Educational Problem Manager Design Manual

Robert L. Walton

August 2, 2020

Notice

Table of Contents

1 Introduction

This document gives design information for EPM system maintainers. This document supplements but does not reiterate documentation in the EPM Help Page for users. Comments in code files in turn supplement but do not, with the exception of parameters files, reiterate this document or the Help Page.

2 Definitions and Rules

2.1 Names

1. Names chosen by the user consists of letters, digits, dash(-), and underscore(_), begin with a letter, and end with a letter or digit. See `/include/parameters.php` `$epm_name_re`.
2. Visible file basenames consist of letters, digits, dash(-), and underscore(_), begin with a letter or digit, and end with a letter or digit. Visible extensions, if present, obey the same rules. See `/include/parameters.php` `$epm_filename_re`.
3. Visible problem file basenames end with the problem name, which may optionally be preceded by a dash(-) but not by any other character.
4. Invisible problem file and directory names begin and end with plus(+).
5. Administrative files may follow other rules. In particular, email addresses have a file with a name that is the URL encoded email address, and browser tickets have a file with a name that is the 32 hex digit ticket itself.

2.2 Times

1. Times are formatted as per `/include/parameters.php` which:
   - defines `$epm_time_format` (defaults to `"%FT%T%Z"`)
   - sets the time zone using `date_default_timezone_set`

2.3 Account IDs

1. Account IDs (AIDs) are user choosen names. They are unique to the account and used for both external and internal identification. Once assigned, they cannot be changed.
2. There are two kinds of AIDs: user UIDs for individual users, and team TIDs for teams.

2.4 Random IDs

1. A random ID is a 32 hexadecimal digit number, or equivalently a 128-bit number. Several are generated from /dev/random the first time the server is used, and thereafter they are generated as a pseudo-random sequence using previously generated values to aes-128-cbc encrypt previous values. See `/include/epm_random.php`.

2. Browser TICKETS are random IDs.

3. The $ID variable is a random ID used to validate both POST and GET requests from pages.

   For each tab, and sometimes for the view window, the first GET for the tab or window generates the first $ID value for the pages that will occupy the tab or window, and also generates a random key that is used to generate a sequence of $ID values for the tab or window by encrypting each $ID to generate the next $ID. Thereafter each request is checked to see if it has the right $ID value, and a new $ID value is generated for the next tab or window contents.

   $ID values are generated and checked by `/page/index.php` which is required by all page files.

2.5 Tabs and Windows

1. There are specific tabs and windows for different kinds of transations. The **`main`** tab is for non-problem specific transations. For each account problem there is a problem-specific tab for transactions on that problem. There is a **`view`** pop-up window for looking at files and information, and a separate **`help`** pop-up window for the Help Page and Guide Page.

2. Pages are assigned to windows. E.g., the Login, Project, User, Manage, List Edit, and Favorites Edit Pages are assigned to the main tab, the Problem, Option, and Run Pages are assigned to problem tabs, and the View and Template Pages are assigned to the view pop-up window.

3. ***Page Rule*** At any given time a tab or window has a current page. The page is initially opened with a GET. Subsequent transactions on that page are done with POSTs. Each POST to a page is checked to be sure that page is the current page for its the tab or window type. So if you have opened the Project Page in the main tab, you cannot POST to the User Page.

4. ***Sequence Rule*** Transactions within a tab are sequenced, so that if a transaction is out of sequence the tab becomes ***orphaned*** and must be closed. Sequencing prevents two main tabs from existing at the same time, or two problem tabs for the same problem existing at the same time.

Sequencing is done by random sequence $IDs that are attached to each page. The next request must contain the current $ID else the tab is orphaned. For the main tab the Login Page initializes the tab's $ID sequence. For problem tabs the Problem Page initializes the sequence.

2.6 Directories

1. There are three main directories:

   H, Home Directory: This is the `/epm` directory which is loaded from `github`.

   W, Web Directory: This is the directory named by the EPM server URL. It contains the `index.php` file that is the first file loaded when a user initially contacts the EPM server.

   D, Data Directory: This is the directory containing all the mutable data for the EPM server.

2. The following subdirectories of H contain the EPM files that are directly visible to web clients:

   H/page: Loadable page files. `W/page` is symbolically linked to this directory.

   H/page/downloads: Example files downloadable by the client.

3. The following subdirectories of H contain the EPM files that are not directly visible to web clients:

   H/include: Files that can be '`require`'ed by loadable page files.

