

Gitlocalize / Github Documentation

ShapeShift DAO - Globalization Workstream

Introduction

In order to translate the user interface of app.shapeshift.com, the Globalization Workstream is using a free tool named “Gitlocalize” (<https://gitlocalize.com>), a Web-based platform which allows translating text elements (strings) of the application.

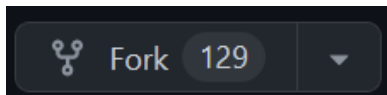
These strings are in text files (using the JSON format) located in [the various Github repositories](#) of the ShapeShift DAO and are maintained in English by the Engineering Workstream. Each string is attributed a unique “key” that is used in the app source code to refer to it when it needs to be displayed. Depending on the current language selected by the user of the app, the appropriate translation is returned. When a translation is not available (or a string is empty), the app will fallback to the English string. If the English string for a key is also missing in English, the key itself will be displayed.

For each language a similar text file using the same format and keys exists in the repository in order to provide these translations. Gitlocalize offers a user-friendly interface, in your browser, to complete these JSON files for each language and push these changes to our engineers

Prerequisites for Gitlocalize Project Managers

[For: Gitlocalize Project Managers]



- A Github account.
- A Git repository fork containing the JSON files to translate. To create a fork, make sure to be logged in on Github, go on the repository page and click the “Fork” icon:



Then create the fork, you will thus be the owner of this fork:


Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Owner *  / **Repository name *** 

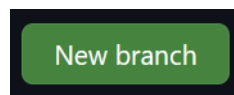
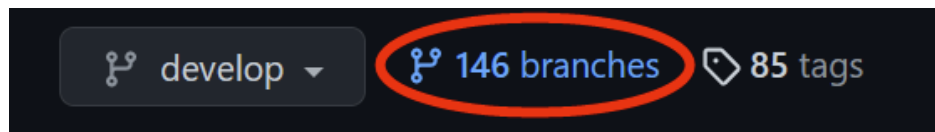
By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

Description (optional)

