TRANSLATIONS

v1.8 2018/02/28

Internationalization of LaTeX 2ε Packages

Clemens Niederberger

https://github.com/cgnieder/translations

contact@mychemistry.eu

Table of Contents

1 Motivation

This package provides means for package authors to have an easy interface for internationalization of their packages. The functionality of this package is in many parts also covered by the package translator [TW19] (part of the beamer bundle). Internationalization is also possible with babel [Bra19] and it's `\addto\captions`⟨*language*⟩ mechanism or KOMA-Script's `\providecaptionname` and similar commands. However, I believe that TRANSLATIONS is more flexible than all of these. Unlike translator it detects the used (babel or polyglossia [Cha19]) language itself and provides expandable retrieving of the translated key. TRANSLATIONS also provides support for language dialects which means package authors can for example distinguish between British, Australian, Canadian and US English.

The first draft of the package was written since I missed an expandable version of translator's `\translate` command. Once I had the package available I began using it in various of my other packages so it got extended to the needs I faced there.

2 License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."

TRANSLATIONS requires the packages etoolbox [Leh19] and scrlfile (part of the KOMA-Script bundle [Koh19]).

3 Usage

3.1 Background

The TRANSLATIONS package enables the author of a package or a class (or a document) to declare translations of key words in different languages and fetch these translations in the document depending on the active language as set by babel or polyglossia. Since TRANSLATIONS checks which language is active it is generally not necessary (although possible) to specify the language for which a translation should be fetched manually.

TRANSLATIONS knows of three types of languages: main languages (see table 2 on page 12), language dialects (see table 3 on page 13), and language aliases (see table 4 on page 13). For the commands declaring or fetching a translation base languages and language aliases are equivalent. Dialects are similar to aliases but there are important differences. An alias can for example be an alias of a dialect.

Figure 1 shows what happens if TRANSLATIONS is asked to fetch a translation for a given key.

What happens if you declare a translation? There are four cases:

1. You declare a translation for a base language: this is the normal case where an internal macro is defined which can be fetched by the `\GetTranslation` command (see section 3.2).
2. You declare a translation for a language alias: this is the very same as the first case since the same internal macro is defined.
3. You declare a translation for a dialect: this is two-fold. Either a translation for the base language exists so only the translation for the dialect is saved. If the translation for the base language does not exist it is defined to be the same as the one for the dialect.
4. You declare a translation for an alias of a dialect: this is the very same as the third case as again the internal macros are the same.

Beware that if the current language is a language using a non-latin font, a translation is missing for said language, and the fallback translation needs a Latin script font then nothing *might be printed.*

3.2 Available Commands

Below the commands provided by TRANSLATIONS are explained. The symbol * means that the command is expandable. Commands without the marker aren't expandable.

