Session Upload Progress

When the session.upload_progress.enabled INI option is enabled, PHP will be able to track the upload progress of individual files being uploaded. This information isn't particularly useful for the actual upload request itself, but during the file upload an application can send a POST request to a separate endpoint (via XHR for example) to check the status.

The upload progress will be available in the `$_SESSION` superglobal when an upload is in progress, and when POSTing a variable of the same name as the session.upload_progress.name INI setting is set to. When PHP detects such POST requests, it will populate an array in the `$_SESSION`, where the index is a concatenated value of the session.upload_progress.prefix and session.upload_progress.name INI options. The key is typically retrieved by reading these INI settings, i.e.

```
<?php
$key = ini_get("session.upload_progress.prefix") .
$_POST[ini_get("session.upload_progress.name")];
var_dump($_SESSION[$key]);
?>
```

It is also possible to *cancel* the currently in-progress file upload, by setting the `$_SESSION[$key]["cancel_upload"]` key to ENTITY. When uploading multiple files in the same request, this will only cancel the currently in-progress file upload, and pending file uploads, but will not remove successfully completed uploads. When an upload is cancelled like this, the `error` key in `$_FILES` array will be set to `UPLOAD_ERR_EXTENSION`.

The session.upload_progress.freq and session.upload_progress.min_freq INI options control how frequent the upload progress information should be recalculated. With a reasonable amount for these two settings, the overhead of this feature is almost non-existent.

Example information

Example of the structure of the progress upload array.

```
<form action="upload.php" method="POST" enctype="multipart/form-data">
 <input type="hidden" name="<?php echo ini_get("session.upload_progress.name"); ?>"
```

```
value="123" />
 <input type="file" name="file1" />
 <input type="file" name="file2" />
 <input type="submit" />
</form>
```

The data stored in the session will look like this:

```
<?php
$_SESSION["upload_progress_123"] = array(
 "start_time" => 1234567890,   // The request time
 "content_length" => 57343257, // POST content length
 "bytes_processed" => 453489,  // Amount of bytes received and processed
 "done" => false,              // true when the POST handler has finished, successfully or not
 "files" => array(
  0 => array(
   "field_name" => "file1",       // Name of the <input/> field
   // The following 3 elements equals those in $_FILES
   "name" => "foo.avi",
   "tmp_name" => "/tmp/phpxxxxxx",
   "error" => 0,
   "done" => true,                // True when the POST handler has finished handling this
file
   "start_time" => 1234567890,    // When this file has started to be processed
   "bytes_processed" => 57343250, // Number of bytes received and processed for this file
  ),
  // An other file, not finished uploading, in the same request
  1 => array(
   "field_name" => "file2",
   "name" => "bar.avi",
   "tmp_name" => NULL,
   "error" => 0,
   "done" => false,
   "start_time" => 1234567899,
   "bytes_processed" => 54554,
  ),
 )
);
```

The web server's request buffering has to be disabled for this to work properly, else PHP may see the file upload only once fully uploaded. Servers such as Nginx are known to buffer larger requests.

The upload progress information is written to the session before any scripts are executed. Therefore changing the session name via `ini_set` or `session_name` will give a session without the upload progress information. This also means that user session handlers are never used for the session information for upload progress information, since there is no code that could have set the session handler to be a user handler by calling `session_set_save_handler`.

□□□ #2
□ □□□ □□□ 2026-10-31 18:02:22 UTC
□ □□□ □□□ 2026-10-31 18:02:24 UTC