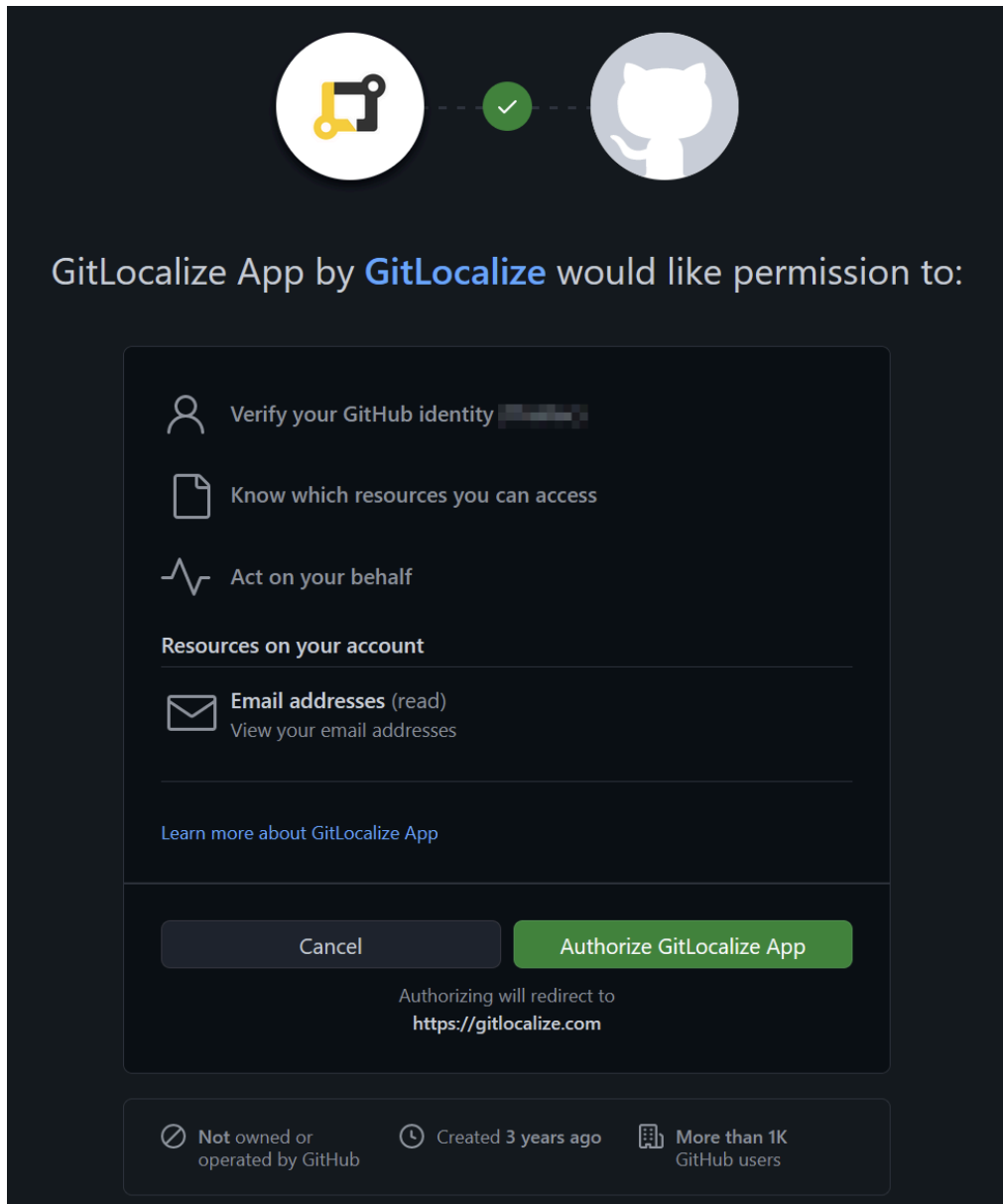
 You are creating a fork in your personal account.

[Create fork](#)

- Creating a specific branch on the repository fork, for this documentation we will name it **“l10n”** (shorthand for localization) for that repository. The changes made on Gitlocalize will be merged in this branch, and then this branch will be used to create a Pull Request on the official ShapeShift repository using Github.
 - To create the branch, go in your fork on Github and click the “Branches” link:



- The click on the “New Branch” button:
 - Name it “l10n”, make sure the default “Source Branch” is selected, in our case “develop” and create it.
- Authorizing Gitlocalize to access your data on Github. For that, login on <https://gitlocalize.com> with your Github account and allow Gitlocalize the following permissions:



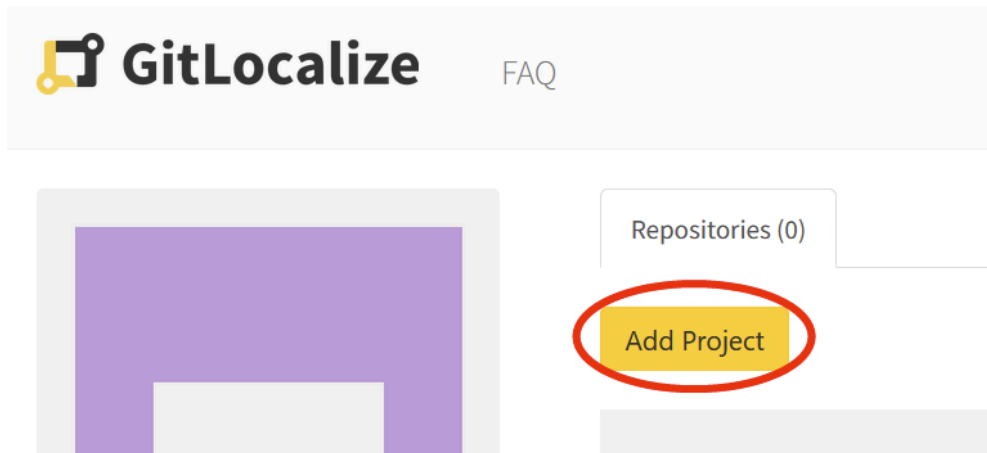
Setting up a new Gitlocalize project

[For: Gitlocalize Project Managers]

