

The Lokalize Handbook

Nick Shaforostoff



The Lokalize Handbook

Contents

1	Introduction	1
2	Projects	3
2.1	Project Overview tab	3
3	Glossary	4
4	Translation Memory	6
5	Translation Synchronization Capabilities	8
5.1	Merging	8
5.2	Replication	9
5.3	Alternate Translations	9
6	Scripting Lokalize	11
7	Credits and License	12

Abstract

Lokalize is a computer-aided translation system that focuses on productivity and quality assurance. It has components usual for CAT tools: 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

When you start Lokalize first time, you will see an empty Project Overview tab. Lokalize workflow implies that you start with creating/opening a project.

If you are working along, chances are you are going to translate OpenDocument file. To do this select Project → Create new project. The wizard will ask you about your source document, convert it to Lokalize internal format, XLIFF, and create project for it. After you finish translating choose Tools → Merge into ODF to integrate your translation into a copy of source document.

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

Translation files are opened in separate tab, with two big multi-line edits as well as a bunch of *toolviews*. 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**.

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 unit list in Translation Units view. Once you are done, press **Page Down** to start moving along the filtered list.

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

The Lokalize Handbook

through the file according to the state of their translation. See Go menu for the shortcuts. When navigating, untranslated units are treated as not ready. Also you may use filtering feature of Translation Units toolview. Pressing **Page Down** actually takes you to the next unit in filtered/sorted list of that toolview.

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 2

Projects

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 a workflow phase Lokalize automatically picks up for files you edit.

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.

2.1 Project Overview tab

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 new tab of the current Lokalize, window.

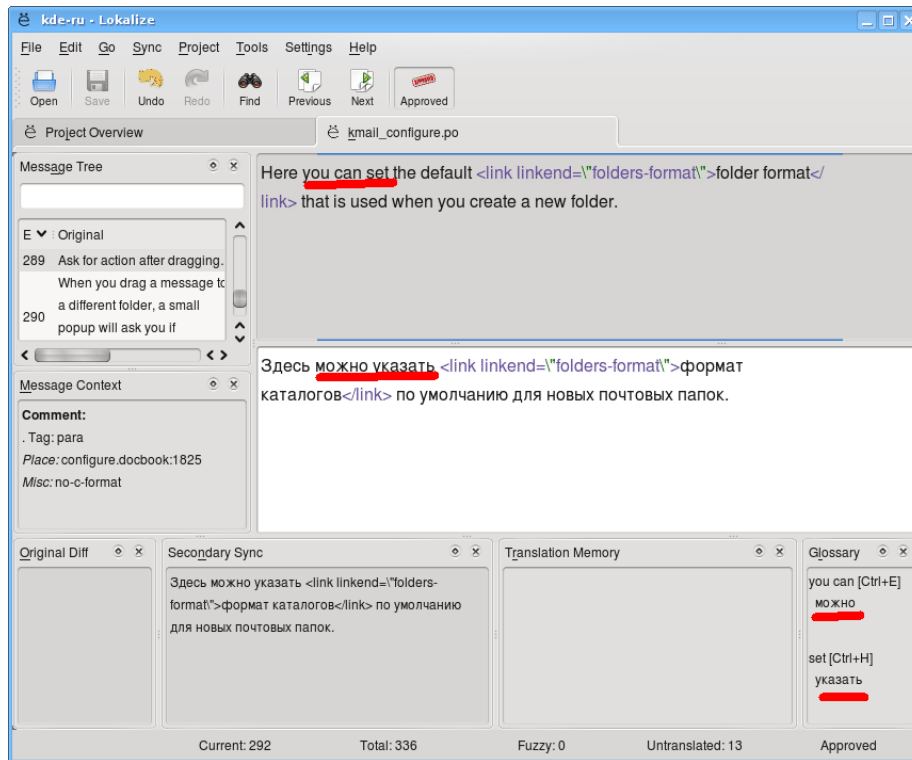
Chapter 3

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.