   H/bin: Binary executables of programs called by loadable pages or used for off-line maintenance.

   H/template: Templates used to compute client problem files from other client or project files.

   H/doc: Off-line documentation files, including this file.

   H/secure: Source code for binary executables involved with security.

   H/src: Source code for binary executables not involved with security.

   H/setup: Initial contents of D, the data directory, during EPM server setup.

2.7 Page Initialization

1. The web directory, W, contains the following:

symbolic link `W/page` ⟹ `H/page`
symbolic link `W/index.php` ⟹ `W/page/index.php`
`W/parameters.php`, modified `H/include/parameters.php`
`W/maintenance_parameters.php`,
modified `H/include/maintenance_parameters.php`,
(only used off-line)

2. When loaded, a page initializes by executing the following steps:

   - Set `$epm_page_type` to indicate the tab or pop-up window or other type. The possible values are:

     `+main+` and `+problem+` for tabs;

     `+view+` for a view pop-up window that POSTs;

     `+no-post+` for a view pop-up window that does not POST;

     `+download+` for pages that download files so that `<script>` in `/page/index.php` which implements the help button is surpressed (these pages do not POSTing and have no buttons).

   - If the page is the first loaded in a tab or popup-window that POSTs, then it must set `$epm_ID_init` to initialize a new $ID sequence for the tab or popup-window. Otherwise the page leaves `$epm_ID_init` unset.

   - The page requires `/page/index.php` using:

     ```
     require __DIR__ .  '/index.php'
     ```

3. Upon being required, `/page/index.php` executes the following:

   - Computes:

     ```
     $epm_root = ROOT
     $epm_self = SELF
     $epm_web = W
     ```

     where the page currently being loaded has the URL

     ```
     http://HOST/ROOT/SELF
     ```

     `SELF` has the form `page/...`, or if not that, the form `index.php`, HOST is the EPM server host name, and `ROOT` is whatever is left over. Here W is the EPM server web directory (p4) and is

     ```
     $_SERVER['DOCUMENT_ROOT'] . ROOT
     ```

   - If `SELF` is either `index.php` or `page/index.php`, re-routes the request to `page/login.php`.

   - Loads `W/parameters.php` which in turn defines H and D: see p4.

   - Runs the following checks and aborts invalid requests:

     – Checks that the session is logged in, unless the page being loaded is `/page/login.php` or `/page/user/php`.

  - Checks that the client request is using the same IP address as was used for login, unless the `$epm_check_ipaddr` parameter is false.
  - Uses `EPM_ABORT` (p11) to check that no other session has been started after this session using the same login name (team and user ID for a team, or user ID for a user).
- Defines functions and error handlers.
- Except for pages of `+download+` and `+no-post+` type, checks for violations of the Page Rule (p3) and aborts violating requests.
- Except for pages of `+download+` and `+no-post+` type, initializes or checks the $ID to enforce the Sequence Rule (p3), and re-routes violating requests to the `/page/orphan.html` page to declare the tab or window orphaned.
- Except for pages of `+download+` type, defines functions and parameters for use in creating buttons and launching widows, including `<script>` functions.
- Note that may parameters and some functions are defined in `W/parameters.php`. See that file. Also see `/page/index.php` for functions it defines and button parameters it defines.

3 Data Files

Name	Format	Description	Creators	Updaters	Readers
admin	dir	administrative files	login	login user	all
admin/+blocking+	dir	email blocking control file	(editor)	(editor)	login
admin/motd.html	html	message of the day	(editor)	(editor)	login
admin/+lock+	lock	administrative lock file	(lockers)	login user	(lockers)
admin/+random+	lock	random number generator	login	login index	login index
admin/+actions+	lines	log of administrative actions p??	(updaters)	user	view
admin/browser	dir	browser tickets	login	login	login
admin/browser/TICKET	1-line	ticket info p10	login		login
admin/email	dir	email files	user	user	login user
admin/email/EMAIL	1-line	email info p10	user	login user	login user
admin/users **admin/teams**	dir	administrative user directories	user	user	user login
admin/users/UID **admin/teams/TID**	dir	administrative UID user files	user	user login	user login
admin/users/UID/ UID.login **admin/teams/TID/ UID.login**	lines	log of logins p10	(updaters)	login user	(index)
admin/users/UID/ UID.inactive **admin/teams/TID/ UID.inactive**	lines	inactive .login files p13	user		
admin/users/UID/ UID.info	json	user info p12	user	user	user
admin/teams/TID/ TID.info	json	team info p??	user	user	user
admin/users/UID/ +actions+ **admin/teams/TID/ +actions+**	lines	log of accounts's administrative actions p??	(updaters)	user	view

