Localization Primer

Vedran Vučić <vvucic@eunet.yu>
Nikola Kotur <kotnick@gmail.com>

Compiled for International Training Seminar
on Localization of Free Software
on Romani language

Linux Center
www.gnucenter.org.yu

Čortanovci, Serbia and Montenegro
December 04 – 08, 2005.
Project Supported by the Open Society Institute, Hungary

Typographical conventions:

a. In this document text is written by using font Bistream Vera Serif 12pt, no Italic and no Bold.
b. Chapter titles are written by using font Bistream Vera Serif 16pt bold, no Italic.
c. Chapter subtitles are written by using font Bistream Vera Serif 12pt bold, no Italic.
d. The figure captions are written by using font Bistream Vera Serif 8pt, no Italic and no Bold.
e. Names of menues, software packages, string content and commands are written by using font Courier 12pt.
f. Sequence of steps to perform certain operations is written by using font Courier 12pt.

Table of contents:

Introduction

Todays world is characterized by high discrepancies of the development of various countries. Modern technology is being considered as a luxury that belongs to the rich and principle of equal chances for the development is not being respected. Proprietary software is by its definition closed and user does not have access to its internals and/or is not allowed to modify the software in anyway. However, success of deployment of software requires ability to address user having in mind their cultural characteristics. On the other side, lessons drawn from many development projects show that the development may be sustainable and well designed and managed if those who are developing themselves are owners of the development. From that point of view, free software is appropriate environment to give users power and ownership on the development tools. Flexible licensing provides liberating legal framework for the good governance of the development.
Indeed, while being English-centric, software and technical documentation was/is hard for deployment and use by people who don't speak English language. That problematic situation raised important questions about the ownership of implementation of information technologies. GNU/Linux and similar free software projects introduced a package of tools which enable users to localize software and documentation to their native languages. This enables users from various cultures to make information and knowledge accessible to their people and to provide them with tools for the development (i.e. improvement of education system, housing, increasing employment opportunities, fostering economic growth etc.). Free software offers to the users ability to translate menus, commands, help files with ease. The process of translation is determined by the following processes:

a) **internationalization** – internationalization is process which enables a certain software to be translated on other languages

b) **localization** – localization is process of translation of software that is written on one language (i.e. English, German, French etc.) on other language (i.e. Romani, Serbian, Hungarian, Slovak, Russian etc.) They do have the same meaning in this document, since this document is concerned with translation that is actually localization.

Thus, software developers themselves should take care on need to enable the users to translate software on their own languages and provide them with the tools to do so. There are, in the world of GNU/Linux several packages that enable users to do translation activities efficiently. However, localizer himself/herself needs to understand what is the technical background and hierarchy of localizing the software. That is not hard, but it is prerequisite in order to do localization efficiently and to help others to enjoy freedom of software and help their own communities. We will show in this manual contained of training sessions we have had at the training

seminar dedicated to translation of software on Romani languages. We used Kbabel translation tool while translating some KDE applications.

Mechanism of translation

We presented below a diagram that explains what is mechanism behind localization support for GNU/Linux and other free software. Persons involved in localization process do not need to be programmers or engineers. However, basic knowledge of technical aspects of localization is desirable. That basic knowledge may be acquired in one short lecture held by a person that is skilled and capable to present technical information in a simple way.

