The GNU Taler tutorial for PHP Web shops

Version 0.2.0
11 November 2016

Marcello Stanisci (marcello.stanisci@inria.fr)
Christian Grothoff (christian.grothoff@inria.fr)

This tutorial is about implementing a merchant frontend to run against a GNU Taler merchant backend (version 0.2.0, 11 November 2016),

Short Contents

Table of Contents

1 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 tutorial

This tutorial is for Web developers and addresses how to integrate GNU Taler with Web shops. It describes how to create a Web shop that processes payments with the help of a GNU Taler merchant *backend*. In the second chapter, you will learn how to trigger the payment process from the Web site, how to communicate with the backend, how to generate a order and process the payment. The third chapter covers the integration of a back office with the backend, which includes tracking payments for orders, matching payments to orders, and persisting and retrieving contracts.

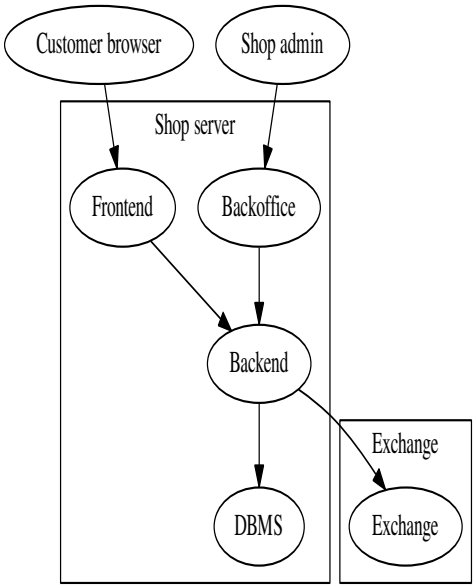
You can download all of the code examples given in this tutorial from `https://git.taler.net/merchant-frontend-examples.git/tree/php/`.

1.3 Architecture overview

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

- A frontend which interacts with the customer's browser. The frontend enables the customer to build a shopping cart and place an order. Upon payment, it triggers the respective business logic to satisfy the order. This component is not included with Taler, but rather assumed to exist at the merchant. This tutorial describes how to develop a Taler frontend.
- A back office application that enables the shop operators to view customer orders, match them to financial transfers, and possibly approve refunds if an order cannot be satisfied. This component is again not included with Taler, but rather assumed to exist at the merchant. This tutorial will describe how to integrate such a component to handle payments managed by Taler. Such integration is shown by adding the back office functionality to the frontend implemented in the second part of this tutorial.
- A Taler-specific payment backend which makes it easy for the frontend to process financial transactions with Taler. For this tutorial, you will use a public backend, but for a production deployment a merchant-specific backend will need to be setup by a system administrator.

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



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

2 Setting up a simple donation page

This section describes how to setup a simple shop, which exposes a button to get donations via Taler. The expected behaviour is that once the "donate" button is clicked, the customer will receive a Taler *proposal* offering him the opportunity to make a fixed donation, for example to donate 1 KUDOS to the charity operating the shop.

All the code samples shown below in the tutorial can be found at `https://git.taler.net/merchant-frontend-examples.git/tree/php`.

Note that if the customer does not have the Taler wallet installed, they should instead be prompted to proceed with a classic dialog for credit card payments.

2.1 Specifying the backend

For many critical operations, the frontend needs to communicate with a Taler backend. Assuming that you do not yet have a backend configured, you can use the public backend provided by the Taler project for testing. This public backend has been set-up at `http://backend.demo.taler.net/` specifically for testing frontends. It uses the currency "KUDOS" and all payments will go into the "Tutorial" account at the Taler "bank" running at `https://bank.demo.taler.net/public-accounts`.

To point the frontend being developed in this tutorial to some backend, it suffices to set the variable `$BACKEND` in `php/config.php` to the desired backend's base URL. You also need to specify the currency used by the backend. For example:

```
// php/config.php
<?php
  // This file is in the public domain.

  // Which backend should we use?  Must end in "/".
  $BACKEND = "http://backend.demo.taler.net/";

  // The currency must match the one used by the backend.
  $CURRENCY = "KUDOS";
?>
```

2.2 Talking to the backend

Given the above configuration, we can now implement two simple functions `get_to_backend` and `post_to_backend` to send requests to the backend. The function `get_to_backend` is in charge of performing HTTP GET requests to the backend, while `post_to_backend` will send HTTP POST requests.

```
// php/backend.php
<?php
  // This file is in the public domain.

  include_once 'config.php';
  include_once 'helpers.php';

  /**
   * 'body' is an object, representing the JSON to POST. NOTE: we do NOT
   * rely on a more structured way of doing HTTP, like the one offered by
   * pecl_http, as its installation was NOT always straightforward.
   */
```

```
  function post_to_backend($backend_uri, $body){
    $json = json_encode($body);
    $c = curl_init(url_join ($GLOBALS['BACKEND'], $backend_uri));
    $options = array(CURLOPT_RETURNTRANSFER => true,
                     CURLOPT_CUSTOMREQUEST => "POST",
                     CURLOPT_POSTFIELDS => $json,
                     CURLOPT_HTTPHEADER =>
                       array('Content-Type:  application/json'));
    curl_setopt_array($c, $options);
    $r = curl_exec($c);
    return array("status_code" => curl_getinfo($c, CURLINFO_HTTP_CODE),
                 "body" => $r);
  }

  /**
   * Sends a GET request to the backend.
   */
  function get_to_backend($backend_url, $args){
    $path = sprintf("%s?%s", $backend_url, http_build_query($args));
    $c = curl_init(url_join($GLOBALS['BACKEND'], $path));

    $options = array(CURLOPT_RETURNTRANSFER => true,
                     CURLOPT_CUSTOMREQUEST => "GET");
    curl_setopt_array($c, $options);
    $r = curl_exec($c);
    return array("status_code" => curl_getinfo($c, CURLINFO_HTTP_CODE),
                 "body" => $r);
  }
?>
```

The given `backend.php` code uses a few helper functions from `php/helpers.php`, which should be self-explanatory.

