Installation - Linux Sourc
e 2-0-Alpha-from-svn

Admin - Reference guide v2-0

Version 27, last updated by meijer at 2016-02-22

Contents

1. Key concepts

FileSender 2.0 is built around the following concepts:

transfers

A transfer consists of one or more files sent to one (zero?) or more recipients

files

recipients

user preferences

guest use (guest invitations)

audit log

email receipts

stats log

2. privacy footprint

- full details of a transfer are logged in the audit log
- once a transfer expires, depending on the config settings and the preferences a user indicated an audit report in html or pdf can be sent to a user by email.
- after this, the audit log for that transfer _can_ be deleted depending on config settings
- all activity is also logged to the statistics log but without any information identifying the particular end user or the details of what was transferred. What is logged? (overview!)
- relevant config directives
- common scenarios: "best privacy"

3. Maintenance mode

Version 2.0 supports a maintenance mode. This allows to interrupt the service for database upgrade or even server restart without breaking uploads. When switched on:

- all pages are replaced with the maintenance page
- webservice returns specific exception to all requests
- clients display a popup explaining what happens
- clients pause uploads and put all requests they were about to make in a stack
- clients starts to query the server on a regular basis to see if maintenance ended (server responding with no exception status)
- when server exits maintenance mode clients restart uploading and run stacked requests and remove maintenance popup

How to switch on maintenance mode

Do <fill out here how to do this>

Where to change the maintenance messages

Change the following language tags in your localised language file(s):
$lang['undergoing_maintenance'] = 'This application is under maintenance'; $lang['maintenance_autoresume'] = 'Your operations will automatically resume when maintenance ends.';

Sizing your installation

How much space do I need?

Just get something that can easily expand, preferably in a matter of hours :)

4. Authentication and user account handling

FileSender has no user database and has no concept of user accounts.

What happens when a login session expires?
There are two expiry timers on a user logon session. One controlled by FileSender on its side, in its simplesamlphp or shibboleth configuration. The other is controlled by the IdP the user uses to authenticate against.
The <...> is set in the SAML2 message received by .
..

- Any uploads started during an active logon session get their own "upload session" authentication token. This allows the upload to finish even if the user's logon session should expire.
- When a login session is expired and a user tries do do something, a message pops up informing the user of the expired session and inviting to re-logon

5. How uploads are handled

Any size with HTML5 chunked upload

fall-back for non-html5 browsers

no longer flash: now normal html post. User can still select multiple files, when uploading a hidden iframe

takes care of the upload. Progress event through polling the server. How often polling is done is defined by legacy_upload_progress_refresh_period.

TeraSender high speed upload module

Use of temporary files

<filesender>/tmp
used for following tmp files:

- instance.secret which is the seed value used in the secure generation of random values used for fileUID, download link etc.
- used by dompdf to store tmp files when generating audit report
- NOT used for tmp storage of chunks: those are written directly in actual file on disk

xsrf token and instance.secret
if instance.secret disappears a user session might break.
question: do we need to implement auto-refresh of the instance.secret?

Difference html5 and non-html5 browsers

- no automatic transfer restart (uses html5 localstorage)
- no chunked upload (=limit of 2 GB per file)

Detection of stalled and corrupted chunks

We open file from zero, seek to chunk offset and write the data received in the chunk. So checks on file size of total file less meaningfull. You could only write last chunk and file size would still be good.
Advantage: can restart any chunk.

