GNU Taler Developer Onboarding Manual

Release 0.8.0pre0

GNU Taler team

Jan 21, 2021

CONTENTS

CHAPTER

ONE

GNU TALER RELEASE CHECKLIST

Release checklists for GNU Taler:

Wallet:

- [ ] build wallet
- [ ] verify wallet works against 'test.taler.net'
- [ ] tag repo.
- [ ] upgrade 'demo.taler.net' to 'test.taler.net'
- [ ] upload new wallet release to app store
- [ ] Update bug tracker (mark release, resolved -> closed)
- [ ] Send announcement to taler@gnu.org
- [ ] Send announcement to info-gnu@gnu.org (major releases only)
- [ ] Send announcement to coordinator@translationproject.org

For exchange:

- [ ] check no compiler warnings at "-Wall"
- [ ] ensure Coverity static analysis passes
- [ ] make check.
- [ ] upgrade 'demo.taler.net' to 'test.taler.net'
- [ ] make dist, make check on result of 'make dist'.
- [ ] Change version number in configure.ac.
- [ ] make dist for release.
- [ ] tag repo.
- [ ] Upload triplet to ftp-upload.gnu.org/incoming/ftp or /incoming/alpha
- [ ] Update bug tracker (mark release, resolved -> closed)
- [ ] Send announcement to taler@gnu.org
- [ ] Send announcement to info-gnu@gnu.org (major releases only)
- [ ] Send announcement to coordinator@translationproject.org

For merchant (C backend):

- [ ] check no compiler warnings at "-Wall"

- [ ] ensure Coverity static analysis passes
- [ ] make check.
- [ ] upgrade 'demo.taler.net' to 'test.taler.net'
- [ ] make dist, make check on result of 'make dist'.
- [ ] Change version number in configure.ac.
- [ ] make dist for release.
- [ ] tag repo.
- [ ] Upload triplet to ftp-upload.gnu.org/incoming/ftp or /incoming/alpha
- [ ] Update bug tracker (mark release, resolved -> closed)
- [ ] Send announcement to taler@gnu.org
- [ ] Send announcement to info-gnu@gnu.org (major releases only)
- [ ] Send announcement to coordinator@translationproject.org

For bank:

- TBD

For Python merchant frontend:

- TBD

For PHP merchant frontend:

- TBD

For auditor:

- TBD

For libebics:

- TBD

CHAPTER

TWO

GNU TALER DEMO UPGRADE CHECKLIST

Post-upgrade checks:

- Run `taler-deployment-arm -I` to verify that all services are running.
- Run the headless wallet to check that services are actually working:

```
$ taler-wallet-cli integrationtest \
    -e https://exchange.demo.taler.net/ \
    -m https://backend.demo.taler.net/ \
    -b https://bank.demo.taler.net \
    -w "KUDOS:10" \
    -s "KUDOS:5"
```

Basics:

- Visit https://demo.taler.net/ to see if the landing page is displayed correctly
- Visit the wallet installation page, install the wallet, and see if the presence indicator is updated correctly.
- Visit https://bank.demo.taler.net/, register a new user and withdraw coins into the browser wallet.

Blog demo:

- Visit https://shop.demo.taler.net/ and purchase an article.
- Verify that the balance in the wallet was updated correctly.
- Go back to https://shop.demo.taler.net/ and click on the same article link. Verify that the article is shown and **no** repeated payment is requested.
- Open the fulfillment page from the previous step in an anonymous browsing session (without the wallet installed) and verify that it requests a payment again.
- Delete cookies on https://shop.demo.taler.net/ and click on the same article again. Verify that the wallet detects that the article has already purchased and successfully redirects to the article without spending more money.

Donation demo:

- Make a donation on https://donations.demo.taler.net
- Make another donation with the same parameters and verify that the payment is requested again, instead of showing the previous fulfillment page.

Survey/Tipping:

- Visit https://survey.demo.taler.net/ and receive a tip.
- Verify that the survey stats page (https://survey.demo.taler.net/survey-stats) is working, and that the survey reserve has sufficient funds.

Note: This manual contains information for developers working on GNU Taler and related components. It is not intended for a general audience.

Table of Contents

CHAPTER

THREE

FUNDAMENTALS

3.1 Bug Tracking

Bug tracking is done with Mantis (https://www.mantisbt.org/). The bug tracker is available at https://bugs.taler.net. A registration on the Web site is needed in order to use the bug tracker, only read access is granted without a login.

3.2 Code Repositories

Taler code is versioned with Git. For those users without write access, all the codebases are found at the following URL:

```
git://git.taler.net/<repository>
```

A complete list of all the existing repositories is currently found at https://git.taler.net/.

3.3 Committing code

Before you can obtain Git write access, you must sign the copyright agreement. As we collaborate closely with GNUnet, we use their copyright agreement – with the understanding that your contributions to GNU Taler are included in the assignment. You can find the agreement on the GNUnet site. Please sign and mail it to Christian Grothoff as he currently collects all the documents for GNUnet e.V.

To obtain Git access, you need to send us your SSH public key. Most core team members have administrative Git access, so simply contact whoever is your primary point of contact so far. You can find instructions on how to generate an SSH key in the Git book. If you have been granted write access, you first of all must change the URL of the respective repository to:

```
ssh://git@git.taler.net/<repository>
```

For an existing checkout, this can be done by editing the `.git/config` file.

The server is configured to reject all commits that have not been signed with GnuPG. If you do not yet have a GnuPG key, you must create one, as explained in the GNU Privacy Handbook. You do not need to share the respective public key with us to make commits. However, we recommend that you upload it to key servers, put it on your business card and personally meet with other GNU hackers to have it signed such that others can verify your commits later.