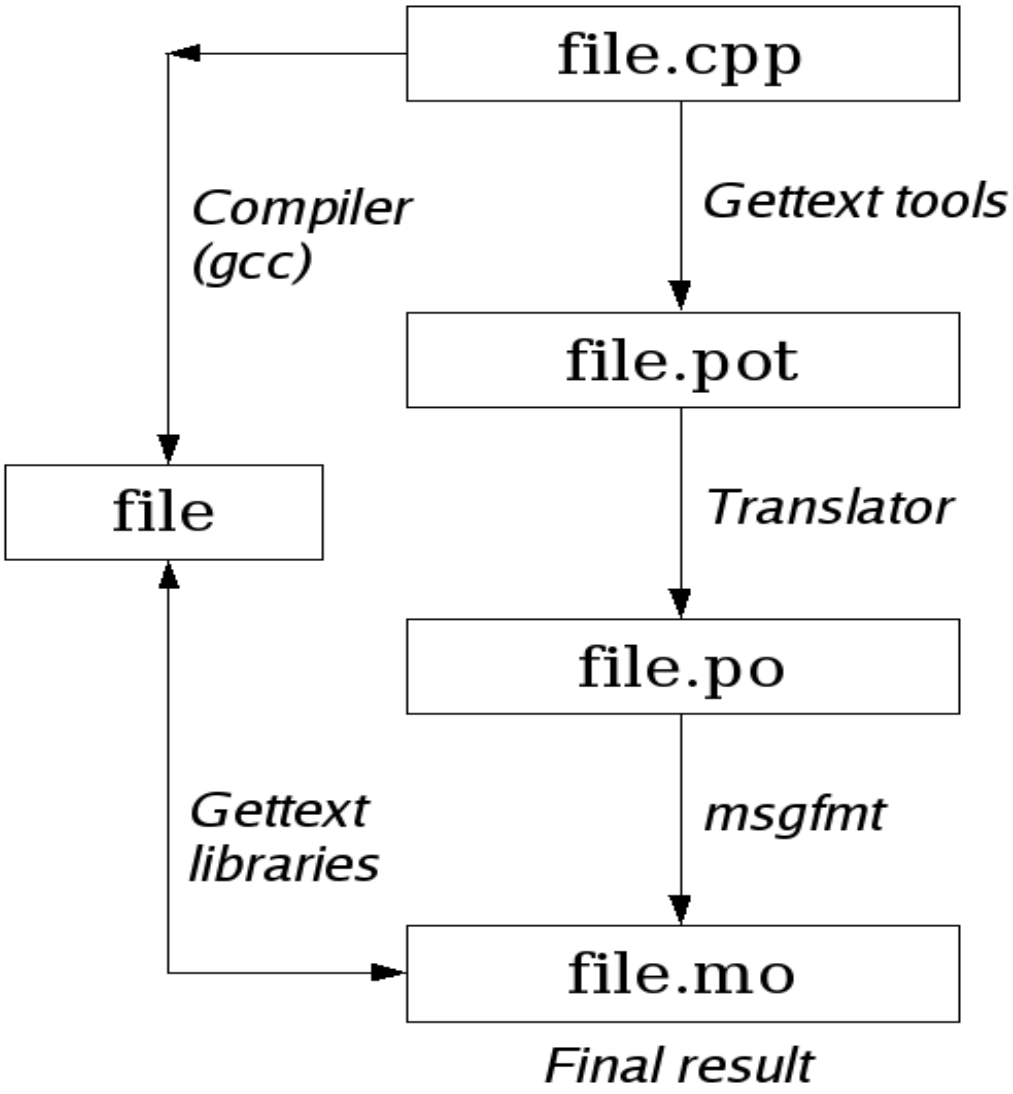

Figure no.1: mechanism of translation

Software developers realized that support for every existing language should be straightforward and simple. Consequently, after several years of development, a library and set of tools named `gettext` became available for deployment. `Gettext` is just a name for several programs and library routines[1] that serve for localization and internationalization of free software. Every free software must be `gettext` aware which is unlikely with proprietary software. That means software developer should develop software capable to be localized. In free software projects people are aware of that very important feature. `Gettext` is standardized way for internationalization and localization which is used all over the world.

Above diagram shows mechanism behind localization process for GNU/Linux and other free software.

Process of Localization

Since free software has its source code open to everyone, localizer can start from the file with the program source code[2]. (see above diagram) In our case, we start from `file.cpp` which is a source code that is `gettext` aware. However, we do not need to translate all code. We translate part of the code that is related with interaction with users. Various `gettext` tools extract strings[3] from source code file, and put them in the file with extension `.pot` let say, `file.pot`. `POT` files are Portable Object Templates; a collection of untranslated strings waiting for humans to translate them to her/his native language.[4] Person takes `POT` file and then translate all the strings, and then saves it as `.po` file named `file.po`, in this example. Difference between `POT` and `PO` file is that `POT` file is not translated at all. But, it is rule which says that even if only one string is translated, then translator should name it as `PO` file, to avoid possible confusion. `PO` file, once translated, is a collection of strings in a text form. A special tool from `gettext` package, named `msgfmt` translates the text strings to binary data, with file extension `mo`, which stands for Meta Object. In our example, that file is called `file.mo`.
This is possible to do by executing a command:

```
msgfmt -o file.mo file.po
```

1 Library routines are special types of programs that other software can use, thus new features and functionality are easily added, since software can relay on those library routines.
2 Software is written in a programming language as source code, which people can read and understand, and then convert to binary, executable format to be used in computers.
3 String is a technical term for one or more lines of text. For example, “I am Sam” is a string.
4 POT files for KDE graphic environment may be found on http://www.kde.org under section download.

Now strings, being binary data, are easily accessed by the program, and when a program gets compiled with gcc (GNU C Compiler[5]) into executable form, it uses gettext libraries to take the strings from a file.mo and display them to user.

Internals of PO files

The structure of PO file should be well understood by the translator. Although the most translators use GUI[6] programs to edit PO files, sometimes an error can occur. In order to avoid errors translators should know internal structure of these files.
PO file is made of many entries, and the most frequently one PO file holds all translations for a specific software package. Each PO file is consisted of entries. This is the format of the entry:

```
white-space
#  translator-comments
#. automatic-comments
#: reference...
#, flag...
msgid untranslated-string
msgstr translated-string
```

Every entry usually begins with a white-space, or a delimiter. This is optional, but all the tools for automatic translation will add a white-space, or an empty line. It is easier to read PO files where entries are divided with empty line.
After white space, delimiter there is a space for two types of comments that are marked with # (hash) symbol. If comment has a white-space right after sign #, then it is a translator's comment. Translator can comment an entry this way when manually editing PO file or when editing by using GUI application. This way, a translator can leave some notes for the next translators, or even to himself. Then, there are comments which have some symbol after sign #, and those are created and maintained by gettext tools. Note that all comments are optional, and translator is not required to use them, but it is a good habit to comment an entry when you find that it is appropriate.
Comments starting with "#:" are for reference. These point the source file and position of the string. (Please check example that is mentioned below).
Special types of comments start with this mark: "#," Gettext programs use it to put the flags which will help with diagnostic messages that serve for

5 Compiler is a program which transform source code into binary code understandable by computers.
6 GUI is an acronym for the Graphic User Interface.

identification of possibly incorrect translation. The most important flag is `fuzzy` flag, and it means that the translation may not be correct.
The original string from source file is introduced with `msgid` keyword, and translation with `msgstr` keyword.
Let's see one `PO` entry as an example:

```
# Check it later, Daniel!
#: komparepart/kompareprefdlg.cpp:39
#, fuzzy
msgid "View Settings"
msgstr "Dikh lačaripe"
```

This entry begins with comment, and it was put there by translator. Comments are for other translators, `gettext` tools don't pay attention to it. The next comment called reference comment, and we can spot it because it starts with "#:". This reference comment in our example says that string "View Settings" is taken from source file named `kompareprefdlg.cpp`, line 39, in `komparepart` directory. The next comment is the comment for flags, and this one is informing us and `gettext` tools that the translation is fuzzy, meaning that it might contain wrong translation. Fuzzy entries occur after automatic translation, or when programmer changes string contents, but not position. The translator should review fuzzy entries, check if it is correct, and act appropriately. Finally, in our example, we have original message (`msgid`) from source file and it's translation (`msgstr`).