2.3 Prompting for payment

Our goal is to trigger a Taler payment once the customer has clicked on a donation button. We will use a button that issues an HTTP GET to the frontend `/donate.php` URL. For this, the HTML would be as follows:

```
// php/index.html
<!DOCTYPE html>
<html lang="en">
  <!-- This file is in the public domain -->
  <head>
    <title>A donation button</title>
  </head>
  <body>
    <form action='/donate.php' method='GET'>
      <input type='submit' value='Donate!'></input>
    </form>
  </body>
</html>
```

When the server-side handler for `/donate.php` receives the form submission, it will return a HTML page that will take care of:

- showing a credit card paywall to the user if no wallet is found, and
- fetching a Taler proposal and passing it to the wallet if one is found

A minimalistic `donate.php` implementation is shown below (in PHP):

```
// php/donate.php
<?php
  // This file is in the public domain.

  // Next two lines offer Taler payment option for Taler wallets:
  http_response_code(402); // 402:  Payment required
  header ('X-Taler-Contract-Url:  /generate-order.php');
?>
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Select payment method</title>
  </head>
  <body>
    Here you should put the HTML for the non-Taler (credit card) payment.
  </body>
</html>
```

Given this response, the Taler wallet will fetch the proposal from `/generate-order.php` and display it to the user.

If the wallet is not present, the HTML body will be shown and the Taler headers and the 402 status code ought to be ignored by the browser.

2.4 A helper function to generate the order

We make distinction between *three* different stages of what it informally called "contract".

In a very first stage, we call it the *order*: that occurs when the frontend generates the first JSON that misses some information that the backend is supposed to add. When the backend completes the order and signs it, we have a *proposal*. The proposal is what the user is prompted with, and allows them to confirm the purchase. Once the user confirms the purchase, the wallet makes a signature over the proposal, turning it into a *contract*.

We first define a helper function `make_order` that will generate a complete Taler order as a nested PHP array. The function takes only the order ID and the timestamp as arguments; all of the other details of the order are hardcoded in this simple example.

The following code generate a order about donating 1 KUDOS to the 'Taler charity program':

```
// php/order.php
<?php
  // This file is in the public domain.

  include_once 'config.php';
  include_once 'helpers.php';

  function make_order($nonce,
                      $order_id,
                      $now){
    $order
      = array(
        'nonce' => $nonce,
        'amount' =>
          array('value' => 0,
                'fraction' => 10000000,
```

```
        'currency' => $GLOBALS['CURRENCY']),
  'max_fee' =>
    array('value' => 0,
          'fraction' => 5000000,
          'currency' => $GLOBALS['CURRENCY']),
'products' =>
  array(array('description' =>
                "Donation to charity program",
                'quantity' => 1,
                'price' =>
                   array ('value' => 0,
                          'fraction' => 10000000,
                       'currency' => $GLOBALS['CURRENCY']),
                'product_id' => 0,
                'taxes' =>
                array(),
                'delivery_date' =>
                "/Date(" .  $now->getTimestamp() .  ")/",
                'delivery_location' =>
                'LNAME1'
             )
       ),
'summary' =>
  "Personal donation to charity program",
'order_id' => $order_id,
  'timestamp' =>
  "/Date(" .  $now->getTimestamp() .  ")/",
  'fulfillment_url' =>
  url_rel("/fulfillment.php?order_id=$order_id"),
  'pay_url' =>
  url_rel("/pay.php"),
  'refund_deadline' =>
    "/Date(" .  $now->getTimestamp() .  ")/",
  'pay_deadline' =>
  "/Date(" .  $now->add(new DateInterval('P2W'))->getTimestamp() .  ")/",
  'merchant' =>
    array('address' =>
           'LNAME2',
        'instance' => "tutorial",
           'name' =>
            "Charity donation shop",
           'jurisdiction' =>
            'LNAME2'),
'locations' =>
    array ('LNAME1' =>
               array ('country' => 'Test Country 1',
                        'city' => 'Test City 1',
                        'state' => 'Test State 1',
                        'region' => 'Test Region 1',
                        'province' => 'Test Province 1',
                        'ZIP code' => 49081,
                        'street' => 'test street 1',
                        'street number' => 201),
           'LNAME2' =>
              array ('country' => 'Test Country 2',
                     'city' => 'Test City 2',
                     'state' => 'Test State 2',
                     'region' => 'Test Region 2',
```

```
                                'province' => 'Test Province 2',
                                'ZIP code' => 49082,
                                'street' => 'test street 2',
                                'street number' => 202)
                ));
    return array ('order' => $order);
  }
?>
```

2.5 Signing and returning the proposal

The server-side handler for `/generate-order.php` has to call `make_order` and then POST the result to the backend at `/proposal`. By POSTing the order to the backend we get a cryptographic signature over its contents. The result is then returned to the wallet.