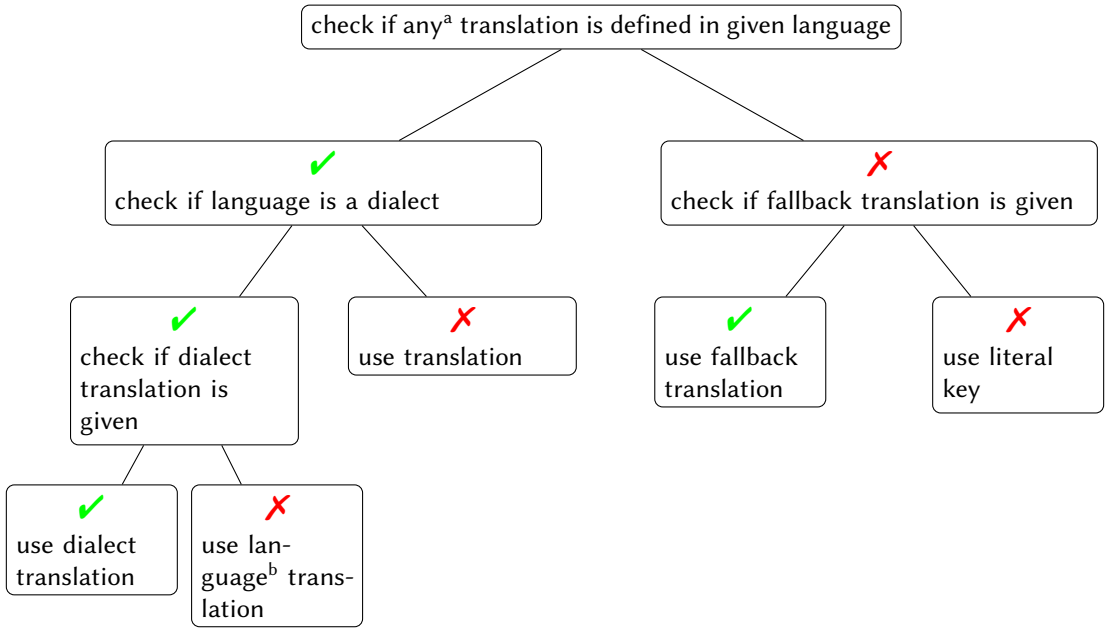

FIGURE 1: Schematic representation of TRANSLATIONS' translating mechansim. Notes: [a] except for a possible fallback translation. [b] *i. e.*, the base language of the dialect.

`\DeclareLanguage{`⟨*lang*⟩`}`
Declare a language that can be used by TRANSLATIONS. If the language already exists it will be silently redefined. This command can only be used in the preamble. It should never be necessary to use this command as TRANSLATIONS already declares loads of languages (section 4). Should you miss one please send me an email and I'll add it to TRANSLATIONS.

`\DeclareLanguageAlias{`⟨*lang2*⟩`}{`⟨*lang1*⟩`}`
Declares ⟨*lang2*⟩ to be an alias of ⟨*lang1*⟩. If ⟨*lang1*⟩ doesn't exist yet a warning will be raised and it will be defined. This command can only be used in the preamble. It should never be necessary to use this command as TRANSLATIONS already declares loads of languages (section 4). Should you miss one please send me an email and I'll add it to TRANSLATIONS.

`\DeclareLanguageDialect{`⟨*dialect*⟩`}{`⟨*lang*⟩`}`
Declares ⟨*dialect*⟩ to be a dialect of language ⟨*lang*⟩. If a translation for ⟨*dialect*⟩ is provided it is used by the translation macros. If there is none the corresponding translation for ⟨*lang*⟩ is used instead. It should never be necessary to use this command as TRANSLATIONS already declares loads of languages (section 4). Should you miss one please send me an email and I'll add it to TRANSLATIONS.

`\NewTranslation{`⟨*lang*⟩`}{`⟨*key*⟩`}{`⟨*translation*⟩`}`
Defines a translation of key ⟨*key*⟩ for the language ⟨*lang*⟩. An error will be raised if a translation of ⟨*key*⟩ in language `{`⟨*lang*⟩`}` already exists. This command can only be used in the preamble.

`\NewTranslationFallback{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
Defines a fallback translation of key ⟨*key*⟩ for the language ⟨*lang*⟩. An error will be raised if a fallback translation of ⟨*key*⟩ already exists. This command can only be used in the preamble.

`\RenewTranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`
Redefines a translation of key ⟨*key*⟩ for the language ⟨*lang*⟩. An error will be raised if no translation of ⟨*key*⟩ in language ⟨*lang*⟩ exists. This command can only be used in the preamble.

`\RenewTranslationFallback{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
Renews a fallback translation. This command can only be used in the preamble.

`\ProvideTranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.2
Provides a translation of key ⟨*key*⟩ for the language ⟨*lang*⟩. If a translation of ⟨*key*⟩ in language ⟨*lang*⟩ already exists it won't be overwritten and no error will be raised. This command can only be used in the preamble.

`\ProvideTranslationFallback{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
Provides a fallback translation. This command can only be used in the preamble.

`\DeclareTranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`
Defines a translation of key ⟨*key*⟩ for the language ⟨*lang*⟩. No error will be raised if a translation of ⟨*key*⟩ already exists. This command can only be used in the preamble.

`\DeclareTranslationFallback{⟨key⟩}{⟨fallback⟩}`
Declares a fallback translation. This command can only be used in the preamble.

`\definetranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
A version of `\NewTranslation` that *can* be used after begin document.