Common examples of strings

This section will cover the most common original strings and the rules for its translation. In the table below, you can see some most often and interesting strings:[7]

The original string	***Explanation***
_:	Strings that start like this are not for translating. They are to inform translator about something `gettext` tools or program's author considered important.
_n:	Strings that have this group of symbols at the start are telling the translator that this string needs to be translated in plural form.
Open %1	All the words that begin with a

7 It is very important to learn to communicate with string signs in order o avoid confusion and provide those who continue translation with enough information that will enable them to translate properly free software.

The original string	***Explanation***
	percentage sign, %, are not to be translated. Program substitute them with another string and must remain intact. Kbabel will colorize these strings into blue color, so we can easily recognize them. So, translator should translate everything else and leave %1 in translation as it is. But, note that 1% is not special at all.
Watch me.\n	\n in most programming languages stands for a new line. A translator have to copy \n into translation, and try to maintain the structure of string, if he/she can.
$KDEDIR/share/apps/khangman/data/%1/%2	As a previous example, we have %1 and %2, so we won't touch them. But this example is special because it is a directory path, and these are not translatable, so this string should be copied into translation.
_: NAME OF TRANSLATORS\n Your names	When you see original string like this, then Kbabel expects from you to enter your name. People in free software like to get credited for doing something, and are very sensitive about it. So, you enter just your real, full name here.
_: EMAIL OF TRANSLATORS\n Your emails	When you see this string, it is required to enter your email address.
disabled	Almost all HTML[8] tags can be found in original strings from source files. KDE uses standard HTML tags to describe certain properties for fonts (in this example, means to bold the text, and present it that way to the users).
&File	Sign & is used to mark the

8 HTML, which stands for HyperText Markup Language is known as the language of the World Wide Web. It uses some tags, or marks, to define formatting of the text and pages.

The original string	***Explanation***
	accelerators for menus in GUI applications. Accelerators offer an easy way to select menu items, and when you see this in a menu: “File”, then you know that you can press ALT key together with a letter F to fast access that menu. Sign & tells program where to put underscore and use the keyboard accelerator.

The original string	*Explanation*
Look && Feel	Unlike previous example, this is not a keyboard accelerator. && gets replaced with a single & sign.
file \"%1\" is a macro file	When you encounter \", you're supposed to copy them into translation just the way it was in the original string. Each \" will be replaced with a single quotation mark.

If a translator find some problem hard to solve, or is not certain about translation, while working on her own and not in the group, he/she can always state his/her problem on a mailing list[9] which is established for translators. All translators then can decide how to address the problem.

Tool for translation

Kbabel

KBabel is one of the most powerful tools for localization. And it is preferred tool for localizing KDE, since the authors of KDE are also authors of Kbabel.

When you start Kbabel, be sure to set it up correctly. Firstly, go to the `Project` menu and choose command `Configure...`

enter information about yourself in section called `Identity`.

```
Project - > Configure -> Identity
```

