
Running the auditor against a the original the production database (without using `taler-auditor-sync`) enables the auditing logic to perform a few additional checks that can detect inconsistencies. These checks are enabled by passing the **-i** option to the `taler-auditor` command. As always, the resulting 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.

13.5 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:


reserves_open_requests [table]
open_request_uuid
reserve_pub
request_timestamp
expiration_date
reserve_sig
reserve_payment_val
reserve_payment_frac
requested_purse_limit
< 1
close_requests [table]
close_request_serial_id
reserve_pub
close_timestamp
reserve_sig
close_val
close_frac
close_fee_val
close_fee_frac
payto_uri
done
< 1
history_requests [table]
history_request_serial_id
reserve_pub
request_timestamp
reserve_sig
history_fee_val
history_fee_frac
< 1
age_withdraw_commitments [table]
age_withdraw_commitment_id
h_commitment
amount_with_fee_val
amount_with_fee_frac
max_age
reserve_pub
reserve_sig
noreveal_index
< 1
1 >
reserves [table]
reserve_uuid
reserve_pub
current_balance_val
current_balance_frac
purses_active
purses_allowed
max_age
expiration_date
gc_date
8 >
reserves_close [table]
close_uuid
reserve_pub
execution_date
wtid
wire_target_h_payto
amount_val
amount_frac
closing_fee_val
closing_fee_frac
close_request_row
< 1
age_withdraw_revealed_coins [table]
age_withdraw_revealed_coins_id
h_commitment
freshcoin_index
denominations_serial
coin_ev
h_coin_ev
ev_sig
ewv
< 2
reserves_in [table]
reserve_in_serial_id
reserve_pub
wire_reference
credit_val
credit_frac
wire_source_h_payto
exchange_account_section
execution_date
< 1
reserves_out [table]
reserve_out_serial_id
h_blind_ev
denominations_serial
denom_sig
reserve_uuid
reserve_sig
execution_date
amount_with_fee_val
amount_with_fee_frac
< 2
1 >
recoup [table]
recoup_uuid
coin_pub
coin_sig
coin_blind
amount_val
amount_frac
recoup_timestamp
reserve_out_serial_id
< 2
exchange_sign_keys [table]
esk_serial
exchange_pub
master_sig
valid_from
expire_sign
expire_legal
1 >
signkey_revocations [table]
signkey_revocations_serial_id
esk_serial
master_sig
< 1
recoup_refresh [table]
recoup_refresh_uuid
coin_pub
known_coin_id
coin_sig
coin_blind
amount_val
amount_frac
recoup_timestamp
rrc_serial
< 3
denomination_revocations [table]
denom_revocations_serial_id
denominations_serial
master_sig
< 1
refunds [table]
refund_serial_id
coin_pub
deposit_serial_id
merchant_sig
rtransaction_id
amount_with_fee_val
amount_with_fee_frac
< 2
deposits [table]
deposit_serial_id
shard
coin_pub
known_coin_id
amount_with_fee_val
amount_with_fee_frac
wallet_timestamp
exchange_timestamp
refund_deadline
wire_deadline
merchant_pub
h_contract_terms
coin_sig
wire_salt
wire_target_h_payto
done
policy_blocked
policy_details_serial_id
< 3
2 >
denominations [table]
denominations_serial
denom_pub_hash
denom_type
age_mask
denom_pub
master_sig
valid_from
expire_withdraw