The Lokalize Handbook



Chapter 4

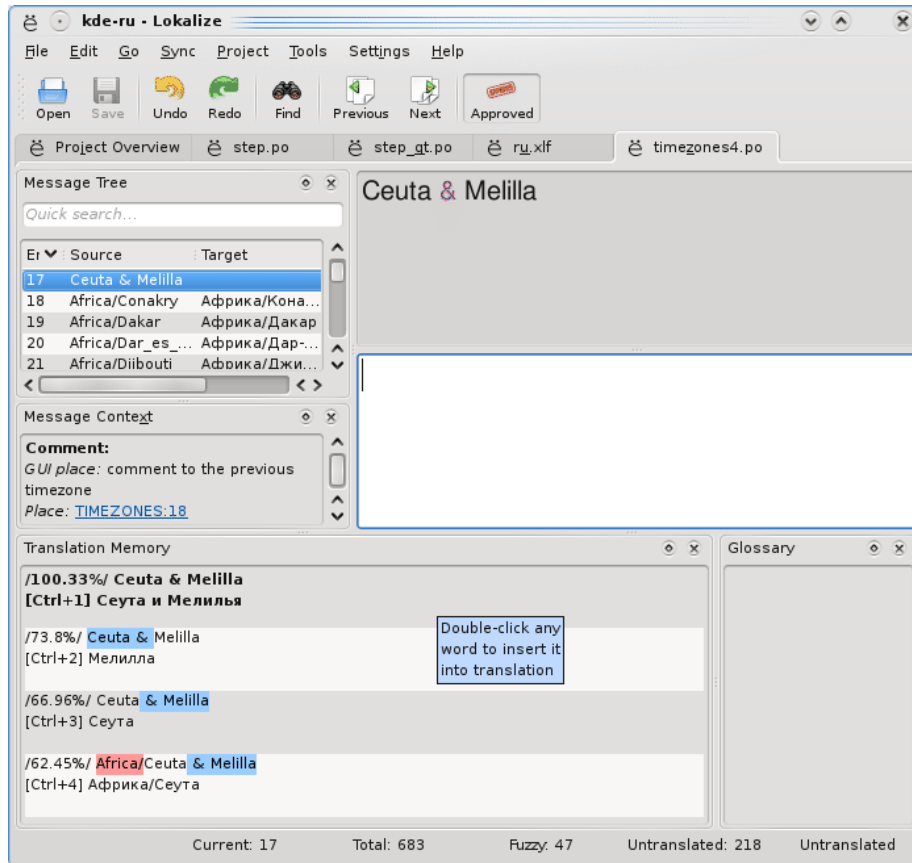
Translation Memory

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.

Pressing F7 will open Translation Memory tab, which allows free querying TM. Clicking a search result will open corresponding file on corresponding 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 just type its name into File mask field accompanied by '*'.

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

The Lokalize Handbook



Chapter 5

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.

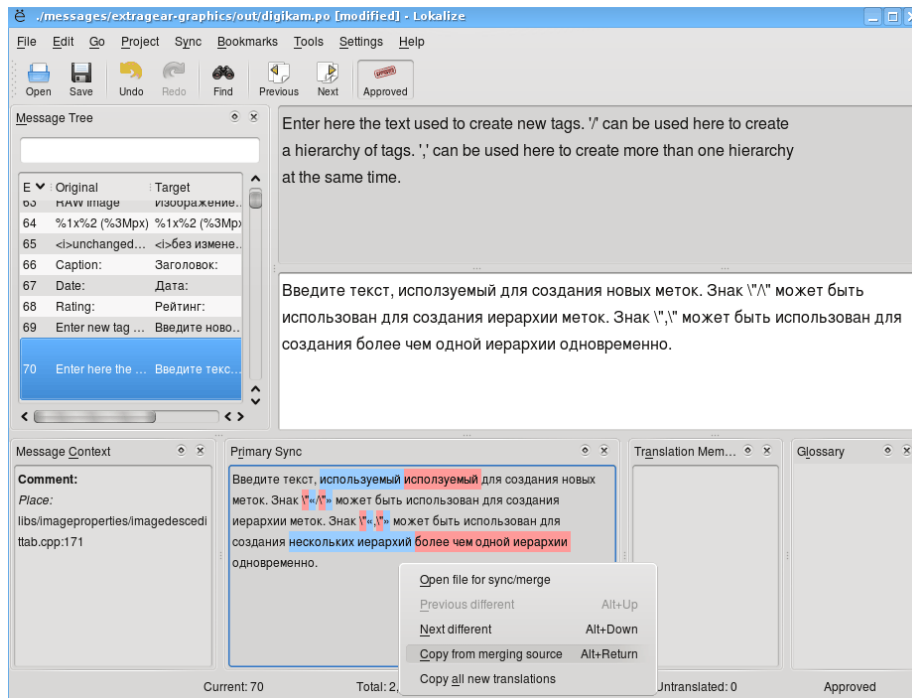
After you copied translation from auxiliary file (*synchronized* it), any subsequent changes made to this unit will be replicated back to auxiliary file.