9 Each Translation Project maintains a mailing list for easy contact with other persons involved in the localization.

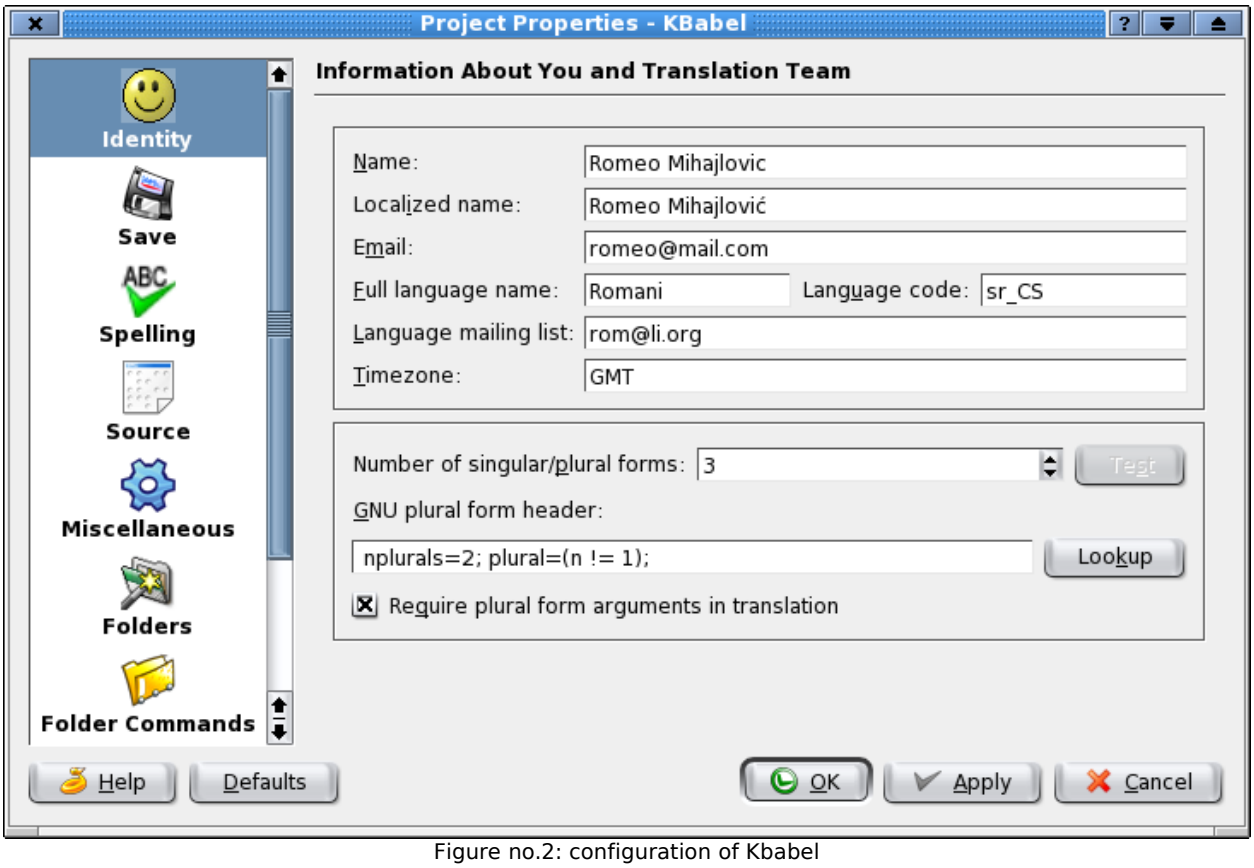

Figure no.2: configuration of Kbabel

Translator should fill the blank fields appropriately according to the following explanation of fields:

Name: Here enter your name using only English alphabet.

Localized name: Enter your name as you write it in your own language.

E-mail: Enter your email. It's important that users can contact a person responsible for a certain localized software.

Full language name: Enter full language name in English (Romani, for Roma Localization Project).

Language code: You can write ISO 639 that is set for Romani language codes.

Number of singular/plural forms: Enter the number of plural forms for your language. For Romani, it is understood that you have to enter number 3. Right after that click on *Lookup* button to generate some code for gettext tools that is helpful for successful performance of *gettext*.

Others fields are not important, and if their contents change, all translators will be informed about it.

It is very important to set up correctly a feature called *Translation Database*. Translation Database is a file on your local hard disk, where Kbabel will collect all the translated strings which you will later use for *Rough Translation* (also one of Kbabel's features). It is highly recommended to keep

this database up to date. When you begin with translation, you have to collect as many as possible already translated PO files, and then go to:

Tools -> Rough Translation

Be sure that only *Translation Database* appears in the *Use* field, and click once on it, than click on *Configure* button. After that you have to click on the *Database* tab and update your database with translated strings. In case you have grouped all the PO files in one directory (or sometimes called folder), then you'll select *Scan Folder...* option. You can also update database with single PO file by clicking on *Scan Single* PO *file.* This database will help translator a lot in the future, since it will search automatically for already translated strings. As more strings you add to it, it will be more capable to translate them for you, due to repetitive nature of software and strings.
After you've done all of the above routines, you can load PO or POT file, and start to translate it. After finishing with file, translator should add it to the *Translation Database* and then mail it to the coordinator[10] for the project. Successful cooperation and communication with coordinator is essential for good quality translation of software and integration of software packages in graphical environments such as KDE, GNOME etc.
Kbabel does have GUI that provides easy to manage workspace and makes translation easy and straightforward process. Upper left filed is reserved for the *stringid* easy while just below it is filed reserved for translation itself. Translator can choose on the right side tabs which offer more precise insight in entires that are being translated or choosing the symbols panel for reinserting in the translation, so even more complex languages may be properly translated and written.

10 Coordinator is a person that is responsible for coordination of localizers, help and data acquisition sent to the coordinator. Coordinator communicate with translators and discuss issues important for quality of translation. Coordinator finally prepares packages and include them in an official repository of translated files.