expire_deposit
expire_legal
coin_val
coin_frac
fee_withdraw_val
fee_withdraw_frac
fee_deposit_val
fee_deposit_frac
fee_refresh_val
fee_refresh_frac
fee_refund_val
fee_refund_frac
6 >
account_merges [table]
account_merge_request_serial_id
reserve_pub
reserve_sig
purse_pub
wallet_h_payto
< 2
aggregation_tracking [table]
aggregation_serial_id
deposit_serial_id
wtid_raw
< 1
known_coins [table]
known_coin_id
denominations_serial
coin_pub
age_commitment_hash
denom_sig
remaining_val
remaining_frac
< 1
8 >
refresh_transfer_keys [table]
rc_serial
rwtt_serial_id
transfer_pub
transfer_priv
< 1
refresh_commitments [table]
rwtt_serial_id
rc
old_coin_pub
old_coin_sig
amount_with_fee_val
amount_with_fee_frac
noreveal_index
< 1
2 >
refresh_revealed_coins [table]
rrc_serial
rwtt_serial_id
freshcoin_index
link_sig
denominations_serial
coin_ev
h_coin_ev
ev_sig
ewv
< 2
1 >
auditors [table]
auditor_uuid
auditor_pub
auditor_name
auditor_url
is_active
last_change
1 >
auditor_denom_sigs [table]
auditor_denom_serial
auditor_uuid
denominations_serial
auditor_sig
< 2
policy_details [table]
policy_details_serial_id
policy_hash_code
policy_json
deadline
commitment_val
commitment_frac
accumulated_total_val
accumulated_total_frac
fee_val
fee_frac
transferable_val
transferable_frac
fulfillment_state
fulfillment_id
< 1
1 >
purse_deposits [table]
purse_deposit_serial_id
partner_serial_id
purse_pub
coin_pub
amount_with_fee_val
amount_with_fee_frac
coin_sig
< 2
partner_accounts [table]
payto_uri
partner_serial_id
partner_master_sig
last_seen
< 1
policy_fulfillments [table]
fulfillment_id
fulfillment_timestamp
fulfillment_proof
h_fulfillment_proof
policy_hash_codes
1 >
partners [table]
partner_serial_id
partner_master_pub
start_date
end_date
next_wad
wad_frequency
wad_fee_val
wad_fee_frac
master_sig
partner_base_url
4 >
wads_out [table]
wad_out_serial_id
wad_id
partner_serial_id
amount_val
amount_frac
execution_time
< 1
1 >
wad_out_entries [table]
wad_out_entry_serial_id
wad_out_serial_id
reserve_pub
purse_pub
h_contract
purse_expiration
merge_timestamp
amount_with_fee_val
amount_with_fee_frac
wad_fee_val
wad_fee_frac
deposit_fees_val
deposit_fees_frac
reserve_sig
purse_sig
< 1
purse_requests [table]
purse_requests_serial_id
purse_pub
merge_pub
purse_creation
purse_expiration
h_contract_terms
age_limit
flags
in_reserve_quota
amount_with_fee_val
amount_with_fee_frac
purse_fee_val
purse_fee_frac
balance_val
balance_frac
purse_sig
was_deleted
2 >
purse_merges [table]
purse_merge_request_serial_id
partner_serial_id
reserve_pub
purse_pub
merge_sig
merge_timestamp
< 2
wads_in [table]
wad_in_serial_id
wad_id
origin_exchange_url
amount_val
amount_frac
arrival_time
1 >
wad_in_entries [table]
wad_in_entry_serial_id
wad_in_serial_id
reserve_pub
purse_pub
h_contract
purse_expiration
merge_timestamp
amount_with_fee_val
amount_with_fee_frac
wad_fee_val
wad_fee_frac
deposit_fees_val
deposit_fees_frac
reserve_sig
purse_sig
< 1
Generated by SchemaSpy