Gitlocalize uses your Git repository on Github as a reference to populate the list of strings available for translation. So for each project on Gitlocalize, there is a Github repository, and any change happening in this repository will be reflected in the Gitlocalize project directly (after a short synchronization time).

Here are the steps to create a new “Project” on Gitlocalize, based on a fork of the “web” repository listed in [Prerequisites for the Gitlocalize Project Managers](#).

1. Once logged in on Gitlocalize, go in your Profile and click one “Add project”:



2. Select the repository, for our example “web” and enter the name of the branch you have created above:

Create New Project

3. Under “Project Paths” you will define the path of the default language file/path and the generic location of the translation files. Select “File” in this case as the project only relates to a single file currently (main.json).
4. In the repository on Github identify where the language JSON files are located, for the “web” repository it is here:

web / src / assets / translations / en / main.json

Based on this enter the path with the in the field “Source Path”, this defines the source/default language file, in our case the English one →

web/src/assets/translations/en/main.json


5. In the “Target Path”, enter a token-based generic path which represents all the other files
→ **web/src/assets/translations/%lang%/main.json**
6. In the “Project Languages” section, you should make sure English is selected as the “Source Language” and then add all the languages you wish to list for the translators. Each language has a default “Language code”, that can be changed if needed, and will be used to build the path in the previous step by replacing the **%lang%** token. For our example the default codes are fine.

You can select “Moderators” for each language by entering their Github usernames, it is important to understand that moderators have the ability to validate Review Requests and create Pull Requests for your fork from Gitlocalize. This power should be given only to trustworthy people.

7. Click on “Submit” to create the project.
8. Click on “Approve, Install, & Authorize” to approve the new permissions asked on your Github account by Gitlocalize, for this specific repository. This will allow Gitlocalize to create Pull Requests automatically on your fork:



Approve, Install, & Authorize GitLocalize App

Approve, Install, & Authorize on your personal account 




suggested installation of this GitHub App now

☐ **All repositories**



This applies to all *current and* future repositories.

☒ **Only select repositories**

Select at least one repository.

 Select repositories ▾

Selected 1 repository.

  /web suggested



with these permissions:

- ✓ Read access to metadata
- ✓ Read and write access to code, pull requests, and repository hooks

User permissions

Installing and authorizing GitLocalize App immediately grants these permissions on your account: **TheRec**.

- ✓ Read access to emails

Approve, Install, & Authorize

Reject

Next: you'll be redirected to https://gitlocalize.com/auth/github_app_login/callback

9. Voilà, your project is created and synchronizing:

Language	Moderator	Translated
Spanish		0%
French		0%
Portuguese		0%

10. Once synchronized it will be possible to start translating strings.

11. Under “Team” you should add all the contributors, here’s the current list:

Admin: thesmithdao
Admin: firebomb1
Translator ES: Jpanam
Translator FR: firebomb1
Translator RU/UK: romko391
Translator PT: guiribabrb
Translator DE: hellhound13
Translator TR: MarkusMeyer1

Translating

[For: Gitlocalize users/translators]

When translating on Gitlocalize, the changes are temporarily saved on the Gitlocalize platform. They will be pushed to your fork using a Pull Request only after a moderator (or your the owner) of the project approves a Review Request created by translators.

As a translator, no real Git/Github knowledge is necessary, here are the steps translators can follow:

1. You need a Github account. Make one if you don’t already have one on <https://www.github.com/>, or create one when offered as you try to connect in step 2.

