The Lokalize Handbook

Nick Shaforostoff

Contents

Abstract

Lokalize is a computer-aided translation system that focuses on productivity and quality assurance. It has components usually included in CAT tools like translation memory, glossary, and also a unique translation merging (synchronization) capability. It is targeted for software translation and also integrates external conversion tools for freelance office document translation.

Chapter 1

Introduction

Usually program messages and documentation are written in English. Using a framework made of a set of tools and libraries, it is possible to have your favorite applications speak your native non-English language. This process of adapting an application to a specific language is known as *localization*. The localization process includes translating the program's interfaces and documentation to the various languages users need and, in some countries or regions, making the inputs and outputs conform to particular conventions. Lokalize is a tool which will assist you in the localization process to make an application's interface speaks many languages.

Every internationalization-aware program makes available for translation one or more message-catalog files. The extension of these files is `.pot`. POT is an acronym for 'Portable Object Template'. Lokalize is an advanced and easy to use PO file (GNU gettext message catalogs) editor. It is a computer-aided translation system for translators, written from scratch using the KDE Platform 4 framework. Aside from basic editing of PO files with nifty auxiliary details, it integrates support for glossary, translation memory, diff-modes for QA, project managing, etc. It has many features like full navigation capabilities, extensive editing functionality, search functions, syntax checking and statistics functions.

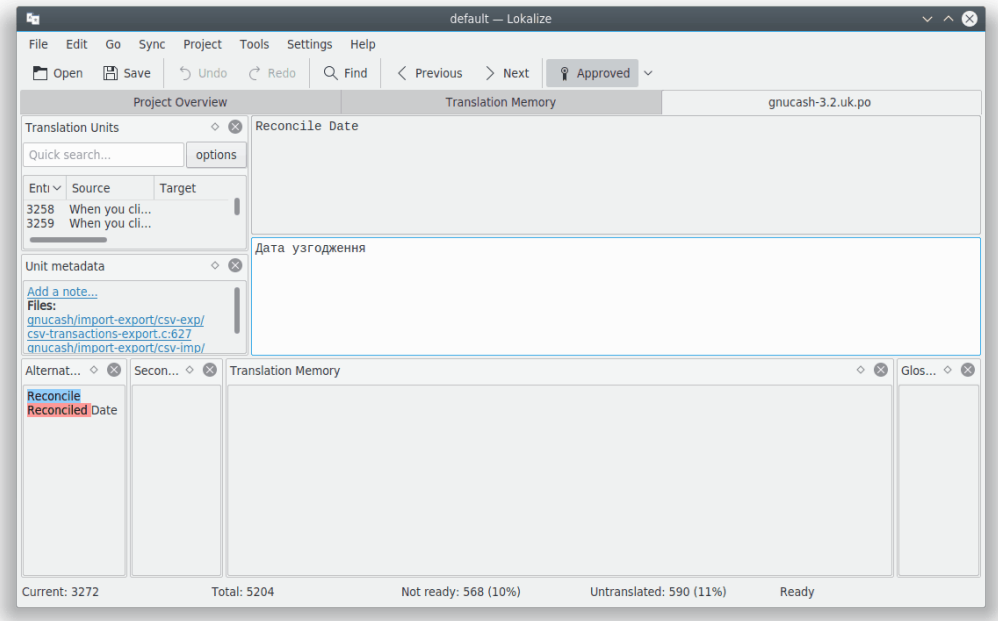
Portable Object (.po) files: Each translator takes a copy of one of these POT templates and begins to fill in the blanks: each message is translated into the language desired. The file containing the translated text is referred to as a PO (Portable Object) file.

Chapter 2

Editor

2.1 Main Window

By default, the main window contains six parts. The upper-right box is read-only and contains the current msgid (source text) field from the opened PO-file. The edit box just below this contains the msgstr (target text) field related to the msgid shown and here you can input or edit the translated text.



The top-left part of the main window shows the Translation Units. Below this, there is Unit Metadata section and it contains comments relevant to the currently displayed source text. In the lower-left, there is a Translation Memory section which shows the suggested translations from the translation memory database for the current source text entry. On the lower-right corner of the window, the glossary section is shown.