`\definetranslationfallback{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
A version of `\NewTranslationFallback` that *can* be used after begin document.

`\redefinetranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
A version of `\RenewTranslation` that *can* be used after begin document.

`\redefinetranslationfallback{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
A version of `\RenewTranslationFallback` that *can* be used after begin document.

`\addtranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
A version of `\ProvideTranslation` that *can* be used after begin document.

`\addtranslationfallback{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
A version of `\ProvideTranslationFallback` that *can* be used after begin document.

`\declaretranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
A version of `\DeclareTranslation` that *can* be used after begin document.

`\declaretranslationfallback{⟨key⟩}{⟨translation⟩}`
Introduced in version 1.4
A version of `\DeclareTranslationFallback` that *can* be used after begin document.

* `\IfTranslation{⟨lang⟩}{⟨key⟩}{⟨true⟩}{⟨false⟩}`

Introduced in version 1.2d

Checks if a translation for ⟨*key*⟩ in language ⟨*lang*⟩ is defined or not and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

* `\GetTranslationFor{⟨lang⟩}{⟨key⟩}`

Fetches and prints the translation of ⟨*key*⟩ for the language ⟨*lang*⟩. This command is expandable.

* `\GetTranslation{⟨key⟩}`

Fetches and prints the translation of ⟨*key*⟩ for the currently active language (as for example set by babel). This command is expandable.

* `\GetLCTranslationFor{⟨lang⟩}{⟨key⟩}`

Introduced in version 1.1

Fetches and prints the translation of ⟨*key*⟩ for the language ⟨*lang*⟩. This command ensures that the fetched translation is set lowercase. This command is expandable (well, sort of: in an `\edef` it leaves `\lowercase{`⟨*translation*⟩`}` in the input stream where ⟨*translation*⟩ is what `\GetTranslationFor` would expand to).

* `\GetLCTranslation{⟨key⟩}`

Introduced in version 1.1

Fetches and prints the translation of ⟨*key*⟩ for the currently active language (as for example set by babel). This command ensures that the fetched translation is set lowercase. This command is expandable (well, sort of: in an `\edef` it leaves `\lowercase{`⟨*translation*⟩`}` in the input stream where ⟨*translation*⟩ is what `\GetTranslation` would expand to).

`\GetTranslationForWarn{⟨lang⟩}{⟨key⟩}`

Introduced in version 1.0

Fetches and prints the translation of ⟨*key*⟩ for the language ⟨*lang*⟩. Issues a warning if no translation is available at the cost of expandability.

`\GetTranslationWarn{⟨key⟩}`

Introduced in version 1.0

Fetches and prints the translation of ⟨*key*⟩ for the currently active language (as for example set by babel). Issues a warning if no translation is available at the cost of expandability.

`\GetLCTranslationForWarn{⟨lang⟩}{⟨key⟩}`

Introduced in version 1.1

Fetches and prints the translation of ⟨*key*⟩ for the language ⟨*lang*⟩. This command ensures that the fetched translation is set lowercase. Issues a warning if no translation is available at the cost of expandability.

`\GetLCTranslationWarn{⟨key⟩}`

Introduced in version 1.1

Fetches and prints the translation of ⟨*key*⟩ for the currently active language (as for example set by babel). This command ensures that the fetched translation is set lowercase. Issues a warning if no translation is available at the cost of expandability.

`\SaveTranslationFor{⟨cmd⟩}{⟨lang⟩}{⟨key⟩}`

Fetches and saves the translation of ⟨*key*⟩ for the language ⟨*lang*⟩ in the macro ⟨*cmd*⟩.

`\SaveTranslation{⟨cmd⟩}{⟨key⟩}`

Fetches and saves the translation of ⟨*key*⟩ for the currently active language (as for example set by babel) in the macro ⟨*cmd*⟩.

`\LoadDictionary{`⟨*name*⟩`}`
Loads a file named ⟨*name*⟩`-`⟨*lang*⟩`.trsl` where ⟨*lang*⟩ corresponds to the lowercase name of the current language as defined with `\DeclareLanguage`. This file should contain the translations for the specified language.

