Creating or maintaining translations

This page explains some common processes that are performed by any user aiming to have its new language file created and updated.

The basic idea

In the PHP scripts (.php files), all the text strings must be written as tra("here goes the string"); which calls the tra function that translates the string.

In Smarty templates (.tpl files), you should use the Smarty \{tr\} block, for example \{tr\}Hello world{/tr}.

This is how you write text in Tiki to be language-aware.

Once you have modified Tiki the get_strings.php script can be used to scan all the .php and .tpl files in the application regenerating/changing the language files. New strings are added to language files untranslated.

Creating a new language

1. Figure out the 2-character RFC1766 code for that language. Please name it using the standard RFC 1766:
   - 2-letter language code iso639 in lower case
   - eventually "." and a 2-letter country code ISO3166 in upper case
     - For example, en = english, es = spanish, pt-BR = brazilian
2. Edit the file lang/langmapping.php to add the new language to it. You must specify the language code, as well as the name of the language.
3. Create a new directory under the "lang" directory. Its name must be the code for that language.
4. Give the adequate write permission to this directory (so that the web server can write).
5. If you are working on a Tiki stable release, copy the language file (language.php) from a language you are familiar with or from the en directory in your directory.
6. If you are working on a Tiki SVN version, run the script get_strings.php?lang=en as admin (change en with the code of your language). This script collects all the strings of the Tiki code.
   - Each text string looks like this:
     - "last modification" => "last modification",
   - Leave the text on the left as it is and insert your new text on the right.
     - For example (in German),
       - "last modification" => "letzte Änderung",
     - In French, "last modification" => "dernière modification",
       - Note that the accents are utf-8 encoding
     - If you find some Smarty variables (ex:{nb}) or some php variables(ex:%s) leave them in both parts
8. Tiki will automatically detect the new language once you save the language.php file.

Don't forget that the file needs to be in UTF-8 encoding to be displayed in Tiki.
You can download [here](#) a simple language.php file with just a few basic Tiki strings to start your translation (around 100 strings).

When testing a new language, keep in mind that strings can be cached so they may not change their language unless you clear the cache. This can be done with the Exterminator on the [System Admin](#) page (see System Admin documentation page).

---

**Updating a language file with the last Tiki code: get_strings.php**

The get_strings.php script will search and add all the untranslated strings enclosed with the translation Smarty block (\{tr\}text{/tr} to all the languages file. Note that the script will create a new document (backup file) for each language and all lang folders will contain: languages.php and languages.php.old.

To update a language file (language.php) you can run the script get_strings.php as admin.

- To update all the languages files: get_strings.php
- To update only one language (ex: en) : get_strings.php?lang=en

Then you can edit / translate the new strings.

To update all the language files (lang/*/language.php) you can also use unix command shell to run the script on all the language files and commit the changed files. **Be sure to have the last revision of the code** from within your branch before committing languages or any file.

```php
get_strings.php
// It is wise to have a quick look at the files and to check that the languages files are ok.
svn commit -m 'Updated all translation strings in language.php files using php get_strings.php' lang/*/language.php
// Once all is good you can delete backup files (languages.php.old)
find . -type f -name 'language.php.old' -delete
```

---

**Understanding language.php internal divisions**

A Tiki language.php file is divided into three groups:

- **Unused strings**
  The strings are not any more found in Tiki: because they have been modified in Tiki or because it is the value of a variable. Actually get_strings.php is not able to distinguish both cases. When you are updating the translation, it is better to keep these lines (they can help to find the close string) but at release time, this section must be deleted (except for the variable values translation).

- **Untranslated strings**
  Each untranslated strings is preceded with "//". As soon as, you have translated it, you need to take the "//", otherwise the next get_strings.php will delete your translation. The reason that they are commented is that otherwise running get_strings.php twice would put untranslated strings in this
section into the possibly untranslated strings section (not very fun if there are a lot of changes and the translated language has similarities with English and/or you don't translate a lot of computer terms).

- **Possibly untranslated strings**
  This section is just for checking. It is all the strings that have both part identical.

### get_strings.php optional parameters

- **close**: you can use the option `close` that proposes you as a comment a string already translated 'close' to the new one: `+get_strings.php?lang=en&close=1`
  //close "previously translated strings"=>'old translation". The algorithm used the levenshtein distance (the minimum letters number that differs)
- **module**: this option generates comments that describes in which .php and/or .tpl module(s) each string was found
- **patch**: this option looks for the file 'language.patch' in the same directory as the corresponding language.php and overrides any strings in language.php - good if a user does not agree with some translations or if only changes are sent to the maintainer
- **comments**: this option generate all comments (equal to close&module).
- **spelling**: this option generates a file `spellcheck_me.txt` that contains all the translated words (only once). You then have to use a traditional spellchecker (e.g. Open Office, Word or Emacs) on this file to find the errors and make the corrections in the original language.php. When running `get_strings.php` in spellcheck mode it is more convenient to not have the close and module options active (i.e. translate first and spellcheck later)
- **groupwrite**: this option makes the generated file `language.php` group writable, so that you can edit the file if you have group access but not root access. Just do not forget to run the script again without the parameter so that the write permission is reset.

### Maintaining translations between different Tiki branches

See [Translation branching strategy](#). Two tools are available to facilitate the merge between trunk and other branches:

- `doc/devtools/mergelang.php` (see the documentation in the file)
- `get_strings.php?lang=xx&patch=lang/xx/language.patch` where the `lang/xx/language.patch` is the language.php with new translations

### Translating as PO files

[+]
Text editor suggestions

Alias

- Code Howto: Add a language
- How to add a language