

TRANSLATIONS

V1.2C 2015/08/29

Internationalization of L^AT_EX 2_ε Packages

Clemens NIEDERBERGER

<http://www.mychemistry.eu/forums/forum/translations/>

contact@mychemistry.eu

Table of Contents

1	Motivation	1	3.5.2	Own Dictionaries . . .	9
2	License and Requirements	2	3.5.3	TRANSLATIONS ' Basic Dictionaries	10
3	Usage	2	4	Defined Languages	13
3.1	Background	2	4.1	Base Languages	13
3.2	Available Commands	3	4.2	Language Dialects	14
3.3	A Small Example	6	4.3	Language Aliases	14
3.4	Usage in Packages	7	5	Implementation	17
3.4.1	Basic Structure	7	Bibliography		38
3.4.2	The 'fallback' language	7	Index		38
3.5	Dictionaries	9			
3.5.1	Background	9			

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 [Tan10] (part of the beamer bundle). Internationalization is also possible with babel [Bra13] 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 [Cha13]) 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 L^AT_EX 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 `cnltx-base` from the `cnltx` bundle [Nie14] and `scrfile` (part of the KOMA-Script bundle [Koh15]).

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 13), language dialects (see table 3 on page 14), and language aliases (see table 4 on page 15). 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.

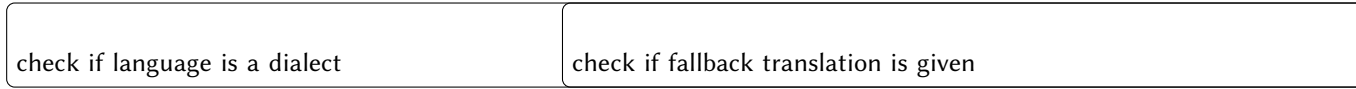


FIGURE 1: Schematic representation of **TRANSLATIONS**' translating mechanism. Notes: ^a except for a possible fallback translation. ^b *i. e.*, the base language of the dialect.

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.

\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* 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⟩` exists. 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⟩` already exists it won't be overwritten and no error will be raised. 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⟩}`

Defines a fallback translation for key `⟨key⟩` that is used in case no translation of `⟨key⟩` for the currently active language has been provided. No error will be raised if a fallback for `⟨key⟩` already exists. This command can only be used in the preamble.

* `\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{ $\langle key \rangle$ }

Introduced in
version 1.1

Fetches and prints the translation of $\langle key \rangle$ 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{ $\langle cmd \rangle$ }{ $\langle lang \rangle$ }{ $\langle key \rangle$ }

Fetches and saves the translation of $\langle key \rangle$ for the language $\langle lang \rangle$ in the macro $\langle cmd \rangle$.

\SaveTranslation{ $\langle cmd \rangle$ }{ $\langle key \rangle$ }

Fetches and saves the translation of $\langle key \rangle$ for the currently active language (as for example set by babel) in the macro $\langle cmd \rangle$.

\LoadDictionary{ $\langle name \rangle$ }

Loads a file named $\langle name \rangle - \langle lang \rangle . trsl$ where $\langle lang \rangle$ corresponds to the lowercase name of the current language as defined with **\DeclareLanguage**. This file should contain the translations for the specified language.

\LoadDictionaryFor{ $\langle lang \rangle$ }{ $\langle name \rangle$ }

Loads a file named $\langle name \rangle - \langle lang \rangle . trsl$.

\NewDictTranslation{ $\langle key \rangle$ }{ $\langle translation \rangle$ }

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 $\langle key \rangle$ or the dictionary entry for the $\langle key \rangle$ already exists.

\RenewDictTranslation{ $\langle key \rangle$ }{ $\langle translation \rangle$ }

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 $\langle key \rangle$ or the dictionary entry for the $\langle key \rangle$ doesn't exist.

\ProvideDictTranslation{ $\langle key \rangle$ }{ $\langle translation \rangle$ }

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{ $\langle key \rangle$ }{ $\langle translation \rangle$ }

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{ $\langle lang \rangle$ }{ $\langle name \rangle$ }[$\langle date \rangle$]

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**{ $\langle lang \rangle$ }{ $\langle name \rangle$ }{ $\langle pre \rangle$ }{ $\langle mid \rangle$ }{ $\langle post \rangle$ }

Introduced in
version 1.0

Prints all entries of dictionary $\langle name \rangle$ in language $\langle lang \rangle$ in the order the entries have been declared. For every entry the code

$\langle pre \rangle \langle key \rangle \langle mid \rangle \langle translation \rangle \langle post \rangle$

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{ $\langle lang \rangle$ }`

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{ $\langle lang \rangle$ }{ $\langle true \rangle$ }{ $\langle false \rangle$ }`

Introduced in
version 1.2

Places $\langle true \rangle$ in the input stream if the current language is $\langle lang \rangle$. Note: a dialect counts as a language of it's own here. `\ifcurrentlanguage{English}` will for example be $\langle false \rangle$ if the current babel language is american.

* `\ifcurrentbaselanguage{ $\langle lang \rangle$ }{ $\langle true \rangle$ }{ $\langle false \rangle$ }`

Introduced in
version 1.2

Places $\langle true \rangle$ in the input stream if the current language is $\langle lang \rangle$. 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 $\langle true \rangle$.

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.

```

1 % in the preamble:
2 % \DeclareTranslation{English}{Kueche}{kitchen}
3 % \DeclareTranslation{German}{Kueche}{K\"uche}
4 % \DeclareTranslation{Spanish}{Kueche}{cocina}
5 % \DeclareTranslation{French}{Kueche}{cuisine}
6
7 \GetTranslation{Kueche}
8 \SaveTranslation\kitchen{Kueche}
9 \SaveTranslationFor\cuisine{french}{Kueche}
10
11 \selectlanguage{ngerman}
12 \GetTranslation{Kueche} \kitchen\ \GetTranslationFor{spanish}{Kueche}
13 \cuisine

```

kitchen
 Küche kitchen cocina cuisine

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:

```

1 % in the preamble:
2 % \DeclareTranslation{English}{farbe}{color}
3 % \DeclareTranslation{British}{farbe}{colour}
4
5 \GetTranslationFor{English}{farbe}
6 \GetTranslationFor{British}{farbe}
7 \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:

```

1 \RequirePackage{translations}
2 \DeclareTranslationFallback{mypackage-title}{Nice Title}
3 \DeclareTranslation{English}{mypackage-title}{Nice Title}
4 \DeclareTranslation{French}{mypackage-title}{Beau Titre}
5 \DeclareTranslation{German}{mypackage-title}{Sch\{"o}ner Titel}
6 ...
7 \def\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:

```

1 \documentclass[margin=5mm]{standalone}
2 \usepackage{translations}
3 \DeclareTranslation{German}{foo-literal}{bar}
4 \begin{document}
5 \GetTranslation{foo-literal} % foo-literal
6 \end{document}

```

foo-literal

```

1 \documentclass[margin=5mm]{standalone}
2 \usepackage{translations}
3 \DeclareTranslationFallback{foo-literal}{foo}
4 \DeclareTranslation{German}{foo-literal}{bar}
5 \begin{document}
6 \GetTranslation{foo-literal} % foo
7 \end{document}

```

foo

```

1 \documentclass[margin=5mm]{standalone}
2 \usepackage[ngerman]{babel}
3 \usepackage{translations}
4 \DeclareTranslation{German}{foo-literal}{bar}
5 \begin{document}
6 \GetTranslation{foo-literal} % bar
7 \end{document}

```

bar

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 as defined with `\DeclareLanguage`. This file should contain the translations for the specified language.

`\LoadDictionaryFor{⟨lang⟩}{⟨name⟩}`

Loads a file named `⟨name⟩-⟨lang⟩.trsl`.