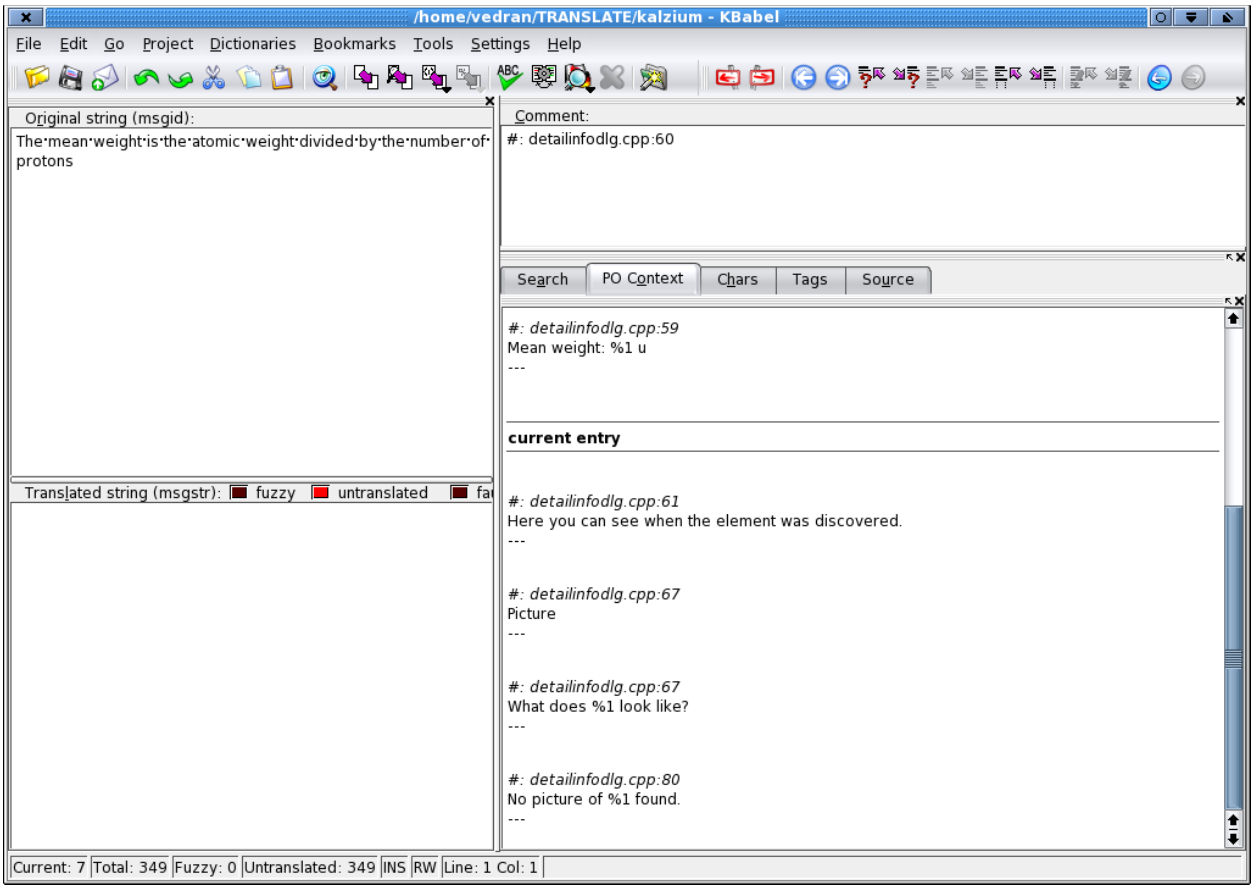

Figure no3: Kbabel's GUI

It is possible in Kbabel, to use various symbols that may be specific to certain languages with ease simply by drag and drop of symbol. Symbol may be taken from the right panel to the left below panel that is aimed for translations of `stringid` strings by choosing the symbol by pressing the left button on the mouse and while keeping the left mouse button pressed to drag and drop it on appropriate place in the word. Thus, symbols from various languages may be easily inserted in the translated text and users may work with their favorite free software without sacrificing their cultural characteristics. Figure no. 4 shows panel that offers various symbols/characters that user may choose in order to stick with proper and accurate translation of strings in a respective software package.[11]

11 Some ethnic groups do live in several countries and they do sometime use various characters and symbols that are specific to that country, yet preserving their native symbols and characters. By using this tool it is possible to respect such variations from usual ways of using symbols and characters. For example, Roma in Bulgaria use Cyrillic letter while in Slovakia use Latin letter with characters typically used in Slovakia.

