Educational Problem Manager Design Manual

Robert L. Walton

September 23, 2020

Notice

Table of Contents

1 Introduction . . . 3
2 Definitions and Rules . . . 3
2.1 Names . . . 3
2.2 Times . . . 3
2.3 Account IDs . . . 4
2.4 Random IDs . . . 4
2.5 Tabs and Windows . . . 4
2.6 Directories . . . 6
2.7 Page Initialization . . . 7
2.8 Locking . . . 8
2.9 Security . . . 11
3 Data Files . . . 12
4 Session Variables . . . 15
5 Account Actions . . . 17
6 Job Templates . . . 17
7 Web Pages . . . 20
7.1 Index Page . . . 20
7.1.1 Index Page File Formats . . . 21
7.1.2 Index Page Session Variables . . . 22
7.1.3 Index Page Transactions . . . 23
7.2 Login Page . . . 24

7.2.1 Login Page File Formats 24
7.2.2 Login Page Session Variables 27
7.2.3 Login Page Transactions 27
7.3 User Page 28
7.3.1 User Page File Formats 28
7.3.2 User Page Session Variables 31
7.3.3 User Page Transactions 31
7.4 Problem Page 32
7.4.1 Problem Page File Formats 33
7.4.2 Problem Page Session Variables 34
7.4.3 Problem Page Transactions 35
7.5 Run Page 36
7.5.1 Run Page File Formats 36
7.5.2 Run Page Transactions 36

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.

Instructions for setting up an EPM server are in the file

`include/maintenance_parameters.php`

2 Definitions and Rules

2.1 Names

1. ***User chosen names*** consists of letters, digits, dash(`-`), and underscore(`_`), begin with a letter, and end with a letter or digit. See `/include/parameters.php` `$epm_name_re`.
2. ***Visible file names*** have basenames that consist of letters, digits, dash(`-`), and underscore(`_`), begin with a letter or digit, and end with a letter or digit, and optional extensions that obey same rules. See `/include/parameters.php` `$epm_filename_re`.
3. ***Visible problem file names*** have basenames that 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.
6. ***User IDs*** and ***team IDs*** are user chosen names. An ***account ID*** is either a user ID or a team ID.
7. ***E-mail addresses*** may not have the characters `<`, `>`, `"`, `:`, or space characters.
8. A ***login name*** is either an e-mail address, or an account ID followed by a `:` followed by an e-mail address.

2.2 Times

1. Times are formatted as per `/include/parameters.php` which:

- defines `$epm_time_format` (defaults to `"%FT%T%Z"`, which produces times such as `2020-09-15T07:40:10EDT`)
- sets the time zone using `date_default_timezone_set`

2.3 Account IDs

1. ***Account IDs*** (AIDs) are user chosen names (2.1.1)). 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 (2.1.6).

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 .php files (2.5.4).

2.5 Tabs and Windows

1. Transactions that make changes to the server files are executed in tabs. The ***main tab*** is for non-problem specific transactions. For each account problem there is a ***problem tab*** for transactions on that problem.

   Popup windows are used to display information about server files, without making any changes. The ***help window*** displays help and guide information. The ***documents***

index and ***downloads index*** windows display indices of available documents and downloads. The ***auxiliary window*** displays files and other information. Pages that are loaded into the auxiliary window by default can be loaded into floating windows instead (just by holding down an ALT key when launching the window): each floating window is separarate and specific to its content.

2. Pages are assigned to tabs or windows. The Login, Logout, Project, User, Manage, List Edit, and Favorites Edit Pages are assigned to the main tab; Problem, Option, and Run Pages are assigned to problem tabs; the View and Template Pages are assigned to the auxiliary window; the Documents Page is assigned to the documents index window; the Downloads Page is assigned to the downloads index window; and Help and Guide Pages are assigned to the help pop-up window.

3. ***Page Rule*** At any given time a tab or window that does POSTs has a current page. A GET can change the current page. All transactions done with POSTs are checked to be sure their page is the current page for its the tab or window type. So, for example, if you have just done a GET to the Project Page, you cannot POST to the User Page. Or if you have just done a GET to the Option Page with problem=PPPP, you cannot do a POST to the Problem Page with problem=PPPP.

   This rule is checked by `index.php` which is required at the beginning of all pages in tabs or windows that access the server state.

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.

   Transactions in a window that does POSTs are also sequenced.

   To initialize a sequence, the page must set $epm_ID_init before it requires index.php.

   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.

   Pages that do no posts set $epm_page_type to **`+no-post+`** and do no sequencing. Pages that do views or downloads set $epm_page_type to **`+download+`** and do no sequencing.

   Other pages that provide read-only views but do POSTs (e.g., the View Page) set $epm_page_type to a value unique to the page (e.g., +view+) and set $epm_ID_init on a GET to initiate sequencing for the page's POSTs.

   The sequence rule is checked by `index.php` which is required at the beginning of all pages in tabs or windows that access the server state.

5. ***Stateless Pages*** Pages that do no POSTs are referred to a stateless pages, as they have no session state of their own (they do use state variables such as $aid and $uid). These include .php pages that do no POSTs and also .html pages.

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 a symbolic link to the `index.php` file that is the first file loaded when a user initially contacts the EPM server. In also contains a symbolic link to the H/page directory and edited versions of the include/parameters.php and include/maintenance_parameters.php files.

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

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

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

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/downloads: Example files that can be downloaded. Indexed by the Downloads Index Page.

   H/documents: Documentation files, including this file. Indexed by the Documents Index Page.

   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 (2.6.1), contains the following:

   symbolic link `W/page` ⟶ `H/page`
   symbolic link `W/index.php` ⟶ `page/index.php`
   `W/parameters.php`, edited copy of `H/include/parameters.php`
   `W/maintenance_parameters.php`,
   edited copy of `H/include/maintenance_parameters.php`,
   (only used off-line)

2. When loaded, a .php 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;

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

     `+download+` for pages that download files so that `<script>` in `/page/index.php` which implements the help button is surpressed (these pages do no POSTing and have no buttons).