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:

```

1 % this is file housing-german.trsl
2 \ProvideDictionaryFor{German}{housing}[<version info>]
3 \ProvideDictTranslation{kitchen (housing)}{K\"uche}
4 \ProvideDictTranslation{bathroom (housing)}{Bad}
5 \ProvideDictTranslation{living room (housing)}{Wohnzimmer}
6 \ProvideDictTranslation{bedroom (housing)}{Schlafzimmer}
7 ...
8 \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⟩}`

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.

Introduced in
version 0.10

Introduced in version 0.10 `\RenewDictTranslation{⟨key⟩}{⟨translation⟩}`
 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.

Introduced in version 0.10 `\ProvideDictTranslation{⟨key⟩}{⟨translation⟩}`
 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 English, French, German and Spanish. This dictionary is loaded automatically if the document language is one of these four. 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.

TABLE 1: All entries of **TRANSLATIONS**' basic dictionary in German.

key	translation
Abstract	Zusammenfassung
Addresses	Adressen
addresses	Adressen
Address	Adresse
address	Adresse
and	und
Appendix	Anhang
Authors	Autoren
authors	Autoren
Author	Autor

continues

key	translation
author	Autor
Bibliography	Literaturverzeichnis
cc	Verteiler
Chapters	Kapitel
chapters	Kapitel
Chapter	Kapitel
chapter	Kapitel
Conclusion	Zusammenfassung
conclusion	Zusammenfassung
Contents	Inhaltsverzeichnis
Continuation	Fortsetzung
continuation	Fortsetzung
cont	Forts
encl (plural)	Anlagen
encl (singular)	Anlage
encl	Anlage(n)
Figures	Abbildungen
figures	Abbildungen
Figure	Abbildung
figure	Abbildung
From	Von
from	von
Glossary	Glossar
Index	Index
Introduction	Einleitung
introduction	Einleitung
List of Figures and Tables	Abbildungs- und Tabellenverzeichnis
List of Figures	Abbildungsverzeichnis
List of Tables	Tabellenverzeichnis
or	oder
Outline	Gliederung
Overview	Übersicht
Pages	Seiten
pages	Seiten
Page	Seite
page	Seite
Paragraphs	Absätze
paragraphs	Absätze
Paragraph	Absatz
paragraph	Absatz
Parts	Teile

continues

key	translation
parts	Teile
Part	Teil
part	Teil
Preface	Vorwort
Proofs	Beweise
proofs	Beweise
Proof	Beweis
proof	Beweis
References	Literatur
Related work	Verwandte Arbeiten
Related Work	Verwandte Arbeiten
Sections	Abschnitte
sections	Abschnitte
Section	Abschnitt
section	Abschnitt
See also	Siehe auch
see also	siehe auch
See	Siehe
see	siehe
Sketch of Proofs	Beweisskizzen
Sketch of proofs	Beweisskizzen
Sketch of Proof	Beweisskizze
Sketch of proof	Beweisskizze
Subsections	Unterabschnitte
subsections	Unterabschnitte
Subsection	Unterabschnitt
subsection	Unterabschnitt
Summary	Zusammenfassung
Tables	Tabellen
tables	Tabellen
Table	Tabelle
table	Tabelle
To	An
to	an
Monday	Montag
Tuesday	Dienstag
Wednesday	Mittwoch
Thursday	Donnerstag
Friday	Freitag
Saturday	Samstag
Sunday	Sonntag

continues

key	translation
January	Januar
February	Februar
March	März
April	April
May	Mai
June	Juni
July	Juli
August	August
September	September
October	Oktober
November	November
December	Dezember

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	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	malay	malayalam	maldivian
marathi	mongolian	norwegian	occitan	piedmontese
pinyin	polish	portuges	romanian	romansh

continues

russian	samin	sanskrit	serbocroatian	slovak
slovenian	sorbian	spanglish	spanish	swedish
tamil	telugu	thai	tibetan	turkish
turkmen	ukrainian	vietnamese	welsh	canadien
acadian	american	australian	british	canadian
newzealand	irish	scottish	austrian	hindi
urdu	indonesian	brazil	serbian	croatian
lowersorbian	uppersorbian	swissgerman		

4.2 Language Dialects

TRANSLATIONS also defines a few dialects of the base 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
canadien	french	acadian	french
american	english	australian	english
british	english	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		

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.

4 Defined Languages

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	Basque	basque
Bengali	bengali	Breton	breton
Bulgarian	bulgarian	Catalan	catalan
Coptic	coptic	coptic egyptian	coptic
Coptic Egyptian	coptic egyptian	Czech	czech
Danish	danish	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

continues

4 Defined Languages

alias	language	alias	language
Latin	latin	Latvian	latvian
lettish	latvian	Lettish	lettish
Lithuanian	lithuanian	Malay	malay
Indonesian	indonesian	indon	indonesian
bahasa meyalu	malay	Bahasa Meyalu	bahasa meyalu
bahasa	bahasa meyalu	Bahasa	bahasa
bahasai	bahasa	bahasam	bahasa
Malayalam	malayalam	Maldivian	maldivian
divehi	maldivian	Divehi	divehi
Marathi	marathi	Mongolian	mongolian
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
Sorbian	sorbian	Lowersorbian	lowersorbian
Uppersorbian	uppersorbian	lsorbian	lowersorbian
usorbian	uppersorbian	lower sorbian	lowersorbian
upper sorbian	uppersorbian	Lower Sorbian	lowersorbian
Upper Sorbian	uppersorbian	Spanglish	spanglish
Spanish	spanish	Swedish	swedish
swiss	swissgerman	Swiss	swiss
Swissgerman	swissgerman	swiss german	swissgerman
Swiss German	swiss german	Tamil	tamil
Telugu	telugu	Thai	thai

continues

alias	language	alias	language
thaicjk	thai	Thaicjk	thaicjk
Tibetan	tibetan	Turkish	turkish
Turkmen	turkmen	Ukrainian	ukrainian
Vietnamese	vietnamese	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 Implementation

```

1 % -----
2 % the TRANSLATIONS package
3 %
4 %   internationalization of LaTeX2e packages
5 %
6 % -----
7 % Clemens Niederberger
8 % Web:   http://www.mychemistry.eu/forums/forum/translations
9 % E-Mail: contact@mychemistry.eu
10 % -----
11 % Copyright 2012-2015 Clemens Niederberger
12 %
13 % This work may be distributed and/or modified under the
14 % conditions of the LaTeX Project Public License, either version 1.3
15 % of this license or (at your option) any later version.
16 % The latest version of this license is in
17 %   http://www.latex-project.org/lppl.txt
18 % and version 1.3 or later is part of all distributions of LaTeX
19 % version 2005/12/01 or later.
20 %
21 % This work has the LPPL maintenance status `maintained'.
22 %
23 % The Current Maintainer of this work is Clemens Niederberger.
24 % -----
25 % If you have any ideas, questions, suggestions or bugs to report, please
26 % feel free to contact me.
27 % -----
28 \def\@trnslt@date{2015/08/29}
29 \def\@trnslt@version{v1.2c}
30 \def\@trnslt@info{internationalization of LaTeX2e packages}
31
32 \ProvidesPackage{translations}[%
33   \@trnslt@date\space
34   \@trnslt@version\space

```

5 Implementation

