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

Language handling and localisation reference v2-0

Version 9, last updated by meijer at 2016-02-22

Translation logic

Filesender includes a translation engine which allows for flexible user language detection and customization.

How to contribute to the FileSender 2.0 translation effort

Starting with version 2.0 we are using www.poeditor.com to manage the different languages FileSender supports. If you want to contribute to a translation please follow the public link to the FileSender 2.0 project in poeditor.com and follow the instructions you get there:

https://poeditor.com/join/project/RqXr9WBJwU

After completing the sign-up process you should be able to add yourself to the language you want to contribute to or add a new language and start translating right away!

User language detection

FileSender builds the list of user languages by order of importance from various sources :

- From the url (`lang` url parameter) : allows for user language switching (only if `lang_url_enabled` set to true in config), if `lang_save_url_switch_in_userpref` is enabled in config and a user session exists the new language is saved in the user preferences so that he doesn't need to switch it again the nex time. If no user session is found the new choice is saved in the PhP session.
- From the browser's `Accept-Language` header : allows for automatic language detection base on the browser config (if `lang_browser_enabled` set to true in config)
- From `default_language` config parameter
- From absolute default `en`

Dictionary loading

Translations are then loaded depending on the user language order, all more-preferred ones overriding less-preferred ones.

One can redefine translations just by creating a `language` directory under the `config` directory.

The `config/language` directory must have the same structure as the root `language` directory.

It is not necessary to copy all files, only the ones needing changes.

Any `lang.php` file under `config/language/some_lang` doesn't need to contain all translations, only the ones needing changes.

One can select available languages and their names by creating a `config/language/locale.php` file (same structure as `language/locale.php`).

Translation process

When a translation is requested (UI display, email sending) the translated version is taken from the dictionary.

Then several replacements can occur (depending on the application code), the translation engine parses a simple markup language described below.

Translation engine language

FileSender UI / emails translation engine includes a simple markup language with variable display, type formatting, conditional statements, loops ...

Variables come from the various replacement calls done on the translated content during it's processing.

Variables available in UI translations can be found in the `en_AU` language files most of the time.

Variables available in emails are described a bit further

Variable display

Printing out the content of a variable is pretty straightforward :

```
{variable_name}
```

If the variable contains an object (such as a `transfer`, a `recipient` ...) properties can be accessed as :

```
{variable_name.property_name} (one can chain as many properties as needed)
```

If the variable is an array possible uses are :

```
{variable_name.2} (get the item with index 2)

{variable_name.foo} (get the item with index foo)

{variable_name.first()} (get the first item)

{variable_name.last()} (get the last item)

{variable_name.nth(3)} (get the 4th item since arrays indexes are 0 based)
```

Variables can also be automatically converted to formatted version (depending on language) :

```
{date:variable_name} (format as date)

{datetime:variable_name} (format as date and time)

{time:variable_name} (format as time)

{size:variable_name} (format as a file size)

{raw:variable_name} (no formatting and no output escaping, MUST BE USED CAREFULLY si
nce it escape injection protection)
```

In case the variable resolves to an array without further key based access the number of items in the array will be returned :

If variable_name = ["foo", "bar"]

then {variable_name}

will print out "2".

All above access methods and formatters can be combined to achieve complex data fetching :

```
{size:transfer.downloads.first().target.size}

{transfer.files.last().downloads}
```

Conditionals

Conditionals can be used to handle plural forms, display part of a text depending on variables ...

Nesting of conditionals is not supported. Variable printing and loops can be used in statements.

Basic syntaxes are (new lines allowed) :

```
{if:condition}foo{endif}

{if:condition}foo{else}bar{endif}
```

`condition` should be a boolean test, it can use available variables against a number of test operators.

Test operators are `==`, `!=`, `<`, `<=`, `>` and `>=` :

```
{if:transfer.files > 1}foo{else}bar{endif}
```

Logical operators can be used to build complex tests : AND (`&`) and OR (`|`) (AND operator takes precedence).

```
{if:transfer.files>1 & recipient.downloads}foo{else}bar{endif}

{if:transfer.files>1 & recipient.downloads | transfer.size>1234567890}foo{else}bar{e
ndif} (here just having transfer.size over 1234567890 bytes leads to "foo" being pri
nted)
```

Logical negation can also be used by putting `!` in front of the test sub-part.

```
{if:transfer.files>1 & !recipient.downloads}foo{else}bar{endif}
```

Loops

Loops can be used to sequentially print a list of items, syntax is :

```
{each:transfer.files}File {item.name} ({size:item.size}){endeach}
```

Item name choosing for better readability (does not change result) :

```
{each:transfer.files as file}File {file.name} ({size:file.size}){endeach}
```

Conditionals can be used inside loops :

```
{each:transfer.files as file}File {file.name} ({size:file.size}) {if:file.size>12345
67890}big file !{endif}{endeach}
```

Nesting of loops is not supported.