`\LoadDictionaryFor{`⟨*lang*⟩`}{`⟨*name*⟩`}`
Loads a file named ⟨*name*⟩`-`⟨*lang*⟩`.trsl`.

`\NewDictTranslation{`⟨*key*⟩`}{`⟨*translation*⟩`}`
Introduced in version 0.10
This command is to be used in a dictionary file and picks up the language of that file. Issues an error if either the translation for the ⟨*key*⟩ or the dictionary entry for the ⟨*key*⟩ already exists.

`\RenewDictTranslation{`⟨*key*⟩`}{`⟨*translation*⟩`}`
Introduced in version 0.10
This command is to be used in a dictionary file and picks up the language of that file. Issues an error if either the translation for the ⟨*key*⟩ or the dictionary entry for the ⟨*key*⟩ doesn't exist.

`\ProvideDictTranslation{`⟨*key*⟩`}{`⟨*translation*⟩`}`
Introduced in version 0.10
This command is to be used in a dictionary file and picks up the language of that file. Only defines the translation and adds a corresponding dictionary entry if they don't exist yet. This command is used in the dictionaries that a part of TRANSLATIONS.

`\DeclareDictTranslation{`⟨*key*⟩`}{`⟨*translation*⟩`}`
This command is to be used in a dictionary file and picks up the language of that file, see section 3.5 for an example. Defines the translation and adds a dictionary entry regardless if they exist or not.

`\ProvideDictionaryFor{`⟨*lang*⟩`}{`⟨*name*⟩`}[`⟨*date*⟩`]`
Needs to be in a dictionary file. This command tells TRANSLATIONS that the file indeed is a dictionary and also sets the language for the dictionary which is used by `\DeclareDictTranslation`.

* `\PrintDictionaryFor{`⟨*lang*⟩`}{`⟨*name*⟩`}{`⟨*pre*⟩`}{`⟨*mid*⟩`}{`⟨*post*⟩`}`
Introduced in version 1.0
Prints all entries of dictionary ⟨*name*⟩ in language ⟨*lang*⟩ in the order the entries have been declared. For every entry the code

⟨*pre*⟩⟨*key*⟩⟨*mid*⟩⟨*translation*⟩⟨*post*⟩

is printed. The dictionary must have been loaded of course. There is probably only a very limited number of use cases for this command. (It was for example used to print table 1.)

* `\baselanguage{`⟨*lang*⟩`}`
Changed in version 1.2a
Returns the (internal) base name of the given language, language alias or language dialect. For a dialect this expands to the name of language it is a dialect of. For a base language (see section 4.1) this usually simply is the lowercase version of the name.

`\baselanguage{English}` ⇒ `english`
`\baselanguage{American}` ⇒ `english`

* `\ifcurrentlanguage{`⟨*lang*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`
Introduced in version 1.2
Places ⟨*true*⟩ in the input stream if the current language is ⟨*lang*⟩. Note: a dialect counts as a

language of it's own here. `\ifcurrentlanguage{English}` will for example be ⟨*false*⟩ if the current babel language is `american`.

* `\ifcurrentbaselanguage{`⟨*lang*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

Introduced in version 1.2

Places ⟨*true*⟩ in the input stream if the current language is ⟨*lang*⟩. Note: a dialect does not count as a language of it's own here. If the current babel language is `american` then `\ifcurrentbaselanguage{English}` will be ⟨*true*⟩.

3.3 A Small Example

This section demonstrates with two short examples how the macros are used. The first example covers the basics: declaring of translations and then retrieving and typesetting them.

```
% in the preamble:
% \DeclareTranslation{English}{Kueche}{kitchen}
% \DeclareTranslation{German}{Kueche}{K\"uche}
% \DeclareTranslation{Spanish}{Kueche}{cocina}
% \DeclareTranslation{French}{Kueche}{cuisine}

\GetTranslation{Kueche}
\SaveTranslation\kitchen{Kueche}
\SaveTranslationFor\cuisine{french}{Kueche}

\selectlanguage{ngerman}
\GetTranslation{Kueche} \kitchen\ \GetTranslationFor{spanish}{Kueche}
\cuisine

\IfTranslation{German}{Kueche}{true}{false} \par
\IfTranslation{Danish}{Kueche}{true}{false}
```