Translation files are opened in separate tabs, with two big multi-line edits as well as a bunch of *tool views*. These views can be stacked (similar to tabs), shown separately, or hidden. Translation files consist of many English-target pairs called *units*. A *unit* typically correspond to a single string in the user interface, or one paragraph in the documentation. The purpose of the first

multi-line edit is to display the original part of the pair. The purpose of the second multi-line edit is to display the translation. You can navigate through the *units* via the **Translation Units** view or by using **Page Down** and **Page Up**.

A unit may be *translated* or *untranslated*. A translation of a translated unit may be *ready* or *not ready* (also called *fuzzy* sometimes). If the unit is not ready, its translation is rendered in italics. Lokalize allows you to easily navigate through the file according to the state of their translation. See **Go** menu for the shortcuts. The status bar at the bottom of the window shows the current string number, total number of strings, total untranslated strings, total not ready (fuzzy) strings and status of the current string respectively. When navigating, untranslated units are treated as not ready. Also, you may use the filtering feature of **Translation Units** toolview. Pressing **Page Down** actually takes you to the next unit in filtered/sorted list of that tool view.

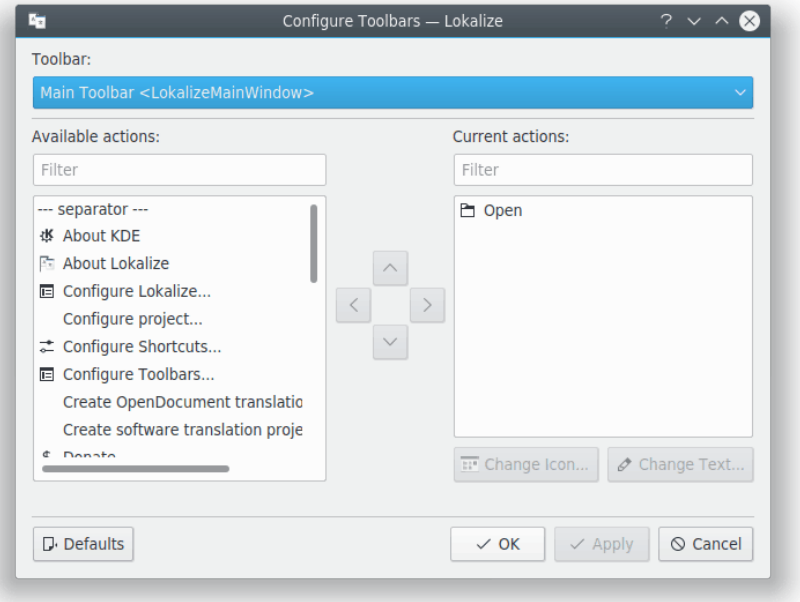
To ease up translation of the console applications where spacers in messages are important you can check the **Visualize separators such as spaces, tabs and new lines in the editor** item on the page **Editing** of the configuration window which can be opened by choosing the **Settings** → **Configure Lokalize...** main menu item.

If you have no access to the original translation template file (it is a common practice on some commercial translation projects) then you can use a translation file from the related language. Just choose the **File** → **Clear all translated entries** menu item or use **Ctrl-Alt-D** shortcut to clear all translation units.

To the main window one can add many more sections like **Alternate Translations**, **Primary Sync**, **Secondary Sync**, **Binary Units** by using **Settings** → **Toolviews** from the main menu.

2.2 Toolbars

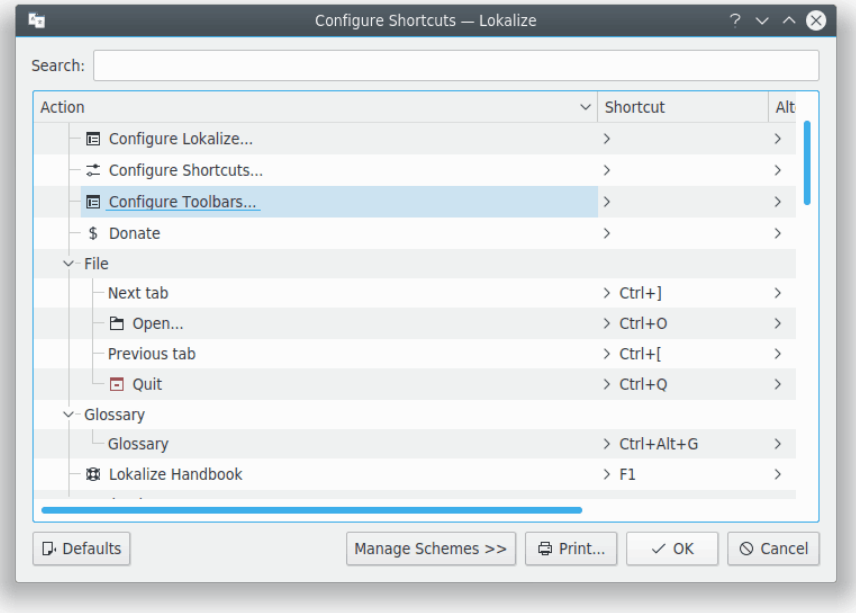
You can add or remove actions in the toolbars using **Settings** → **Configure Toolbars...** from the main menu.