A simple `/generate-order.php` handler may thus look like this:

```
// php/generate-order.php
<?php
  // This file is in the public domain.

  include 'order.php';
  include 'backend.php';
  include 'error.php';

  $order_id = "tutorial-" .  dechex(rand(0,99999999)) .  date("-H_i_s");
  session_start();
  $_SESSION["order_id"] = $order_id;
  if(!isset($_GET["nonce"]))
    return build_error(array("body" => null),
                       "no nonce given",
                       400);
  $order = make_order($_GET["nonce"],
                      strval($order_id),
                      new DateTime('now'));
  // Here the frontend POSTs the proposal to the backend
  $response = post_to_backend("/proposal", $order);
  // We always return verbatim what the backend returned
  http_response_code($response["status_code"]);
  if (200 != $response["status_code"]) {
    echo build_error($response,
                     "Failed to generate proposal",
                     $response['status_code']);
    return;
  }
  echo $response["body"];
?>
```

Note that in practice the frontend might want to generate a monotonically increasing series of order IDs to avoid a chance of collisions. Order IDs must be in the range of $[0 : 2^{51})$.

2.6 Handling errors

In the above example, the helper function `build_error` is used to generate an error response in the case that the backend somehow failed to respond properly to our request.

The function `build_error` is shown below, it returns JSON data matching a particular format for reporting errors, see `http://api.taler.net/api-common.html#errordetail`:

```
// php/error.php
```

```
<?php
  // This file is in the public domain.

  function build_error($response, $hint, $http_code){
    http_response_code($http_code);
    return json_encode(array(
                       'error' => "internal error",
                       'hint' => $hint,
                       'detail' => $response["body"]),
                       JSON_PRETTY_PRINT);
  }
?>
```

2.7 Initiating the payment process

After the browser has fetched the proposal, the user will be given the opportunity to affirm the payment. Assuming the user affirms, the browser will navigate to the "fulfillment_url" that was specified in the proposal.

The fulfillment page can be called by users that have already paid for the item, as well as by users that have not yet paid at all. The fulfillment page must thus use the HTTP session state to detect if the payment has been performed already, and if not request payment from the wallet.

For our example, the fulfillment URL will contain the order id of the donation, like in the following example:

```
https://shop.com/fulfillment.php?order_id=<ORDER_ID>
```

The fulfillment handler at `/fulfillment.php` will use this information to check if the user has already paid, and if so confirm the donation. If the user has not yet paid, it will instead return another "402 Payment Required" header, requesting the wallet to pay:

```
// php/fulfillment.php
<?php
  // This file is in the public domain.

  include 'helpers.php';

  session_start();

  if(pull($_SESSION, 'paid', false)){
    echo sprintf("<p>Thanks for your donation!</p>
                  <br><p>The order ID is:  <b>%s</b>; use it to
                  <a href=\"backoffice.html\">track</a> your money,
                  or make <a href=\"/\">another donation!</a></p>",
                  $_SESSION['order_id']);
    session_destroy();
    return;
  }

  // The user needs to pay, instruct the wallet to send the payment.
  http_response_code(402);
  header('X-Taler-Contract-Url:  '  .  url_rel('/generate-order.php'));
  header('X-Taler-Contract-Query:  '  .  "fulfillment_url");
  header('X-Taler-Offer-Url:  '  .  url_rel('/donate.php'));
  return;
?>
```

Here, this second 402 response contains the following Taler-specific headers:

`X-Taler-Contract-Url`
: The URL that generated the proposal that led to this payment. The wallet may need to reconstruct the proposal.

`X-Taler-Contract-Query`
: The way the wallet should lookup for replayable payments. NOTE that for each payment done, the wallet stores the coins it spent for it in an internal database. And each set of used coins is associated to the fulfillment page where they have been spent. So whenever an already known fulfillment page requests a payment, the wallet will pick those coins it spent on that fulfillment page and resend them (therefore *replaying* the payment). In other words, new coins are used only on unknown fulfillment pages. This header is supposed to be removed in future versions of the wallet though, as it only works with the value `"fulfillment_url"`.

`X-Taler-Offer-Url`
: In case that the wallet does not know about this payment already, i.e. because a user shared the URL with another user, this tells the wallet where to go to retrieve a fresh offer.

2.8 Receiving payments via Taler

The final next step for the frontend is to accept the payment from the wallet. For this, the frontend must implement a payment handler at the URI specified in the `pay_url` proposal field, as explained above.

The role of the `/pay.php` handler is to receive the payment from the wallet and forward it to the backend. If the backend reports that the payment was successful, the handler needs to update the session state with the browser to remember that the user paid.