To sign all commits, you should run

```
$ git config --global commit.gpgsign true
```

You can also sign individual commits only by adding the `-S` option to the `git commit` command. If you accidentally already made commits but forgot to sign them, you can retroactively add signatures using:

```
$ git rebase -S
```

Whether you commit to a personal branch (recommended: `dev/$USER/...`), a feature branch or to `master` should depend on your level of comfort and the nature of the change. As a general rule, the code in `master` must always build and tests should always pass, at least on your own system. However, we all make mistakes and you should expect to receive friendly reminders if your change did not live up to this simple standard. We plan to move to a system where the CI guarantees this invariant in the future.

In order to keep a linear and clean commits history, we advise to avoid merge commits and instead always rebase your changes before pushing to the `master` branch. If you commit and later find out that new commits were pushed, the following command will pull the new commits and rebase yours on top of them.

```
# -S instructs Git to (re)sign your commits
$ git pull --rebase -S
```

3.4 Observing changes

Every commit to the `master` branch of any of our public repositories (and almost all are public) is automatically sent to the gnunet-svn@gnu.org mailinglist. That list is for Git commits only, and must not be used for discussions. It also carries commits from our main dependencies, namely GNUnet and GNU libmicrohttpd. While it can be high volume, the lists is a good way to follow overall development.

3.5 Communication

We use the #taler channel on the Freenode IRC network and the taler@gnu.org public mailinglist for discussions. Not all developers are active on IRC, but all developers should probably subscribe to the low-volume Taler mailinglist. There are separate low-volume mailinglists for gnunet-developers (@gnu.org) and for libmicrohttpd (@gnu.org).

CHAPTER

FOUR

LANGUAGE-SPECIFIC GUIDELINES

- Python Guidelines

CHAPTER

FIVE

TALER DEPLOYMENT ON GV.TALER.NET

This section describes the GNU Taler deployment on `gv.taler.net`. `gv` is our server at BFH. It hosts the Git repositories, Web sites, CI and other services. Developers can receive an SSH account and e-mail alias for the system. As with Git, ask your primary team contact for shell access if you think you need it.

Our old server, `tripwire`, is currently offline and will likely go back online to host `production` systems for operating real Taler payments at BFH in the future.

5.1 DNS

DNS records for taler.net are controlled by the GNU Taler maintainers, specifically Christian and Florian. If you need a sub-domain to be added, please contact one of them.

5.2 User Acccounts

On `gv.taler.net`, there are four system users that are set up to serve Taler on the Internet:

- `taler-test`: serves `*.test.taler.net` and gets automatically built by Buildbot.
- `taler-internal`: serves `*.int.taler.net`, and does *NOT* get automatically built.

The following two users are *never* automatically built, and they both serve `*.demo.taler.net`. At any given time, only one is active and serves the HTTP requests from the outside; the other one can so be compiled without any downtime. If the compilation succeeds, the inactive user can be switched to become active (see next section), and vice versa.

- `demo-blue`
- `demo-green`

CHAPTER

SIX

DEMO UPGRADE PROCEDURE

Upgrading the `demo` environment should be done with care, and ideally be coordinated on the mailing list before. It is our goal for `demo` to always run a "working version" that is compatible with various published wallets.

Before deploying on `demo`, the same version of all components **must** be deployed *and* tested on `int`.

Please use the *demo upgrade checklist* to make sure everything is working.

6.1 Tagging components

All Taler components must be tagged with git before they are deployed on the `demo` environment, using a tag of the following form:

```
demo-YYYY-MM-DD-SS
YYYY = year
MM = month
DD = day
SS = serial
```

6.2 Environment Layout

Environments have the following layout:

```
$HOME/
  deployment (deployment.git checkout)
  envcfg.py  (configuration of the Taler environment)
  activate   (bash file, sourced to set environment variables)
  logs/      (log files)
  local/     (locally installed software)
  sources/   (sources repos of locally build components)
  sockets/   (unix domain sockets of running components)
  taler-data (on-disk state, public and private keys)
  .config/taler.conf (main Taler configuration file)
```

On `demo-blue` and `demo-green`, `taler-data` is a symlink pointing to `$HOME/demo/shared-data` instead of a directory.

6.3 Using envcfg.py

The `$HOME/envcfg.py` file contains (1) the name of the environment and (2) the version of all components we build (in the form of a git rev).

The `envcfg.py` for demo looks like this:

```
env = "demo"
tag = "demo-2019-10-05-01:
tag_gnunet = tag
tag_libmicrohttpd = tag
tag_exchange = tag
tag_merchant = tag
tag_bank = tag
tag_twister = tag
tag_landing = tag
tag_donations = tag
tag_blog = tag
tag_survey = tag
tag_backoffice = tag
tag_sync = tag
```

Currently only the variables `env` and `tag_${component}` are used.

When deploying to `demo`, the `envcfg.py` should be committed to `deployment.git/envcfg/envcfg-demo-YYYY-MM-DD-SS.py`.

6.4 Bootstrapping an Environment

```
$ git clone https://git.taler.net/deployment.git ~/deployment
$ cp ~/deployment/envcfg/$ENVCFGFILE ~/envcfg.py
$ ./deployment/bin/taler-deployment bootstrap
$ source ~/activate
$ taler-deployment build
$ taler-deployment-prepare
$ taler-deployment-start
$ taler-deployment-arm -I # check everything works
# The following command sets up the 'blog' and 'donations' instances.
$ taler-config-instances
```

6.5 Upgrading an Existing Environment

```
$ rm -rf ~/sources ~/local
$ git -C ~/deployment pull
$ cp ~/deployment/envcfg/$ENVCFGFILE ~/envcfg.py
$ taler-deployment bootstrap
$ taler-deployment build
$ taler-deployment-prepare
$ taler-deployment-restart
$ taler-deployment-arm -I # check everything works
```

6.6 Switching Demo Colors

As the `demo` user, to switch to color `${COLOR}`, run the following script from `deployment/bin`:

```
$ taler-deployment switch-demo
```

CHAPTER

SEVEN

ENVIRONMENTS AND BUILDERS ON TALER.NET

7.1 Buildbot implementation

GNU Taler uses a buildbot implementation (front end at https://buildbot.taler.net) to manage continuous integration. Buildbot documentation is at https://docs.buildbot.net/.