Name	Format	Description	Creators	Updaters	Readers
admin/users/UID/ manager	1-line	teams that UID manages p??	user	user	user
admin/users/UID/ member	1-line	teams of which UID is a member p??	user	user	user
admin/teams/TID/ +read-write+	UID	current read-write user p??	+main+	+main+	+main+
accounts	dir	holds account subdirectories	user	user	all
accounts/AID	dir	account subdirectory	user	problem project	all
accounts/AID/ +lists+	dir	holds account problem lists	list	list favorites	+main+ view
accounts/AID/ +actions+	lines	log of account problem related actions	(updaters)	project run	view
accounts/AID/ PROBLEM	dir	account problem directory	project	+problem+ project	+problem+ project
accounts/AID/ PROBLEM/ +actions+	lines	log of problem related actions	(updaters)	project run	view
accounts/AID/ PROBLEM/ +altered+	empty	alteration indicator p??	(updaters)	problem run	(updaters)
accounts/AID/ PROBLEM/ +changes+	lines	log of changes made by pulls	project	project	
accounts/AID/ PROBLEM/ +work+	dir	working directory for jobs	problem run	problem run	problem run
accounts/AID/ PROBLEM/ +run+	dir	working directory for runs	run	run	run
accounts/AID/ PROBLEM/ ...	various	files visible to users	+problem+	+problem+	+problem+

Name	Format	Description	Creators	Updaters	Readers
`projects`	dir	p??	login	maint	
`solutions`	dir	p??	login	maint	

4 Session Variables

Name	Description	Creators	Updaters	Readers
`EPM_EMAIL`	login email	login		all pages
`EPM_AID`	account ID	login user		all pages
`EPM_UID`	user ID	login user		login user
`EPM_IPADDR`	session IP address	login		index login user
`EPM_TIME`	session time	login		index login user
`EPM_ID_GEN`	$ID generation	index	index	index
`EPM_ABORT`	session abort info	login user		index

5 Web Pages

5.1 Login Page

Login Page Requires

`page/index.php`
`include/epm_random.php`

Login Page Files

`admim/ticket/TICKET`	create	-	read
`admim/email/EMAIL`	-	update	read
`admim/users/AID/UID.login`	-	append	stat

Login Page Session Data

EPM_EMAIL	create	-	-
EPM_AID	create	-	read
EPM_UID	create	-	-
EPM_IPADDR	create	-	read
EPM_TIME	create	-	read
EPM_ABORT	create	-	-
EPM_RW	create	-	-

5.1.1 Login Page File Formats

admin/browser/TICKET (ticket file): T TID EMAIL

TICKET	ticket proper; 32 hexadecimal digit ticket number
T	ticket type; 'c' for confirmation number; 'a' for automatic
TID	team ID or '-' if ticket is for user account
EMAIL	Email address (identifying user account)

- When a person initially logs in to create an account, the UID is not known when the ticket is created.

admin/email/EMAIL (regular email file): UID ACOUNT ATIME

EMAIL	Email address encoded with PHP rawurlencode
UID	user ID
ACOUNT	Number of auto-login periods completed so far.
ATIME	Start time of newest (incomplete) auto-login period.

admin/email/EMAIL (pre-login email file): - TID ...

EMAIL	Email address encoded with PHP rawurlencode
TID	Team user ID (may be more than one)

- This form of email file is created by the Team Page when a team member is assigned the given EMAIL before the member has an account or EMAIL has been added to an existing account. The TID's list all the team user IDs that might have a member which is this EMAIL and not a UID. A TID might be listed whose TID.info file no longer contains the EMAIL.

  When the pre-login form is converted to a regular form, the list of TID's is used to convert any matching EMAIL members in TID.info files to UID(EMAIL) members.

admin/users/AID/UID.login (login log):

Lines of format: TIME EMAIL IPADDR BROWSER

AID	Account ID; equals TID for teams, UID for users
UID	User ID
TIME	Session time (EPM_TIME) for login
EMAIL	Email address (EPM_EMAIL) used for login
IPADDR	IP address (EPM_IPADDR) for session
BROWSER	$_SERVER['HTTP_USER_AGENT'] with '(...)'s removed and horizontal spaces replaced by '**;**'s

- A login with full name AID:UID is valid iff this file exists.
- This file name and modification time is stored in EPM_ABORT and used to abort a session if another session logs in with the same AID:UID.