For more information read the section about Toolbars of the KDE Fundamentals.

2.3 Shortcut keys

You can save time by using shortcut keys during translation. To configure shortcut keys, use **Settings** → **Configure Shortcuts...** from the main menu.



For more information read the section about Shortcuts of the KDE Fundamentals.

2.4 General Tips

If you are doing translations for KDE, then either you will already have Lokalize project file in your language folder (usually named `index.lokalize`), or you can select **Project** → **Create new project** and the wizard will download translation files for your language and will create project for you.

> TIP
> It is recommended that you get used to the keyboard shortcuts instead of the menus and toolbars for increased productivity. For example, use the **Ctrl**-**L** to focus **Quick search** input line to filter the unit list in **Translation Units** view. Once you are done, press **Page Down** to start moving along the filtered list.

If you are working with translation files in XLIFF format (definitely the case when you translate OpenDocument), then extended states are available (*new*, *needs review*, *approved*, etc.). You may select them in drop-down menu of **Approved** button in the toolbar. Classification of the state as *ready* or *not ready* depends on the current *workflow phase* (*translation*, *review*, *approval*). A default phase for you depends on your *role* in the project (set in project settings). Each unit usually contains information about phase it was changed last time, and for each phase its owner is logged to the file.

Chapter 3

Projects

3.1 General Notes

The projects are one of the main concepts in Lokalize. A project is represented by a file that contains paths, folders with translations, templates, and other files: glossary file, automation scripts, translation memories. Whenever Lokalize opens a file without a project loaded, it will search for a project file in the parent folders (up to four levels). Alternatively, you can specify the project file via the **`--project`** flag when starting Lokalize from the command line.

For each project you select your role in it (*translator*, *reviewer*, *approver*), which in turn affects the workflow phase Lokalize automatically picks up for files you edit.

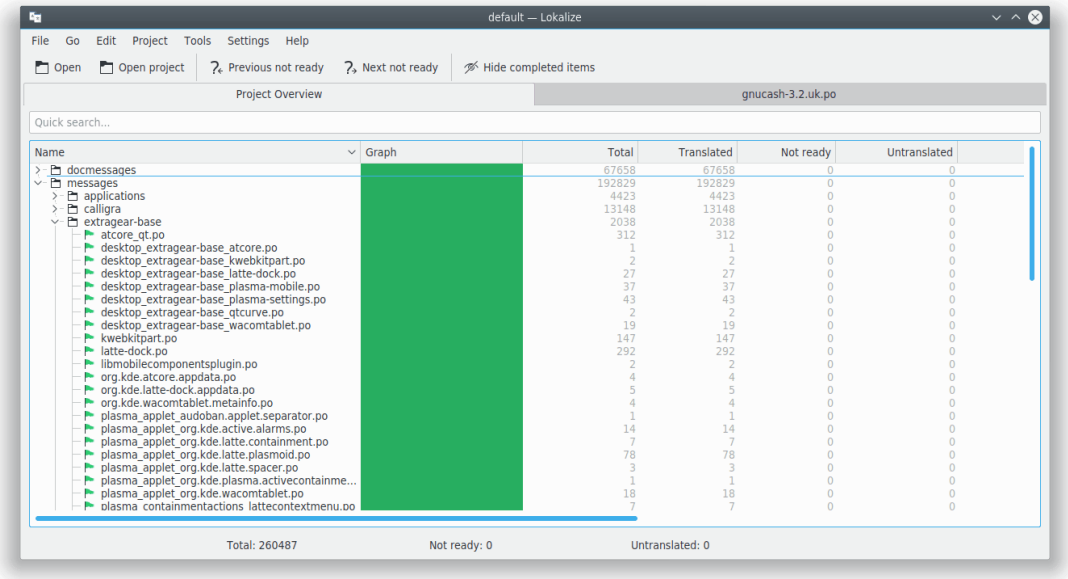
It is possible to define a word wrapping position on a project level. Just use the **Project** → **Configure project...** menu item then go to the **Advanced** page.