The `jmeasures` JSON in the `legitimization_measures` table is of type *LegitimizationMeasures*:

```
interface LegitimizationMeasures {

  // Array of legitimization measures that
  // are to be applied.
  measures: MeasureInformation[];

  // True if the client is expected to eventually satisfy all requirements.
  // Default (if missing) is false.
  is_and_combinator?: boolean;
}
```

13.6 Database upgrades

Before installing a new exchange version, you should probably make a backup of the existing database and study the release notes on migration. In general, the way to migrate is to stop all existing Taler exchange processes and run:

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

This will migrate the existing schema to the new schema. You also may need to grant Taler exchange processes the rights to the new tables (see last step of database setup).

Note: The **taler-exchange-dbconfig** tool can be used to automate the database migration. In general, simply invoking it again should trigger the migration including **taler-exchange-dbinit** and setting the permissions.

If you do not want to keep any data from the previous installation, the exchange database can be fully re-initialized using:

```
$ taler-exchange-dbinit --reset
```

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. You still also need to then grant the permissions to the other exchange processes again.

CHAPTER

FOURTEEN

TEMPLATE CUSTOMIZATION

The Exchange comes with various HTML templates that are shown to guide users through the KYC process. The Exchange uses C implementation of mustache as the templating engine. This section describes the various templates. In general, the templates must be installed to the `share/taler/exchange/templates/` directory. The file names must be of the form `$NAME.$LANG.must` where `$NAME` is the name of the template and `$LANG` is the 2-letter language code of the template. English templates must exist and will be used as a fallback. If the browser (user-agent) has provided language preferences in the HTTP header and the respective language exists, the correct language will be automatically served.

The following subsections give details about each of the templates. Most subsection titles are the `$NAME` of the respective template.

14.1 Generic Errors Templates

A number of templates are used for generic errors. These are:

- kyc-proof-already-done (KYC process already completed)
- kyc-bad-request (400 Bad Request)
- kyc-proof-endpoint-unknown (404 Not Found for KYC logic)
- kyc-proof-internal-error (500 Internal Server Error)
- kyc-proof-target-unknown (404 Not Found for KYC operation)

All of these templates are instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- message: String; optional, extended human-readable text provided to elaborate on the error, should be shown to provide additional context

14.2 kycaid-invalid-request

The KYCaid plugin does not support requests to the `/kyc-proof/` endpoint (HTTP 400 bad request).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- error: String; error code from the server
- error_details: String; optional error description from the server
- error_uri: optional URI with further details about the error from the server

14.3 oauth2-authentication-failure

The OAuth2 server said that the request was not properly authenticated (HTTP 403 Forbidden).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error

14.4 oauth2-authorization-failure

The OAuth2 server refused to return the KYC data because the authorization code provided was invalid (HTTP 403 Forbidden).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- error: String; error code from the server
- error_message: String; error message from the server

14.5 oauth2-authorization-failure-malformed

The server refused the authorization, but then provided a malformed response (HTTP 502 Bad Gateway).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- debug: Bool; true if we are running in debug mode and are allowed to return HTML with potentially sensitive information
- server_response: Object; could be NULL; this includes the (malformed) OAuth2 server response, it should be shown to the use if "debug" is true

14.6 oauth2-bad-request

The client made an invalid request (HTTP 400 Bad Request).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- message: String; additional error message elaborating on what was bad about the request

14.7 oauth2-conversion-failure

Converting the KYC data into the exchange's internal format failed (HTTP 502 Bad Gateway).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- debug: Bool; true if we are running in debug mode and are allowed to return HTML with potentially sensitive information
- converter: String; name of the conversion command that failed which was used by the Exchange
- attributes: Object; attributes returned by the conversion command, often NULL (after all, conversion failed)
- message: error message elaborating on the conversion failure

14.8 oauth2-provider-failure

We did not get an acceptable response from the OAuth2 provider (HTTP 502 Bad Gateway).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- message: String; could be NULL; text elaborating on the details of the failure

14.9 persona-exchange-unauthorized

The Persona server refused our request (HTTP 403 Forbidden from Persona, returned as a HTTP 502 Bad Gateway).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- data: Object; data returned from Persona service, optional
- persona_http_status: Integer; HTTP status code returned by Persona

14.10 persona-load-failure

The Persona server refused our request (HTTP 429 Too Many Requests from Persona, returned as a HTTP 503 Service Unavailable).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- data: Object; data returned from Persona service, optional
- persona_http_status: Integer; HTTP status code returned by Persona

14.11 persona-exchange-unpaid

The Persona server refused our request (HTTP 402 Payment REquired from Persona, returned as a HTTP 503 Service Unavailable).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- data: Object; data returned from Persona service, optional
- persona_http_status: Integer; HTTP status code returned by Persona

14.12 persona-logic-failure

The Persona server refused our request (HTTP 400, 403, 409, 422 from Persona, returned as a HTTP 502 Bad Gateway).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- data: Object; data returned from Persona service, optional
- persona_http_status: Integer; HTTP status code returned by Persona

14.13 persona-invalid-response

The Persona server refused our request in an unexpected way; returned as a HTTP 502 Bad Gateway.