```

35 \@trnslt@info\space (CN)]
36 \RequirePackage{cnltx-base,scrfile}
37
38 % -----
39 % message handling
40 % generic help message:
41 \def\@trnslt@error@message{%
42   For details have a look at the `translations' manual.%
43 }
44
45 % create message macros:
46 \cnltx@create@generic@message{@trnslt}{translations}{Error}{\@trnslt@error@message}
47 \cnltx@create@generic@message{@trnslt}{translations}{Warning}{}
48 \cnltx@create@generic@message{@trnslt}{translations}{WarningNoLine}{}
49 \cnltx@create@generic@message{@trnslt}{translations}{Info}{}
50
51 % specific errors:
52 \newrobustcmd*\@trnslt@err@unknown@lang[1]{\@trnslt@error{Unknown language `#1'}}
53 \newrobustcmd*\@trnslt@warn@unknown@lang[1]{\@trnslt@warning{Unknown language `#1'}}
54 \newrobustcmd*\@trnslt@err@already@defined[2]{%
55   \@trnslt@error{The #2 translation for `#1' is already defined.}%
56 }
57 \newrobustcmd*\@trnslt@err@not@defined[2]{%
58   \@trnslt@error{%
59     The \@trnslt@language{#2} translation for `#1' is not defined yet.%
60   }%
61 }
62 \newrobustcmd*\@trnslt@err@dict@already@defined[2]{%
63   \@trnslt@error{The #2 dictionary entry for `#1' is already defined.}%
64 }
65 \newrobustcmd*\@trnslt@err@dict@not@defined[2]{%
66   \@trnslt@error{%
67     The \@trnslt@language{#2} dictionary entry for `#1' is not defined yet.%
68   }%
69 }
70
71 % -----
72 % check if babel or polyglossia is used
73 \AtEndPreamble{
74   \@ifpackageloaded{babel}{}{
75     \@ifpackageloaded{polyglossia}{}
76     {\@trnslt@info{No language package found. I am going to use `english'
77       as default language.}}
78   }
79   % define \language if not defined yet:
80   \providecommand*\language{english}%
81   \def\@trnslt@current@language{\language}%
82   % define \bbl@afterfi if not defined yet:
83   \ifdef\bbl@afterfi{}

```

5 Implementation

```

84   {\long\def\bbl@afterfi#1\fi{\fi#1}}
85 }
86
87 % -----
88 % book keeping: the following macros will be used as `etoolbox' lists that
89 % keep record of defined languages, dialects and aliases
90 \newcommand*\@trnslt@languages{}% all languages
91 \newcommand*\@trnslt@aliases@pair{}% all aliases and their base
92 \newcommand*\@trnslt@aliases@single{}% all aliases
93 \newcommand*\@trnslt@dialects@pair{}% all dialects and their base
94 \newcommand*\@trnslt@dialects@single{}% all dialects
95
96 % -----
97 % define \@trnslt@if@<name> conditionals that don't leave the checked macro as
98 % \relax behind and check for \@trnslt@<name>@#1. These conditionals should
99 % also be expandable in an \edef-like context. Thanks to e-TeX there's
100 % \ifcsname:
101 \newrobustcmd*\@trnslt@new@check[1]{%
102   \csdef{@trnslt@if@#1}##1{%
103     \ifcsname @trnslt@#1@##1\endcsname
104       \expandafter\@firstoftwo
105     \else
106       \expandafter\@secondoftwo
107     \fi
108   }%
109 }
110
111 % -----
112 % leaves lowercase full expansion of #1 in the input stream (except that it
113 % doesn't, because nothing is expanded... :p)
114 \newcommand\@trnslt@get@lowercase[1]{%
115   \lowercase\expandafter{\romannumeral-\`Q#1}%
116 }
117
118 % -----
119 % \DeclareLanguage
120 % #1: language
121 \newrobustcmd*\DeclareLanguage[1]{%
122   \@trnslt@declare@language{#1}}
123 \@onlypreamble\DeclareLanguage
124
125 \newrobustcmd*\@trnslt@declare@language[1]{%
126   \@trnslt@if@language{#1}
127   {}{%
128     \csdef{@trnslt@language@#1}{#1}%
129     \listadd\@trnslt@languages{#1}%
130   }%
131 }
132

```

5 Implementation

```

133 % get the name declared by \DeclareLanguage: the underlying name of a given
134 % language; this is not the base of a dialect!
135 \newcommand*\@trnslt@language[1]{\csuse{@trnslt@language@#1}}
136
137 % a user-level equivalent:
138 % \newcommand*\baselanguage[1]{\@trnslt@language{#1}}
139
140 % define \@trnslt@if@language{#1}{}{} that actually checks the existence of
141 % \@trnslt@language@#1:
142 \@trnslt@new@check{language}
143
144 % -----
145 % \DeclareLanguageDialect
146 % #1: dialect
147 % #2: language
148 \newrobustcmd*\DeclareLanguageDialect[2]{%
149   \@trnslt@declare@language@dialect{#1}{#2}}
150 \onlypreamble\DeclareLanguageDialect
151
152 \newrobustcmd*\@trnslt@declare@language@dialect[2]{%
153   \@trnslt@if@language{#2}
154   {}{%
155     \@trnslt@warn@unknown@lang{#2}%
156     \@trnslt@declare@language{#2}%
157   }%
158   \@trnslt@if@dialect{#1}
159   {% => ist schon als dialect definiert => irgendwelche weiteren checks?
160   }
161   {%
162     \@trnslt@if@alias{#2}
163     {%
164       \csdef{@trnslt@dialect@#1}{\@trnslt@alias{#2}}{#1}}%
165       \@trnslt@declare@language{#1}%
166       \listeadadd\@trnslt@dialects@single{#1}%
167       \listeadadd\@trnslt@dialects@pair{{#1}{\@trnslt@alias{#2}}}%
168     }
169     {%
170       \csdef{@trnslt@dialect@#1}{#2}{#1}}%
171       \@trnslt@declare@language{#1}%
172       \listeadadd\@trnslt@dialects@single{#1}%
173       \listeadadd\@trnslt@dialects@pair{{#1}{#2}}}%
174     }%
175   }%
176 }
177
178 % dialect equivalent of \@trnslt@language
179 \newcommand*\@trnslt@dialect[1]{\csuse{@trnslt@dialect@#1}}
180
181 % this macros fetches the base language for a given dialect, expandably:

```

5 Implementation

```

182 \newcommand*\@trnslt@dialect@of[1]{%
183   \@trnslt@if@dialect{#1}
184   {%
185     \expandafter\expandafter\expandafter
186     \@firstoftwo
187     \csname @trnslt@dialect@#1\endcsname
188   }{}%
189 }
190
191 % get the underlying name of a language or the name of the base language of a
192 % dialect:
193 \newcommand*\baselanguage[1]{%
194   \@trnslt@if@dialect{#1}
195   {\@trnslt@language{\@trnslt@dialect@of{#1}}}
196   {\@trnslt@language{#1}}%
197 }
198
199 % define \@trnslt@if@dialect{#1}{}{} that actually checks the existence of
200 % \@trnslt@dialect@#1:
201 \@trnslt@new@check{dialect}
202
203 % -----
204 % \DeclareLanguageAlias
205 % #1: alias
206 % #2: language
207 \newrobustcmd*\DeclareLanguageAlias[2]{%
208   \@trnslt@declare@languagealias{#1}{#2}}
209 \onlypreamble\DeclareLanguageAlias
210
211 \newrobustcmd*\@trnslt@declare@languagealias[2]{%
212   \@trnslt@if@language{#2}
213   {}{%
214     \@trnslt@warn@unknown@lang{#2}%
215     \@trnslt@declare@language{#2}%
216   }%
217   \csletcs{@trnslt@language@#1}{@trnslt@language@#2}%
218   \@trnslt@if@dialect{#2}
219   {\csletcs{@trnslt@dialect@#1}{@trnslt@dialect@#2}}
220   {}%
221   \ifinlist{#1}\@trnslt@aliases@single
222   {}{%
223     \csdef{@trnslt@alias@#1}{#2}%
224     \listadd\@trnslt@aliases@pair{{#1}{#2}}%
225     \listadd\@trnslt@aliases@single{#1}%
226   }%
227 }
228
229 % alias equivalent of \@trnslt@language
230 \newcommand*\@trnslt@alias[1]{\csuse{@trnslt@alias@#1}}

```

5 Implementation