     OTHER for auxiliary pages that POST. E.g., `+view+` for the View Page.
   - 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 in order:

   - Compute:

     `$epm_root = ROOT` (begins with `/`)
     `$epm_self = SELF` (begins with `/`)
     `$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 (p6) and is

     ```
     $_SERVER['DOCUMENT_ROOT'] . ROOT
     ```
   - If `SELF` is either `/index.php` or `/page/index.php`, re-routes the request to `page/login.php`. The request must be a 'GET' else it is not accepted.
   - Loads `W/parameters.php` which in turn defines H and D (2.6.1).

- Starts the session and clears the file status cache. Sets umask to 07 and Cache-Control header to no-store.
- Runs the following checks and aborts invalid requests:
  - Checks that the client request is using the same IP address as was used for login, unless the `$epm_check_ipaddr` parameter is false.
  - Checks that the session is logged in, unless the page being loaded is `/page/login.php` or `/page/user.php`.
  - Uses `EPM_ABORT` (p27) to check that no other session has been started after this session using the same `AID:UID` login name.
- If the session has logged in, defines:

  $aid = $_SESSION['EPM_AID']
  $uid = $SESSION['EPM_UID']
  $lname = <login name>
  $is_team = $_SESSION['EPM_IS_TEAM']
  $rw = true if page is read-write; false if read-only
  (see `page/index.php` and `include/epm_rw.php` for details)
  $RW _BUTTON= a button to be included in the upper right corner of pages that allow the read-write mode of a session to be changed; set to `''` if not team member login
- Defines functions and error handlers.
- Except for pages of `+download+` and `+no-post+` type, checks for violations of the Page Rule (p5) and aborts violating requests.
- Except for pages of `+download+` and `+no-post+` type, initializes or checks the $ID to enforce the Sequence Rule (p5), and re-routes violating requests to the `/page/orphan.html` page to declare the tab or window orphaned.
- Except for pages of `+download+` type and xhttp requests, and pages run before login is complete, sets up shutdown function that will write statistics into `accounts/AID/+read-write+` or `accounts/AID/+read-only+`.
- Except for pages of `+download+` type and xhttp requests, defines `<script>` functions that handle the refresh keys and launch popup windows.
- 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 more details on the above.

2.8 Locking

In EPM each request is an independent transaction. Locking is needed to keep two requests from interferring with each other.

Some EPM operations consist of multiple requests. However, only the last of these requests modifies EPM server state (that is not in a working subdirectory dedicated to the operation). So the strategy is to have this last request check whether other conflicting requests happened during the operation, and if yes, the last request aborts, does not change EPM state, and produces an error message.

1. ***Session Locking*** At the beginning of each request the PHP `session_start()` function is called. This locks the session file (where session data is stored). As a consequence, given two requests to the same session, one must complete before the other starts.

2. ***Atomic Files*** Some files are shared between sessions and need to be read and written atomically, so that they maintain their format specifications, but need no other locking:

   - `.list` files containing problem lists. Only one session can write such a file, but many may read it.
   - `+priv+` files containing privileges. Its possible but rare for such a file to be writtable by several sessions if it has multiple owners, but if these collide, 'last-writter-wins' is an acceptable implementation. These files can have many readers.

3. ***Tab Uniqueness*** A session is logged into a particular account. Each tab has a type, either 'main' or the name of a problem in the session account. The Sequence Rule (2.5.4) ensures that there is at most one tab of each type at a given time.

   More specifically, if a second tab of a give type is opened by the user for the session, the second tab gets a new sequence of $ID numbers for the tab type, and when the first tab makes a request, its now obsolete $ID number is detected (by `index.php`) and the tab contents are replaced by the `orphan.php` page which announces that the tab is ***orphaned*** and should be closed.

   A similar thing happens if two windows (not-tabs) exist whose pages execute POSTs and have the same $epm_page_type (and consequently are the same .php page). Although such pages make no changes to the EPM file system, they do have session state that must be managed between their original GET and subsequent POSTs.

4. ***Tab Independence*** The pages in each tab, for the most part, operate on different data from the pages in other tabs. A problem tab operates mostly on its problem directory in the account, and the 'main' tab operates mostly on everything else. Therefore, since there is at most one tab of each type, by Tab Uniqueness, most requests are independent of each other.

5. ***Administrative Locking*** Administrative files are those in the `admin` directory tree. Only the Login Page and User Page access most administrative files. These pages both begin by getting an exclusive lock on the `admin` directory using the LOCK function in `parameters.php`.

Administrative files with other or extra considerations are:

- `admin/teams/TID/+read-write+` files: These are themselves locked by `page/index.php` and `include/epm_rw.php`.
- `admin/users/UID/UID.info` files. These are written atomically by the User Page and read atomically by the Problem Page in order to ensure the integrity of their format.

6. ***Read-Write Locking*** For team member logins the `admin/teams/TID/+read-write+` file is locked at the beginning of a request. If it is determined that the account is currently read-only, this file is unlocked immediately, else it is not unlocked until the end of the request (even if the request makes the session read-only).

   If a read-only request attempts to become read-write, the file is re-locked and remains locked till the end of the request (even if the request to become read-write is not successful).

   This sequences requests from read-write team logins for the same team, even if the requests are made by different team members in different sessions.

7. ***Project Problem Locking*** A project problem directory is locked using the LOCK function in `parameters.php` during a push or pull involving the directory. For pushing this is an exclusive lock; for pulling it is a shared lock.

   A push or pull can involve several requests: the first to compile actions and the last to execute them (there can be a request in between that simply presents information stored in the session data by the first request). If the project problem directory changes between requests, because of a push to the directory by another user, the last request could cause data inconsistency. To prevent this the exclusive LOCK time of the directory is monitored to check if some other session has exclusively locked the directory between the first and last requests of an operation. If so, the last request is aborted with an error message and does not execute.