Here are some highlights:

- The WORKER is the config that that lives on a shell account on a localhost (taler.net), where this host has buildbot-worker installed. The WORKER executes the commands that perform all end-functions of buildbot.
- The WORKER running buildbot-worker receives these commands by authenticating and communicating with the buildbot server using parameters that were specified when the worker was created in that shell account with the `buildbot-worker` command.
- The buildbot server's master.cfg file contains FACTORY declarations which specify the commands that the WORKER will run on localhost.
- The FACTORY is tied to the WORKER in master.cfg by a BUILDER.
- The master.cfg also allows for SCHEDULER that defines how and when the BUILDER is executed.
- Our master.cfg file is checked into git, and then periodically updated on a particular account on taler.net (ask Christian for access if needed). Do not edit this file directly/locally on taler.net, but check changes into Git.

Best Practices:

- When creating a new WORKER in the `master.cfg` file, leave a comment specifying the server and user account that this WORKER is called from. (At this time, taler.net is the only server used by this implementation, but it's still good practice.)
- Create a worker from a shell account with this command: `buildbot-worker create-worker <workername> localhost <username> <password>`

Then make sure there is a WORKER defined in master.cfg like: `worker.Worker("<username>", "<password>")`

7.2 Documentation Builder

All the Taler documentation is built by the user `docbuilder` that runs a Buildbot worker. The following commands set the `docbuilder` up, starting with a empty home directory.

```
# Log-in as the 'docbuilder' user.

$ cd $HOME
$ git clone git://git.taler.net/deployment
$ ./deployment/bootstrap-docbuilder

# If the previous step worked, the setup is
# complete and the Buildbot worker can be started.

$ buildbot-worker start worker/
```

7.3 Website Builder

Taler Websites, `www.taler.net` and `stage.taler.net`, are built by the user `taler-websites` by the means of a Buildbot worker. The following commands set the `taler-websites` up, starting with a empty home directory.

```
# Log-in as the 'taler-websites' user.

$ cd $HOME
$ git clone git://git.taler.net/deployment
$ ./deployment/bootstrap-sitesbuilder

# If the previous step worked, the setup is
# complete and the Buildbot worker can be started.

$ buildbot-worker start worker/
```

7.4 Code coverage

Code coverage tests are run by the `lcovworker` user, and are also driven by Buildbot.

```
# Log-in as the 'lcovworker' user.

$ cd $HOME
$ git clone git://git.taler.net/deployment
$ ./deployment/bootstrap-taler lcov

# If the previous step worked, the setup is
# complete and the Buildbot worker can be started.

$ buildbot-worker start worker/
```

The results are then published at `https://lcov.taler.net/`.

7.5 Service Checker

The user `demo-checker` runs periodic checks to see if all the `*.demo.taler.net` services are up and running. It is driven by Buildbot, and can be bootstrapped as follows.

```
# Log-in as the 'demo-checker' user

$ cd $HOME
$ git clone git://git.taler.net/deployment
$ ./deployment/bootstrap-demochecker

# If the previous step worked, the setup is
# complete and the Buildbot worker can be started.

$ buildbot-worker start worker/
```

7.6 Tipping reserve top-up

Both 'test' and 'demo' setups get their tip reserve topped up by a Buildbot worker. The following steps get the reserve topper prepared.

```
# Log-in as <env>-topper, with <env> being either 'test' or 'demo'

$ git clone git://git.taler.net/deployment
$ ./deployment/prepare-reservetopper <env>

# If the previous steps worked, then it should suffice to start
# the worker, with:

$ buildbot-worker start worker/
```

7.7 Producing auditor reports

Both 'test' and 'demo' setups get their auditor reports compiled by a Buildbot worker. The following steps get the reports compiler prepared.

```
# Log-in as <env>-auditor, with <env> being either 'test' or 'demo'

$ git clone git://git.taler.net/deployment
$ ./deployment/prepare-auditorreporter <env>

# If the previous steps worked, then it should suffice to start
# the worker, with:

$ buildbot-worker start worker/
```

7.8 Database schema versioning

The Postgres databases of the exchange and the auditor are versioned. See the 0000.sql file in the respective directory for documentation.

Every set of changes to the database schema must be stored in a new versioned SQL script. The scripts must have contiguous numbers. After any release (or version being deployed to a production or staging environment), existing scripts MUST be immutable.

Developers and operators MUST NOT make changes to database schema outside of this versioning.

CHAPTER

EIGHT

RELEASES

8.1 Release Process and Checklists

Please use the *release checklist*

This document describes the process for releasing a new version of the various Taler components to the official GNU mirrors.

The following components are published on the GNU mirrors

- taler-exchange (exchange.git)
- taler-merchant (merchant.git)
- talerdonations (donations.git)
- talerblog (blog.git)
- taler-bank (bank.git)
- taler-wallet-webex (wallet-webex.git)

8.2 Tagging

Tag releases with an **annotated** commit, like

```
$ git tag -a v0.1.0 -m "Official release v0.1.0"
$ git push origin v0.1.0
```

8.3 Database for tests

For tests in the exchange and merchant to run, make sure that a database *talercheck* is accessible by *$USER*. Otherwise tests involving the database logic are skipped.

8.4 Exchange, merchant

Set the version in `configure.ac`. The commit being tagged should be the change of the version.

Update the Texinfo documentation using the files from docs.git:

```
# Get the latest documentation repository
$ cd $GIT/docs
$ git pull
$ make texinfo
# The *.texi files are now in _build/texinfo
#
# This checks out the prebuilt branch in the prebuilt directory
$ git worktree add prebuilt prebuilt
$ cd prebuilt
# Copy the pre-built documentation into the prebuilt directory
$ cp -r ../_build/texinfo .
# Push and commit to branch
$ git commit -a -S -m "updating texinfo"
$ git status
# Verify that all files that should be tracked are tracked,
# new files will have to be added to the Makefile.am in
# exchange.git as well!
$ git push
# Remember $REVISION of commit
#
# Go to exchange
$ cd $GIT/exchange/doc/prebuilt
# Update submodule to point to latest commit
$ git checkout $REVISION
```

Finally, the Automake `Makefile.am` files may have to be adjusted to include new `*.texi` files or images.