---

kitchen
Küche kitchen cocina cuisine
true
false

The next example demonstrates the use of dialects and how they fall back to the translation for the main language if no extra translation was declared:

```
% in the preamble:
% \DeclareTranslation{English}{farbe}{color}
```

```
% \DeclareTranslation{British}{farbe}{colour}

\GetTranslationFor{English}{farbe}
\GetTranslationFor{British}{farbe}
\GetTranslationFor{American}{farbe}
```

---

color colour color

3.4 Usage in Packages

3.4.1 Basic Structure

A typical usage in a package would look as follows:

```
\RequirePackage{translations}
\DeclareTranslationFallback{mypackage-title}{Nice Title}
\DeclareTranslation{English}{mypackage-title}{Nice Title}
\DeclareTranslation{French}{mypackage-title}{Beau Titre}
\DeclareTranslation{German}{mypackage-title}{Sch\"{o}ner Titel}
...
\newcommand*\mypackage@title{\GetTranslation{mypackage-title}}
```

That is, a package defines some unique key for an expression and at least defines a fallback translation. Additionally translations for as many languages as the author wants are defined. A user then may add `\DeclareTranslation{`⟨*language*⟩`}{`⟨*translation*⟩`}` if they find their translation missing.

3.4.2 The 'fallback' language

If a user has neither loaded babel nor polyglossia TRANSLATIONS will use English as language and translate to English if the translation was provided. If the user *has* loaded one of the language packages but has chosen a language for which no translation is defined the language 'fallback' will be used, *i. e.*, the translation provided with `\DeclareTranslationFallback`. If no fallback translation is provided either, the translation will expand to the literal string.

The following three examples should make this concept clear:

```
\documentclass[margin=5mm]{standalone}
\usepackage{translations}
\DeclareTranslation{German}{foo-literal}{foo-german}
\begin{document}
```

```
\GetTranslation{foo-literal} % foo-literal
\end{document}
```

foo-literal

```
\documentclass[margin=5mm]{standalone}
\usepackage{translations}
\DeclareTranslationFallback{foo-literal}{foo}
\DeclareTranslation{German}{foo-literal}{foo-german}
\begin{document}
\GetTranslation{foo-literal} % foo
\end{document}
```

foo

```
\documentclass[margin=5mm]{standalone}
\usepackage[ngerman]{babel}
\usepackage{translations}
\DeclareTranslationFallback{foo-literal}{foo}
\DeclareTranslation{German}{foo-literal}{foo-german}
\begin{document}
\GetTranslation{foo-literal} % foo-german
\end{document}
```

foo-german

3.5 Dictionaries

3.5.1 Background

TRANSLATIONS provides the means to write dictionary files that can be loaded by packages or in a document. Dictionaries can be loaded for the currently active language with \LoadDictionary or for a specific language with \LoadDictionaryFor.

`\LoadDictionary{`⟨*name*⟩`}`
Loads a file named ⟨*name*⟩`-`⟨*lang*⟩`.trsl` where ⟨*lang*⟩ corresponds to the lowercase name of the current language or base language as defined with `\DeclareLanguage`. This file should contain the translations for the specified language. This means if ⟨*lang*⟩ is a dialect and a dictionary exists the dictionary is used. If it doesn't exist but there is a dictionary for the base language that one is used.

`\LoadDictionaryFor{`⟨*lang*⟩`}{`⟨*name*⟩`}`
Loads a file named ⟨*name*⟩`-`⟨*lang*⟩`.trsl` where ⟨*lang*⟩ corresponds to the lowercase name of the current language or base language as defined with `\DeclareLanguage`. This file should contain the translations for the specified language. This means if ⟨*lang*⟩ is a dialect and a dictionary exists the dictionary is used. If it doesn't exist but there is a dictionary for the base language that one is used.

`\LoadDictionaryForDialect{`⟨*lang*⟩`}{`⟨*name*⟩`}`
Introduced in version 1.3
Loads a file named ⟨*name*⟩`-`⟨*lang*⟩`.trsl` where ⟨*lang*⟩ corresponds to the lowercase name of the current language as defined with `\DeclareLanguage`. This file should contain the translations for the specified language. This command *does not look* for a base language dictionary.