8. ***Local Problem Locking*** When a file is uploaded into a problem or made from another file in the problem, a background job is executed. Similarly when a `.run` file is run, the run is a background job. The local problem directory is not modified by a background job until the job finished, at which time some files may be saved in the local problem directory.

   If the problem has a parent, a shared lock is obtained on the parent using LOCK at the start of the background job, and the LOCK time is checked at the end of the job to be sure it has not changed. If it has, a push to the parent was done during the job, an error is declared for the job, and the job results are not saved in the problem local directory.

   Pulls to a local problem directory can be run in the 'main' tab while a background job is run in the local problem directory in its problem tab. To prevent conflict, every time

the problem directory is altered its `+altered+` file is touched. The modification time of this is checked to detect conflicts and abort either the execution request for a pull or the finishing of an otherwise sucessful background job. Local problem directories can be altered when a background job keeps files, when the Problem Page deletes files, or when the Project Page pulls to the local problem.

2.9 Security

There are two ways to breach an EPM server:

- ***Session Hi-Jacking*** The session is identified by the cookie which is a random number. To hi-jack a session, the hacker must intercept the cookie. A good way to protect against this is to get a certificate for the EPM server so the server uses https.

  As alternate protection, the `parameters.php` file contains a parameter which if set will cause the session to insist that all requests made to it come from the same IP address. This might cause problems for legitimate mobile browsers, but should prevent session hi-jacking.

- ***Illegal Requesting*** Since it is easy to get a user account on an EPM server, a user can try to breach the server by issuing an illegal request from a legitimate session. Therefore each request must be checked to be sure it is legal. If not, an exit with 'UNACCEPTABLE HTTP ...' message is executed.

  An EPM session is definitely not stateless. Not only is there session state, such as the current user logged into the session, but there is state in the EPM server data file system.

  The Page Rule (2.5.3) and Sequence Rule (2.5.4) work together to ensure that a page POSTed to will be be the same page loaded by the last GET to the tab or window doing the POST. This simplifies request checking.

  Request checking is just about checking the request type and request parameters to ensure that the request is legal given the current state of the session, server, and page loaded by the last request for the tab or window.

3 Data Files

for template files see p17 and p??
[xxx] means file modification time is read by xxx

Name	Format	Description	Creators	Updaters	Readers
error.log	lines	error log p21	(all)	(all)	
debug.log	lines	debugging log p22	(all)	(all)	
admin	dir	administrative files	login	login user	index login user
admin/+blocking+	dir	email blocking control file p24	(editor)	(editor)	login
admin/motd.html	html	message of the day p25	(editor)	(editor)	login
admin/+lock+	time	administrative lock file p25	(updaters)	login user	(updaters)
admin/+random+	binary	random number data p25	login	login index	login index
admin/+actions+	lines	log of administrative actions p30	(updaters)	user	view
admin/browser	dir	browser tickets	login	login	login
admin/browser/TICKET	1-line	ticket info p25	login		login
admin/email	dir	email files	user	user	login user
admin/email/EMAIL	1-line	email info p25	user	login user	login user
admin/users **admin/teams**	dir	administrative user/team directories	user	user	user login
admin/users/UID **admin/teams/TID**	dir	administrative account files	user	user	user login
admin/users/UID/ **UID.login** **admin/teams/TID/** **UID.login**	lines	log of logins p26	(updaters)	login user	[index]
admin/users/UID/ **UID.inactive** **admin/teams/TID/** **UID.inactive**	lines	inactive .login files p30	user		

Name	Format	Description	Creators	Updaters	Readers
admin/users/UID/ UID.info	json	user info p28	user	user	user problem
admin/teams/TID/ TID.info	json	team info p29	user	user	user
admin/users/UID/ +actions+ admin/teams/TID/ +actions+	lines	log of accounts's administrative actions p30	(updaters)	user	view
admin/users/UID/ manager	1-line	teams that UID manages p30	user	user	user
admin/users/UID/ member	1-line	teams of which UID is a member p30	user	user	user
admin/teams/TID/ +rw+	UID	current read-write user p30	+main+	+main+	+main+ index
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 p17	(updaters)	project run	view
accounts/AID/ +read-write+	lines	log of account read-write mode requests p22	index	index	[+main+]
accounts/AID/ +read-only+	lines	log of account read-only mode requests p22	index	index	[+main+]

Name	Format	Description	Creators	Updaters	Readers
accounts/AID/ PROBLEM	dir	account problem directory	project	+problem+ project	+problem+ project
accounts/AID/ PROBLEM/ +actions+	lines	log of problem related actions p17	(updaters)	project run	view
accounts/AID/ PROBLEM/ +altered+	empty	alteration indicator p33	(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 p34	problem run	problem run	problem run
accounts/AID/ PROBLEM/ +run+	dir	working directory for runs p36	run	run	run
accounts/AID/ PROBLEM/ ...	various	files visible to users p32	+problem+	+problem+	+problem+
projects	dir	p??	login	maint	
lists	dir	links to published lists	list	list	favorites list project view manage
default	dir	default program binaries	setup		
+web-save+	dir	backup of W	backup	backup	backup
+web+	link	link to W	setup		

setup is setup function of epm/bin/epm
backup is backup function of epm/bin/epm

4 Session Variables

Important: See Global Variables defined by index.php on p21.