For bootstrap, you will need to install GNU Recutils.

For the exchange test cases to pass, `make install` must be run first. Without it, test cases will fail because plugins can't be located.

```
$ ./bootstrap
$ ./configure # add required options for your system
$ make dist
$ tar -xf taler-$COMPONENT-$VERSION.tar.gz
$ cd taler-$COMPONENT-$VERSION
$ make install check
```

8.5 Wallet WebExtension

The version of the wallet is in *manifest.json*. The `version_name` should be adjusted, and *version* should be increased independently on every upload to the WebStore.

```
$ ./configure
$ make dist
```

8.6 Upload to GNU mirrors

See https://www.gnu.org/prep/maintain/maintain.html#Automated-FTP-Uploads

Directive file:

```
version: 1.2
directory: taler
filename: taler-exchange-0.1.0.tar.gz
```

Upload the files in **binary mode** to the ftp servers.

8.7 Creating Debian packages

Our general setup is based on https://wiki.debian.org/DebianRepository/SetupWithReprepro

First, update at least the version of the Debian package in debian/changelog, and then run:

```
$ dpkg-buildpackage -rfakeroot -b -uc -us
```

in the respective source directory (GNUnet, exchange, merchant) to create the `.deb` files. Note that they will be created in the parent directory. This can be done on gv.taler.net, or on another (secure) machine.

Next, the `*.deb` files should be copied to gv.taler.net, say to `/root/incoming`. Then, run

```
# cd /var/www/repos/apt/debian/
# reprepro includedeb sid /root/incoming/*.deb
```

to import all Debian files from `/root/incoming/` into the `sid` distribution. If Debian packages were build against other distributions, reprepro may need to be first configured for those and the import command updated accordingly.

Finally, make sure to clean up `/root/incoming/` (by deleting the now imported `*.deb` files).

CHAPTER

NINE

CONTINUOUS INTEGRATION

CI is done with Buildbot (https://buildbot.net/), and builds are triggered by the means of Git hooks. The results are published at https://buildbot.taler.net/ .

In order to avoid downtimes, CI uses a "blue/green" deployment technique. In detail, there are two users building code on the system, the "green" and the "blue" user; and at any given time, one is running Taler services and the other one is either building the code or waiting for that.

There is also the possibility to trigger builds manually, but this is only reserved to "admin" users.

CHAPTER

TEN

INTERNATIONALIZATION

Internationalization (a.k.a "Translation") is handled with Weblate (https://weblate.org) via our instance at https://weblate.taler.net/ .

At this time, this system is still very new for Taler.net and this documentation may be incorrect and is certainly incomplete.

10.1 Who can Register

At this time, anyone can register an account at https://weblate.taler.net/ to create translations. Registered users default to the **Users** and **Viewers** privilege level.

10.2 About Privilege Levels

This is the breakdown of privilege levels in Weblate:

- **Users/Viewers** = Can log in, view Translations (*applies to new users*)
- **Reviewers** = Can contribute Translations to existing *Components*
- **Managers** = Can create new *Components* of existing *Projects*
- **Superusers** = Can create new *Projects*

10.3 Upgrading Privileges

To upgrade from **Users/Viewers**, a superuser must manually augment your privileges. At this time, superusers are Christian, Florian, and Buck.

10.4 How to Create a Project

The *GNU Taler* project is probably the correct project for most Components and Translations falling under this guide. Please contact a superuser if you need another Project created.

10.5 How to Create a Component

Reference: https://docs.weblate.org/en/weblate-4.0.3/admin/projects.html#component-configuration

In Weblate, a *Component* is a subset of a *Project* and each Component contains N translations. A Component is generally associated with a Git repo.

To create a Component, log into https://weblate.taler.net/ with your *Manager* or higher credentials and choose **+ Add** from the upper-right corner.

What follows is a sort of Wizard. You can find detailed docs at https://docs.weblate.org/. Here are some important notes about connecting your Component to the Taler Git repository:

Under *https://weblate.taler.net/create/component/vcs/*:

- **Source code repository** - Generally `git+ssh://git@git.taler.net/<reponame>`. Check with `git remote -v`.
- **Repository branch** - Choose the correct branch to draw from and commit to.
- **Repository push URL** - This is generally `git+ssh://git@git.taler.net/<reponame>` Check with `git remote -v`.
- **Repository browser** - This is the www URL of the Git repo's file browser. Example `https://git.taler.net/<repositoryname>.git/tree/{{filename}}?h={{branch}}#n{{line}}` where `<repositoryname>` gets replaced but `{{filename}}` and other items in braces are actual variables in the string.
- **Merge style** - *Rebase*, in line with GNU Taler development procedures
- **Translation license** - *GNU General Public License v3.0 or Later*
- **Adding new translation** - Decide how to handle adding new translations

10.6 How to Create a Translation

1 - Log into https://weblate.taler.net

2 - Navigate to *Projects* > *Browse all projects*

3 - Choose the *Project* you wish to contribute to.

4 - Choose the *Component* you wish to contribute to.

5 - Find the language you want to translate into. Click "Translate" on that line.

6 - Find a phrase and translate it.

You may also wish to refer to https://docs.weblate.org/ .

10.7 Translation Standards and Practices

By default, our Weblate instance is set to accept translations in English, French, German, Italian, Russian, Spanish, and Portuguese. If you want to contribute a translation in a different language, navigate to the *Component* you want to translate for, and click "Start new translation" to begin. If you require a privilege upgrade, please contact a superuser with your request.

When asked, set the license to GPLv3 or later.