```
231
232 % define \@trnslt@if@alias{#1}{}{} that actually checks the existence of
233 % \@trnslt@alias@#1:
234 \@trnslt@new@check{alias}
235
236 % -----
237 % check the current language, expandably:
238 % #1: language
239 % #2: true
240 % #3: false
241 \newcommand*\@trnslt@iflanguage[1]{%
242   \cnltx@braced@expanded@fully
243   \cnltx@braced@expanded@fully
244   \cnltx@firstofone
245   \cnltx@ifstrequal
246   {\@trnslt@language{#1}}
247   {\@trnslt@language{\language}}}%
248 }
249
250 % check the current /base/ language, expandably:
251 \newcommand*\@trnslt@ifbaselanguage[1]{%
252   \@trnslt@if@dialect{\language}
253   {%
254     \cnltx@braced@expanded@fully
255     \cnltx@braced@expanded@fully
256     \cnltx@firstofone
257     \cnltx@ifstrequal
258     {\@trnslt@language{\@trnslt@dialect@of{\language}}}
259     {\@trnslt@language{#1}}}%
260   }
261   {\@trnslt@iflanguage{#1}}}%
262 }
263
264 % user commands for the above:
265 \newcommand*\ifcurrentlanguage[1]{%
266   \@trnslt@iflanguage{#1}%
267 }
268 \newcommand*\ifcurrentbaselanguage[1]{%
269   \@trnslt@ifbaselanguage{#1}%
270 }
271
272 % -----
273 % Now that we can define languages we also want to be able to define
274 % translations:
275 % \DeclareTranslation, \NewTranslation, \RenewTranslation and
276 % \ProvideTranslation
277 % #1: language
278 % #2: word
279 % #3: replacement
```

5 Implementation

```

280 \newrobustcmd*\DeclareTranslation[3]{%
281   \@trnslt@declare@translation{#2}{#1}{#3}%
282 }
283 \@onlypreamble\DeclareTranslation
284
285 \newrobustcmd*\DeclareTranslationFallback[2]{%
286   \@trnslt@declare@translation{#1}{fallback}{#2}%
287 }
288 \@onlypreamble\DeclareTranslationFallback
289
290 \newrobustcmd*\NewTranslation[3]{%
291   \@trnslt@new@translation{#2}{#1}{#3}%
292 }
293 \@onlypreamble\NewTranslation
294
295 \newrobustcmd*\RenewTranslation[3]{%
296   \@trnslt@renew@translation{#2}{#1}{#3}%
297 }
298 \@onlypreamble\RenewTranslation
299
300 \newrobustcmd*\ProvideTranslation[3]{%
301   \@trnslt@provide@translation{#2}{#1}{#3}%
302 }
303 \@onlypreamble\ProvideTranslation
304
305 % #1: word
306 % #2: language
307 % #3: replacement
308 \newrobustcmd*\@trnslt@declare@translation[3]{%
309   \@trnslt@if@language{#2}
310   {%
311     \@trnslt@if@dialect{#2}
312     {%
313       \csdef{@trnslt@word@detokenize{#1}@@trnslt@dialect{#2}}{#3}%
314       \@trnslt@if@word\@trnslt@dialect@of{#1}{#2}
315       {}
316       {\csdef{@trnslt@word@detokenize{#1}@@trnslt@dialect@of{#2}}{#3}}%
317     }
318     {\csdef{@trnslt@word@detokenize{#1}@@trnslt@language{#2}}{#3}}%
319     % save the <word> as <word>:
320     \csdef{@trnslt@word@detokenize{#1}@literal}{#1}%
321   }
322   {\@trnslt@err@unknown@lang{#2}}%
323 }
324
325 \newrobustcmd*\@trnslt@new@translation[3]{%
326   \@trnslt@if@word\@trnslt@language{#1}{#2}
327   {\@trnslt@err@alreadydefined{#1}{#2}}
328   {\@trnslt@declare@translation{#1}{#2}{#3}}%

```

5 Implementation

```

329 }
330
331 \newrobustcmd*\@trnslt@renew@translation[3]{%
332   \@trnslt@if@word\@trnslt@language{#1}{#2}
333   {\@trnslt@declare@translation{#1}{#2}{#3}}
334   {\@trnslt@err@not@defined{#1}{#2}}}%
335 }
336
337 \newrobustcmd*\@trnslt@provide@translation[3]{%
338   \@trnslt@if@word\@trnslt@language{#1}{#2}
339   {}
340   {\@trnslt@declare@translation{#1}{#2}{#3}}}%
341 }
342
343 % -----
344 % let's go through some trouble to check if a translation exists:
345 \newcommand*\@trnslt@if@word[3]{%
346   \ifcsname @trnslt@word@detokenize{#2}@#1{#3}\endcsname
347   \expandafter\@firstoftwo
348   \else
349   \expandafter\@secondoftwo
350   \fi
351 }%
352
353 \newcommand*\@trnslt@if@translation[2]{%
354   \@trnslt@if@word\@trnslt@language{#1}{#2}
355   {\expandafter\@firstoftwo}
356   {%
357     \@trnslt@if@dialect{#2}
358     {%
359       \@trnslt@if@word\@trnslt@dialect{#1}{#2}
360       {\expandafter\@firstoftwo}
361       {%
362         \@trnslt@if@word\@trnslt@dialect@of{#1}{#2}
363         {\expandafter\@firstoftwo}
364         {\expandafter\@secondoftwo}}%
365       }
366     }
367     {\expandafter\@secondoftwo}%
368   }%
369 }
370
371 % -----
372 % We also need to be able to look up a translation:
373 % \GetTranslationFor and \GetTranslation
374 % these need to be expandable!
375 % #1: language
376 % #2: word
377 \newcommand*\GetTranslationFor[2]{%

```


5 Implementation

```

378 \@trnslt@checkandget@translation@for{#2}{#1}}
379
380 \newcommand*\GetTranslation[1]{%
381 \@trnslt@checkandget@translation@for{#1}{\@trnslt@current@language}}
382
383 % those will print the translation lowercase... they're also expandable; that
384 % is: nearly...
385 % \DeclareTranslation{German}{foo}{Bla}
386 % \edef\foo{\GetLCTranslationFor{German}{foo}}
387 % \show\foo
388 % > \foo=macro:
389 % ->\lowercase {Bla}.
390 \newcommand*\GetLCTranslationFor[2]{%
391 \@trnslt@get@lowercase{\@trnslt@checkandget@translation@for{#2}{#1}}%
392 }
393
394 \newcommand*\GetLCTranslation[1]{%
395 \@trnslt@get@lowercase{%
396 \@trnslt@checkandget@translation@for{#1}{\@trnslt@current@language}%
397 }%
398 }
399
400 % unexpandable version of the commands that raise a warning if no translation
401 % is available:
402 \newrobustcmd*\GetTranslationForWarn[2]{%
403 \@trnslt@getandwarn@translation@for{#2}{#1}%
404 }
405
406 \newrobustcmd*\GetTranslationWarn[1]{%
407 \@trnslt@getandwarn@translation@for{#1}{\@trnslt@current@language}%
408 }
409
410 \newrobustcmd*\GetLCTranslationForWarn[2]{%
411 \@trnslt@getandwarn@lctranslation@for{#2}{#1}%
412 }
413
414 \newrobustcmd*\GetLCTranslationWarn[1]{%
415 \@trnslt@getandwarn@lctranslation@for{#1}{\@trnslt@current@language}%
416 }
417
418 % #1: word #2: language
419 \newcommand*\@trnslt@get@translation@for[2]{%
420 \@trnslt@if@dialect{#2}
421 {%
422 \ifcsdef{\@trnslt@word@detokenize{#1}@@trnslt@dialect{#2}}
423 {\csuse{\@trnslt@word@detokenize{#1}@@trnslt@dialect{#2}}}
424 {\csuse{\@trnslt@word@detokenize{#1}@@trnslt@dialect@of{#2}}}%
425 }
426 {\csuse{\@trnslt@word@detokenize{#1}@@trnslt@language{#2}}}%

```

5 Implementation