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 manage
`EPM_IS_TEAM`	true iff AID is team ID	login user		index
`EPM_PAGE[id_type]`	current session page	index	index	index
`EPM_IPADDR`	session IP address	login		index login user
`EPM_TIME`	session time	login		index login user
`EPM_ID_GEN[id_type]`	$ID generation	index	index	index
`EPM_ABORT`	session abort info	login user		index

id_type is $epm_page_type if this is not
+problem+, +no-post+, or +download+,
or is PROBLEM if $epm_page_type is +problem+

Name	Description	Creators	Updaters	Readers
EPM_PROJECT	permanent data for project page	project	project	project
EPM_USER	permanent data for user page	user	user	user
EPM_MANAGE	permanent data for manage page	manage	manage	manage
EPM_VIEW	permanent data for view page	view	view	view
EPM_PROBLEM[problem]	permanent data for problem page	problem	problem	problem
EPM_WORK[problem]	data for current background task	problem	problem	problem
EPM_RUN[problem]	data for current background run	run	run	run

'problem' is the problem name of the tab

5 Account Actions

Actions are recorded in `+actions+` files. The following describes actions affecting `accounts` directory contents. For actions affecting the `admin` directory contents, see p30.

An account `+actions+` file consists of lines of format:

TIME AID TYPE PROJECT NAME DATA...

TIME	Session time for login (EPM_TIME)
AID	AID of account performing action (may be team ID or user ID)
TYPE	see below
PROJECT	project involved in action, or `-` if none
NAME	problem involved in action, or name of list involved in action
DATA	TYPE-specific data, or empty

The possible TYPEs and their corresponding DATA... fields are:

submit	BASENAME SOLUTION-TIME SCORE...
push	
pull	
create-problem	
creat-list	
publish-list	
unpublish-list	
update-priv	

For a 'submit' action, the .run file is BASENAME.run and the run output file is BASENAME.rout, the SOLUTION-TIME is the maximum number of CPU seconds taken by the solution for any one test case .in file, and the SCORE... is the submission score ('Completely Correct' or otherwise).

All account actions are appended to the `accounts/AID/+actions+` file.

If a PROJECT is involved in an action, the action is also appended to the `project/PROJECT/+actions+` file.

If a PROJECT and PROBLEM are involved in an action, the action is also appended to both the `project/PROJECT/PROBLEM/+actions+` file and the `accounts/AID/PROBLEM/+actions+` file.

6 Job Templates

Job templates are used to make a file XXXX.dext from a file XXXX.sext. A job template is a `.tmpl` file in the `H/template` directory that encodes a PHP array in JSON.

A `.tmpl` file is used to make a source file from a destination file and has a name of the form:

`BBBBBBBB.SSS:BBBBBBBB.DDD:QQQQQQQQ.tmpl`

where `BBBBBBBB` is the basename of the source and destination file
`.SSS` is the extension of the source file
`.DDD` is the extension of the destination file and may be empty
`QQQQQQQQ` is a qualifier, and may be any text, such as '`JAVA`', '`JAVA-UPLOAD`', '`SUBMIT`', etc.

Some examples are:

`PPPP.cc:PPPP:.tmpl`
make PPPP binary from PPPP.cc C++ source

`XXXX-PPPP.in:XXXX-PPPP.score:.tmpl`
make .score file from .in file using PPPP binary

`XXXX-PPPP.in:XXXX-PPPP.score:JAVA.tmpl`
make .score file from .in file using PPPP.jar binary

Sequences of 4 identical capital letters in a row in the basename are replaced by text from the source file name when a template file is used. `PPPP` is always replaced by the problem name. For example, if given the source file `01-000-reverser.in`, the destination file `01-000-reverser.score` may be made by either of the last two template file named above. Which if the two template files is used is determined by the contents of the files and the problem directory, as described below.

A `.tmpl` file JSON-encodes a PHP array with the following components:

`'COMMANDS' => ['line',... ]`
A list of command lines that is parsed into a sequence of commands. A command consists of a consecutive sequence of lines all but the last of which end in \.
In JSON each line is enclosed in double quotes " and \is represented by \\.

`'LOCAL-REQUIRES' => ['filename',... ]`
Locally required files. A list of files in the 'local' `accounts/AID/PROBLEM` directory that must exist if the template is to be used. These files are linked into the working directory in which the commands are executed.
For example, `XXXX-PPPP.in:XXXX-PPPP.score:.tmpl`
lists `PPPP` as locally required, whereas
`XXXX-PPPP.in:XXXX-PPPP.score:JAVA.tmpl`
lists `PPPP.jar` as locally required.

`'REMOTE-REQUIRES' => ['filename',... ]`
Remote required files. Like locally required files but the files must be in the 'remote' `accounts/AID/PROBLEM/+parent+` directory.

`'REQUIRES' => ['filename',... ]`

Required files. Like above but files can be in either the local or remote directories; the local directory version is perferred if both exist.

`'CREATABLE' => ['filename',... ]`
Creatable files. If file does not otherwise exist and is listed in REQUIRES or LOCAL-REQUIRES, the file will be automatically created in the local directory. Note that a REQUIRES file which exists in the remote directory will be used from that directory and not created.

`'KEEP' => ['filename',... ]`
Files that are to be 'kept', that is, moved to the local problem directory at the end of the job, if there are no errors or failed checks (see CHECKS).

`'CHECKS' => ['check',... ]`
Checks to be run upon job completion before keeping any files. A check has the form ['filename',...]. For all but the last check, this means that the first named file should be non-existant or empty, else the check fails, and if the check fails, the files listed in the check should be shown to the user as proof that the check failed.

An example check is `["PPPP.cerr","PPPP.cc"]` which checks that the compiler .cerr error output file is empty and if not shows it and the .cc source code file.

The last check is the list of files that should be shown to the user if the job has no error, none of the previous checks fail, and the KEEP files are moved to the local problem directory. The last check is typically empty, that is [], but the previous checks cannot be empty.

A check can also be just 'filename', which is equivalent to ['filename'].

`'CONDITIION' => 'condition'`
A condition to be satisfied if the template is to be used. Most templates have no CONDITION. The possible conditions are:

'UPLOAD filename'
The job is being run to verify the named file which has just been uploaded using the Problem Page.