> NOTE
> Translation memories (unlike project files, glossary and scripts) are not shared between the translation team members, as they are created and stored under the user's home folder, meaning that the translation memories for all of the projects are stored in the same folder and thus can be used when other projects are opened.

3.2 Project Overview tab

When you start Lokalize first time, you will see an empty **Project Overview** tab. Project Overview is a file manager view, which helps you keep an overview of your PO files. The Lokalize suite will help you to translate quickly and also to keep translations consistent. Lokalize workflow implies that you start with creating/opening a project. The Project Overview tab displays a file tree with statistics with current project, such as percentage of the translated units completed and the last translator. It allows you to open a selected file in a tab of current Lokalize window.

To create a new project, use **Project** → **Create new project**. This will guide through the steps to create a new project. In **Project** menu you can also find options like **Project overview**, **Configure project**, **Open project**, and **Open recent project**.



The **Project Overview** tab displays a file tree with statistics for a current project, such as the percentage of translated units completed and the last translator. It allows you to open a selected file in a new tab of the current Lokalize, window.

> NOTE
> You can switch off the completely translated files from the **Project Overview** using the **Hide completed items** button on the main toolbar or **Ctrl-T** default shortcut.

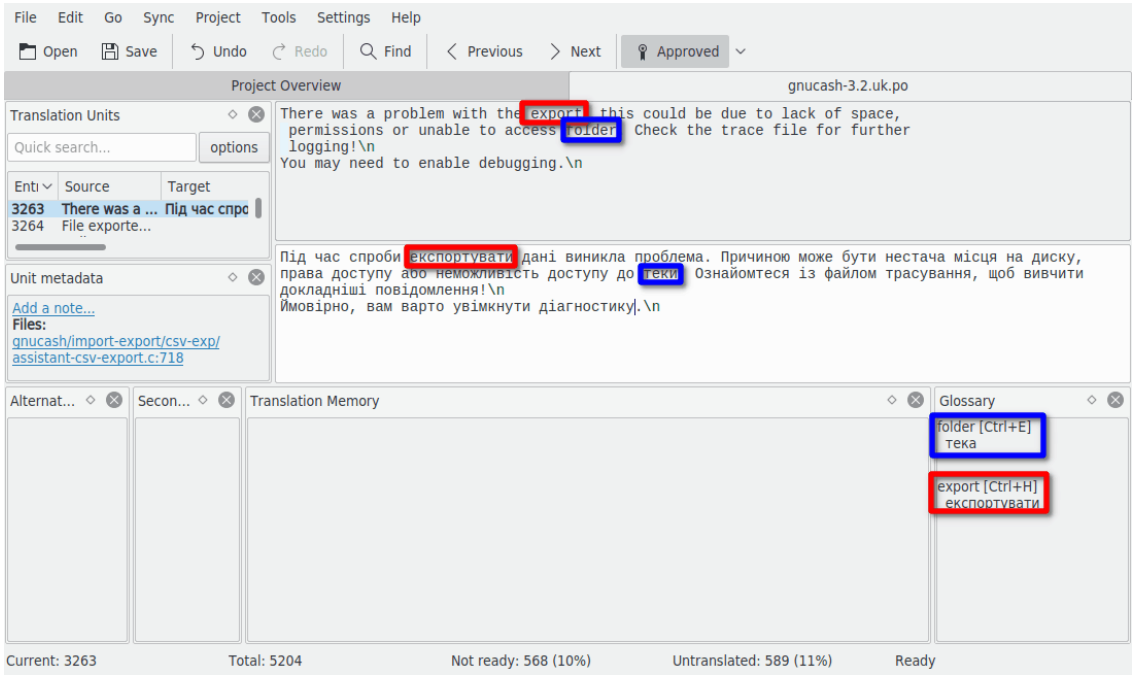
When you start Lokalize the next time it will show the last opened file by default. You can switch off this behavior with the **Restore the previously opened files when opening Lokalize** item on the page **General** of the configuration window which can be opened by choosing the **Settings** → **Configure Lokalize...** main menu item. The same page can be used to configure Lokalize behavior when you switch to the next/previous tab in it. It is possible to go through the tab list according to the tab positions or according to their activation order.