```

427 }
428
429 \newcommand*\@trnslt@checkandget@translation@for[2]{%
430   \@trnslt@if@translation{#1}{#2}
431   {\@trnslt@get@translation@for{#1}{#2}}
432   {%
433     \@trnslt@if@translation{#1}{fallback}
434     {\csuse{@trnslt@word@\detokenize{#1}@fallback}}
435     {\csuse{@trnslt@word@\detokenize{#1}@literal}}}%
436   }%
437 }
438
439 % this is not expandable!
440 \newrobustcmd*\@trnslt@getandwarn@translation@for[2]{%
441   \@trnslt@if@translation{#1}{#2}
442   {\@trnslt@get@translation@for{#1}{#2}}
443   {%
444     \@trnslt@warning{Translation for `#1' in #2 unknown. You may try to use
445       \string\DeclareTranslation{#2}{#1}{ ... } in your preamble.}%
446     \@trnslt@if@translation{#1}{fallback}
447     {%
448       \@trnslt@info{Using fallback translation for `#1'}%
449       \csuse{@trnslt@word@\detokenize{#1}@fallback}
450     }
451     {\csuse{@trnslt@word@\detokenize{#1}@literal}}}%
452   }%
453 }
454
455 % lowercase version for translation with warnings:
456 \newrobustcmd*\@trnslt@getandwarn@lc@translation@for[2]{%
457   \@trnslt@if@translation{#1}{#2}
458   {\@trnslt@get@lowercase{\@trnslt@get@translation@for{#1}{#2}}}
459   {%
460     \@trnslt@warning{Translation for `#1' in #2 unknown. You may try to use
461       \string\DeclareTranslation{#2}{#1}{ ... } in your preamble.}%
462     \@trnslt@if@translation{#1}{fallback}
463     {%
464       \@trnslt@info{Using fallback translation for `#1'}%
465       \@trnslt@get@lowercase{\csuse{@trnslt@word@\detokenize{#1}@fallback}}%
466     }
467     {\@trnslt@get@lowercase{\csuse{@trnslt@word@\detokenize{#1}@literal}}}%
468   }%
469 }
470
471 % -----
472 % no idea if the following ones really are needed... let's define them
473 % anyway
474 % \SaveTranslationFor and \SaveTranslation
475 \newrobustcmd*\SaveTranslationFor[3]{%

```

5 Implementation

```
476 \@trnslt@save@translation@for{#1}{#3}{#2}%
477 }
478
479 \newrobustcmd*\SaveTranslation[2]{%
480 \@trnslt@save@translation@for{#1}{#2}{\@trnslt@current@language}%
481 }
482
483 \newrobustcmd*\@trnslt@save@translation@for[3]{%
484 \edef#1{%
485 \@trnslt@if@translation{#2}{#3}
486 {%
487 \unexpanded\expandafter\expandafter\expandafter
488 {\csname @trnslt@word@detokenize{#2}@@trnslt@language{#3}\endcsname}%
489 }
490 }%
491 }%
492 }
493
494 % -----
495 % The basics are set. What's missing? Right: dictionaries! Those will be files
496 % that contain translations for a specific language.
497 % -----
498 % load dictionaries and check for existing ones:
499 % \LoadDictionary and \LoadDictionaryFor
500 \newrobustcmd*\LoadDictionary[1]{%
501 \@trnslt@load@dictionary@for{#1}{\@trnslt@current@language}}
502 \onlypreamble\LoadDictionary
503
504 \newrobustcmd*\LoadDictionaryFor[2]{%
505 \@trnslt@load@dictionary@for{#2}{#1}}
506 \onlypreamble\LoadDictionaryFor
507
508 % #1: name
509 % #2: lang
510 \newrobustcmd*\@trnslt@load@dictionary@for[2]{%
511 \AtBeginDocument{%
512 \InputIfFileExists{#1-\@trnslt@language{#2}.trsl}
513 {\@trnslt@check@dictionary{#1}{#2}}
514 {%
515 \@trnslt@warning{dictionary file `#1-\@trnslt@language{#2}.trsl' not
516 found.}%
517 }%
518 }%
519 }
520
521 \newrobustcmd*\@trnslt@check@dictionary[2]{%
522 \AfterFile{#1-\@trnslt@language{#2}.trsl}
523 {%
524 \ifcsdef{@trnslt@dictionary@#1@trnslt@language{#2}}
```

5 Implementation

```

525     {\@trnslt@info{loading dictionary `#1' for `#2'.}}
526     {%
527         \@trnslt@warning{file `#1-\@trnslt@language{#2}.trsl' does not
528             appear to be a dictionary}%
529     }%
530 }%
531 }
532
533 \newcommand*\@trnslt@if@dictionary[2]{\IfFileExists{#1-#2.trsl}}
534
535 \newcommand*\@trnslt@load@dictionary@silent@for[2]{\InputIfFileExists{#1-#2.trsl
536     }{}}{}
537 % -----
538 % the contents of a dictionary; let's declare that is one.
539 \newrobustcmd*\ProvideDictionaryFor[2]{%
540     \@trnslt@provide@dictionary@for{#1}{#2}}
541 \@onlypreamble\ProvideDictionaryFor
542
543 \newrobustcmd*\@trnslt@provide@dictionary@for[2]{%
544     \def\@trnslt@dictionary@name{#2}%
545     \edef\@trnslt@dictionary@lang{\@trnslt@language{#1}}%
546     % this macro can be used to check if we have a dictionary and will also be
547     % used as a list for the dictionary entries:
548     \csdef{\@trnslt@dictionary@\@trnslt@dictionary@name \@trnslt@dictionary@lang}{}%
549     \@ifnextchar[
550         {\@trnslt@provide@dictionary@version}
551         {
552             \ProvidesFile
553                 {#2-\@trnslt@dictionary@lang.trsl}%
554                 [(\@trnslt@dictionary@lang\space translation file `#2')]
555         }%
556     }
557
558 \protected\def\@trnslt@provide@dictionary@version[#1]{%
559     \ProvidesFile
560         {\@trnslt@dictionary@name-\@trnslt@dictionary@lang.trsl}%
561         [(\@trnslt@dictionary@lang\space translation file \@trnslt@dictionary@name')
562         #1]}
563
564 % change this test (we can't use braces inside the item with \ifinlist):
565 \newcommand*\@trnslt@check@dictionary@entry[2]{%
566     \csltx@ifinlistcs
567         {{#1}{#2}}
568         {\@trnslt@dictionary@\@trnslt@dictionary@name \@trnslt@dictionary@lang}%
569     }
570
571 \newrobustcmd*\@trnslt@add@dictionary@entry[2]{%
572     \listcsadd

```

5 Implementation