2. Connect on <https://gitlocalize.com/> using your Github account. You need to allow the Gitlocalize application when asked during your first login process.
3. Once connected you can go to the repository URL provided by the moderator/owner of the project, and choose the language you are going to translate to.

Tip: Before you start translating, you can have an overview of the number of words to translate by hovering with your mouse on the completion percentage of a language:

Turkish		Total: 92275 chars 13019 words Untranslated: 5095 chars 792 words Translated: 87180 chars 12227 words	Incor
Ukrainian			Incor
Chinese	94%		Incor

4. Then select the file on which you need to work (you might need to click on folders to find files). It will show a list of strings (texts) that are present in this file in the left column, and their translations in the corresponding language on the right.

Tip: You can filter the strings if you click on the small funnel icon on the top right of the



list: and for example only display “Untranslated segments” to only see the entries that need a translation.

5. For each entry you can input your translation by clicking on the text field in the right column, then entering the translated text.

You can help yourself with the "Machine Translate" button, **but be aware that you should still always check the machine translation carefully as it's not perfect.**

Always make sure to preserve of the following special parts:

- * **Asset identifiers/ticker name (eg. BTC, FOX, etc.)**
- * **Brand and Product names**
- * **Tags, for example `<tag_name>translatable content</tag_name>`. Apply the tags around the literal translation of the content within them if possible, if not surround the related part of your translation.**
- * **Placeholders for variables such as `%{variable_name}`**
- * **Markdown syntax, for example: `[text](url)` (in that case the “text” can be translated if it isn’t part of the exceptions above, the “url” should not be translated)**

“Machine Translate” has a tendency to add spaces, mangle slashes/special characters, be careful especially about these special parts if you use this feature.

If you have any doubt ask in the Globalization channels/threads on Discord.

Tip: Certain words might need some context for a proper translation, refer to the gray column which lists the keys. It should give some hints as to where the translations will be used and if it's not enough ask in the Globalization channels/threads on Discord.

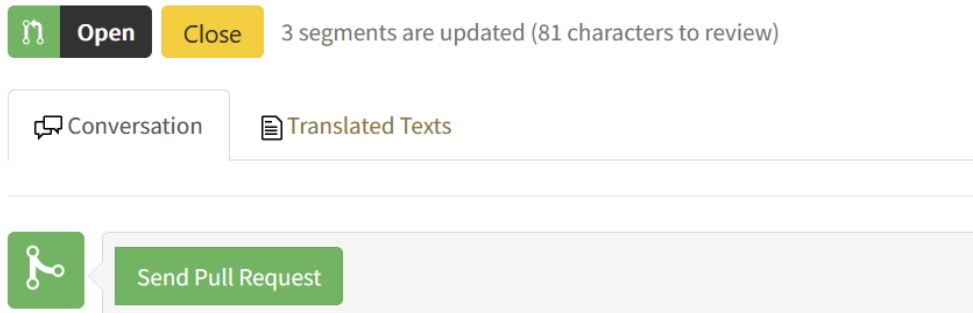
6. Once an entry is properly translated, click on "Submit". You can always go back and edit them later, they are saved on the platform for now.
7. When you're done, and all the strings in the file are translated you can click on the "Create Review Request", enter comments if you have any and submit it.
8. Then notify whoever is in charge of the language/project that you are done. They will review your work and eventually will push them to Git.