Chapter 4

Glossary

Have you ever become tired of typing the same long text sequence several times just because it would take more time to find its translation for a copy and paste? Now you will only have to find the (frequent) word sequence in the **Glossary View**, and then insert it by pressing a shortcut.

Of course the glossary should be populated with word sequences first. Lokalize has a handy glossary editor that allows an explicit search over the entire glossary.



Chapter 5

Translation Memory

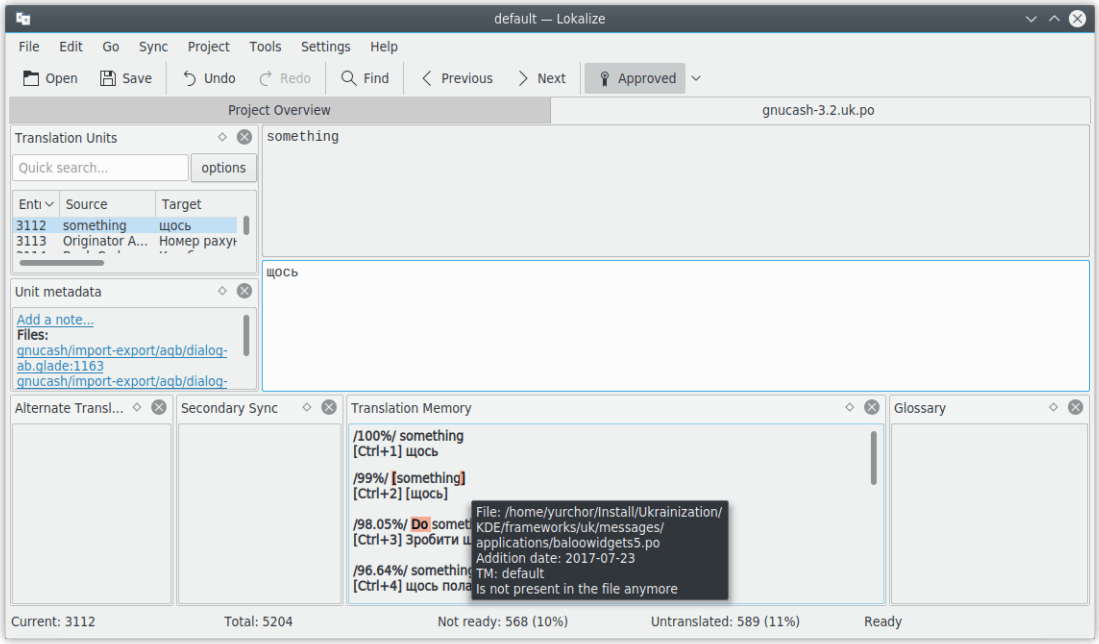
The **Translation Memory** view allows you to drag and drop a folder with translation files from say Dolphin into the view, and then, within few minutes, translation suggestions will be shown automatically on the unit switch. To insert the translation suggestions into the file, use **Ctrl-1**, **Ctrl-2** and so on, depending on the number of suggestion.

Use **Tools** → **Manage translation memories** to add/manage projects to your Translation Memory. Here you can also import or export data from `tmx` file format.

Pressing **F7** will open **Translation Memory** tab, which allows you to query the TM freely. Clicking a search result will open the corresponding file and unit. If you want to quickly open some file in the project (and it is added to TM), then instead of browsing **Project Overview** you can just type its name into the **File mask** field, accompanied by '*'.

The TM engine indexes all entries, including non-ready and untranslated ones. This allows it to completely replace the Search-in-Files feature which required scanning every file in the project each time a search is done.

> NOTE
> The outdated TM entries will be deleted from the Lokalize translation memory on rescan or clicking a missing entry if you check the **Delete missing files from translation memory on Rescan or when clicking a missing entry** item on the page **Translation Memory** of the configuration window which can be opened by choosing the **Settings** → **Configure Lokalize...** main menu item.



Batch Translation:

To insert the exactly matching suggestion automatically from the translation memory database, use **Tools** → **Fill in all exact suggestions** OR **Fill in all exact suggestions and mark as fuzzy**. This feature is similar rough translation feature in KBabel.