Set commit/push to manual only.

10.8 GPG Signing of Translations

weblate.taler.net signs GPG commits with the GPG key CD33CE35801462FA5EB0B695F2664BF474BFE502, and the corresponding public key can be found at https://weblate.taler.net/keys/.

This means that contributions made through weblate will not be signed with the individual contributor's key when they are checked into the Git repository, but with the weblate key.

CHAPTER

ELEVEN

ANDROID APPS

11.1 Android App Nightly Builds

There are currently three Android apps in the official Git repository:

- Wallet [CI]
- Merchant PoS Terminal [CI]
- Cashier [CI]

Their git repositories are mirrored at Gitlab to utilize their CI and F-Droid's Gitlab integration to publish automatic nightly builds for each change on the `master` branch.

All three apps publish their builds to the same F-Droid nightly repository (which is stored as a git repository): https://gitlab.com/gnu-taler/fdroid-repo-nightly

You can download the APK files directly from that repository or add it to the F-Droid app for automatic updates by clicking the following link (on the phone that has F-Droid installed).

> GNU Taler Nightly F-Droid Repository

Note: Nightly apps can be installed alongside official releases and thus are meant **only for testing purposes**. Use at your own risk!

11.2 Building apps from source

Note that this guide is different from other guides for building Android apps, because it does not require you to run non-free software. It uses the Merchant PoS Terminal as an example, but works as well for the other apps if you replace `merchant-terminal` with `wallet` or `cashier`.

First, ensure that you have the required dependencies installed:

- Java Development Kit 8 or higher (default-jdk-headless)
- git
- unzip

Then you can get the app's source code using git:

```
# Start by cloning the Android git repository
$ git clone https://git.taler.net/taler-android.git

# Change into the directory of the cloned repository
$ cd taler-android

# Find out which Android SDK version you will need
$ grep -i compileSdkVersion merchant-terminal/build.gradle
```

The last command will return something like `compileSdkVersion 29`. So visit the Android Rebuilds project and look for that version of the Android SDK there. If the SDK version is not yet available as a free rebuild, you can try to lower the `compileSdkVersion` in the app's `merchant-terminal/build.gradle` file. Note that this might break things or require you to also lower other versions such as `targetSdkVersion`.

In our example, the version is `29` which is available, so download the "SDK Platform" package of "Android 10.0.0 (API 29)" and unpack it:

```
# Change into the directory that contains your downloaded SDK
$ cd $HOME

# Unpack/extract the Android SDK
$ unzip android-sdk_eng.10.0.0_r14_linux-x86.zip

# Tell the build system where to find the SDK
$ export ANDROID_SDK_ROOT="$HOME/android-sdk_eng.10.0.0_r14_linux-x86"

# Change into the directory of the cloned repository
$ cd taler-android

# Build the merchant-terminal app
$ ./gradlew :merchant-terminal:assembleRelease
```

If you get an error message complaining about build-tools

> **Failed to install the following Android SDK packages as some licences have not been accepted.**
> build-tools;29.0.3 Android SDK Build-Tools 29.0.3

you can try changing the `buildToolsVersion` in the app's `merchant-terminal/build.gradle` file to the latest "Android SDK build tools" version supported by the Android Rebuilds project.

After the build finished successfully, you will find your APK in `merchant-terminal/build/outputs/apk/release/`.

CHAPTER

TWELVE

CODE COVERAGE

Code coverage is done with the Gcov / Lcov (http://ltp.sourceforge.net/coverage/lcov.php) combo, and it is run nightly (once a day) by a Buildbot worker. The coverage results are then published at https://lcov.taler.net/ .

CHAPTER

THIRTEEN

CODING CONVENTIONS

GNU Taler is developed primarily in C, Kotlin, Python and TypeScript.

13.1 Components written in C

These are the general coding style rules for Taler.

- Baseline rules are to follow GNU guidelines, modified or extended by the GNUnet style: https://docs.gnunet.org/handbook/gnunet.html#Coding-style

13.1.1 Naming conventions

- include files (very similar to GNUnet):
  - if installed, must start with "`taler_`" (exception: platform.h), and MUST live in src/include/
  - if NOT installed, must NOT start with "`taler_`" and MUST NOT live in src/include/ and SHOULD NOT be included from outside of their own directory
  - end in "_lib" for "simple" libraries
  - end in "_plugin" for plugins
  - end in "_service" for libraries accessing a service, i.e. the exchange
- binaries:
  - taler-exchange-xxx: exchange programs
  - taler-merchant-xxx: merchant programs (demos)
  - taler-wallet-xxx: wallet programs
  - plugins should be libtaler_plugin_xxx_yyy.so: plugin yyy for API xxx
  - libtalerxxx: library for API xxx
- logging
  - tools use their full name in GNUNET_log_setup (i.e. 'taler-exchange-keyup') and log using plain 'GNUNET_log'.
  - pure libraries (without associated service) use 'GNUNET_log_from' with the component set to their library name (without lib or '.so'), which should also be their directory name (i.e. 'util')
  - plugin libraries (without associated service) use 'GNUNET_log_from' with the component set to their type and plugin name (without lib or '.so'), which should also be their directory name (i.e. 'exchangedb-postgres')

  - libraries with associated service) use 'GNUNET_log_from' with the name of the service, which should also be their directory name (i.e. 'exchange')
  - for tools with `-l LOGFILE`, its absence means write logs to stderr
- configuration
  - same rules as for GNUnet
- exported symbols
  - must start with TALER_[SUBSYSTEMNAME]_ where SUBSYSTEMNAME MUST match the subdirectory of src/ in which the symbol is defined
  - from libtalerutil start just with `TALER_`, without subsystemname
  - if scope is ONE binary and symbols are not in a shared library, use binary-specific prefix (such as TMH = taler-exchange-httpd) for globals, possibly followed by the subsystem (TMH_DB_xxx).
- structs:
  - structs that are 'packed' and do not contain pointers and are thus suitable for hashing or similar operations are distinguished by adding a "P" at the end of the name. (NEW) Note that this convention does not hold for the GNUnet-structs (yet).
  - structs that are used with a purpose for signatures, additionally get an "S" at the end of the name.
- private (library-internal) symbols (including structs and macros)
  - must not start with `TALER_` or any other prefix
- testcases
  - must be called "test_module-under-test_case-description.c"
- performance tests
  - must be called "perf_module-under-test_case-description.c"

13.2 Shell Scripts

Shell scripts should be avoided if at all possible. The only permissible uses of shell scripts in GNU Taler are:

- Trivial invocation of other commands.
- Scripts for compatibility (e.g. `./configure`) that must run on as many systems as possible.