Creating and Merging a Pull Request from the Review Request

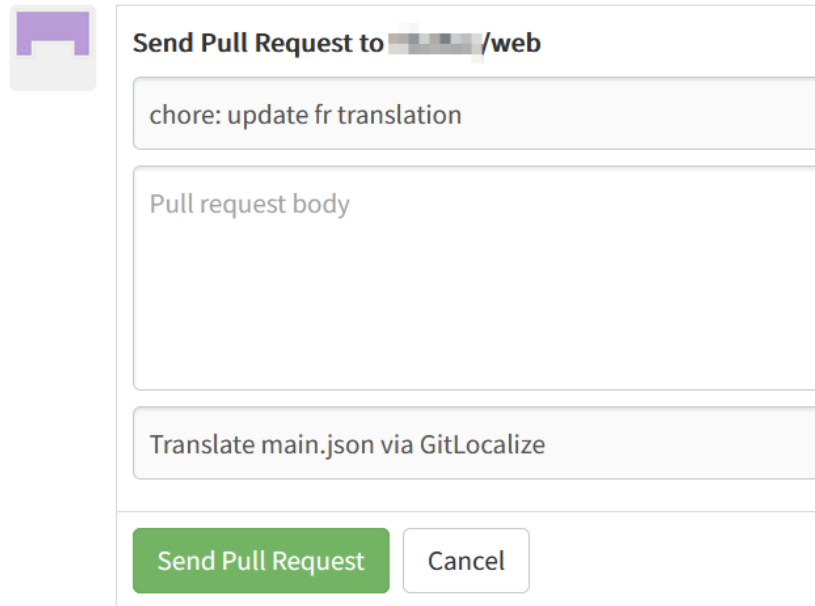
[For: Gitlocalize Project Managers and Moderators]

As the owner and/or moderator of a Gitlocalize project, you have the power and duty to review the translations suggested by translators before creating a Pull Request. You have the ability to view all the changes by clicking on the "Translated Texts" tab inside a Review Request (created by the translator when they are done working).

src/assets/translations/fr/main.json

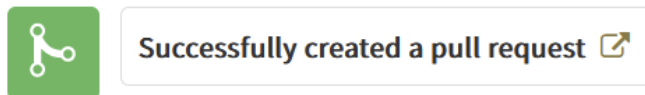


1. Once the review process is done, you can click on the "Send Pull Request" button.
2. Fill the form with a commit message, it is required to conform to the [Contribution Guidelines](#) of the ShapeShift repository for the commit messages:



In this example I used the commit message “chore: update fr translation” that follows the guidelines.

3. Click on “Send Pull Request” again.
Once the Pull Request is created this confirmation will show up, with a link to Github:



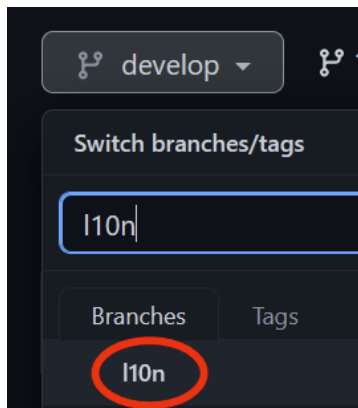
4. As an owner of the Github repository fork and branch for which this Pull Request has been created you have the power to Merge these changes in your repository.
On the Github linked page at the previous step you can review the changes once again if you want, and then click the “Merge pull request” button. You can use the default commit message and click on “Confirm merge”. Once this operation is done the language in Gitlocalize will be listed as “Synced”.
5. Repeat this process for every language that is completed by translators, this way you “accumulate” the commits/merges for all the languages in this branch of your fork. And when you will be ready to push all these changes to the Engineering Workstream for review/merge in the app you will be able to create a Pull Request for the ShapeShift origin repository.

Preparing a Pull Request for the ShapeShift repositories

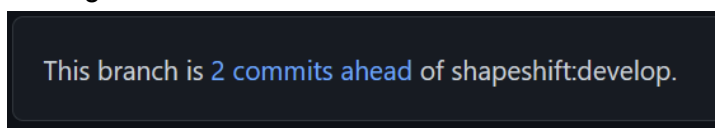
[For: Gitlocalize Project Managers]

Once you have gathered all the new translations in the form of commit/merge from Pull Requests in your “l10n” branch in your forked repository, it will be time to create a Pull Request for the origin repository, in order for the Engineering Workstream to review all these changes and eventually merge them.