Chapter 6

Translation Synchronization Capabilities

The **Sync Mode** (previously known as **Merge Mode**) saves a great deal of time for the editors, and for cases when two or more translators are working simultaneously on the same file, or when one has to maintain translations for several branches of software.

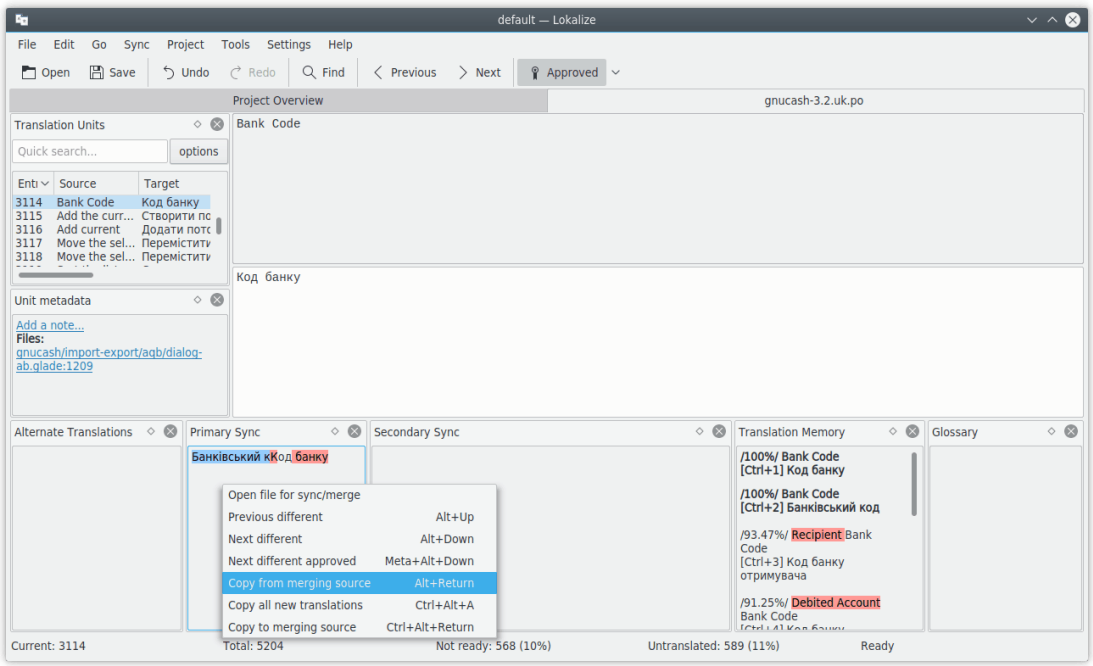
Lokalize allows quick navigation through units that differ, and displays word-by-word differences. Also, Lokalize has two Sync views - **Primary Sync** and **Secondary Sync**. They are identical, but the former is usually used to merge translations and second to keep in sync translations for two software branches.

After you copied translation from auxiliary file (*synchronized* it), any subsequent changes made to this unit will be replicated back to auxiliary file.

6.1 Merging

One use of **Sync Mode** is reviewing changes made by (new) contributors, when you cannot be sure of the quality of the work done.

Open a base file, then drop its changed version into the **Primary Sync** view, followed by **Alt-Down** or **Alt-Up** (remember that shortcuts may be modified in a usual way for all KDE applications) to navigate through entries that are different.