The following code implements this in PHP:

```
// php/pay.php
<?php
  // This file is in the public domain.

  include "backend.php";
  include "error.php";

  session_start();
  if(!isset($_SESSION["paid"])){
    echo "<p>No session active.  Aborting.</p>";
    return;
  }
  // Get coins.
  $body = json_decode(file_get_contents("php://input"));

  $response = post_to_backend("/pay", $body);
  $proposal_data = json_decode($response["body"])->proposal_data;
  /**
   * NOTE: the order id is fetched from the data returned by the
   * backend.  This information is then shown in the final page from
   * the fulfillment URL. This way, if a malicious wallet sends a
   * old deposit permission for a new donation, then the user can
```

```
   * still detect that, since the old order id would be shown on
   * the fulfillment page.
   */
  $_SESSION["order_id"] = $proposal_data->order_id;
  http_response_code($response["status_code"]);

  if (200 != $response["status_code"]){
    echo build_error($response,
                     "Could not send payment to backend",
                     $response["status_code"]);
    return;
  }
  // Payment went through!
  $_SESSION["paid"] = true;
  return;
?>
```

Do not be confused by the `isset` test for the session state. In our simple example, it will be set to "false" by the fulfillment URL which the browser actually always visits first.

After the `pay.php` handler has affirmed that the payment was successful, the wallet will refresh the fulfillment page, this time receiving the message that the donation was successful. If anything goes wrong, the wallet will handle the respective error.

3 Integration with the back office

This chapter shows how to implement the back office Web interface.

We will use the term *transaction* to refer to the business transaction that a merchant has with a customer (and the related communication with the Taler exchange for payment), and the term *wire transfer* to refer to the exchange settling its accounts with the merchant.

However, from the frontend's perspective, any transaction is denoted by the *order id* contained in the proposal that led to the transaction.

Given that Taler deals with microtransactions, not every customer payment processed with Taler will necessarily correspond to a wire transfer. The Taler exchange may aggregate multiple payments from transactions into one larger wire transfer. The *refund_deadline* and the *pay_deadline* values in the contract specify the timeframes within which the exchange is permitted to perform such aggregations, see ⟨undefined⟩ [Taler contracts], page ⟨undefined⟩.

In this chapter, we will see how a merchant can obtain the mapping from transactions to wire transfers and vice versa. Additionally, we will describe how to obtain a list of all transactions known to the backend.

3.1 Entry page

Given this charge, the back office's main tasks are:

- Allow the back office operator to specify a order id, and display the corresponding wire transfer identifiers.
- Allow the back office operator to specify a wire transfer ID, and display all of the corresponding order ids.
- Allow the back office operator to obtain a list of all transactions.

We implement these with a simple HTML form. For simplicity, we have one single page for gathering input data for both tracking directions:

```
// php/backoffice.html
<!DOCTYPE html>
<html lang="en">
  <!-- This file is in the public domain -->
  <head>
    <title>Minimal merchant back office</title>
    <script src="/history.js" type="application/javascript"></script>
  </head>
  <body>
    <form action='/track-transaction.php' method='GET'>
      <input type='text' name='order_id' placeholder='Order ID'></input>
      <input type='text' name='instance' value='tutorial' hidden></input>
      <input type='submit' value='Track transaction'></input>
    </form>
    <form action='/track-transfer.php' method='GET'>
      <input type='text' name='wtid' placeholder='Wire transfer ID'></input>
      <input type='text' name='exchange' placeholder='Exchange URL'></input>
      <input type='text' name='instance' value="tutorial" hidden></input>
      <input type='submit' value='Track transfer'></input>
    </form>
    <form id="history_form" action="" method="GET">
      <a href="#" onclick="submit_history()">Get transactions list</a>
```

```
      <br>
      <small>Works only if JavaScript enabled</small>
    </form>
    <br/>
    <table id="history" style="visibility:  hidden;">
      <tr>
        <th>Order ID</th>
        <th>Date</th>
        <th>Amount</th>
      </tr>
    </table>
  </body>
</html>
```

The imported script `history.js` is responsible for dynamically get the list of known transactions. See below.

3.2 Tracking a transaction

The `track-transaction.php` script is now responsible for taking all the URL query parameter and use them on the `/track/transaction` request to the backend, see `http://api.taler.net/api-merchant.html#get--track-transaction`. The parameters are the *order_id* and the *instance* (see Section 4.5 [Instances], page 22) of this merchant. Note that the backend may then request this information from the exchange, or retrieve it from its own cache if it has already obtained it. The backend will also check signatures from the exchange, persist the information obtained, and complain if the exchange ever changes its facts in an inconsistent manner.

```
// php/track-transaction.php
<?php
  // This file is in the public domain.

  include 'error.php';
  include 'backend.php';


  $response = get_to_backend("/track/transaction", $_GET);

  if (!in_array($response["status_code"], array(200, 202, 424))){
    echo build_error($response,
                     "Backend error",
                     $response["status_code"]);
    return;
  }

  // Report conflict
  if (424 == $response["status_code"]){
    $body = json_decode($response["body"]);
    echo sprintf("<p>Exchange provided conflicting information about
                  transaction '%s':  what claimed by the exchange does
                  not match what stored in our DB.</p>",
                  $_GET["order_id"]);
    return;
  }

  // Render HTML
  http_response_code($response["status_code"]);
```

```
  $decoded = json_decode($response["body"]);
  if (202 == $response["status_code"]){
    $pretty_date = get_pretty_date($decoded->details->execution_time);
    echo "<p>The exchange accepted the transaction.
          The estimated time for when the related wire transfer
          is to be performed is:  $pretty_date</p>";
    return;
  }

  echo "<ul>";
  foreach ($decoded as $entry){
    $pretty_date = get_pretty_date($entry->execution_time);
    echo sprintf("<li>Wire transfer ID: %s, date:  %s</li>",
                 $entry->wtid,
                 $pretty_date);
  }
  echo "</ul>";
?>
```

If the backend returned an HTTP status code `202` (Accepted), this means that the exchange simply did not yet perform the wire transfer. This is usually the case before *pay_deadline*, as the exchange is waiting for additional opportunities to aggregate transactions. In this case, we tell the user when to retry this operation later.