1. Firstly **make sure there are no more pending Review Requests and that translators are done translating**, this is important as work that has not been turned into a Pull Request on the fork can be lost during string updates. If there is pending work, you have these strings into Review Requests and then create Pull Requests. Then and only then, perform an update of the strings as described in [“Updating strings in an existing Gitlocalize project”](#). It might make the percentage coverage drop below “100%” for languages if new strings have been added, but it is important to make sure the fork repository is “synchronized” with the latest changes before trying to push changes to the origin repository.
2. Go in your fork repository on Github and choose the “l10n” branch.



3. At the top of the file list, Github will inform you of the number of “commits” ahead from the origin branch like this:



Click on the link to review all these commits and on the page that appear you will see a button labeled “Create pull request”, click on it.

4. This opens a form to create a Pull Request on the origin repository and branch, the form is usually pre-filled, but it is necessary to once again use a proper commit message as a title. Let's say we are trying to push Spanish, Portuguese and French at the same time, one could use “chore: updating es/pt/fr translations”.
5. Make sure to complete all the form requirements (ticking the “[]” boxes by using “[x]” when needed). And add any relevant comment you might want to communicated to the

Engineering Workstream.

6. Once everything is completed you can click on “Create pull request”.
7. The Engineering Workstream will eventually notice your Pull Request (commonly abbreviated PR) and if everything is correct will merge it in the origin repository. The translated strings will then appear on the official app once a new “Release” is done (you can follow the progress in the Operations Workstream channels on Discord).
8. It is important to follow the PR on Github, as engineers will review and potentially ask for updates/changes.

To add changes/modify your PR you can:

- a. Use the Github interface in the PR to modify the files (approve suggestions).
 - b. Modify files directly in your “l10n” branch using Git (commit/push) or even the Github features to modify files, any change to the l10n branch in your fork will be added to the current PR as long as it is open.
 - c. Lastly you can also ask translators to do changes themselves on Gitlocalize, and repeat the Review Request -> PR in your fork -> Merge in your fork described above... and the changes will be added to the PR for the ShapShift repository once you merge your translator’s PR.
9. **Once a PR is merged in the official repository, not before, otherwise you will erase translators’ work**, in order to keep the history of the “l10n” branch clean and avoid keeping the list of changes done in the past reappearing in your future PRs you can/should run the following commands in order to “reset” the branch and force push it to your repository:

```
git checkout develop && git checkout -B l10n && git push --force
```

Note: “develop” is the name of the main/development branch in this case. It’s a sanity check that will warn you if you have pending changes in your repository.

Updating strings in an existing Gitlocalize project

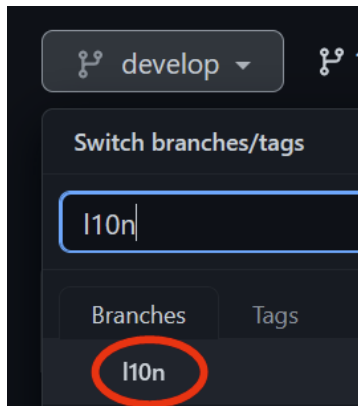
[For: Gitlocalize Project Managers]

After a release, engineers and product managers will continue to improve the app, add new features, fix bugs, and thus new strings are going to be added (and some strings will be updated). As the person responsible for managing the Gitlocalize project, periodically you have to check if there is a need/want for a new “batch” of translations to be done. You also need to potentially organize bounties for the languages for which our DAO has no hired translators currently.

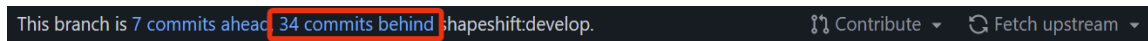
In all these cases you will need to first update the list of strings on Gitlocalize, so translators work on the most updated list. The operation is simple and mostly automated by Gitlocalize in a non-destructive way.