Magic

Config parameters

It is possible to use config values anywhere in the translations using any of the following syntaxes :

```
{config:parameter_name}
{conf:parameter_name}
{cfg:parameter_name}
```

It is possible to use automatic formatting as well :

```
{size:cfg:max_transfer_size}
```

Email parts

When translating an email (file like `some_id.mail.php` or just `some_id.mail`, same effect) the translation file can contain several parts.

Headers

Anything before the first double line break is considered as headers. Headers a expressed like in a `key: value` format.

As of now only `subject` header is supported, if several subject lines are provided the last non-empty (after variables replacement) subject is used.

```
subject: Foo
subject: {bar}

< mail contents >
```

In this exemple the sent mail subject will be "Foo" except if the `bar` variable contains something.

Using other translation syntaxes in headers is supported (only rule : headers have to be on a single line).

```
subject: Foo
subject: {if:bar & foo.bar != bar}Alt subject : {bar}{endif}

< mail contents >
```

If no `subject` header is found the `site_name` config parameter will be used as subject, it is considered good practice to use a subject that tells what the email is about.

Additionally the `email_subject_prefix` config parameter value will be prepended to the subject.

Content parts

Mail translation can include either or both `text/plain` and `text/html` alternatives (they will be used depending on the configuration).

Start of `text/plain` part is signaled using `{alternative:plain}`

Start of `text/html` part is signaled using `{alternative:html}`

If the first part after the headers does not have a starting `alternative` tag its type will be set depending on what comes after in the mail :

- `plain` if an other part has a `{alternative:html}` starting tag
- `html` if an other part has a `{alternative:plain}` starting tag
- `html` if no other part is found but some html tags can be detected in the part
- `plain` if no other part is found and no html tags can be detected

It is usually considered a good practice to create both `html` and `plain` parts when translating FileSender emails.

When translating an email the newly created translation REPLACES the basic one, there is no parts overriding so even if one wants to only tweak the `html` part the `plain` part must be copied as well otherwise it will be missing when translating.

Email variables

Here are the names of the variables that are available for use in the emails translations :

bounce_report

Sent to transfer / guest owner when a bounce is received.

- `bounces` : Tracking Events (array)

daily_summary

Sent to transfer owners for transfers that needs a daily summary of what happened.

- `events` : summary events (array)
- `transfer` : Transfer (object)
- `user` : recipient as a User (object)

download_complete

Sent to a recipient who downloaded some file(s) to notify him that download ended.

- `files` : Files that were downloaded (array)
- `recipient` : Recipient who downloaded (object)
- `result` : flag indicating if everything went well (bool)

email_feedback

Sent to admin when receiving an unknown feedback (depending on config).

- `target` : feedback's target (object)
- `target_id` : identifier of the feedback's target (int)
- `target_type` : type of the feedback's target (string)

file_deleted

Sent to transfer recipients when a file is removed from the transfer.

- `file` : File (object)
- `recipient` : Recipient (object)

files_downloaded

Sent to a transfer owner when some file(s) is/are downloaded.

- `files` : Files that were downloaded (array)
- `recipient` : Recipient who downloaded (object)
- `result` : flag indicating if everything went well (bool)
- `user` : owner as a User (object)

guest_access_upload_page

Sent to a guest owner when his guest accesses the upload page.

- `guest` : Guest (object)
- `user` : guest owner as a User (object)

guest_canceled

Sent to a guest when his owner removes the guest token.

- `guest` : Guest (object)

guest_created

Sent to a guest when it becomes available.

- `guest` : Guest (object)

guest_created_receipt

Sent to a guest owner when his guest becomes available.

- `guest` : Guest (object)
- `user` : guest owner as a User (object)

guest_expired

Sent to a guest when it expires.

- `guest` : Guest (object)

guest_reminder

Sent to a guest when his owner sends him a reminder.

- `guest` : Guest (object)

guest_upload_complete

Sent to a guest owner when his guest finishes to upload a transfer.

- `guest` : Guest (object)
- `user` : guest owner as a User (object)

guest_upload_start

Sent to a guest owner when his guest starts to upload a transfer.

- `guest` : Guest (object)
- `user` : guest owner as a User (object)

recipient_deleted

Sent to a recipient when it is removed from a transfer.

- `recipient` : Recipient (object)

recipient_feedback

Sent to a transfer owner when an unknown feedback is received for one of his recipients (depending on config).

- `target` : feedback's target (object)
- `target_id` : identifier of the feedback's target (int)
- `target_type` : type of the feedback's target (string)

report_attached

Sent to a transfer owner when reporting transfer activity using attached file (depending on config).