5.1 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.

The Lokalize Handbook



5.2 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).

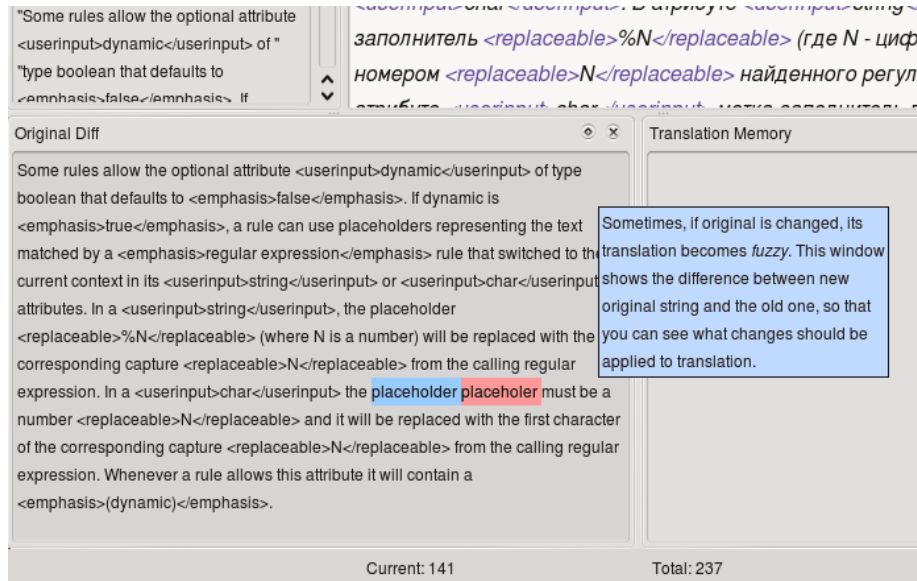
For example, if you work on KDE translation, you can checkout trunk to `/home/xx/hacking/kde/trunk/110n-kde4/YOUR_LANG` and branch to `/home/xx/hacking/kde/branches/stable/110n-kde4/YOUR_LANG`. Create Lokalize project: `/home/xx/hacking/kde/trunk/110n-kde4/YOUR_LANG/project.lokalize` and set `BranchDir=../../branches/stable/110n-kde4/YOUR_LANG`, then work via this project, and commit changes in both trunk and branch folders.

5.3 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.

The Lokalize Handbook

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 life of translator and highlights parts of original text that were changed in Alternate Translations view.



Chapter 6

Scripting Lokalize

Lokalize is extensible using scripts in several interpreted languages, including Python and JavaScript. Scripts are usually integrated into Lokalize UI as menu actions (to which you may assign a keyboard shortcut). The location and name of 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 are tried both (actually they *should* be relative if you want to share `.rc` file with other people in your project). So you for example can specify `../../scripts/lokalize/opensrc.py` to load script from [global kde4-110n scripts folder](#) (i.e. not specific to your language).

Examples of `.rc` files may be found in Lokalize install folder (usually `/usr/share/kde4/apps/lokalize/scripts/`) and in [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

Chapter 7

Credits and License

Lokalize

Program Copyright (c) 2007-2009, Nick Shaforostoff shaforostoff@kde.ru

Some code was taken from KBabel, the Lokalize predecessor.

Documentation Copyright (c) 2007-2009 Nick Shaforostoff shaforostoff@kde.ru

Author:

- Nick Shaforostoff shaforostoff AT kde.ru

See the [Lokalize homepage](#) for more details.

This documentation is licensed under the terms of the [GNU Free Documentation License](#).

This program is licensed under the terms of the [GNU General Public License](#).