In the `foreach` loop, we construct the list of all the wire transfers which paid back transaction *order_id*. For simplicity, the list will report only two values: the wire transfer identifier and the date when the transfer occurred. Nonetheless, the data returned by the backend contains more information that can be shown to the user.

3.3 Tracking a wire transfer

To track a wire transfer, the frontend just needs to forward the request it got from the Web form, to the backend. Again, the backend may request missing information from the exchange, verify signatures, persist the result and complain if there are inconsistencies.

In the case that the backend detects inconsistencies, an HTTP status code of `402` is returned. In this case, we report this situation to the user, who should now report this situation to the exchange's auditors as the exchange is misbehaving.

In the usual case where everything went fine, we first output the amount that was supposed to have been transfered under the given `wtid`, and when it was performed (according to the exchange). Finally, in the `foreach` loop, we construct the list of the order ids paid by the wire transfer:

```
<?php
  // This file is in the public domain.

  include 'error.php';
  include 'backend.php';


  $response = get_to_backend("/track/transfer", $_GET);

  if (!in_array($response["status_code"], array(200, 424))){
    echo build_error($response,
                     "Backend error",
                     $response["status_code"]);
```

```
    return;
  }

  // Render HTML
  http_response_code($response["status_code"]);
  if (424 == $response["status_code"]){
    $body = json_decode($response["body"]);
    echo sprintf("<p>The backend detected that the amount wire
                  transferred by the exchange for coin '%s', differs
                  from the coin's original amount.</p>",
                  $body->coin_pub);
    return;
  }

  $json_response = json_decode($response["body"]);
  $pretty_date = get_pretty_date($json_response->execution_time);
  $amount = get_amount($json_response->total);
  echo "<p>$amount transferred on $pretty_date.  The list of involved
       transactions is shown below:</p>";
  echo "<ul>";

  foreach ($json_response->deposits_sums as $entry){
    echo sprintf("<li>Order id:  %s", $entry->order_id);
  }
  echo "</ul>";
?>
```

3.4 Listing all transactions

In order to track transactions, order ids are needed as input. To that purpose, the frontend needs to make a HTTP GET request to `/history`, which is offered by the backend.

The returned data lists the transactions from the youngest back to the oldest.

The `/history` API is actually more rich, as it offers the way to skim results based on time or on index, but that goes beyond the scope of this tutorial.

Our example frontend implements this feature by orchestrating two parts:

- A JavaScript function, imported within `backoffice.html`, that issues the HTTP GET to `/history.php?instance=tutorial` and modifies the current page by adding a table showing the result.
- The `history.php` script, which is in charge of serving the request coming from the JavaScript. Its task is to relay that request to the backend, and echo the result back.

See below both parts:

```
// ../history.js
var FRACTION = 100000000;

// Stringify amounts.  Take a {value:  x, fraction:  y, currency:  "Z"}
// and return a "a.b Z" form.
function parse_amount(amount){
  var v = amount.value + (amount.fraction/FRACTION);
  return v + " " + amount.currency;
}

// Parse Taler date ("/Date(TIMESTAMP)/") string and
// return a JavaScript Date object.
```

```
function get_date(date){
  var split = date.match(/Date\((.*)\)/);
  var seconds;
  if(isNaN(seconds = Number(split[1]))){
    console.error("Malformed date gotten from backend");
    return;
  }
  console.log("Extracting timestamp", split[1]);
  var d = new Date(seconds * 1000);
  return d;
}


// Perform the call to /history.php?instance=tutorial.
// It also takes care of cleaning/filling the table showing
// the results.
function submit_history(){

  // Clean the table showing old results
  var table = document.getElementById("history");
  /* We don't want to kill the first child */
  for (var i = 2; i < table.childNodes.length; i++)
    table.removeChild(table.childNodes[i]);
  var req = new XMLHttpRequest();
  var get_column = function(value){
    var column = document.createElement("td");
    column.textContent = value;
    return column;
  };
  var on_error = function(){
    table.innerHTML = "<p>Could not get transactions list from server</p>"
  };
  req.open("GET", "/history.php?instance=tutorial", true);
  req.onload = function(){
    if(req.readyState == 4 && req.status == 200){
      console.log("Got history:", req.responseText);
      var history = JSON.parse(req.responseText);
      if(!history)
        console.log("Got invalid JSON");
      if(0 == history.length){
        table.innerHTML = "<p>No transaction was that young!</p>";
      }
      // Fill the table with fresh results
      for (var i=0; i<history.length; i++){
          var entry = history[i];
        var tr = document.createElement("tr");
        tr.appendChild(get_column(entry.order_id));
          var date = get_date(entry.timestamp);
          tr.appendChild(get_column(date.toLocaleDateString()));
          tr.appendChild(get_column(parse_amount(entry.amount)))
          table.appendChild(tr);
      }
      table.style.visibility = "";
    }
    else{
      on_error();
    }
  };
```

```
  req.send();
}
// ../history.php
<?php

  include "helpers.php";
  include "backend.php";
  include "error.php";

  // Just relay the request we got from the JavaScript
  $response = get_to_backend("/history", $_GET);

  if (200 != $response["status_code"]){
    echo build_error($response,
                     "Backend error",
                     $response["status_code"]);
    return;
  }

  // Give the response "verbatim" back.
  echo $response["body"];
?>
```

4 Advanced topics

4.1 Reacting to the presence of a Taler wallet

Taler offers the way to the frontend developer to detect whether a user has the wallet installed in their browser, and take actions accordingly.