```

572   {@trnslt@dictionary@\@trnslt@dictionary@name @\@trnslt@dictionary@lang}
573   {{#1}{#2}}}%
574 }
575
576 % -----
577 % now we should be able to add translations without repeatedly type the
578 % language ...
579 \newrobustcmd*\NewDictTranslation[2]{%
580   \@trnslt@check@dictionary@entry{#1}{#2}
581   {\@trnslt@err@dict@already@defined{#1}{#2}}}%
582   {%
583     \@trnslt@add@dictionary@entry{#1}{#2}
584     \@trnslt@new@translation{#1}{\@trnslt@dictionary@lang}{#2}%
585   }%
586 }
587 \@onlypreamble\NewDictTranslation
588
589 \newrobustcmd*\RenewDictTranslation[2]{%
590   \@trnslt@check@dictionary@entry{#1}{#2}
591   {\@trnslt@renew@translation{#1}{\@trnslt@dictionary@lang}{#2}}
592   {\@trnslt@err@dict@not@defined{#1}{#2}}}%
593 }
594 \@onlypreamble\RenewDictTranslation
595
596 \newrobustcmd*\DeclareDictTranslation[2]{%
597   \@trnslt@check@dictionary@entry{#1}{#2}
598   {}
599   {\@trnslt@add@dictionary@entry{#1}{#2}}}%
600   \@trnslt@declare@translation{#1}{\@trnslt@dictionary@lang}{#2}%
601 }
602 \@onlypreamble\DeclareDictTranslation
603
604 \newrobustcmd*\ProvideDictTranslation[2]{%
605   \@trnslt@check@dictionary@entry{#1}{#2}
606   {}
607   {%
608     \@trnslt@add@dictionary@entry{#1}{#2}%
609     \@trnslt@provide@translation{#1}{\@trnslt@dictionary@lang}{#2}%
610   }%
611 }
612 \@onlypreamble\ProvideDictTranslation
613
614 % -----
615 % last not least let's inspect which entries a dictionary contains of:
616 % \PrintDictionaryFor
617 % #1: lang
618 % #2: name
619 % #3: pre
620 % #4: mid

```

5 Implementation

```
621 % #5: post
622 \newcommand*\PrintDictionaryFor[5]{%
623   \@trnslt@print@dictionary@for{#1}{#2}{#3}{#4}{#5}%
624 }
625
626 % #1: lang
627 % #2: name
628 % #3: pre
629 % #4: mid
630 % #5: post
631 \newcommand*\@trnslt@print@dictionary@for[5]{%
632   \forlistcsloop
633     {\@trnslt@print@dictionary@entry{#3}{#4}{#5}}
634     {\@trnslt@dictionary@#2@\@trnslt@language{#1}}%
635 }
636
637 % #1: pre
638 % #2: mid
639 % #3: post
640 % #4: (key)(translation)
641 \newcommand*\@trnslt@print@dictionary@entry[4]{%
642   \@trnslt@print@dictionary@entry@aux{#1}{#2}{#3}#4%
643 }
644
645 % #1: pre
646 % #2: mid
647 % #3: post
648 % #4: key
649 % #5: translation
650 \newcommand*\@trnslt@print@dictionary@entry@aux[5]{#1#4#2#5#3}
651
652 % =====
653 % Now that the package is finished let's use the above commands to provide a
654 % basis for usage; we need all languages known to 'babel' and 'polyglossia'
655 % as well as aliases to allow different spellings/names; we also need dialects
656 % where it makes sense (e.g. British as a dialect of English)
657
658 % -----
659 % dummy language 'fallback' -- it is needed for the whole mechanism provided
660 % earlier (\DeclareFallbackTranslation...)
661 \DeclareLanguage{fallback}
662 \DeclareLanguageAlias{Fallback}{fallback}
663
664 % -----
665 % predefined languages
666 \DeclareLanguage{afrikaans}
667 \DeclareLanguage{albanian}
668 \DeclareLanguage{amharic}
669 \DeclareLanguage{arabic}
```

```

670 \DeclareLanguage{armenian}
671 \DeclareLanguage{asturian}
672 \DeclareLanguage{basque}
673 \DeclareLanguage{bengali}
674 \DeclareLanguage{breton}
675 \DeclareLanguage{bulgarian}
676 \DeclareLanguage{catalan}
677 \DeclareLanguage{coptic}
678 \DeclareLanguage{czech}
679 \DeclareLanguage{danish}
680 \DeclareLanguage{dutch}
681 \DeclareLanguage{english}
682 \DeclareLanguage{esperanto}
683 \DeclareLanguage{estonian}
684 \DeclareLanguage{ethiop}
685 \DeclareLanguage{farsi}
686 \DeclareLanguage{finnish}
687 \DeclareLanguage{french}
688 \DeclareLanguage{friulan}
689 \DeclareLanguage{gaelic}
690 \DeclareLanguage{galician}
691 \DeclareLanguage{german}
692 \DeclareLanguage{greek}
693 \DeclareLanguage{hebrew}
694 \DeclareLanguage{hindustani}
695 \DeclareLanguage{hungarian}
696 \DeclareLanguage{icelandic}
697 \DeclareLanguage{interlingua}
698 \DeclareLanguage{italian}
699 \DeclareLanguage{japanese}
700 \DeclareLanguage{kannada}
701 \DeclareLanguage{korean}
702 \DeclareLanguage{ladin}
703 \DeclareLanguage{lao}
704 \DeclareLanguage{latin}
705 \DeclareLanguage{latvian}
706 \DeclareLanguage{lithuanian}
707 \DeclareLanguage{malay}
708 \DeclareLanguage{malayalam}
709 \DeclareLanguage{maldivian}
710 \DeclareLanguage{marathi}
711 \DeclareLanguage{mongolian}
712 % polyglossia seems to support this one but it is unclear which language is
713 % actually meant by it:
714 % \DeclareLanguage{nko}
715 \DeclareLanguage{norwegian}
716 \DeclareLanguage{occitan}
717 \DeclareLanguage{piedmontese}
718 \DeclareLanguage{pinyin}

```

5 Implementation

```
719 \DeclareLanguage{polish}
720 \DeclareLanguage{portuges}
721 \DeclareLanguage{romanian}
722 \DeclareLanguage{romansh}
723 \DeclareLanguage{russian}
724 \DeclareLanguage{samin}
725 \DeclareLanguage{sanskrit}
726 \DeclareLanguage{serbocroatian}
727 \DeclareLanguage{slovak}
728 \DeclareLanguage{slovenian}
729 \DeclareLanguage{sorbian}
730 % not sure about this: isn't it either a Spanish or English dialect?
731 \DeclareLanguage{spanglish}
732 \DeclareLanguage{spanish}
733 \DeclareLanguage{swedish}
734 % polyglossia seems to support this one but it is unclear which language is
735 % actually meant by it:
736 % \DeclareLanguage{syriac}
737 \DeclareLanguage{tamil}
738 \DeclareLanguage{telugu}
739 \DeclareLanguage{thai}
740 \DeclareLanguage{tibetan}
741 \DeclareLanguage{turkish}
742 \DeclareLanguage{turkmen}
743 \DeclareLanguage{ukrainian}
744 \DeclareLanguage{vietnamese}
745 \DeclareLanguage{welsh}
746
747 % -----
748 % aliases and dialects:
749 \DeclareLanguageAlias {Afrikaans}{afrikaans}
750 \DeclareLanguageAlias {Albanian}{albanian}
751 \DeclareLanguageAlias {Amharic}{amharic}
752 \DeclareLanguageAlias {Arabic}{arabic}
753 \DeclareLanguageAlias {Armenian}{armenian}
754 \DeclareLanguageAlias {Asturian}{asturian}
755 \DeclareLanguageAlias {astur-leonese}{asturian}
756 \DeclareLanguageAlias {Astur-Leonese}{astur-leonese}
757 \DeclareLanguageAlias {asturian-leonese}{asturian}
758 \DeclareLanguageAlias {Asturian-Leonese}{asturian-leonese}
759 \DeclareLanguageAlias {Basque}{basque}
760 \DeclareLanguageAlias {Bengali}{bengali}
761 \DeclareLanguageAlias {Breton}{breton}
762 \DeclareLanguageAlias {Bulgarian}{bulgarian}
763 \DeclareLanguageAlias {Catalan}{catalan}
764 \DeclareLanguageAlias {Coptic}{coptic}
765 \DeclareLanguageAlias {coptic egyptian}{coptic}
766 \DeclareLanguageAlias {Coptic Egyptian}{coptic egyptian}
767 \DeclareLanguageAlias {Czech}{czech}
```


5 Implementation