'SUBMIT'
The job is being run as part of a Run Page submission.

7 Web Pages

7.1 Index Page

The Index Page is required by every other EPM .php page and does initial setup for all EPM .php pages. It also functions as the initial file for accessing the EPM server and reroutes these accesses to the Login Page.

Index Page Requires

`include/parameters.php`	
`include/epm_abort.php`	only if aborting
`include/epm_random.php`	only for 'GET's to pages setting $epm_ID_init

Index Page Files

`error.log`	create	append	-
`debug.log`	-	-	-
`admin/teams/AID/+rw+`	create	lock	read
`accounts/AID/+read-write+`	create	append	-
`accounts/AID/+read-only+`	create	append	-

Index Page Session Data

`EPM_IPADDR`	-	-	read
`EPM_AID`	-	-	read
`EPM_UID`	-	-	read
`EPM_EMAIL`	-	-	read
`EPM_IS_TEAM`	-	-	read
`EPM_ABORT`	-	-	read
`EPM_TIME`	-	-	read
`EPM_PAGE[$id_type*]`	create	update	read
`EPM_ID_GEN[$id_type*]`	create	update	read

* $id_type is "+main+" for main tab page,
PROBLEM name for problem tab page, and
"+view+" for view window page

Index Page Global Data

The following are global variables set just before index.php is required by another page.

$epm_page_type	One of:	+main+	main tab page
		+problem+	some problem tab page
		+view+	view window page
		+no-post+	view window page that does no POSTs
		+download+	page that just downloads a file
$epm_ID_init	If set re-initializes $ID generator; see $ID below.		

The following are global variables defined by index.php when it is required at the beginning of an EPM .php page, and usable by the remainder of that page. For other such parameters, see the include/parameters.php file.

$epm_method	the request method, either 'GET' or 'POST'
$epm_root	ROOT and SELF, where the URL used to access a page
$epm_self	is HOST/ROOT/SELF and SELF either has the form /page/... or the form /index.php
$epm_web	the EPM web directory W (2.6.1)
$rw	true if request is being processed in read-write mode; false if in read-only mode
$aid	AID ($_SESSION['EPM_AID']) if set
$uid	UID ($_SESSION['EPM_UID']) if set
$lname	login name, either AID if AID == UID, or AID:EMAIL if AID != UID
$is_team	true iff AID is a team ID so login is a team member login ($_SESSION['EPM_IS_TEAM'])
$ID	the identifier which must be presented by the next request (as ?id=$ID) unless the next page requested sets $epm_ID_init; see EPM_ID_GEN on p22
$data	same as $_SESSION['EPM_PAGE'][$id_type]; used for per tab or view window data: see p22
$state	same as $data['STATE']; set to 'normal' by index.php on 'GET'

7.1.1 Index Page File Formats

error.log:

Records all PHP error messages that would normally be in the HTTP server log, including messages generated by the ERROR and WARN functions (see index.php).

Contains lines of format: CLASS ERRNO SELF AID (TIME)

followed by a stack trace. Here

CLASS	{USER,EPM,SYSTEM}_{WARNING,NOTICE,ERROR}
ERRNO	PHP error number
SELF	page name relative to W (EPM_SELF)
AID	account ID, or EMAIL if account ID not available
TIME	session time (EPM_TIME), if available

debug.log:
Contains lines output by the DEBUG function in parameters.php. Not actually specific to index.php or any page. See $epm_debug and the DEBUG function in parameters.php.

admin/teams/AID/+rw+: See p30

accounts/AID/+read-only+:
accounts/AID/+read-write+:
One line is appended at the end of each http request, of the form:

BEGIN-TIME END-TIME SELF AID UID

where

BEGIN-TIME	request processing start time in seconds
END-TIME	request processing end time in seconds
SELF	page name relative to W (EPM_SELF)
AID	account ID
UID	user ID

Times are typically to the nearest microsecond.

The +read-write+ file is written if the request is read-write at its end, or the +read-only+ file is written if the request is read-only. The modification time of the +read-write+ file is used to determine the last time the account made a read-write request.

Download and xhttp requests are not recorded.

7.1.2 Index Page Session Variables

In the following $id_type is $epm_page_type when the latter is '+main+' or '+view+' and the problem name when the latter is '+problem+'.

EPM_PAGE[$id_type]:
Data specific to a particular tab or to the view window. Same as **$data** global variable. Re-initialized on a 'GET' by index.php to:

['SELF' => $epm_self, 'STATE' => 'normal']

Checked by index.php for 'POST's to be sure they target the page of the last 'GET' for the given $id_type.

EPM_ID_GEN[$id_type]:

The list [VALUE, KEY, IV] used to generate $ID values. The next $ID value is VALUE. When it is set, a new VALUE is generated by encrypting the old VALUE by KEY with IV as the initialization vector (it is always 0 and is included here as an optimization). Re-initialized for pages that set $epm_ID_init; checked for other pages of $epm_page_type "+main", "+problem+", or "+view+".

For session variables just read by index.php, see other EPM pages.

7.1.3 Index Page Transactions

See Page Initialization (2.7.3).