4.1.1 The no-JavaScript way

The follwing example shows all that is needed to perform the detection without using JavaScript:

```
<!DOCTYPE html>
<html lang="en" data-taler-nojs="true">
  <head>
    <title>Tutorial</title>
    <link rel="stylesheet"
          type="text/css"
          href="/web-common/taler-fallback.css"
          id="taler-presence-stylesheet" />
  </head>
  <body>
    <p class="taler-installed-hide">
      No wallet found.
    </p>
    <p class="taler-installed-show">
      Wallet found!
    </p>
  </body>
</html>
```

The `taler-fallback.css` is part of the Taler's *web-common* repository, available at `https://git.taler.net/web-common.git`. Please adjust the `href` attribute in order to make it work with your Web site.

The detection works by `taler-fallback.css` hiding any tag from the `taler-installed-show` class, in case no wallet is installed. If otherwise the wallet is installed, the wallet takes action by hiding any tag from the `taler-installed-hide` class and overriding `taler-fallback.css` logic by showing any tag from the `taler-installed-show` class.

4.1.2 The JavaScript way

`taler-wallet-lib.js` helps the frontend, by providing the way to register two callbacks: one to be executed if a wallet is present, the other if it is not. See the example below:

```
<html lang="en">
  <head>
    <script src="/web-common/taler-wallet-lib.js" type="application/javascript">
    </script>
  </head>
  <body>
    <div id="content">
    </div>
    <script type="application/javascript">

      content = document.getElementById("content");
```

```
        p = document.createElement("p");

        function walletInstalled()
          p.textContent = "Wallet installed!";
          content.appendChild(p);

        function walletNotInstalled()
          p.textContent = "Wallet not found.";
          content.appendChild(p);

        taler.onPresent(wallerInstalled);
        taler.onAbsent(wallerNotInstalled);
      </script>
    </body>
  </html>
```

`taler-wallet-lib.js` exports the `taler` object that exposes the `onPresent` and the `onAbsent` functions needed to register the frontend's callbacks. Thus the function `walletInstalled` will be executed whenever a wallet is installed, and `walletNotInstalled` if not. Note that since now we can use JavaScript we can register callbacks that do more than just showing and hiding elements.

4.2 Building Taler proposals

A Taler proposal can specify many details about the transaction. This section describes each of the fields in depth.

amount Specifies the total amount to be paid to the merchant by the customer. The amount is broken up into a *value*, a *fraction* (100.000.000 *fraction* units correspond to one *value*) and the *currency*. For example, `EUR 1.50` would be represented as the tuple `value = 1, fraction = 50000000, currency = "EUR"`.

max_fee This is the maximum total amount of deposit fees that the merchant is willing to pay. If the deposit fees for the coins exceed this amount, the customer has to include it in the payment total. The fee is specified using the same triplet used for *amount*.

max_wire_fee
Maximum wire fee accepted by the merchant (customer share to be divided by the 'wire_fee_amortization' factor, and further reduced if deposit fees are below 'max_fee'). Default if missing is zero.

wire_fee_amortization
Over how many customer transactions does the merchant expect to amortize wire fees on average? If the exchange's wire fee is above 'max_wire_fee', the difference is divided by this number to compute the expected customer's contribution to the wire fee. The customer's contribution may further be reduced by the difference between the 'max_fee' and the sum of the actual deposit fees. Optional, default value if missing is 1. 0 and negative values are invalid and also interpreted as 1.

pay_url Which URL accepts payments. This is the URL where the wallet will POST coins.

fulfillment_url
: Which URL should the wallet go to for obtaining the fulfillment, for example the HTML or PDF of an article that was bought, or an order tracking system for shipments, or a simple human-readable Web page indicating the status of the contract.

order_id
: Alphanumeric identifier, freely definable by the merchant. Used by the merchant to uniquely identify the transaction.

summary
: Short, human-readable summary of the contract. To be used when displaying the contract in just one line, for example in the transaction history of the customer.

timestamp
: Time at which the offer was generated.

pay_deadline
: Timestamp of the time by which the merchant wants the exchange to definitively wire the money due from this contract. Once this deadline expires, the exchange will aggregate all deposits where the contracts are past the *refund_deadline* and execute one large wire payment for them. Amounts will be rounded down to the wire transfer unit; if the total amount is still below the wire transfer unit, it will not be disbursed.

refund_deadline
: Timestamp until which the merchant willing (and able) to give refunds for the contract using Taler. Note that the Taler exchange will hold the payment in escrow at least until this deadline. Until this time, the merchant will be able to sign a message to trigger a refund to the customer. After this time, it will no longer be possible to refund the customer. Must be smaller than the *pay_deadline*.