- `content` : array with `plain` and `html` entries, not used
- `file` : File the report is about if report `target` is a file, not provided otherwise (object)
- `recipient` : Recipient the report is about if report `target` is a recipient, not provided otherwise (object)
- `target` : array describing the report target in a common way, it has a `type` entry containing the target type as a string and an `id` entry containing the target identifier
- `transfer` : Transfer the report is about if report `target` is a transfer, not provided otherwise (object)
- `user` : recipient as a User (object)

report_inline

Sent to a transfer owner when reporting transfer activity in the mail body (depending on config).

- `content` : array with `plain` and `html` entries, contains report text for both formats
- `file` : File the report is about if report `target` is a file, not provided otherwise (object)
- `recipient` : Recipient the report is about if report `target` is a recipient, not provided otherwise (object)
- `target` : array describing the report target in a common way, it has a `type` entry containing the target type as a string and an `id` entry containing the target identifier
- `transfer` : Transfer the report is about if report `target` is a transfer, not provided otherwise (object)
- `user` : recipient as a User (object)

storage_usage_warning

Sent to the admins when storage usage is too high (depending on config).

- `warnings` : storage blocks warnings (array)

transfer_available

Sent to a recipient when a transfer becomes available.

- `recipient` : Recipient (object)
- `transfer` : Transfer (object)

transfer_autoreminder_receipt

Sent to a transfer owner when recipients that did not download files were reminded.

- `recipients` : Recipients reminders were sent to (array)
- `transfer` : Transfer (object)
- `user` : owner as a User (object)

transfer_deleted

Sent to a recipient when a transfer is closed by it's owner.

- `recipient` : Recipient (object)
- `transfer` : Transfer (object)

transfer_deleted_receipt

Sent to a transfer owner when he deletes it.

- `transfer` : Transfer (object)
- `user` : owner as a User (object)

transfer_expired

Sent to a recipient when a transfer expires.

- `recipient` : Recipient (object)
- `transfer` : Transfer (object)

transfer_expired_receipt

Sent to a transfer owner when it expires.

- `transfer` : Transfer (object)
- `user` : owner as a User (object)

transfer_reminder

Sent to a recipient when a transfer owner triggers a remind.

- `recipient` : Recipient (object)
- `transfer` : Transfer (object)

translate_email_footer

Sent along any email that can be translated.

- `translatableemail` : related TranslatableEmail (object)

upload_complete

Sent to a transfer owner when it's transfer is made available.

- `transfer` : Transfer (object)
- `user` : owner as a User (object)

Core objects properties

AuditLog

- `id` : unique identifier (int)
- `event` : type of event (string)
- `target_type` : type of the event target ("Transfer", "File" ... depends on event type) (string)
- `target_id` : identifier of the event target (int)
- `author_type` : type of the event author ("User", "Guest", "Recipient" ... depends on event type)

(string)
- `author_id` : identifier of the event author (int)
- `created` : event epoch date (int)
- `ip` : IP address of remote user which triggered the event (string)
- `target` : shortcut to get the event target (object)
- `author` : shortcut to get the event author (object)
- `time_taken` : time it took to get to the event (depending on event type, empty when it doesn't make sense) (in seconds) (int)

File

- `id` : unique identifier (int)
- `transfer_id` : id of the Transfer the file is part of (int)
- `uid` : file uid (string)
- `name` : file name (string)
- `mime_type` : file Mime type as provided by uploader (string)
- `size` : size of the file in bytes (int)
- `upload_start` : epoch date when the first file started being uploaded (int)
- `upload_end` : epoch date when the last file finished uploading (int)
- `sha1` : not used
- `transfer` : shortcut to the Transfer the file is part of (object)
- `owner` : shortcut to the User who owns the transfer the file is part of (object)
- `auditlogs` : file AuditLogs (array)
- `downloads` : download related AuditLogs (array)
- `upload_time` : time it took to upload the file (in seconds) (int)

Guest

- `id` : unique identifier (int)
- `user_id` : identifier of the creator (string)
- `user_email` : email of the creator (string)
- `token` : upload token (used to build upload link) (string)
- `email` : email of the guest (string)
- `transfer_count
- `subject` : subject given by creator (string)
- `message` : message given by creator (string)
- `options` : guest options (array of strings)
- `transfer_options` : created transfer options (array of strings)
- `status` : guest status (string)
- `created` : epoch creation date (int)
- `expires` : epoch date when the guest will expire (int)
- `last_activity` : epoch date when the guest was last active (int)
- `user` : alias of `owner` (object)
- `owner` : shortcut to the User who owns the transfer (object)
- `transfers` : Transfers the guest uploaded (array)
- `tracking_events` : TrackingEvents encountered when sending emails to the guest (array)
- `errors` : TrackingEvents describing errors encountered when sending emails to the guest (array)
- `identity` : alias to `email` (string)
- `name` : local part of `email` (string)