7.2 Login Page

Login Page Requires

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

Login Page Files

`admin/+blocking+`	-	-	read
`admin/motd.html`	-	-	read
`admin/+lock+`	create	update	read
`admin/+random+`	create	update	read
`admin/browser/TICKET`	create	-	read
`admin/email/EMAIL`	-	update	read
`admin/users/UID/UID.login`	-	append	stat
`admin/users/UID/GID.login`	-	append	stat
`admin/teams/TID/MID.login`	-	append	stat

Login Page Session Data

`EPM_EMAIL`	create	-	-
`EPM_AID`	create	-	read
`EPM_UID`	create	-	-
`EPM_IS_TEAM`	create	-	-
`EPM_IPADDR`	create	-	read
`EPM_TIME`	create	-	read
`EPM_ABORT`	create	-	-

7.2.1 Login Page File Formats

`admin/+blocking+`:
Lines of format: SIGN RE
SIGN `+` to not block, `-` to block
RE regular expression matched to the entire email name (e.g., `.*` matches any email name and `.*\.edu` matches any email name ending in `.edu`)

- The lines are read in order and the first line with RE matching the login name EMAIL is used to not block or block the EMAIL. If no line matches, the EMAIL is not blocked.
- Blank lines and whose first non-whitespace character is `#` are ignored. Various forms of within-line whitespace are equivalent, and whitespace at beginning or end of a line is ignored.

admin/motd.html:
An HTML file that is included inside a `<div>...</div>` block that gives the 'message of the day' on the Login Page. Typically this file consists of some `<p>` paragraphs. If the file does not exist, the `<div>...</div>` block is not created.

admin/+lock+:
All transactions within the `admin` directory (i.e., all http requests that access files or subdirectories within `admin`) begin by calling the `parameters.php LOCK` function to lock the `admin` directory. This function locks the directory by creating if necessary and locking this `+lock+` file for the course of the transaction. Note there are no EPM transactions longer than a single http request.

admin/+random+:
The pseudo-random number generator in `include/epm_random.php` exclusively creates, updates, an reads this file.

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

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

- When a user 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 User 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 IDs that might in their `TID.info` file have a member which has this EMAIL and no 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/UID/UID.login` (login log):
`admin/teams/TID/UID.login` (login log):
`admin/users/UID/GID.login` (login log):

Lines of format: TIME EMAIL IPADDR BROWSER

UID	User ID
TID	Team ID
GID	Guest User ID
TIME	Session time for login (EPM_TIME)
EMAIL	Email address used for login (EPM_EMAIL)
IPADDR	IP address for session (EPM_IPADDR)
BROWSER	$_SERVER['HTTP_USER_AGENT'] with '(...)'s removed and horizontal spaces replaced by '`;`'s

- A login with name AID:EMAIL is valid iff the file `.../AID/UID.login` exists for UID the user ID associated with EMAIL.
- Upon login, a line is appended to the appropriate the `.login` file, and then that file's name and modification time are stored in EPM_ABORT and used to abort a session if another session logs in with the same AID:EMAIL and appends to the file, thus changing its modification time.

7.2.2 Login Page Session Variables

`EPM_EMAIL`	EMAIL entered by user into browser; set by Login Page when either (1) EMAIL is to be transferred to user.php for a new user, or (2) browser sends TICKET which identifies EMAIL and EPM_UID is being 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. This equals EPM_UID for a user login, is the team ID for team member login, and is the host user ID of the EMAIL guest for a guest login.
`EPM_UID`	User ID associated with EPM_EMAIL. Set when EPM_AID is set.
`EPM_IS_TEAM`	True iff EPM_AID is team ID; Set when EPM_AID is set.
`EPM_IPADDR`	Set to $_SERVER['REMOTE_ADDR'] by Login Page when 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.

7.2.3 Login Page Transactions

1. If regular form admin/emails/EMAIL exists log existing user in and go to Project Page. The browser first gets a ticket which it sends to the server, and the ticket specifies the EMAIL.
   (a) Browser can look EMAIL up in browser's local memory to get ticket to send to server, or if this ticket does not exist or is invalid,
   (b) Browser can send EMAIL to server and get confirmation number back to use as a ticket.
2. Otherwise, if no regular form admin/emails/EMAIL exists, give the browser a confirmation number to use as ticket, and upon receiving this set EPM_EMAIL and give the browser a new automatic ticket and instruct the browser to go to User Page to create new user.

7.3 User Page

User Page Files

`admin/email/EMAIL`	create	update	read
`admin/users/UID/UID.info`	create	update	read
`admin/teams/TID/TID.info`	create	update	read
`admin/users/UID/UID.login`	-	append	stat
`admin/users/UID/GID.login`	-	append	stat
`admin/teams/TID/UID.login`	-	append	stat
`admin/users/UID/UID.inactive`	create	-	-
`admin/users/UID/GID.inactive`	create	-	-
`admin/teams/TID/UID.inactive`	create	-	-
`admin/users/UID/manager`	create	update	read
`admin/users/UID/member`	create	update	read
`admin/teams/TID/+rw+`	create	update	read
`admin/users/UID/+actions+`	create	append	-
`admin/teams/TID/+actions+`	create	append	-
`admin/+actions+`	create	append	-

User Page Session Data

`EPM_USER`	create	update	read
`EPM_EMAIL`	-	-	read
`EPM_AID`	create	-	read
`EPM_UID`	create	-	read
`EPM_IS_TEAM`	create	-	read
`EPM_IPADDR`	-	-	read
`EPM_TIME`	-	-	read
`EPM_ABORT`	create	-	-

7.3.1 User Page File Formats

`admin/email/EMAIL`: see p25

`admin/users/UID/UID.info` (user info file):
JSON file with the following components:

`'uid'`	UID
`'emails'`	`[` EMAIL { `,` EMAIL }* `]`
`'guests'`	`[` GID { `,` GID }* `]` (may be missing)
`'full_name'`	TEXT
`'organization'`	TEXT
`'location'`	TEXT

where

UID	user ID (i.e., an account ID) for user; cannot be changed once account is created
EMAIL	e-mail address for user
GID	UID for guest of user
TEXT	plain text (with a minimum and maximum allowed length)

- 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/TID/TID.info` (user info file):

JSON file with the following components:

`'tid'`	TID
`'manager'`	MANAGER
`'members'`	**`[`** MEMBER { **`,`** MEMBER }* **`]`** (may be missing)
`'full_name'`	TEXT
`'organization'`	TEXT
`'location'`	TEXT

where

TID	team ID (i.e., an account ID) for team; cannot be changed once account is created
MANAGER	UID of the manager of team
MEMBER	one of: MID (EMAIL) MID(EMAIL)
MID	UID of member of team
EMAIL	EMAIL of member of team as of time member was added to team
TEXT	plain text (with a minimum and maximum allowed length)