products
: Array of products that are being sold to the customer. Each entry contains a tuple with the following values:

  description
  : Description of the product.

  quantity
  : Quantity of the items to be shipped. May specify a unit (`1 kg`) or just the count.

  price
  : Price for *quantity* units of this product shipped to the given *delivery_location*. Note that usually the sum of all of the prices should add up to the total amount of the contract, but it may be different due to discounts or because individual prices are unavailable.

  product_id
  : Unique ID of the product in the merchant's catalog. Can generally be chosen freely as it only has meaning for the merchant, but should be a number in the range $[0, 2^{51})$.

  taxes
  : Map of applicable taxes to be paid by the merchant. The label is the name of the tax, i.e. *VAT*, *sales tax* or *income tax*, and the

value is the applicable tax amount. Note that arbitrary labels are permitted, as long as they are used to identify the applicable tax regime. Details may be specified by the regulator. This is used to declare to the customer which taxes the merchant intends to pay, and can be used by the customer as a receipt. The information is also likely to be used by tax audits of the merchant.

delivery_date
: Time by which the product is to be delivered to the *delivery_location*.

delivery_location
: This should give a label in the *locations* map, specifying where the item is to be delivered.

Values can be omitted if they are not applicable. For example, if a purchase is about a bundle of products that have no individual prices or product IDs, the *product_id* or *price* may not be specified in the contract. Similarly, for virtual products delivered directly via the fulfillment URI, there is no delivery location.

merchant

address
: This should give a label in the *locations* map, specifying where the merchant is located.

name
: This should give a human-readable name for the merchant's business.

jurisdiction
: This should give a label in the *locations* map, specifying the jurisdiction under which this contract is to be arbitrated.

locations
: Associative map of locations used in the contract. Labels for locations in this map can be freely chosen and used whenever a location is required in other parts of the contract. This way, if the same location is required many times (such as the business address of the customer or the merchant), it only needs to be listed (and transmitted) once, and can otherwise be referred to via the label. A non-exhaustive list of location attributes is the following:

  country
  : Name of the country for delivery, as found on a postal package, i.e. "France".

  state
  : Name of the state for delivery, as found on a postal package, i.e. "NY".

  region
  : Name of the region for delivery, as found on a postal package.

  province
  : Name of the province for delivery, as found on a postal package.

  city
  : Name of the city for delivery, as found on a postal package.

  ZIP code
  : ZIP code for delivery, as found on a postal package.

  street
  : Street name for delivery, as found on a postal package.

street number
: Street number (number of the house) for delivery, as found on a postal package.

name receiver name for delivery, either business or person name.

Note that locations are not required to specify all of these fields, and it is also allowed to have additional fields. Contract renderers must render at least the fields listed above, and should render fields that they do not understand as a key-value list.

4.3 Inlining proposals in HTTP headers

In the example in Section 2.3 [Prompting for payment], page 4, we told the wallet the URL from where it should fetch the proposal. Instead of specifying the proposal via an URL, it is also possible to inline short proposals directly.

That may be accomplished by the frontend including the `X-Taler-Proposal` HTTP header, whenever the requested page is meant to trigger a payment. The example shown below may be an alternative version of `/donate.php`, which implements inline contracts.

```
// ../inline.php
<?php
  // This file is in the public domain.

  include "order.php";

  $order_id = rand(1,90000); // simplified, do not do this!
  $order = make_order($order_id, new DateTime("now"));

  $response = post_to_backend("/proposal", $order);
  $ret = $response["body"];

  if (200 != $response["status_code"]) {
    $ret =  build_error($response,
                        "Failed to generate proposal",
                        $response["status_code"]);
  }

  http_response_code(402); // Payment required
  header ("X-Taler-Proposal:  $ret"); // Inline proposal
?>
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Select payment method</title>
  </head>
  <body>
    Here you should put the HTML for the non-Taler (credit card) payment.
  </body>
</html>
```

The wallet will then look for the `X-Taler-Proposal` header, and fetch the inlined contract if this header is found. Please note that this feature is not yet implemented in the wallet.

4.4 Design considerations for the fulfillment page

The fulfillment page mechanism is designed to provide the following two properties:

1. Taler payments *can* be implemented in DB-less frontends.
2. Taler payments are replayable, meaning that each purchase is associated with a URL (the fulfillment URL) that shows the product each time it gets visited (and of course, only the first time takes the user's money).

Both properties are gotten "for free" by the way replayable payments are implemented. Since `pay.php` simply relays payments to the backend, if the latter returns "200 OK", then the frontend delivers what is mentioned in the backend's response. Note that along with the "200 OK" response, the backend returns the whole proposal associated with the fulfillment URL that triggered the payment, so the frontend has all the information useful to pick the right product to deliver. The "payment" relayed to the backend contains the *order id*, that allows the backend to perform all the integrity checks on the payment. This way, the frontend doesn't need any database to replay payments.

4.5 Instances

Taler's design allows a single backend to manage multiple frontends. In other words, we might have multiple shops relying on the same backend. As of terminology, we call *instance* any of the frontends accounted by the same backend.

The backend's RESTful API allows frontends to specify which instance they are. However, this specification is optional, and a "default" instance will be used whenever the frontend does not specify one.