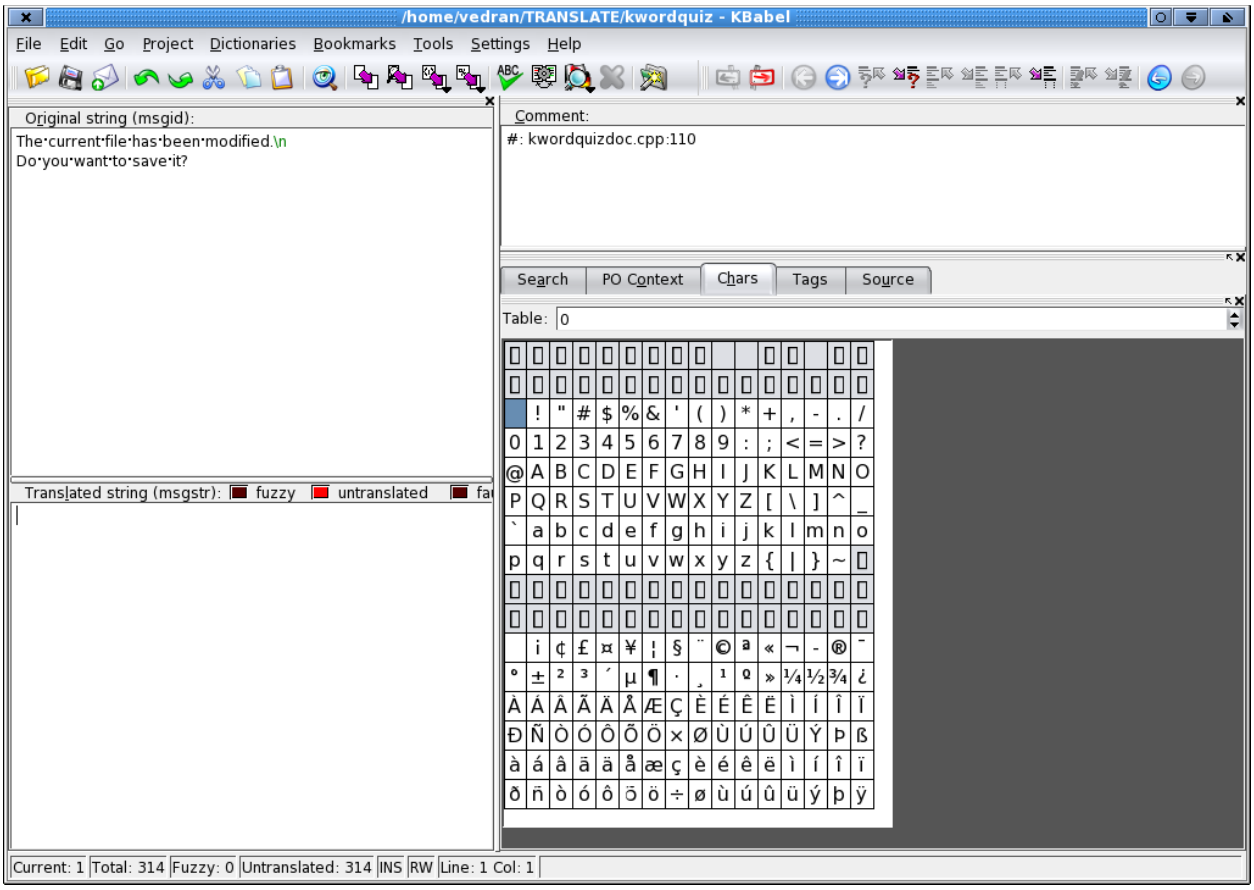

Figure no 4: panel with symbols/characters

Translation rules

Before translation you should read already translated PO files. In that way, you will familiarize yourself with most common terms and syntax.
During the first localization effort, where a certain number of applications have been translated, a dictionary should be assembled. Dictionary should serve as a reminder, and you should always follow the terms agreed in the dictionary. Any objections should be noted, and don't hesitate to tell them. But, until a certain term has been changed, the dictionary must be respected. In that way, translators will preserve the coherency and consistency of the translation, which is important for users, since they tend to get used to some terminology, even though it is not best chosen.
Do not translate the names of applications.
Always keep translation database (one of Kbabel's features) up to date, and update it contuinuously as translation proceed. Having that in mind, always first do rough translation, so already translated strings won't have to be typed in again.

Follow grammatic rules, and remember to always put a space after all interpunction signs (comma, dot, colon, etc.).
Translation of software and help files are integral part of the software and license that apply to the software apply to the translated material too.A file with the translated terms and strings gathered during the initial translation will be attached to this document, or will be otherwise available to potential translators and whoever may it concern.

Organizing translation teams

However, translation process itself may be done by one person as well as by teams of translators gathering in localization camps or localization marathons. Larger software packages with long text strings should be usually translated in pairs in a way that one person is typing and another is translating. However, they may divide job differently according to their skills, academic background etc. When teams are organized, translation itself may be realized in three possible scenarios.

Scenario no. 1

Each translator translates the same software from the beginning to the end. This scenario is good for training purposes when people translate software for the first time. This is good for mastering required skills and developing team cooperation and good habits. (Figure no. 5 explain scenario no. 1)

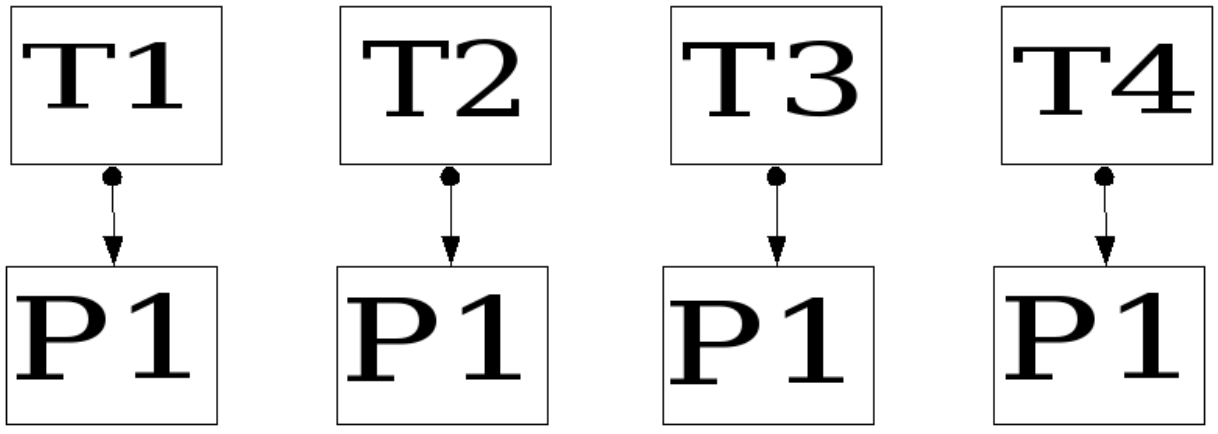

Figure no. 5 Each translator (T1,T2...) translates the same program (P1)

Scenario no. 2

In scenario 2 each translator translate different software/program from the beginning to the end. This scenario is good for increasing efficiency and fostering team to discuss meaning of words that may appear in different contexts. This scenario is good when translators translate software that belong to the same group of software packages such as education, multimedia, office etc. (Figure no.6 explain scenario no. 2)

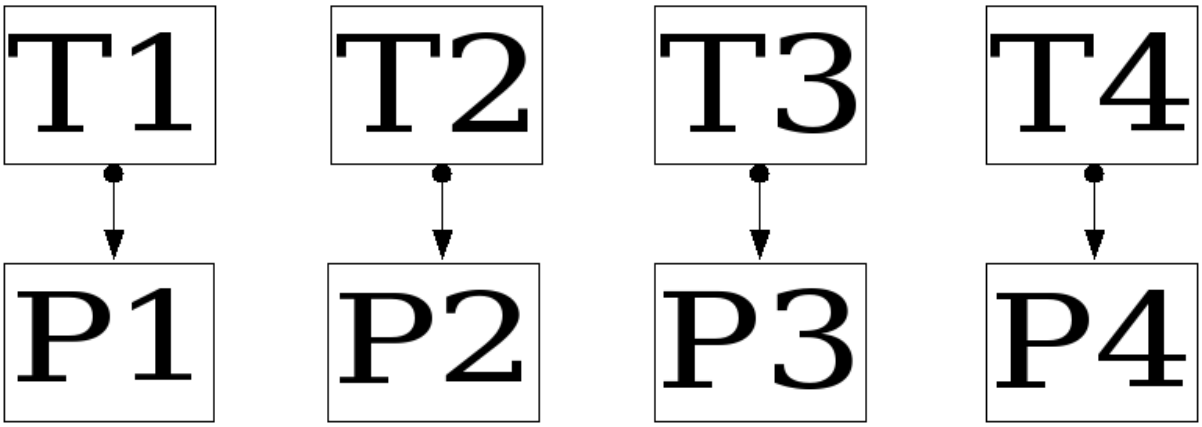

Figure no.6. Each translator (T1,T2...) translates different program (P1,P2...)