6.2 Replication

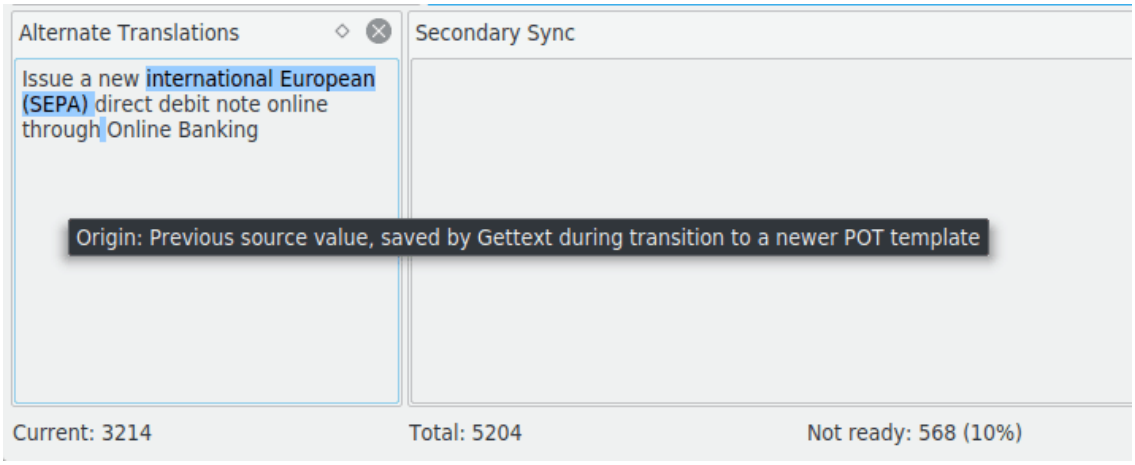
Sync Mode may also be used to make changes to translation for two branches simultaneously. Set **Branch folder** path in your project options to the path that corresponds to base folder of the branch, and **Secondary Sync** view will automatically open files from branch. Then, each time you make changes in files of your main branch, they will automatically be replicated to the branch (of course, if it contains the same English string).

For example, if you work on KDE translation, you can checkout trunk to `/home/xx/hacking/kde/trunk/l10n-kf5/YOUR_LANG` and branch to `/home/xx/hacking/kde/branches/stable/l10n-kde4/YOUR_LANG`. Create Lokalize project: `/home/xx/hacking/kde/trunk/l10n-kf5/YOUR_LANG/project.lokalize` and set `BranchDir=../../../branches/stable/l10n-kf5/YOUR_LANG`, then work via this project, and commit changes in both trunk and branch folders.

6.3 Alternate Translations

Each unit may have several *alternate translations* associated with it. Such translations may appear during file update, when the source string is slightly changed. In this case the old translation with it's (old) source is moved to alternate translations list, so that they are not lost.

When translating software, usually gettext tools are used to prepare translation files. When original text changes, gettext tools update translation files and mark entries with changed original text as *fuzzy* (or *non-ready* in other terminology). They store previous original text so that translators could see what changes exactly were made. Lokalize simplifies the life of the translator and highlights parts of the original text that were changed in the **Alternate Translations** view.


Alternate Translations
Secondary Sync
Issue a new international European (SEPA) direct debit note online through Online Banking
Origin: Previous source value, saved by Gettext during transition to a newer POT template
Current: 3214
Total: 5204
Not ready: 568 (10%)

Chapter 7

Scripting Lokalize

Lokalize is extensible using scripts in several interpreted languages, including Python and JavaScript. Scripts are usually integrated into the Lokalize UI as menu actions (to which you may assign a keyboard shortcut). The location and name of the menu entry for the script is defined in its accompanying .rc file. On each project open, Lokalize scans `PROJECTDIR/lokalize-scripts` folder for .rc files and adds them to a *cache* file called `PROJECTDIR/lokalize-scripts/scripts.rc` (so you shouldn't generally want to add it project's version control system). RC files also contain script paths, which may be relative to .rc file folder, or to a system scripts folder - they both are tried (though they *should* be kept in a relative location if you want to share them with other people in your project). For example, you can specify `../../scripts/lokalize/opensrc.py` to load a script from the global kde4-l10n scripts folder (i.e. not specific to your language).

Examples of .rc files may be found in Lokalize install folder (usually `/usr/share/lokalize/scripts/`) and in the KDE repository. Here you can find more script examples, including JavaScript-based `check-gui.js` that runs automatically on each file save (this is achieved via special option in .rc file). If you're familiar with Python or JavaScript, the code should be self-explanatory.

Below are links to API references. Everything marked as *Q_SCRIPTABLE* may be used from scripts.

- Editor object API reference
- Lokalize object API reference
- Project object API reference

7.1 Pology

One of the best scripting capabilities application is the use of Pology in Lokalize.

The interaction with Pology can be configured using the page **Pology** of the configuration window which can be opened by choosing the **Settings** → **Configure Lokalize...** main menu item.

For more information, please refer to Pology homepage.

Chapter 8

Credits and License

Lokalize

Some code was taken from KBabel, the Lokalize predecessor.

Author:

- Nick Shaforostoff shaforostoff AT kde.ru; Shankar Prasad svenkate AT redhat.com; Sweta Kothari swkothar AT redhat.com

See the Lokalize homepage for more details.