Please note that in this stage of development, the backend's REST call `/history` returns records for *any* instance. The rationale behind is to allow grouping "public" business entities under the same backend.

This way, a single frontend can expose multiple donation buttons for multiple receivers, and still operate against one backend. So in this scenario, there is no harm if the operator of instance 'a' sees history entries related to instance 'b'.

See `https://donations.demo.taler.net/`, which uses this functionality.

4.6 Normalized base URLs

Exchanges and merchants have a base URL for their service. This URL **must** be in a canonical form when it is stored (e.g. in the wallet's database) or transmitted (e.g. to a bank page).

- The URL must be absolute. This implies that the URL has a schema.
- The path component of the URL must end with a slash.
- The URL must not contain a fragment or query.

When a user enters a URL that is, technically, relative (such as "alice.example.com/exchange"), wallets *may* transform it into a canonical base URL ("http://alice.example.com/exchange/"). Other components *should not* accept URLs that are not canonical.

Rationale: Joining non-canonical URLs with relative URLs (e.g. "exchange.example.com" with "reserve/status") results in different and slightly unexpected behavior in some URL handling libraries. Canonical URLs give more predictable results with standard URL joining.

5 Reference

5.1 Headers for HTTP 402

The HTTP status code `402 Payment Required` can be used by the merchant frontend to trigger operations related to payments in the user agent. The user agent associates at most one proposal with every URL via the proposal's `fulfillment_url` field. The associated proposal is either missing (in case it doesn't exist), paid (in case the payment for it was successfully sent to the merchant) or unpaid. If the associated proposal is unpaid, `402 Payment Required` will cause the user agent to pay for the associated proposal.

The following headers for `402 Payment Required` are recognized by Taler and further influence the processing:

`X-Taler-Refund-Url`
: If this header present, the value of this header must be a URL that the user agent can use to request and process refunds.

`X-Taler-Contract-Url`
: If there is no associated proposal, the user agent will fetch a proposal from this URL and process it. This typically prompts the user to agree to pay.

`X-Taler-Offer-Url`
: If there is no associated proposal and `X-Taler-Contract-Url` is not specified, the browser will navigate to this URL.

5.2 JavaScript API

The following functions are defined in the `taler` namespace of the `taler-wallet-lib` helper library available at `https://git.taler.net/web-common.git/tree/taler-wallet-lib.js`.

`onPresent(callback: () => void)`
: Add a callback to be called when support for Taler payments is detected.

`onAbsent(callback: () => void)`
: Add a callback to be called when support for Taler payments is disabled.

`pay({contract_url: string, offer_url: string})`
: Results in the same action as a `402 Payment Required` with `contract_url` in the `X-Taler-Contract-Url` header and `offer_url` in the `X-Taler-Payment-Url` header.

`refund(refund_url: string)`
: Results in the same action as a `402 Payment Required` with `refund_url` in the `X-Taler-Refund-Url` header.

5.3 Stylesheet-based presence detection

Stylesheet-based presence detection will be applied on all pages that have the `data-taler-nojs` attribute of the `html` element set `true`. The default/fallback stylesheet, that will be

taken over by the wallet once installed, must be included with the id `taler-presence-stylesheet`, like this:

The following CSS classes can be used:

`taler-installed-hide`
: A CSS rule will set the `display` property for this class to `none` once the Taler wallet is installed and enabled. If the wallet is not installed, `display` will be `inherit`.

`taler-installed-show`
: A CSS rule will set the `display` property for this class to `inherit` once the Taler wallet is installed and enabled. If the wallet is not installed, `display` will be `none`.

GNU-LGPL

Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts
as the successor of the GNU Library Public License, version 2, hence the
version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program

by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the *Lesser* General Public License because it does *Less* to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

   A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
   a. The modified work must itself be a software library.
   b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
   c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
   d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

      (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply

to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

   Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

   This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

   If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

   However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

   When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

   You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

   a. Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

   b. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

   c. Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

   d. If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

   e. Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

   For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components

(compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

   b. Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

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

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public 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.

    Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN

WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the library's name and an idea of what it does.
Copyright (C) year  name of author

This library is free software; you can redistribute it and/or modify it
under the terms of the GNU Lesser General Public License as published by
the Free Software Foundation; either version 2.1 of the License, or (at
your option) any later version.

This library is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301,
USA.
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the library
'Frob' (a library for tweaking knobs) written by James Random Hacker.

signature of Ty Coon, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!

GNU-FDL

Version 1.3, 3 November 2008

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 non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

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

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

1. APPLICABILITY AND DEFINITIONS

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

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

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

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

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

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

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

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.

2. VERBATIM COPYING

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

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

3. COPYING IN QUANTITY

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

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

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

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

4. MODIFICATIONS

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

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

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

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

D. Preserve all the copyright notices of the Document.

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

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

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

H. Include an unaltered copy of this License.

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

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

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

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

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

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

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

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

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

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

5. COMBINING DOCUMENTS

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

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

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

6. COLLECTIONS OF DOCUMENTS

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

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

7. AGGREGATION WITH INDEPENDENT WORKS

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

8. TRANSLATION

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

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

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided 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.

10. FUTURE REVISIONS OF THIS LICENSE

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

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

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.

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. . . Texts." line with this:

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

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

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

Concept Index