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- debug: Bool; true if we are running in debug mode and are allowed to return HTML with potentially sensitive information

- server_response: Object; could be NULL; this includes the (malformed) OAuth2 server response, it should be shown to the use if "debug" is true

14.14 persona-network-timeout

The Persona server refused our request (HTTP 408 from Persona, returned as a HTTP 504 Gateway Timeout).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- data: Object; data returned from Persona service, optional
- persona_http_status: Integer; HTTP status code returned by Persona

14.15 persona-kyc-failed

The Persona server indicated a problem with the KYC process, saying it was not completed.

This template is instantiated using the following information:

- persona_inquiry_id: String; internal ID of the inquiry within Persona, useful for further diagnostics by staff
- data: Object; could be NULL; this includes the server response, it contains extensive diagnostics, see Persona documentation on their `/api/v1/inquiries/$ID`.
- persona_http_status: Integer; HTTP status code returned by Persona

14.16 persona-provider-failure

The Persona server refused our request (HTTP 500 from Persona, returned as a HTTP 502 Bad Gateway).

This template is instantiated using the following information:

- ec: Integer; numeric Taler error code, should be shown to indicate the error compactly for reporting to developers
- hint: String; human-readable Taler error code, should be shown for the user to understand the error
- data: Object; data returned from Persona service, optional
- persona_http_status: Integer; HTTP status code returned by Persona

CHAPTER

FIFTEEN

BENCHMARKING

This chapter describes how to run various benchmarks against a Taler exchange. These 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 (optionally) auditor.

Real benchmarks that are intended to demonstrate the scalability of GNU Taler should not use the tools presented in this section: they may be suitable for microbenchmarking and tuning, but the setup is inherently not optimzied for performance or realism, both for the load generation and the server side. Thus, we do not recommend using these performance numbers to assess the scalability of GNU Taler. That said, the tools can be useful to help identify performance issues.

The `taler-unified-setup.sh` script can be used to launch all required services and clients. However, the resulting deployment is simplistic (everything on the local machine, one single-threaded process per service type) and not optimized for performance at all. However, this can still be useful to assess the performance impact of changes to the code or configuration.

The various configuration files used in the code snippets in this section can be found in the `src/benchmark/` directory of the exchange. These are generally intended as starting points. Note that the configuration files ending in `.edited` are created by `taler-unified-setup.sh` and contain some options that are determined at runtime by the setup logic provided by `taler-unified-setup.sh`.

15.1 Choosing a bank

For the bank, both a fakebank (`-f`) and libeufin-based (`-ns`) bank deployment are currently supported by all benchmark tools and configuration templates.

Fakebank is an ultra-fast in-memory implementation of the Taler bank API. It is suitable when the goal is to benchmark the core GNU Taler payment system and to ignore the real-time gross settlement (RTGS) system typically provided by an existing bank. When using the fakebank, `taler-unified-setup.sh` must be started with the `-f` option and be told to use the right exchange bank account from the configuration files via `-u exchange-account-1`.

```
$ dropdb talercheck; createdb talercheck
$ taler-unified-setup.sh -emwt -c $CONF -f -u exchange-account-1
```

libeufin is GNU Taler's adapter to the core banking system using the EBICS banking protocol standard. It uses a Postgres database to persist data and is thus much slower than fakebank. If your GNU Taler deployment uses libeufin in production, it likely makes sense to benchmark with libeufin. When using the fakebank, `taler-unified-setup.sh` must be started with the `-ns` options (starting libeufin-nexus and libeufin-bank) and be told to use the right exchange bank account from the configuration files via `-u exchange-account-2`. Note that `taler-unified-setup.sh` currently cannot reset a libeufin database, and also will not run if the database is already initialized. Thus, you must re-create the database every time before running the command:

```
$ dropdb talercheck; createdb talercheck
$ taler-unified-setup.sh -emwt -c $CONF -ns -u exchange-account-2
```