```
768 \DeclareLanguageAlias {Danish}{danish}
769 \DeclareLanguageAlias {Dutch}{dutch}
770 \DeclareLanguageAlias {Farsi}{farsi}
771 \DeclareLanguageAlias {Finnish}{finnish}
772 \DeclareLanguageAlias {français}{french}
773 \DeclareLanguageAlias {Français}{français}
774 \DeclareLanguageDialect{canadien}{french}
775 \DeclareLanguageAlias {Canadien}{canadien}
776 \DeclareLanguageAlias {French}{french}
777 \DeclareLanguageDialect{acadian}{french}
778 \DeclareLanguageAlias {Acadian}{acadian}
779 \DeclareLanguageAlias {frenchle}{french}
780 \DeclareLanguageDialect{american}{english}
781 \DeclareLanguageAlias {American}{american}
782 \DeclareLanguageDialect{australian}{english}
783 \DeclareLanguageAlias {Australian}{australian}
784 \DeclareLanguageDialect{british}{english}
785 \DeclareLanguageAlias {British}{british}
786 \DeclareLanguageDialect{canadian}{english}
787 \DeclareLanguageAlias {Canadian}{canadian}
788 \DeclareLanguageAlias {English}{english}
789 \DeclareLanguageAlias {UKenglish}{british}
790 \DeclareLanguageAlias {USenglish}{american}
791 \DeclareLanguageDialect{newzealand}{english}
792 \DeclareLanguageAlias {Newzealand}{newzealand}
793 \DeclareLanguageAlias {Ethiop}{ethiop}
794 \DeclareLanguageAlias {Esperanto}{esperanto}
795 \DeclareLanguageAlias {Estonian}{estonian}
796 \DeclareLanguageAlias {Friulan}{friulan}
797 \DeclareLanguageAlias {Gaelic}{gaelic}
798 \DeclareLanguageDialect{irish}{gaelic}
799 \DeclareLanguageDialect{scottish}{gaelic}
800 \DeclareLanguageAlias {Irish}{irish}
801 \DeclareLanguageAlias {irish gaelic}{irish}
802 \DeclareLanguageAlias {Irish Gaelic}{irish}
803 \DeclareLanguageAlias {Scottish}{scottish}
804 \DeclareLanguageAlias {scottish gaelic}{scottish}
805 \DeclareLanguageAlias {Scottish Gaelic}{scottish}
806 \DeclareLanguageAlias {Galician}{galician}
807 \DeclareLanguageAlias {German}{german}
808 \DeclareLanguageAlias {germanb}{german}
809 \DeclareLanguageAlias {ngerman}{german}
810 \DeclareLanguageDialect{austrian}{german}
811 \DeclareLanguageAlias {Austrian}{austrian}
812 \DeclareLanguageAlias {naustrian}{austrian}
813 \DeclareLanguageAlias {Greek}{greek}
814 \DeclareLanguageAlias {polutonikogreek}{greek}
815 \DeclareLanguageAlias {ibygreek}{greek}
816 \DeclareLanguageAlias {bgreek}{greek}
```

5 Implementation

```
817 \DeclareLanguageAlias {Hebrew}{hebrew}
818 \DeclareLanguageAlias {Hindustani}{hindustani}
819 \DeclareLanguageAlias {hindi-urdu}{hindustani}
820 \DeclareLanguageAlias {Hindi-Urdu}{hindi-urdu}
821 \DeclareLanguageDialect{hindi}{hindustani}
822 \DeclareLanguageAlias {Hindi}{hindi}
823 \DeclareLanguageDialect{urdu}{hindustani}
824 \DeclareLanguageAlias {Urdu}{urdu}
825 \DeclareLanguageAlias {Hungarian}{hungarian}
826 \DeclareLanguageAlias {magyar}{hungarian}
827 \DeclareLanguageAlias {Magyar}{magyar}
828 \DeclareLanguageAlias {Icelandic}{icelandic}
829 \DeclareLanguageAlias {Interlingua}{interlingua}
830 \DeclareLanguageAlias {Italian}{italian}
831 \DeclareLanguageAlias {Japanese}{japanese}
832 \DeclareLanguageAlias {Kannada}{kannada}
833 \DeclareLanguageAlias {Korean}{korean}
834 \DeclareLanguageAlias {Ladin}{ladin}
835 \DeclareLanguageAlias {Lao}{lao}
836 \DeclareLanguageAlias {laotian}{lao}
837 \DeclareLanguageAlias {Laotian}{laotian}
838 \DeclareLanguageAlias {Latin}{latin}
839 \DeclareLanguageAlias {Latvian}{latvian}
840 \DeclareLanguageAlias {lettish}{latvian}
841 \DeclareLanguageAlias {Lettish}{lettish}
842 \DeclareLanguageAlias {Lithuanian}{lithuanian}
843 % hopefully someone who knows better than me can comment on these
844 \DeclareLanguageAlias {Malay}{malay}
845 \DeclareLanguageDialect{indonesian}{malay}
846 \DeclareLanguageAlias {Indonesian}{indonesian}
847 \DeclareLanguageAlias {indon}{indonesian}
848 \DeclareLanguageAlias {bahasa meyalu}{malay}
849 \DeclareLanguageAlias {Bahasa Meyalu}{bahasa meyalu}
850 \DeclareLanguageAlias {bahasa}{bahasa meyalu}
851 \DeclareLanguageAlias {Bahasa}{bahasa}
852 \DeclareLanguageAlias {bahasai}{bahasa}
853 \DeclareLanguageAlias {bahasam}{bahasa}
854 \DeclareLanguageAlias {Malayalam}{malayalam}
855 \DeclareLanguageAlias {Maldivian}{maldivian}
856 \DeclareLanguageAlias {divehi}{maldivian}
857 \DeclareLanguageAlias {Divehi}{divehi}
858 \DeclareLanguageAlias {Marathi}{marathi}
859 \DeclareLanguageAlias {Mongolian}{mongolian}
860 % \DeclareLanguageAlias {Syriac}{syriac}
861 \DeclareLanguageAlias {norsk}{norwegian}
862 \DeclareLanguageAlias {Norsk}{norsk}
863 \DeclareLanguageAlias {Norwegian}{norwegian}
864 \DeclareLanguageAlias {nynorsk}{norwegian}
865 \DeclareLanguageAlias {Nynorsk}{nynorsk}
```

5 Implementation