- A MEMBER may be specified as an EMAIL or a MID. If specified as an EMAIL, and a regular `admin/email/EMAIL` (p25) exists, the MID is added. If specified as an EMAIL, and no regular `admin/email/EMAIL` file exists, the TID is added to a pre-login `admin/email/EMAIL` (p25), which is created if it does not exist.
- When a user initially creates an account with an EMAIL for which a pre-login `admin/email/EMAIL` file exists, the TID.info files for all TIDs listed in the pre-login file are searched for any MEMBERs of the form '(EMAIL)', and when one is found, its MID is added to it. Similarly if EMAIL is added to an existing user account.

admin/users/UID/UID.login (login log):
admin/teams/TID/UID.login (login log):
admin/users/UID/GID.login (login log): see p26

admin/teams/TID/UID.inactive:
admin/users/UID/GID.inactive:
Inactive `.login` file, made by renaming `.login` file when UID is no longer a member of TID team or GID is no longer a guest of UID. May be reactivated by renaming to `.login` file.

admin/users/UID/manager:
A a list of single space separated TIDs of the teams of which user UID is a manager.

admin/users/UID/member:
A a list of single space separated TIDs of the teams of which user UID is a member.

admin/teams/TID/+rw+:
Either a single UID of the team member whose login currently has read-write mode, or blank if no such. This file is locked by itself for exclusive use by team member login requests, and this is independent of any `+lock+` file locking. The lock is released immediately for read-only requests, but is held to the end of read-write requests.

admin/users/UID/+actions+:
admin/teams/TID/+actions+:
Lines of format: TIME AID info KEY OP VALUE

TIME	Session time for login (EPM_TIME)
AID	equals UID or TID from file name
KEY	`.info` file JSON key
OP	`=` if non-list KEY reset, `+` if addition to KEY's list, `-` if deletion from KEY's list
VALUE	value given to non-list KEY, added to KEY's list, or deleted from KEY's list

- Updates to `AID.info` file are logged by writting lines to `admin/*/AID/+actions+` file.

admin/+actions+:
Any line written to an `admin/*/AID/+actions+` file is also written to this file.

7.3.2 User Page Session Variables

EPM_USER User Page Permanent State:
- **'UID' =>** currently selected user
- **'TID' =>** currently selected team
- **'TID_LIST' =>** currently selected team list; one of:
  - **'all'** all teams
  - **'manager'** teams for which UID is the manager
  - **'member'** teams for which UID is a member

Other see Login Page Session Variables, p27

7.3.3 User Page Transactions

1. If EPM_UID not set, get data for new user and create new user account if data acceptable. Otherwise, or after creating new user account, display `.info` data for all users and all teams.
2. Allow the current user to edit their own `.info` data.
3. If the current user is the manager of a team, allow that team's `.info` data to be edited.
4. Allow the current user to create a new team of which the current user is a manager.
5. NOTE: team member and guest logins cannot edit user or team `.info` or create new teams.
6. Allow a current read-only user to force a switch to read-write.

7.4 Problem Page

Problem Page Visible Files

`accounts/AID/PROBLEM`	create	update	read	delete
`accounts/AID/PROBLEM/PROBLEM.tex`	upload	-	read	delete
`accounts/AID/PROBLEM/PROBLEM.pdf`	create	-	read	delete
`accounts/AID/PROBLEM/PROBLEM.c`	upload	-	read	delete
`accounts/AID/PROBLEM/PROBLEM.cc`	upload	-	read	delete
`accounts/AID/PROBLEM/PROBLEM.java`	upload	-	read	delete
`accounts/AID/PROBLEM/PROBLEM.py`	upload	-	read	delete
`accounts/AID/PROBLEM/PROBLEM`	create	-	read	delete
`accounts/AID/PROBLEM/PROBLEM.jar`	create	-	read	delete
`accounts/AID/PROBLEM/PROBLEM.pyc`	create	-	read	delete
`accounts/AID/PROBLEM/YYYY-PROBLEM.c`	upload	-	read	delete
`accounts/AID/PROBLEM/YYYY-PROBLEM.cc`	upload	-	read	delete
`accounts/AID/PROBLEM/YYYY-PROBLEM.java`	upload	-	read	delete
`accounts/AID/PROBLEM/YYYY-PROBLEM.py`	upload	-	read	delete
`accounts/AID/PROBLEM/YYYY-PROBLEM`	create	-	read	delete
`accounts/AID/PROBLEM/YYYY-PROBLEM.jar`	create	-	read	delete
`accounts/AID/PROBLEM/YYYY-PROBLEM.pyc`	create	-	read	delete
`accounts/AID/PROBLEM/XXXX-PROBLEM.in`	upload	-	read	delete
`accounts/AID/PROBLEM/XXXX-PROBLEM.sin`	create	-	read	delete
`accounts/AID/PROBLEM/XXXX-PROBLEM.sout`	create	-	read	delete
`accounts/AID/PROBLEM/XXXX-PROBLEM.fout`	create	-	read	delete
`accounts/AID/PROBLEM/XXXX-PROBLEM.dout`	create	-	read	delete
`accounts/AID/PROBLEM/XXXX-PROBLEM.ftest`	create	-	read	delete
`accounts/AID/PROBLEM/XXXX-PROBLEM.score`	create	-	read	delete
`accounts/AID/PROBLEM/ZZZZ-PROBLEM.run`	upload	-	read	delete
`accounts/AID/PROBLEM/XXXX-PROBLEM.rout`	-	-	read	delete

Special values for YYYY: **`generate`**, **`filter`**, **`monitor`**, **`display`**