When shell scripts are used, they `MUST` begin with the following `set` command:

```
# Make the shell fail on undefined variables and
# commands with non-zero exit status.
$ set -eu
```

13.3 Kotlin

We so far have no specific guidelines, please follow best practices for the language.

13.4 Python

13.4.1 Supported Python Versions

Python code should be written and build against version 3.7 of Python.

13.4.2 Style

We use yapf to reformat the code to conform to our style instructions. A reusable yapf style file can be found in `build-common`, which is intended to be used as a git submodule.

13.4.3 Python for Scripting

When using Python for writing small utilities, the following libraries are useful:

- `click` for argument parsing (should be preferred over argparse)
- `pathlib` for path manipulation (part of the standard library)
- `subprocess` for "shelling out" to other programs. Prefer `subprocess.run` over the older APIs.

CHAPTER

FOURTEEN

TESTING LIBRARY

This chapter is a VERY ABSTRACT description of how testing is implemented in Taler, and in NO WAY wants to substitute the reading of the actual source code by the user.

In Taler, a test case is a array of `struct TALER_TESTING_Command`, informally referred to as `CMD`, that is iteratively executed by the testing interpreter. This latter is transparently initiated by the testing library.

However, the developer does not have to defined CMDs manually, but rather call the proper constructor provided by the library. For example, if a CMD is supposed to test feature `x`, then the library would provide the `TALER_TESTING_cmd_x ()` constructor for it. Obviously, each constructor has its own particular arguments that make sense to test `x`, and all constructor are thoroughly commented within the source code.

Internally, each CMD has two methods: `run ()` and `cleanup ()`. The former contains the main logic to test feature `x`, whereas the latter cleans the memory up after execution.

In a test life, each CMD needs some internal state, made by values it keeps in memory. Often, the test has to *share* those values with other CMDs: for example, CMD1 may create some key material and CMD2 needs this key material to encrypt data.

The offering of internal values from CMD1 to CMD2 is made by *traits*. A trait is a `struct TALER_TESTING_Trait`, and each CMD contains a array of traits, that it offers via the public trait interface to other commands. The definition and filling of such array happens transparently to the test developer.

For example, the following example shows how CMD2 takes an amount object offered by CMD1 via the trait interface.

Note: the main interpreter and the most part of CMDs and traits are hosted inside the exchange codebase, but nothing prevents the developer from implementing new CMDs and traits within other codebases.

```
/* Without loss of generality, let's consider the
 * following logic to exist inside the run() method of CMD1 */
...

struct TALER_Amount *a;
/**
 * the second argument (0) points to the first amount object offered,
 * in case multiple are available.
 */
if (GNUNET_OK != TALER_TESTING_get_trait_amount_obj (cmd2, 0, &a))
  return GNUNET_SYSERR;
...

use(a); /* 'a' points straight into the internal state of CMD2 */
```

In the Taler realm, there is also the possibility to alter the behaviour of supposedly well-behaved components. This is needed when, for example, we want the exchange to return some corrupted signature in order to check if the merchant backend detects it.

This alteration is accomplished by another service called *twister*. The twister acts as a proxy between service A and B, and can be programmed to tamper with the data exchanged by A and B.

Please refer to the Twister codebase (under the `test` directory) in order to see how to configure it.

CHAPTER

FIFTEEN

USER-FACING TERMINOLOGY

This section contains terminology that should be used and that should not be used in the user interface and help materials.

15.1 Terms to Avoid

Refreshing Refreshing is the internal technical terminology for the protocol to give change for partially spent coins

Use instead: "Obtaining change"

Coin Coins are an internal construct, the user should never be aware that their balance is represented by coins if different denominations.

Use instead: "(Digital) Cash" or "(Wallet) Balance"

Consumer Has bad connotation of consumption.

Use instead: Customer or user.

Proposal The term used to describe the process of the merchant facilitating the download of the signed contract terms for an order.

Avoid. Generally events that relate to proposal downloads should not be shown to normal users, only developers. Instead, use "communication with mechant failed" if a proposed order can't be downloaded.

Anonymous E-Cash Should be generally avoided, since Taler is only anonymous for the customer. Also some people are scared of anonymity (which as a term is also way too absolute, as anonymity is hardly ever perfect).

Use instead: "Privacy-preserving", "Privacy-friendly"

Payment Replay The process of proving to the merchant that the customer is entitled to view a digital product again, as they already paid for it.

Use instead: In the event history, "re-activated digital content purchase" could be used. (FIXME: this is still not nice.)

Session ID See Payment Replay.

Order Too ambiguous in the wallet.

Use instead: Purchase

Fulfillment URL URL that the serves the digital content that the user purchased with their payment. Can also be something like a donation receipt.

15.2 Terms to Use

Auditor Regulatory entity that certifies exchanges and oversees their operation.

Exchange Operator The entity/service that gives out digital cash in exchange for some other means of payment.

In some contexts, using "Issuer" could also be appropriate. When showing a balance breakdown, we can say "100 Eur (issued by exchange.euro.taler.net)". Sometimes we may also use the more generic term "Payment Service Provider" when the concept of an "Exchange" is still unclear to the reader.

Refund A refund is given by a merchant to the customer (rather the customer's wallet) and "undoes" a previous payment operation.