A package could provide dictionary files for its language dependent settings and include the needed one at begin document. The basics for creating a dictionary file are explained in section 3.5.2.

TRANSLATIONS already provides a few basic dictionary files. If the main document language fits to one of the provided files the corresponding basic dictionary is loaded at begin document by TRANSLATIONS, see section 3.5.3 for more on this.

3.5.2 Own Dictionaries

A typical dictionary file should look as follows:

```
% this is file housing-german.trsl
\ProvideDictionaryFor{German}{housing}[<version info>]
\ProvideDictTranslation{kitchen (housing)}{K\"uche}
\ProvideDictTranslation{bathroom (housing)}{Bad}
\ProvideDictTranslation{living room (housing)}{Wohnzimmer}
\ProvideDictTranslation{bedroom (housing)}{Schlafzimmer}
...
\endinput
```

The usage is similar to the one in a package: unique keys are given translations, this time for the language the dictionary file is declared for only. Translations can be declared by one of the following commands:

`\NewDictTranslation{`⟨*key*⟩`}{`⟨*translation*⟩`}`

Introduced in version 0.10

This command is to be used in a dictionary file and picks up the language of that file. Issues an error if either the translation for the ⟨*key*⟩ or the dictionary entry for the ⟨*key*⟩ already exists.

`\RenewDictTranslation{`⟨*key*⟩`}{`⟨*translation*⟩`}`

Introduced in version 0.10

This command is to be used in a dictionary file and picks up the language of that file. Issues an error if either the translation for the ⟨*key*⟩ or the dictionary entry for the ⟨*key*⟩ doesn't exist.

`\ProvideDictTranslation{`⟨*key*⟩`}{`⟨*translation*⟩`}`

Introduced in version 0.10

This command is to be used in a dictionary file and picks up the language of that file. Only defines the translation and adds a corresponding dictionary entry if they don't exist yet. This command is used in the dictionaries that a part of TRANSLATIONS.

`\DeclareDictTranslation{`⟨*key*⟩`}{`⟨*translation*⟩`}`

This command is to be used in a dictionary file and picks up the language of that file, see section 3.5 for an example. Defines the translation and adds a dictionary entry regardless if they exist or not.

Every dictionary file *must* contain the declaration `\ProvideDictionaryFor`:

`\ProvideDictionaryFor{`⟨*lang*⟩`}{`⟨*name*⟩`}[`⟨*date*⟩`]`

Needs to be in a dictionary file. This command tells TRANSLATIONS that the file indeed is a dictionary and also sets the language for the dictionary which is used by `\NewDictTranslation` or similar commands.

3.5.3 TRANSLATIONS' Basic Dictionaries

TRANSLATIONS already provides a basic dictionary for the languages

- Catalan,
- English,
- Dutch,
- French,
- German, and
- Spanish.

The corresponding dictionary[1] is loaded automatically if the document language is one of these languages.

If you'd like to contribute and add the basic dictionary in your language this is more than welcome and highly appreciated! The easiest way to do this would be to copy one of the existing files `translations-basic-dictionary-`⟨*lang*⟩`.trsl` and modify the file accordingly. You can then send me the file via email and I'll add it to TRANSLATIONS.

Table 1 lists all words provided by the basic dictionary for German.

1. Or dictionaries if more than one of these languages are loaded in a document. This works since v0.18.

Table 1: All entries of translations' basic dictionary in German.

key	translation

4 Defined Languages

4.1 Base Languages

Quite a number of languages already are defined, either directly or via an alias. So, before you define a language you should take a look at the tables below if the language doesn't already exist. Table 2 lists all base languages, "fallback" being a dummy language used for fallback translations. Tables 2, 3 and 4 list *all* language names known to translations. However, they're not sorted alphabetically but listed in the order they have been defined. I tried to make the definitions in an alphabetical order but sometimes rather grouped related language names together.

If you miss a language or recognize a language that has falsely been declared as an alias but should rather be a dialect or base language itself (or any variation of this theme) please let me know, preferably with a short explanation what's wrong and why.