Scenario no. 3

In scenario 3. each translator translates the same software package but different part of it. This scenario is good when translators do have already some experience in translation and when they already know each other and consult each other on terms appearing in different contexts. However, this is desirable practise when translators are already familiar with software they translate so they know its features and functionality. (Figure no. 7 explain scenario no.3) However, all parts should be merged after translation into one piece. (Figure no. 8 explain merging in scenario no.3)Although merging is not complicated process it should be done by skilled person. Proper merge will avoid any confusion and potential loss of integrity of translation itself. Merged file should be installed on all machines that translators use so they can check whole software and check quality of translation of all its parts. They should discuss translation and consistency of the translation as a whole and correct parts that may confuse the users of the software. Linguists, writers, journalists, experts from certain fields should be consulted and if possible present during the translation itself.

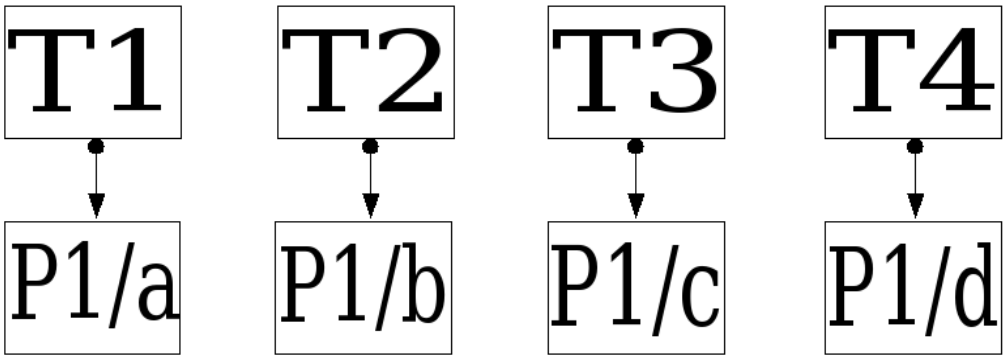

Figure no 7. Each translator (T1,T2,..) translates only part of program (P1/a,P1/b...)

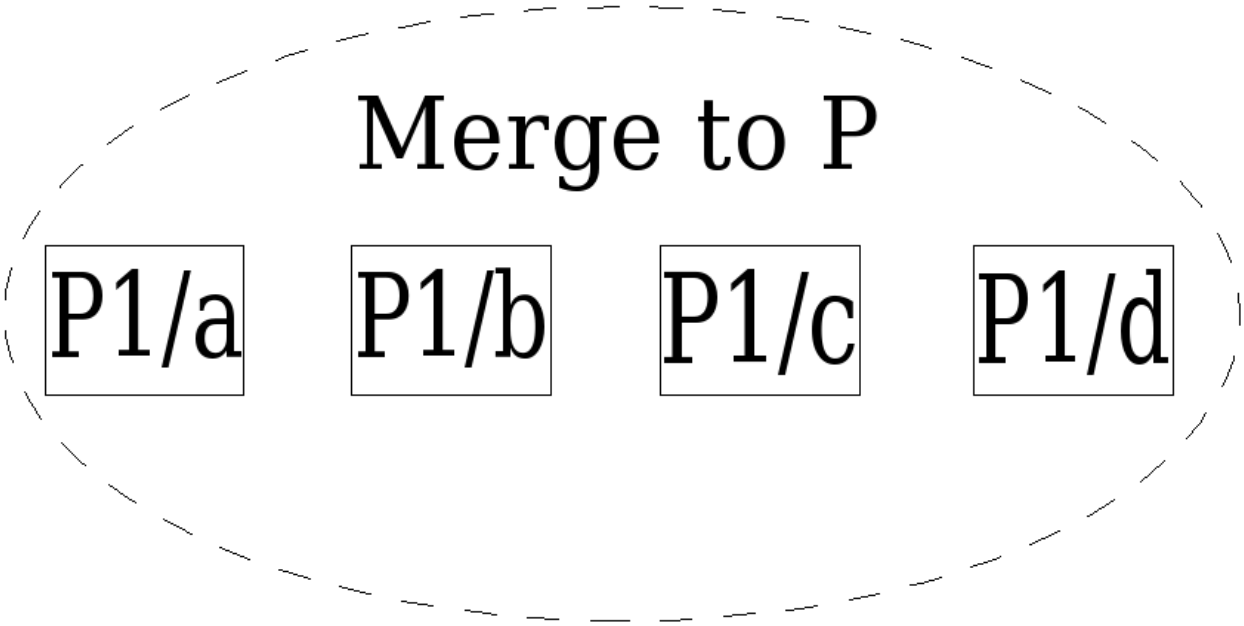

Figure no. 8 Each part of translated software package (P1/a, P1/b..) is merged into translation of software package P

Each translation regardless of possible scenario requires checking of consistency and overall quality of translation. However, some languages such as Romani does have a variety of dialects. Although those dialects do not differ very much it is needed to be sure that translation of one dialect is done properly and that it may serve as referent translation to be used for translation on other dialects. After checking overall quality and consistency of translation on one dialect it is much easier to translate the same software on other dialects by defining the words that are different and replacing them with words from other dialect. (Figure no. 9 explain process of translation to other dialects.) Since differences between dialects are oftenly not significant

it may be useful to crerate a dictionary of the words that are specific for some dialects. This will make much easier to replace words from one dialect into another.