Payment The act of sending digital cash to a merchant to pay for an order.

Purchase Used to refer to the "result" of a payment, as in "view purchase". Use sparsingly, as the word doesn't fit for all payments, such as donations.

Contract Terms Partially machine-readable representation of the merchant's obligation after the customer makes a payment.

Merchant Party that receives a payment.

Wallet Also "Taler Wallet". Software component that manages the user's digital cash and payments.

CHAPTER

SIXTEEN

DEVELOPER GLOSSARY

This glossary is meant for developers. It contains some terms that we usually do not use when talking to end users or even system administrators.

absolute time method of keeping time in *GNUnet* where the time is represented as the number of microseconds since 1.1.1970 (UNIX epoch). Called absolute time in contrast to *relative time*.

aggregate the *exchange* combines multiple payments received by the same *merchant* into one larger *wire transfer* to the respective merchant's *bank* account

auditor trusted third party that verifies that the *exchange* is operating correctly

bank traditional financial service provider who offers wire *transfers* between accounts

buyer individual in control of a Taler *wallet*, usually using it to *spend* the *coins* on *contracts* (see also *customer*).

close

closes

closed

closing operation an *exchange* performs on a *reserve* that has not been *drained* by *withdraw* operations. When closing a reserve, the exchange wires the remaining funds back to the customer, minus a *fee* for closing

coin

coins coins are individual token representing a certain amount of value, also known as the *denomination* of the coin

commitment

refresh commitment data that the wallet commits to during the *melt* stage of the *refresh* protocol where it has to prove to the *exchange* that it is deriving the *fresh* coins as specified by the Taler protocol. The commitment is verified probabilistically (see: *kappa*) during the *reveal* stage.

contract

contracts formal agreement between *merchant* and *customer* specifying the *contract terms* and signed by the merchant and the *coins* of the customer

contract terms the individual clauses specifying what the buyer is purchasing from the *merchant*

customer individual that directs the buyer (perhaps the same individual) to make a purchase

denomination unit of currency, specifies both the currency and the face value of a *coin*, as well as associated fees and validity periods

denomination key (RSA) key used by the exchange to certify that a given *coin* is valid and of a particular *denomination*

deposit

deposits

depositing operation by which a merchant passes coins to an exchange, expecting the exchange to credit his bank account in the future using an *aggregate wire transfer*

dirty

dirty coin a coin is dirty if its public key may be known to an entity other than the customer, thereby creating the danger of some entity being able to link multiple transactions of coin's owner if the coin is not refreshed

drain

drained a *reserve* is being drained when a *wallet* is using the reserve's private key to *withdraw* coins from it. This reduces the balance of the reserve. Once the balance reaches zero, we say that the reserve has been (fully) drained. Reserves that are not drained (which is the normal process) are *closed* by the exchange.

exchange Taler's payment service operator. Issues electronic coins during withdrawal and redeems them when they are deposited by merchants

expired

expiration Various operations come with time limits. In particular, denomination keys come with strict time limits for the various operations involving the coin issued under the denomination. The most important limit is the deposit expiration, which specifies until when wallets are allowed to use the coin in deposit or refreshing operations. There is also a "legal" expiration, which specifies how long the exchange keeps records beyond the deposit expiration time. This latter expiration matters for legal disputes in courts and also creates an upper limit for refreshing operations on special zombie coin

fakebank implementation of the *bank* API in memory to be used only for test cases.

fee an *exchange* charges various fees for its service. The different fees are specified in the protocol. There are fees per coin for *withdrawing*, *depositing*, *melting*, and *refunding*. Furthermore, there are fees per wire transfer for *closing* a *reserve*: and for *aggregate wire transfers* to the *merchant*.

fresh

fresh coin a coin is fresh if its public key is only known to the customer

GNUnet Codebase of various libraries for a better Internet, some of which GNU Taler depends upon.

json

JSON

JavaScript Object Notation serialization format derived from the JavaScript language which is commonly used in the Taler protocol as the payload of HTTP requests and responses.

kappa security parameter used in the *refresh* protocol. Defined to be 3. The probability of successfully evading the income transparency with the refresh protocol is 1:kappa.

LibEuFin FIXME: explain

link

linking specific step in the *refresh* protocol that an exchange must offer to prevent abuse of the *refresh* mechanism. The link step is not needed in normal operation, it just must be offered.

master key offline key used by the exchange to certify denomination keys and message signing keys

melt

melted

melting step of the *refresh* protocol where a *dirty coin* is invalidated to be reborn *fresh* in a subsequent *reveal* step.

merchant party receiving payments (usually in return for goods or services)

message signing key key used by the exchange to sign online messages, other than coins

order FIXME: to be written!

owner a coin is owned by the entity that knows the private key of the coin

planchet precursor data for a *coin*. A planchet includes the coin's internal secrets (coin private key, blinding factor), but lacks the RSA signature of the *exchange*. When *withdrawing*, a *wallet* creates and persists a planchet before asking the exchange to sign it to get the coin.

privacy policy Statement of an operator how they will protect the privacy of users.

proof Message that cryptographically demonstrates that a particular claim is correct.

proposal a list of *contract terms* that has been completed and signed by the merchant backend.

purchase Refers to the overall process of negotiating a *contract* and then making a payment with *coins* to a *merchant*.

recoup Operation by which an exchange returns the value of coins affected by a *revocation* to their *owner*, either by allowing the owner to withdraw new coins or wiring funds back to the bank account of the *owner*.

refresh

refreshing operation by which a *dirty coin* is converted into one or more *fresh* coins. Involves *melting* the *dirty coin* and then *revealing* so-called *transfer keys*.

refund

refunding operation by which a merchant steps back from the right to funds that he obtained from a *deposit* operation, giving the right to the funds back to the customer

refund transaction id unique number by which a merchant identifies a *refund*. Needed as refunds can be partial and thus there could be multiple refunds for the same *purchase*.

relative time method of keeping time in *GNUnet* where the time is represented as a relative number of microseconds. Thus, a relative time specifies an offset or a duration, but not a date. Called relative time in contrast to *absolute time*.

reserve accounting mechanism used by the exchange to track customer funds from incoming *wire transfers*. A reserve is created whenever a customer wires money to the exchange using a well-formed public key in the subject. The exchange then allows the customer's *wallet* to *withdraw* up to the amount received in *fresh coins* from the reserve, thereby draining the reserve. If a reserve is not drained, the exchange eventually *closes* it.