5.1.2 Login Page Session Variables

EPM_EMAIL	EMAIL entered by user into browser; set by Login Page when either (1) sent by browser, or (2) browser sends TICKET which identifies EMAIL. Generally set before EPM_UID set.
EPM_AID	Account ID, either user or team; set by Login Page when a valid TICKET is received, and set by User Page for new users.
EPM_UID	User ID; equals EPM_AID if that is a user ID. Otherwise the full account name is AID:UID where AID is a team ID. Set when EPM_AID is set.
EPM_IPADDR	Set to $_SERVER['REMOTE_ADDR'] by Login Page if EPM_AID is not yet set.
EPM_TIME	Set to $_SERVER['REQUEST_TIME'] formatted by $epm_format_time by Login Page if EPM_AID is not yet set.
EPM_ABORT	Set to [FILE,MTIME] where MTIME is the mod time of $epm_data/FILE and the session must abort if the mod time of this file changes. Here FILE is admin/users/AID/UID.login to which a line is appended whenever EPM_AID is set for a session.
EPM_RW	Set to false for team login and true for user login. Set whenever EPM_AID is set for a session.

5.1.3 Login Page Transactions

1. If regular form admin/emails/EMAIL exists for EMAIL provided by user to browser, log existing user in and go to Project Page.

2. Otherwise, give the browser a valid ticket, set EPM_EMAIL, and go to User Page.

5.2 User

User Page Files

`admim/browser/BID`	create	update	read
`admim/email/EMAIL`	create	update	read
`admim/users/AID/UID.login`	-	append	stat
`admim/users/AID/UID.inactive`	-	append	stat

User Page Session Data

`EPM_EMAIL`	-	-	read
`EPM_UID`	create	-	read
`EPM_IPADDR`	-	-	read
`EPM_TIME`	-	-	read

5.2.1 User Page File Formats

`admin/users/UID/UID.info` (user info file):

JSON file with the following components:

`'uid'`	UID (PID for person, team UID for team)
`'sponsor'`	PID; missing if not team
`'manager'`	PID; missing if not team
`'emails'`	`[` EMAIL { `,` EMAIL }* `]`; missing if team
`'members'`	`[` MID { `,` MID }* `]`; missing if not team
`'full_name'`	TEXT
`'organization'`	TEXT
`'location'`	TEXT

where

MID	member ID; PID if available or EMAIL not yet assigned to a user account otherwise (EMAIL has '`@`' and PID does not)
TEXT	plain text

- When a team UID.info file is created, MIDs are specified as EMAILs which are resolved if possible to PIDs.

- When a person initially creates an account, all UID.info files are searched and if any have MIDs matching the new account EMAIL, they are resolved to PIDs.

`admin/users/UID/PID.inactive`:
Inactive `.login` file, made by renaming `.login` when UID is no longer a member of AID team. May be reactivated.

5.2.2 User Page Transactions

1. If EPM_UID not set, get data for new user and create new user account if data acceptable.
2. If EPM_UID exists for a user account, display user data and allow it to be edited.
3. If EPM_UID exists for a team account (as discovered by reading UID.info), go to Team Page.

5.3 Problem

Problem Page Files

`users/UID/PROBLEM`	create	update	read	delete
`users/UID/PROBLEM/PROBLEM.tex`	upload	-	read	delete
`users/UID/PROBLEM/PROBLEM.pdf`	create	-	read	delete
`users/UID/PROBLEM/PROBLEM.c`	upload	-	read	delete
`users/UID/PROBLEM/PROBLEM.cc`	upload	-	read	delete
`users/UID/PROBLEM/PROBLEM.java`	upload	-	read	delete
`users/UID/PROBLEM/PROBLEM.py`	upload	-	read	delete
`users/UID/PROBLEM/PROBLEM`	create	-	read	delete
`users/UID/PROBLEM/PROBLEM.class`	create	-	read	delete
`users/UID/PROBLEM/PROBLEM.pyc`	create	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.c`	upload	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.cc`	upload	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.java`	upload	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.py`	upload	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM`	create	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.class`	create	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.pyc`	create	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.in`	upload	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.sin`	create	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.sout`	create	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.fout`	create	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.ftest`	create	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.dout`	create	-	read	delete
`users/UID/PROBLEM/XXXX-PROBLEM.score`	create	-	read	delete

Problem Page Session Data

`EPM_EMAIL`	-	-	read
`EPM_UID`	-	-	read
`EPM_PROBLEM`	create	update	read

6 Overview

Here we list adminstrative files and transactions, and give a brief description of each.