Table 2: Base languages defined by translations, from left to right in the order of definition.

fallback	afrikaans	albanian	amharic	arabic
armenian	asturian	azerbaijani	basque	bengali
breton	bulgarian	catalan	coptic	czech
danish	dutch	english	esperanto	estonian
ethiop	farsi	finnish	french	friulan
gaelic	galician	german	greek	hebrew
hindustani	hungarian	icelandic	interlingua	italian
japanese	kannada	korean	ladin	lao
latin	latvian	lithuanian	macedonian	malay
malayalam	maldivian	marathi	mongolian	norwegian
occitan	piedmontese	pinyin	polish	portuges
romanian	romansh	russian	samin	sanskrit
serbocroatian	slovak	slovenian	sorbian	spanglish
spanish	swedish	tamil	telugu	thai
tibetan	turkish	turkmen	ukrainian	vietnamese
welsh				

4.2 Language Dialects

translations also defines a few dialects of thebase languages. They are listed in table 3. The decision what is a dialect and what is an alias is not always clear. I am no linguist so I looked up information available on the internet. A language that was described as "standardized register"

was always defined as a dialect. For some other languages it seemed to make sense, such as British or Austrian. The decisions are open for debate.

TABLE 3: All dialects defined by TRANSLATIONS, from left to right in the order of definition.

dialect	language	dialect	language
british	english	australian	english
american	english	acadian	french
canadien	french	canadian	english
newzealand	english	irish	gaelic
scottish	gaelic	austrian	german
hindi	hindustani	urdu	hindustani
indonesian	malay	brazil	portuges
serbian	serbocroatian	croatian	serbocroatian
lowersorbian	sorbian	uppersorbian	sorbian
swissgerman	german	swissfrench	french
swissitalian	italian	swissromansh	romansh

4.3 Language Aliases

To most of the base languages and dialects at least one alias exists, the uppercase variant. This is due to the fact that it is common to write language names uppercased. For a number of languages aliases were defined in order to match babel's or polyglossia's names for the languages. Others are defined because there apparently exist more than one name for the same language. The decisions are not consistent. For example it could be argued that "deutsch" is an alias of "German". I am open to suggestions and improvements. All defined aliases are listed in table 4.

TABLE 4: All language aliases defined by TRANSLATIONS, from left to right in the order of definition.

alias	language	alias	language
Fallback	fallback	Afrikaans	afrikaans
Albanian	albanian	Amharic	amharic
Arabic	arabic	Armenian	armenian
Asturian	asturian	astur-leonese	asturian
Astur-Leonese	astur-leonese	asturian-leonese	asturian
Asturian-Leonese	asturian-leonese	Azerbaijani	azerbaijani
Basque	basque	Bengali	bengali
Breton	breton	Bulgarian	bulgarian
Catalan	catalan	Coptic	coptic
coptic egyptian	coptic	Coptic Egyptian	coptic egyptian
Czech	czech	Danish	danish

continues

alias	language	alias	language
Dutch	dutch	Farsi	farsi
Finnish	finnish	francais	french
Francais	francais	Canadien	canadien
French	french	Acadian	acadian
frenchle	french	American	american
Australian	australian	British	british
Canadian	canadian	English	english
UKenglish	british	USenglish	american
Newzealand	newzealand	Ethiop	ethiop
Esperanto	esperanto	Estonian	estonian
Friulan	friulan	Gaelic	gaelic
Irish	irish	irish gaelic	irish
Irish Gaelic	irish	Scottish	scottish
scottish gaelic	scottish	Scottish Gaelic	scottish
Galician	galician	German	german
germanb	german	ngerman	german
Austrian	austrian	naustrian	austrian
Greek	greek	polutonikogreek	greek
ibygreek	greek	bgreek	greek
Hebrew	hebrew	Hindustani	hindustani
hindi-urdu	hindustani	Hindi-Urdu	hindi-urdu
Hindi	hindi	Urdu	urdu
Hungarian	hungarian	magyar	hungarian
Magyar	magyar	Icelandic	icelandic
Interlingua	interlingua	Italian	italian
Japanese	japanese	Kannada	kannada
Korean	korean	Ladin	ladin
Lao	lao	laotian	lao
Laotian	laotian	Latin	latin
Latvian	latvian	lettish	latvian
Lettish	lettish	Lithuanian	lithuanian
Macedonian	macedonian	Malay	malay
bahasa malaysia	malay	Bahasa Malaysia	bahasa malaysia
bahasa melayu	bahasa malaysia	Bahasa Melayu	bahasa melayu
bahasa	bahasa melayu	Bahasa	bahasa
bahasai	bahasa	Bahasai	bahasai
bahasam	bahasa	Bahasam	bahasam
Indonesian	indonesian	indon	indonesian
Malayalam	malayalam	Maldivian	maldivian
divehi	maldivian	Divehi	divehi
Marathi	marathi	Mongolian	mongolian