Other definition: Funds set aside for future use; either the balance of a customer at the exchange ready for withdrawal, or the funds kept in the exchange;s bank account to cover obligations from coins in circulation.

reveal

revealing step in the *refresh* protocol where some of the transfer private keys are revealed to prove honest behavior on the part of the wallet. In the reveal step, the exchange returns the signed *fresh* coins.

revoke

revocation exceptional operation by which an exchange withdraws a denomination from circulation, either because the signing key was compromised or because the exchange is going out of operation; unspent coins of a revoked denomination are subjected to recoup.

sharing users can share ownership of a *coin* by sharing access to the coin's private key, thereby allowing all co-owners to spend the coin at any time.

spend

spending operation by which a customer gives a merchant the right to deposit coins in return for merchandise

terms the general terms of service of an operator, possibly including the *privacy policy*. Not to be confused with the *contract terms* which are about the specific purchase.

transaction method by which ownership is exclusively transferred from one entity

transfer

transfers

wire transfer

wire transfers method of sending funds between *bank* accounts

transfer key

transfer keys special cryptographic key used in the *refresh* protocol, some of which are revealed during the *reveal* step. Note that transfer keys have, despite the name, no relationship to *wire transfers*. They merely help to transfer the value from a *dirty coin* to a *fresh coin*

user any individual using the Taler payment system (see *customer*, *buyer*, *merchant*).

version Taler uses various forms of versioning. There is a database schema version (stored itself in the database, see *-0000.sql) describing the state of the table structure in the database of an *exchange*, *auditor* or *merchant*. There is a protocol version (CURRENT:REVISION:AGE, see GNU libtool) which specifies the network protocol spoken by an *exchange* or *merchant* including backwards-compatibility. And finally there is the software release version (MAJOR.MINOR.PATCH, see https://semver.org/) of the respective code base.

wallet software running on a customer's computer; withdraws, stores and spends coins

WebExtension Cross-browser API used to implement the GNU Taler wallet browser extension.

wire gateway FIXME: explain

wire transfer identifier

wtid Subject of a wire transfer from the exchange to a merchant; set by the aggregator to a random nonce which uniquely identifies the transfer.

withdraw

withdrawing

withdrawal operation by which a *wallet* can convert funds from a *reserve* to fresh coins

zombie

zombie coin coin where the respective *denomination key* is past its *deposit expiration* time, but which is still (again) valid for an operation because it was *melted* while it was still valid, and then later again credited during a *recoup* process

CHAPTER

SEVENTEEN

DEVELOPER TOOLS

This section describes various internal programs to make life easier for the developer.

17.1 taler-config-generate

taler-config-generate - tool to simplify Taler configuration generation

taler-config-generate [**-C** *CURRENCY* | **—currency=***CURRENCY*] [**-c** *FILENAME* | **—config=***FILENAME*] [**-e** | **—exchange**] [**-f** *AMOUNT* | *—wirefee=AMOUNT*] [**-h** | **—help**] [**-J** *JSON* | **—wire-json-exchange=***JSON*] [**-j** *JSON* | **—wire-json-merchant=***JSON*] [**-L** *LOGLEVEL* | **—loglevel=***LOGLEVEL*] [**-m** | **—merchant**] [**-t** | **—trusted**] [**-v** | **—version**] [**-w** *WIREFORMAT* | **—wire** *WIREFORMAT*] [**—bank-uri**] [**—exchange-bank-account**] [**—merchant-bank-account**]

taler-config-generate can be used to generate configuration files for the Taler exchange or Taler merchants.

-C *CURRENCY* | —currency=*CURRENCY* Which currency should we use in the configuration.

-c *FILENAME* | —config=*FILENAME* Location where to write the generated configuration. Existing file will be updated, not overwritten.

-e | —exchange Generate configuration for a Taler exchange.

-f *AMOUNT* | *-wirefee=AMOUNT* Setup wire transfer fees for the next 5 years for the exchange (for all wire methods).

-h | —help Shows this man page.

-J *JSON* | —wire-json-exchange=*JSON* Wire configuration to use for the exchange.

-j *JSON* | —wire-json-merchant=*JSON* Wire configuration to use for the merchant.

-L *LOGLEVEL* | —loglevel=*LOGLEVEL* Use LOGLEVEL for logging. Valid values are DEBUG, INFO, WARNING and ERROR.

-m | —merchant Generate configuration for a Taler merchant.

-t | —trusted Setup current exchange as trusted with current merchant. Generally only useful when configuring for testcases.

-v | —version Print version information.

-w *WIREFORMAT* | —wire *WIREFORMAT* Specifies which wire format to use (i.e. “test” or “sepa”)

—bank-uri Alternative to specify wire configuration to use for the exchange and merchant for the “test” wire method. Only useful if WIREFORMAT was set to “test”. Specifies the URI of the bank.

—exchange-bank-account Alternative to specify wire configuration to use for the exchange for the “test” wire method. Only useful if WIREFORMAT was set to “test”. Specifies the bank account number of the exchange.

—merchant-bank-account Alternative to specify wire configuration to use for the merchant for the “test” wire method. Only useful if WIREFORMAT was set to “test”. Specifies the bank account number of the merchant.

APPENDIX

A

GNU FREE DOCUMENTATION LICENSE

Version 1.3, 3 November 2008

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 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 General Public License, to permit their use in free software.

INDEX

S

T

U

V

W

Z