Problem Page Parent Files

`projects/PROJECT/PROBLEM`	-	-	read	-
`projects/PROJECT/PROBLEM/PROBLEM.pdf`	-	-	read	-
`projects/PROJECT/PROBLEM/PROBLEM.optn`	-	-	read	-
`projects/PROJECT/PROBLEM/generate-PROBLEM`	-	-	read	-
`projects/PROJECT/PROBLEM/filter-PROBLEM`	-	-	read	-
`projects/PROJECT/PROBLEM/display-PROBLEM`	-	-	read	-
`projects/PROJECT/PROBLEM/monitor-PROBLEM`	-	-	read	-
`projects/PROJECT/PROBLEM/XXXX-PROBLEM.in`	-	-	read	-
`projects/PROJECT/PROBLEM/XXXX-PROBLEM.ftest`	-	-	read	-
`projects/PROJECT/PROBLEM/ZZZZ-PROBLEM.run`	-	-	read	-

Problem Page Maintenance Files

`accounts/AID/PROBLEM/+parent+`	-	-	read	-
`accounts/AID/PROBLEM/PROBLEM.optn`	-	-	read	-
`accounts/AID/PROBLEM/+altered+`	create	touch	stat	-

Problem Page Work Files

`accounts/AID/PROBLEM/+work+/XXXX-PROBLEM.sh`	create	-	-	-
`accounts/AID/PROBLEM/+work+/XXXX-PROBLEM.shout`	create	append	read	-
`accounts/AID/PROBLEM/+work+/XXXX-PROBLEM.sherr`	create	append	-	-
`accounts/AID/PROBLEM/+work+/XXXX-PROBLEM.*stat`	create	update	read	-

Other files are linked into the `+work+` directory as per templates
or are created by template commands

Problem Page Session Data

`EPM_PROBLEM[problem]`	create	update	read

problem is the value of the `'problem'`
parameter to page GETs and POSTs

7.4.1 Problem Page File Formats

Visible Files: see Help Page

Parent Files: see Help Page and following

accounts/AID/PROBLEM/+parent+:
symbolic link to `projects/PROJECT/PROBLEM`

accounts/AID/PROBLEM/PROBLEM.optn: see p**??**

accounts/AID/PROBLEM/+altered+:

empty file; only modification time is used; this file is touched by every transaction that creates, deletes, or modifies a file in the problem directory

accounts/AID/PROBLEM/+work+:
directory created when commands from a template are to be executed; the commands run with this directory as the current directory

accounts/AID/PROBLEM/+work+/XXXX-PROBLEM.sh:
This is the bash script that is executed to run the template determined job. Before each bash command `n` is set to specifiy the command. The commands in order are:

- n=B bash initialization commands setting traps for signals and command errors and writing PID in first output line
- n=# template command beginning on template command line number #
- n=D '`exit 0`' command after template commands

Upon starting, the script writes to the .shout file the line:

pid `PID`

where *pid* is the ID of the process running the script (which can be used to send the script a stop signal).

On termination the script writes to the .shout file the line:

`::`*n e* `DONE`

where *n* is the value of `n` at the beginning of the last bash command executed, and *e* is the exit code.

See include/epm_make.php for more details.

accounts/AID/PROBLEM/+work+/XXXX-PROBLEM.shout:
This is the standard output of the bash shell script. Template commands redirect their standard output so it does not appear here.

accounts/AID/PROBLEM/+work+/XXXX-PROBLEM.sherr:
This is the standard error of the bash shell script. Template commands redirect their standard error so it does not appear here.

7.4.2 Problem Page Session Variables

EPM_PROBLEM[problem]:
Problem Page Permanent State

EPM_PROBLEM[problem]['ORDER']: currently selected problem list order (see Help Page); one of: **'extension'**
'lexical'
'recent'

7.4.3 Problem Page Transactions

1. Display visible problem files.
2. Display visible +work+ directory files.
3. Display commands last executed.
4. Update command list with execution times and error messages.
5. Change order of visible files in listings (lexical, by-extension, or most-recent-first)
6. Make XXXX.FFF from XXXX.EEE using template.
7. Upload file XXXX.EEE and make XXXX.FFF from it using template (FFF is a function of EEE).
8. Link XXXX.EEE to YYYY-XXXX.EEE.
9. Start run of XXXX.run file (run finished by Run Page)
10. Delete selected visible files.
11. Delete problem.
12. Delete +work+ directory.

7.5 Run Page

Run Page Visible Files

`accounts/AID/PROBLEM/ZZZZ-PROBLEM.run`	-	-	read	-
`accounts/AID/PROBLEM/XXXX-PROBLEM.rout`	create	append	read	-
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.rerr`	create	append	read	-

Run Page Run Files

`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.sh`	create	-	-	-
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.shout`	create	append	read	-
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.sherr`	create	append	-	-
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.stat`	create	append	read	-
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.run`	link	-	read	-
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.rout`	create	append	read	-
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.rerr`	create	append	read	-

7.5.1 Run Page File Formats

`accounts/AID/PROBLEM/+run+`:
: directory created when a run is to be executed; the run executes with this directory as the current directory

`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.sh`:
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.shout`:
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.sherr`:
: These have the same format and purpose as the similar `+work+` files (p34). Also see Run Page Transactions Below.

`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.run`:
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.rout`:
`accounts/AID/PROBLEM/+run+/XXXX-PROBLEM.rerr`:
: See Run Page Transactions Below.

7.5.2 Run Page Transactions

1. Display visible run files.
2. Execute runs. A run is executed in a `+run+` directory using a template consisting of the single command:

```
$BIN/epm_run s XXXX-PROBLEM.run \
     accounts/AID/PROBLEM/+work+ XXXX-PROBLEM.stat \
     > XXXX-PROBLEM.rout 2> XXXX-PROBLEM.rerr
```

The resulting .sh, .shout, .sherr, .rout, and .rerr files are as they would be for commands executed from a template in the `+work` directory.

See epm_run document for more details.