Checks to ensure file integrity

- **No hashing yet (slows things down too much)**
- **For each chunk: is received size identical to what client sent as "sent chunk length"? Client sends this in a header (not content-length header!)**
- **Check chunk size against max. chunksize set in config: can be smaller but not bigger than this**
- **At end of file upload we check the entire file size?**
- **check whether filesize on server is filesize client**
- **on wish list: chunk map or file integrity. For big files this map would be big. For each chunk sent client side we keep chunk offset and chunk data length. At end of upload, hash that, server does same during upload and check if all are equal. Means keeping data both sides, can become huge. Can store in flat file, put it in tmp directory "transferid.map", one line per chunk. When transfer is labeled as "failed" after no data has come for multiple days, just remove the file. Writing entry in map file can only be done once chunk is successfully written to disk. At end of upload read map to check if all expected offsets have been uploaded. This is not really a map but more like an "uploaded offset list". As long as we can detect holes in this list we can detect missing chunks.**
- **observation: no hash is browser limitation. But we have API. Researchers are the most likely to use API with either command line client or their own code _and_ most likely to send the big files that you would like to integrity check. Can implement hashing server-side and make available via API in preparation for browser fucntionality supporting client-side hashing.**
- **reason we don't do file integrity check now: with sending blob we pass on a pointer to the blob to the browser's http client. We don't have access to the data. If we want to hash it, we need to read it, it's an intensive process and memory consuming. Workers time chunk size. Slows down upload considerably. File reader also doesn't really support binary, so get string encoded. You have to create a byte array from that on which you can hash.**

6. Language and internationalisation

- UTF8, multiple languages
- Uses browser preference for automatic language selection.

  - describe how language is selected (use browser indication, use default language, hard coded default back to EN_AU
  - describe what happens with language tags that are not translated

- User can select language in UI as well, if this is enabled in config
-
- Can use multiple languages in emails as well (ask Etienne on implementation status and implemented algorithm)

- default email language: use what sender is using in UI
-user can select other language but applies to _all_ emails (?)

Overriding the default language files

- describe how to modify language
  - local overrides (directory in config directory
-

Creating and changing default language files

  - improve language file that ships with filesender: send to filesender team as patch?
  - add language
  - which script to use to see which tags are not translated
-

-

7. UI Customisation

header footer logo

skinning using style sheets

<filesender>/www/skin

Each template uses “Foundation”. There are template overrides.

When you want to override css or add script, like for Fonts: create skin directory in www/skin and put it there. CSS must be named “styles.css”. In skin: skin/script.js and styles.css are immediately interpreted by FileSender. Can have other scripts, but must include them in your files or tweak templates/header.php to include them. Can copy it in config/templates/header.php and tweak there. Start of page with HTML headers etc. Can add scripts and styles you want there.

html template engine

8. Email bounce handling and SPF

<Etienne to write>
examples of sendmail and postfix

Add possibility to forward feedback with non-detectable type but identified related target (recipient, guest) to person of choice.

Related config parameter is "relay_unknown_feedbacks" (string, defaults to "sender") :

- "sender" : relay to recipient's transfer owner or guest owner
- "admin" : relay to admin emails
- "support" : relay to help_url if it is in the form of "mailto:someaddress@domain.tld"
- "someaddress@domain.tld" : choosen email address

Received feedback is forwarded as a message/rfc822 attachment.

9. Multi-tenant hosting

Installation

Database initialisation and updates

Describe how the database is initialised for version 2.0 (from classes definiation) and how it's updated automatically!

Workarounds implemented to work around various client quirks

mac_unzip_link (to work around Mac unsigned 32 bit default unzip client)

recipient Id includes transfer Id

Debugging

- increase logging (log levels)
- firebug (!!)

Storage block concept

Can create storage blocks from different chunks of storage with filesender without lvm
Use multiple mount points.

Revisit lateron:
progress updates in web workers (for slow uploads with terasender on)

Log file analysis

In debug, why do I see error reports about simplesaml classes not being found?

1. Class SimpleSAML_Auth_Simple is needed
2. PHP runs autoloaders
3. FileSender's autoloader look for SimpleSAML_Auth_Simple class in its classes sub-folder according the the layout we defined
4. It does not find it
5. He reports it in case this is a problem
6. PHP moves onto next autoloader (SimpleSamlPHP's)
7. SimpleSamlPHP looks for the class by its own logic
8. It finds it
9. PHP is happy

PHP cannot magically consider a class to be part of a software from its name, not reliable enough, so it tries to resolve it using its default classpaths and provided autoloaders.

Application security

Securing your FileSender instance

-security mechanisms
* cross site scripting protection: csrf cookies (security-token)
* sp-session (authentication)
* chunk-upload-security token
* fileUID random generated,
* download URL protection
* API: api access key

- clickjacking
-secure webserver config