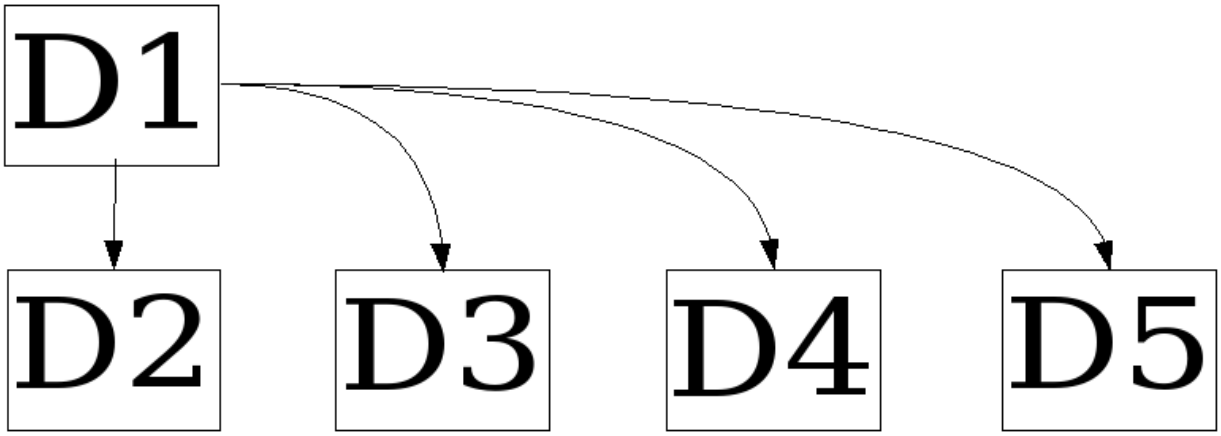

Figure no.9: Translation on one dialect (D1) serves as reference for translation on other dialects (D2,D3,D4,D5....)

Appendix A: Content of CDROMs

This manual is accompanied with two CDROMs. On the first CDROM there is complete bootable installation of Vector GNU/Linux SOHO 5.01 (http://www.vectorlinux.com). You can install it by enabling boot from CDROM in your computer's BIOS setup. Maybe first install may be done with assistance of skilled person, but surely, in the future one can do it on his/her own.

The second CDROM is divided in the following folders:

- **documentation:** In this folder there is this manual and template for OpenOffice for creation of documentation. In this folder you can find English - Romani dictionary of 600 Romani words. Dictionary is created in software package KwordQuiz and it is readable by that software and it is saved in the file `engrom.kvtml`.
- **i18n:** In this folder you can find packages in which there are files with translated KDE on several languages. For example, file that is named kde-i18n-sk-3.3.2.tgz correspond to translation on Slovak language. It can be installed easily by inserting CDROM in your CDROM device, and typing the following commands:
  - `cd /mnt/cdrom/i18n`
  - `pkg -i kde-i18n-sk-3.3.2.tgz`

  You can install packages for other languages too by following above mentioned procedure.
- **pot-files:** In this folder you can find `POT` files of some packages that you may want to translate, by opening them in Kbabel and translating as `PO` file according to instructions written above. In this folder you can find also `PO` files with translations at the training seminar on localization of free software on Romani language.
- **soft-lib:** In this folder you will find packages that are essential for successful translation. You will find in this folder the following packages:
  `kdeedu-3.3.2.tgz`
  `kdesdk-3.3.2.tgz`
  `koffice-1.3.5-i486.tgz`
  You can install them by the following commands:
  - `cd /mnt/cdrom/soft-lib`
  - `pkg -i kdeedu-3.3.2.tgz`
  - `pkg -i kdesdk-3.3.2.tgz`
  - `pkg -i koffice-1.3.5-i486.tgz`

Creative Commons License

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

g. **"Collective Work"** means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

h. **"Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.

i. **"Licensor"** means the individual or entity that offers the Work under the terms of this License.

j. **"Original Author"** means the individual or entity who created the Work.

k. **"Work"** means the copyrightable work of authorship offered under the terms of this License.

l. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the

Licensor to exercise rights under this License despite a previous violation.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;
b. to create and reproduce Derivative Works;
c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;
d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works.
e. For the avoidance of doubt, where the work is a musical composition:
    i. **Performance Royalties Under Blanket Licenses**. Licensor waives the exclusive right to collect, whether individually or via a performance rights society (e.g. ASCAP, BMI, SESAC), royalties for the public performance or public digital performance (e.g. webcast) of the Work.
    ii. **Mechanical Rights and Statutory Royalties**. Licensor waives the exclusive right to collect, whether individually or via a music rights agency or designated agent (e.g. Harry Fox Agency), royalties for any phonorecord You create from the Work ("cover version") and distribute, subject to the compulsory license created by 17 USC Section 115 of the US Copyright Act (or the equivalent in other jurisdictions).
f. **Webcasting Rights and Statutory Royalties**. For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any credit as required by clause 4(b), as requested. If You create a Derivative Work, upon notice from any Licensor You must, to the extent practicable, remove from the Derivative Work any credit as required by clause 4(b), as requested.

b. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Derivative Works or Collective Works, You must keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or (ii) if the Original Author and/or Licensor designate another party or parties (e.g. a sponsor institute, publishing entity, journal) for attribution in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; the title of the Work if supplied; to the extent reasonably practicable, the Uniform Resource Identifier, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and in the case of a Derivative Work, a credit identifying the use of the Work in the Derivative Work (e.g., "French translation of the Work by Original Author," or "Screenplay based on

original Work by Original Author"). Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Derivative Work or Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Derivative Works or Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. Each time You distribute or publicly digitally perform a Derivative Work, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.