6.1 Administrative Files

EPM uses administrative files to direct generation of files from other files, to keep track of visibility permissions, and for other things. All these files are in JSON format. EPM does not use any data base (like MYSQL).

Administrative files are not uploadable by the user. They are made by commands issued by the user to web pages, e.g., the **PPPP.score** file is made by the user commanding the **PPPP.run** file, and the user can make their own **...-PPPP.run** file using a web page. When administrative commands are made using a web page, the code associated with the page checks that the file being made does not violate security.

The following administrative files the user will encounter. Unless otherwise noted, these are visible to the user.

XXXX.mk
: File specifying how to make the file **XXXX**, when this last file is generated from other files. E.g., the **UUUU/PPPP/00-000-PPPP.out.mk** file tells how to make the **UUUU/PPPP/00-000-PPPP.out** from the **system/PPPP/00-000-PPPP.in**, the **system/PPPP/generate-PPPP**, and **UUUU/PPPP/PPPP**.

  A **XXXX.mk** file also tells how to make itself, so there is no need for **XXXX.mk.mk** files. **XXXX.mk** files are made from template files.

system/template/X.E->Y.F.tpl
: Template file used to make **.mk** or other administrative files. A typical template file name ends with **X.cc->X.tmpl** and is used to make a **X.mk** file that specifies how to make a **X** file from a **X.cc** file, where **X** is a parameter to the template file. In general single upper case letters are used as parameter names for template files, and may also appear in the names of the template files themselves.

  Template files are located and used by code in web pages that make administrative files.

UUUU/credentials
: Credential file for user. Specifies user email addresses, ip addresses, dates ip addresses last certified and last used.

UUUU/logins
: Login history of user.

UUUU/PPPP/uploads
: Upload history of user for problem **PPPP**. Note that a user's uploads for a problem are automatically checked into a per-user, per-problem **git** database which can be cloned by the user and partially inspected (in particular to obtain difference listings) using EPM web pages.

UUUU/PPPP/runs
: Run history of user for problem **PPPP**.

DDDD/user.perm
: Permission control file for arbitrary users for all files in directory **DDDD** (e.g., in **system**, **system/PPPP**) and its subdirectories.

DDDD/UUUU.perm

Permission control file for user **UUUU** and files in directory **DDDD** (e.g., in directory **UUUU**).

In the above, **system/** is actually a project directory. There can be many project, and the **UUUU/PPPP/UUUU.perm** file can point to any of them, in place of **system/**.

6.2 Transactions

The only way a user can interact with EPM is via transactions. Each transaction is executed by entering small amounts of text on a web page and clicking appropriately.

The common transations are:

download
Download any file visible to the user.

inspect
Inspect various visible JSON administrative files in a more readable format.

directory
Create or destroy problem directories and designate a current problem directory.

upload
Upload any program source (**.c**, **.cc**, **.java**, **.py**, or **.lsp**) file or any program input (**.in**) file into the current problem directory.

make
Create, edit, and destroy **.mk** files, whose existence causes the associated derived files to be made upon demand, in the current problem directory.

run
Create, edit, and destroy **.run** files, which define runs that cause batches of derived files to be created, in the currrent problem directory. Execute designated runs.

permission
Create, edit, and destroy **.perm** files that control permissions and visibility.

credentials
Inspect and remove credentials of the current user.

project
Create or destroy project directories and designate project directories that are visible to the current problem directory.

move
Move files between the current problem directory and a project directory.

Program source files do not have to be problem solution files. Any program can be run so long as it opens no files and does all its input/output via file descriptors. Program output can be put into **.out**, **.fout**, **.debug**, **.info**, or **.disp** files. The last kind of file encodes

X-windows commands that can be displayed in an X-window or pdf window or placed in a **.pdf** file. Programs an also interact with terminal windows; for example, a program can be written to calculate combinations (i.e., N choose K).

Problem solution programs are run without arguments, unless they are begin debugged, in which case their output is put into **.debug** files. Other programs can be run with or without arguments.