15.2 taler-bank-benchmark

This is the simplest benchmarking tool, simulating only the bank interaction.

```
$ CONF="benchmark-cs.conf"
$ # or with libeufin
$ dropdb talercheck; createdb talercheck
$ taler-unified-setup.sh -emwt -c "$CONF" -f -u exchange-account-1
$ # Once <<READY>>, in another shell (remember to set $CONF):
$ time taler-bank-benchmark -c "$CONF" -r 40 -p 4 -P4 -u exchange-account-1 -f
$ # or with libeufin
$ dropdb talercheck; createdb talercheck
$ taler-unified-setup.sh -emwt -c "$CONF" -ns -u exchange-account-2
$ # Once <<READY>>, in another shell (remember to set $CONF):
$ time taler-bank-benchmark -c "$CONF" -r 40 -p 1 -P1 -u exchange-account-2
```

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. Processing is then handled by *parallel* (`-P`) service workers.

15.3 taler-exchange-benchmark

This is the benchmarking tool simulates a number of clients withdrawing, depositing and refreshing coins. Operations that are not covered by the `taler-exchange-benchmark` tool today include closing reserves, refunds, recoups and P2P payments.

```
$ CONF="benchmark-cs.conf" # -rsa also makes sense
$ # With fakebank
$ dropdb talercheck; createdb talercheck
$ taler-unified-setup.sh -aemwt -c "$CONF" -f -u exchange-account-1
$ # Once <<READY>>, in another shell (remember to set $CONF):
$ taler-exchange-benchmark -c "$CONF".edited -u exchange-account-1 -n 1 -p1 -r 5 -f
$ #
$ # With libeufin
$ dropdb talercheck; createdb talercheck
$ taler-unified-setup.sh -aemwt -c "$CONF" -ns -u exchange-account-2
$ # Once <<READY>>, in another shell (remember to set $CONF):
$ taler-exchange-benchmark -c "$CONF".edited -u exchange-account-2 -L WARNING -n 1 -p1 -
↪r 5
```

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**.

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.

15.4 taler-aggregator-benchmark

This is another simple benchmark tool that merely prepares an exchange database to run a stand-alone benchmark of the `taler-exchange-aggregator` tool. After preparing a database and running the tool, you can then run one or more `taler-exchange-aggregator` processes and measure how quickly they perform the aggregation work.

```
$ CONF=benchmark-rsa.conf
$ taler-exchange-dbinit -c "$CONF" --reset
$ ./taler-aggregator-benchmark -c "$CONF" -m 500 -r 10 -d 100
$ time taler-exchange-aggregator -c "$CONF" --test
```

This above commands will first create 100 deposits with 10 refunds into each of 500 merchant accounts using randomized time stamps. Afterwards, it will time a single aggregator process in `--test` mode (asking it to terminate as soon as there is no more pending work).

CHAPTER

SIXTEEN

FIXMES

- We should have some summary with the inventory of services that should be running. Systemd by default doesn't show this nicely. Maybe suggest running "systemd list-dependencies taler-exchange.target"?
- What happens when the TWG doesn't like one particular outgoing transaction? How to recover from that as a sysadmin when it happens in practice?

APPENDIX

A

GNU FREE DOCUMENTATION LICENSE

Version 1.3, 3 November 2008

Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. https://fsf.org/

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".

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

F

`fee`, 42

M

`maintenance`, 20, 43

W

`wire fee`, 42