Important: When “Review Requests” on Gitlocalize have not been turned into a Pull Request (PR) on Github or if strings have been changed by translators, it is possible that updating strings will overwrite the current changes translators have made. So **ALWAYS** make sure to create all the PRs from Review Requests before following the next procedure.

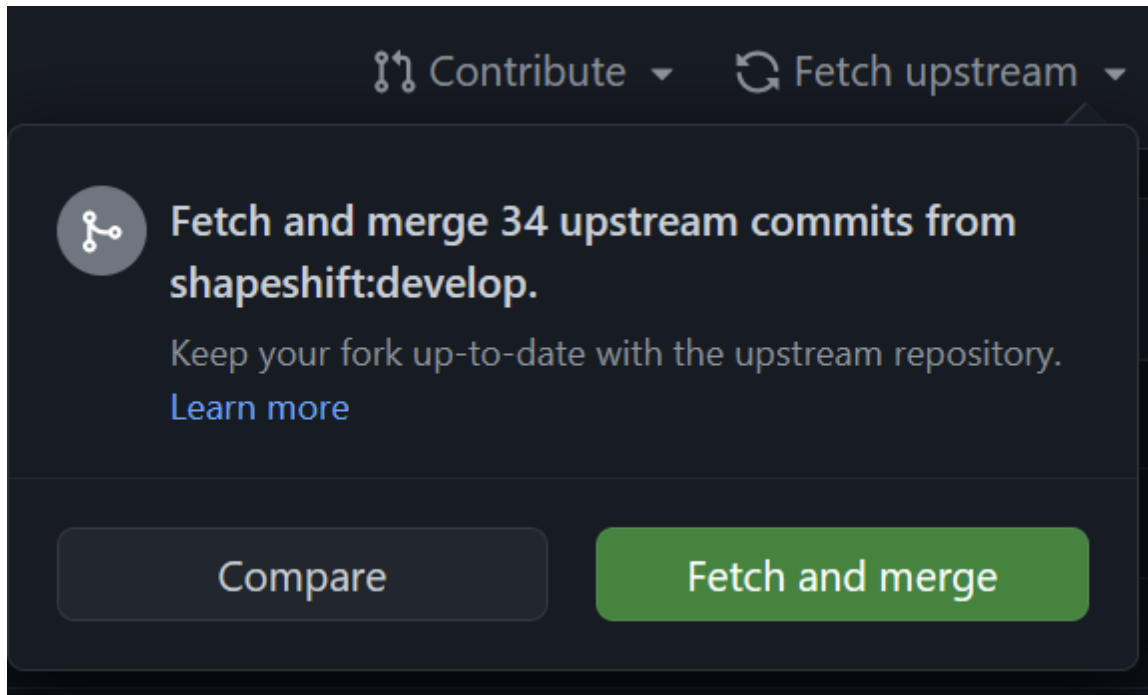
1. Go on your repository page, select the “l10n” branch:



2. At the top of the file list, Github will inform you of the number of “commits” **behind** relative to the origin branch like this:



3. To retrieve these changes from the origin branch, just click on “Fetch upstream” and then you can either review those changes and manually merge them (set a message) or just “Fetch and merge” to do it automatically.



Note: Currently the only way to review strings that have been updated by the Engineering Workstream is to use the “Compare” feature, and check the keys which have are “updates” (deletion of a line, addition of the same line with changes).

It is currently in discussion (with @gomes from the Engineering WS) to have a way for engineers to automate the “emptying” of the strings that are changed in English, in order for translators to be prompted to re-translate them in the next release (filter by “untranslated” strings).

For now this process is manual and needs to be done by engineers (and for now is rarely done in my experience).

Until this is fixed, from time to time it is a good idea to review the whole list of strings, even the ones that were already translated. This will obviously become harder and harder the more we add strings to the app, so this automated process will have to be implemented eventually, unless the DAO wants to incur the fees of having translators reviewing thousands of strings over and over again.

4. Go on the Gitlocalize project and the strings will be updated and ready to be translated.