continues

alias	language	alias	language
norsk	norwegian	Norsk	norsk
Norwegian	norwegian	nynorsk	norwegian
Nynorsk	nynorsk	Occitan	occitan
lenga d'oc	occitan	langue d'oc	occitan
Piedmontese	piedmontese	piemontese	piedmontese
Piemontese	piemontese	piemonteis	piedmontese
Piemonteis	piemonteis	Pinyin	pinyin
Polish	polish	Brazil	brazil
brazilian	brazil	Brazilian	brazilian
Portuges	portuges	portuguese	portuges
Portuguese	portuguese	Romanian	romanian
Romansh	romansh	Romansch	romansh
Rumantsh	romansh	Rumantsch	romansh
Romanche	romansh	Russian	russian
Samin	samin	north sami	samin
North Sami	north sami	northern sami	north sami
Northern Sami	northern sami	Sanskrit	sanskrit
Serbocroatian	serbocroatian	serbo-croatian	serbocroatian
Serbo-Croatian	serbocroatian	Serbian	serbian
serbianc	serbian	Croatian	croatian
Slovak	slovak	Slovenian	slovenian
slovene	slovenian	Slovene	slovene
Sorbian	sorbian	Lowersorbian	lowersorbian
Uppersorbian	uppersorbian	lsorbian	lowersorbian
usorbian	uppersorbian	lower sorbian	lowersorbian
upper sorbian	uppersorbian	lower Sorbian	lowersorbian
upper Sorbian	uppersorbian	Lower Sorbian	lowersorbian
Upper Sorbian	uppersorbian	Spanglish	spanglish
Spanish	spanish	Swedish	swedish
Swissgerman	swissgerman	swiss german	swissgerman
Swiss German	swissgerman	Swissfrench	swissfrench
swiss french	swissfrench	Swiss French	swissfrench
Swissitalian	swissitalian	swiss italian	swissitalian
Swiss Italian	swissitalian	Swissromansh	swissromansh
swiss romansh	swissromansh	Swiss Romansh	swissromansh
swiss	swissgerman	Swiss	swiss
Tamil	tamil	Telugu	telugu
Thai	thai	thaicjk	thai
Thaicjk	thaicjk	Tibetan	tibetan
Turkish	turkish	Turkmen	turkmen
Ukrainian	ukrainian	Vietnamese	vietnamese

continues

alias	language	alias	language
Welsh	welsh		

These languages *should* cover all languages which are currently covered by babel and polyglossia but very likely this is not the case. Should you miss a language please send me an email so I can add it to TRANSLATIONS.

5 References

[Bra19] Johannes BRAAMS. *babel*. Version 3.33. July 19, 2019.
URL: http://mirror.ctan.org/macros/latex/required/babel/.

[Cha19] François CHARETTE. *polyglossia*. Version 1.44. Apr. 4, 2019.
URL: http://mirror.ctan.org/macros/latex/contrib/polyglossia/.

[Koh19] Markus KOHM. *KOMA-Script*. Version 3.25. Jan. 14, 2019.
URL: http://mirror.ctan.org/macros/latex/contrib/koma-script/.

[Leh19] Philipp LEHMAN. *etoolbox*. Version 2.5h. Sept. 21, 2019.
URL: http://mirror.ctan.org/macros/latex/contrib/etoolbox/.

[TW19] Till TANTAU and Joseph WRIGHT. *translator*. Version 1.12a. May 31, 2019.
URL: http://mirror.ctan.org/macros/contrib/beamer/base/translator/.