Recipient

- `id` : unique identifier (int)
- `transfer_id` : id of the Transfer the recipient is part of (int)
- `email` : email of the recipient (string)
- `token` : download token (string)
- `created` : epoch creation date (int)

- `last_activity` : epoch date when the recipient was last active (int)
- `options` : recipient options (array of strings), not used (?)
- `transfer` : shortcut to the Transfer the recipient is part of (object)
- `owner` : shortcut to the User who owns the transfer the recipient is part of (object)
- `auditlogs` : recipient AuditLogs (array)
- `download_link` : recipient download url (string)
- `downloads` : download related AuditLogs (array)
- `tracking_events` : TrackingEvents encountered when sending emails to the recipient (array)
- `errors` : TrackingEvents describing errors encountered when sending emails to the recipient (array)
- `identity` : recipient email or "Anonymous" for "get_a_link" recipients (string)
- `name` : recipient email's local part or "Anonymous" for "get_a_link" recipients (string)

Storage blocks warning

- `filesystem` : identifier of related filesystem, "main" if hashing not configured (string)
- `free_space` : free space on the block in bytes (int)
- `free_space_pct` : human readable free space percentage, floored, without unit (int)
- `paths` : paths that are stored in the block (array of strings)
- `total_space` : total space on the block in bytes (int)

Summary event

- `who` : email of the event's author (string)
- `what` : event target type (string, "file" or "archive")
- `what_name` : name of the file if `what` is set to "file", empty otherwise (string)
- `when` : event epoch date (int)

TrackingEvent

- `id` : unique identifier (int)
- `type` : type of event (string)
- `target_type` : type of the event target (string)
- `target_id` : identifier of the event target (int)
- `details` : details about the event (string)
- `created` : event epoch date (int)
- `reported` : event report epoch date (int)
- `date` : alias of `created`
- `target` : shortcut to th target object (object)

Transfer

- `id` : unique identifier (int)
- `status` : transfer status (string)
- `user_id` : owner id (string)
- `user_email` : email the user used when he uploaded (string)
- `guest_id` : guest identifier if created by a guest, empty otherwise (int)
- `subject` : subject given by owner (string)
- `message` : message given by owner (string)
- `created` : epoch creation date (int)
- `made_available` : epoch date when the transfer became available (int)
- `expires` : epoch date when the transfer will expire (int)
- `expiry_date_extension` : number of days the transfer expiry date can be extended by (int)
- `options` : transfer options (array of strings)
- `lang` : lang set by the owner for the recipients (string)
- `user` : alias of `owner` (object)
- `owner` : shortcut to the User who owns the transfer (object)
- `guest` : shortcut to the Guest who uploaded, null otherwise (object)
- `files` : Files in the transfer (array)

- `size` : total size of the transfer in bytes (int)
- `recipients` : Recipients the transfer was sent to (array, order not predictable)
- `first_recipient` : shortcut to the first Recipients (order not predictable)
- `recipients_with_error` : Recipients of the transfer for whom email sending encountered an error (array, order not predictable)
- `auditlogs` : transfer AuditLogs (array)
- `downloads` : download related AuditLogs (array)
- `is_expired` : expired flag (bool)
- `made_available_time` : total time it took to create, upload and make the transfer available (in seconds) (int)
- `upload_start` : epoch date when the first file started being uploaded (int)
- `upload_end` : epoch date when the last file finished uploading (int)
- `upload_time` : time it took to upload all files (in seconds) (int)

TranslatableEmail

- `id` : unique identifier (int)
- `context_type` : context object type of the sent message (string)
- `context_id` : context object identifier of the sent message (string)
- `token` : translation token for viewing (string)
- `translation_id` : identifier of translation template (string)
- `variables` : variables used for translation (array)
- `created` : epoch creation date (int)
- `context` : shortcut to the context (object)
- `link` : url to use to access the translation UI (string)

User

- `id` : unique identifier (string)
- `additional_attributes` : additional attributes collected during authentication (depending on config) (array)
- `lang` : user preferred language (depending on config) (string)
- `aup_ticked` : flag telling if the user ticked aup checkbox the for previous upload (bool)
- `aup_last_ticked_date` : epoch date the user ticked the aup checkbox last (int)
- `auth_secret` : remote authentication secret if enabled (string)
- `transfer_preferences` : last chosen transfer preferences (array)
- `guest_preferences` : last chosen guest preferences (array)
- `frequent_recipients` : list of most frequent recipients email addresses (array)
- `created` : epoch creation date (int)
- `last_activity` : epoch date when the user was last active (int)
- `email_addresses` : user email addresses as returned by authentication (array)
- `name` : user name as returned by authentication (string)
- `quota` : copy of last user quota in bytes (int)
- `email` : user first email address as returned by authentication (array)
- `remote_config` : user remote access config if enabled (string)
- `identity` : alias to `email` (string)