```

866 \DeclareLanguageAlias {Occitan}{occitan}
867 \DeclareLanguageAlias {lenga d'oc}{occitan}
868 \DeclareLanguageAlias {langue d'oc}{occitan}
869 \DeclareLanguageAlias {Piedmontese}{piedmontese}
870 \DeclareLanguageAlias {piemontese}{piedmontese}
871 \DeclareLanguageAlias {Piemontese}{piemontese}
872 \DeclareLanguageAlias {piemonteis}{piedmontese}
873 \DeclareLanguageAlias {Piemonteis}{piemonteis}
874 \DeclareLanguageAlias {Pinyin}{pinyin}
875 \DeclareLanguageAlias {Polish}{polish}
876 \DeclareLanguageDialect{brazil}{portuges}
877 \DeclareLanguageAlias {Brazil}{brazil}
878 \DeclareLanguageAlias {brazilian}{brazil}
879 \DeclareLanguageAlias {Brazilian}{brazilian}
880 \DeclareLanguageAlias {Portuges}{portuges}
881 \DeclareLanguageAlias {portuguese}{portuges}
882 \DeclareLanguageAlias {Portuguese}{portuguese}
883 \DeclareLanguageAlias {Romanian}{romanian}
884 \DeclareLanguageAlias {Romansh}{romansh}
885 \DeclareLanguageAlias {Romansch}{romansh}
886 \DeclareLanguageAlias {Rumantsh}{romansh}
887 \DeclareLanguageAlias {Rumantsch}{romansh}
888 \DeclareLanguageAlias {Romanche}{romansh}
889 \DeclareLanguageAlias {Russian}{russian}
890 \DeclareLanguageAlias {Samin}{samin}
891 \DeclareLanguageAlias {north sami}{samin}
892 \DeclareLanguageAlias {North Sami}{north sami}
893 \DeclareLanguageAlias {northern sami}{north sami}
894 \DeclareLanguageAlias {Northern Sami}{northern sami}
895 \DeclareLanguageAlias {Sanskrit}{sanskrit}
896 % this one isn't needed, or is it? vvv
897 \DeclareLanguageAlias {Serbocroatian}{serbocroatian}
898 \DeclareLanguageAlias {serbo-croatian}{serbocroatian}
899 \DeclareLanguageAlias {Serbo-Croatian}{serbocroatian}
900 \DeclareLanguageDialect{serbian}{serbocroatian}
901 \DeclareLanguageAlias {Serbian}{serbian}
902 \DeclareLanguageAlias {serbianc}{serbian}
903 \DeclareLanguageDialect{croatian}{serbocroatian}
904 \DeclareLanguageAlias {Croatian}{croatian}
905 \DeclareLanguageAlias {Slovak}{slovak}
906 \DeclareLanguageAlias {Slovenian}{slovenian}
907 \DeclareLanguageAlias {Sorbian}{sorbian}
908 \DeclareLanguageDialect{lowersorbian}{sorbian}
909 \DeclareLanguageDialect{uppersorbian}{sorbian}
910 \DeclareLanguageAlias {Lowersorbian}{lowersorbian}
911 \DeclareLanguageAlias {Uppersorbian}{uppersorbian}
912 \DeclareLanguageAlias {lsorbian}{lowersorbian}
913 \DeclareLanguageAlias {usorbian}{uppersorbian}
914 \DeclareLanguageAlias {lower sorbian}{lowersorbian}

```

5 Implementation

```

915 \DeclareLanguageAlias {upper sorbian}{uppersorbian}
916 \DeclareLanguageAlias {Lower Sorbian}{lowersorbian}
917 \DeclareLanguageAlias {Upper Sorbian}{uppersorbian}
918 \DeclareLanguageAlias {Spanglish}{spanglish}
919 \DeclareLanguageAlias {Spanish}{spanish}
920 \DeclareLanguageAlias {Swedish}{swedish}
921 \DeclareLanguageDialect{swissgerman}{german}
922 % this is to be discussed: swiss could also be an alias of french, italian or
923 % romansh:
924 \DeclareLanguageAlias {swiss}{swissgerman}
925 \DeclareLanguageAlias {Swiss}{swiss}
926 \DeclareLanguageAlias {Swissgerman}{swissgerman}
927 \DeclareLanguageAlias {swiss german}{swissgerman}
928 \DeclareLanguageAlias {Swiss German}{swiss german}
929 \DeclareLanguageAlias {Tamil}{tamil}
930 \DeclareLanguageAlias {Telugu}{telugu}
931 \DeclareLanguageAlias {Thai}{thai}
932 \DeclareLanguageAlias {thaicjk}{thai}
933 \DeclareLanguageAlias {Thaicjk}{thaicjk}
934 \DeclareLanguageAlias {Tibetan}{tibetan}
935 \DeclareLanguageAlias {Turkish}{turkish}
936 \DeclareLanguageAlias {Turkmen}{turkmen}
937 \DeclareLanguageAlias {Ukrainian}{ukrainian}
938 \DeclareLanguageAlias {Vietnamese}{vietnamese}
939 \DeclareLanguageAlias {Welsh}{welsh}
940
941 % =====
942 % OK, we have everything, do we? No, wait: let's load the basic dictionary
943 % that is part of this package if it is available for the document language
944 % -----
945 % load basic dictionary if available
946 \AtBeginDocument{%
947   \@trnslt@if@dialect{\@trnslt@current@language}
948   {%
949     \@trnslt@if@dictionary
950     {translations-basic-dictionary}
951     {\@trnslt@language{\@trnslt@current@language}}%
952     {%
953       \@trnslt@load@dictionary@silent@for
954       {translations-basic-dictionary}
955       {\@trnslt@language{\@trnslt@current@language}}%
956     }%
957     {%
958       \@trnslt@load@dictionary@silent@for
959       {translations-basic-dictionary}
960       {\@trnslt@dialect@of{\@trnslt@current@language}}%
961     }
962   }
963   {%

```

5 Implementation

```
964 \@trnslt@load@dictionary@silent@for
965 {translations-basic-dictionary}
966 {\@trnslt@language{\@trnslt@current@language}}%
967 }%
968 }
969
970 \endinput
971
972 % =====
973 % HISTORY:
974 2012/09/30 v0.2beta - first version (as part of the `exsheets' bundle)
975 2012/10/05 v0.2 - \LoadDictionary and \LoadDictionaryFor added and loads of
976 languages defined.
977 2013/03/10 v0.8 - basic dictionaries for English, German, French and Spanish
978 - new command \DeclareDictTranslation
979 2013/04/04 v0.8a - bug fix in \DeclareDictTranslation
980 2013/04/07 v0.9 - slightly improved messages
981 2013/04/08 v0.9a - changed fallback warning into info
982 - synchronized version number with `exsheets' until now but
983 won't any more
984 2013/06/22 v0.9b - added Swiss
985 2013/06/28 v0.10 - declaring aliases of dialects now works as expected
986 - declarings dialects of an alias now correctly declares
987 the dialect to the correct base language
988 - corrected a few erroneous language declarations
989 2013/07/12 v0.10a - \GetTranslation gets two-folded fallback: use
990 fallback-translation if no translation for the current
991 language has been defined; use literal string if /no/
992 language is used - this should never happen but /will/
993 happen if neither `babel' nor `polyglossia' have been
994 loaded, i.e., no language has been chosen /and/ the
995 package writer did not provide an English translation
996 2013/07/16 v1.0 - removed from `exsheets' bundle - `translations' should
997 be a package of it's own
998 - load basic dictionary automatically if available
999 - rudimentary check in \LoadDictionary if loaded file is a
1000 dictionary
1001 - new command \PrintDictionaryFor
1002 - redefined conditionals; they still seemed to make
1003 trouble in some cases
1004 2013/08/05 v1.1 - added /loads/ of languages, now the list of babel and
1005 polyglossia languages hopefully is complete
1006 - a few languages had falsely been declared as dialect
1007 instead of an alias
1008 - added weekday names and month names to basic dictionary
1009 - new command \baselanguage
1010 - new commands \GetLCTranslation, \GetLCTranslationFor,
1011 \GetLCTranslationWarn and \GetLCTranslationForWarn
1012 - load basic dictionary also for dialects and if it
```

1013 doesn't exist load it for the corresponding base
1014 language instead
1015 2013/09/30 v1.1a - Bug fix in `\NewTranslation` und `\RenewTranslation`
1016 2014/01/10 v1.2 - `\ifcurrentlanguage`, `\ifcurrentbaselanguage`
1017 - require `cnltx-base`
1018 - change the 'no language package' warning into an info
1019 - `\ProvideTranslation`
1020 - `\NewDictTranslation`, `\RenewDictTranslation`,
1021 `\ProvideDictTranslation`
1022 - translations in dictionaries are provided
1023 - `\baselanguagename`
1024 2014/01/23 v1.2a - add `\detokenize{}` so that translation keys with
1025 non-ascii chars can safely be used
1026 - fix bug in dialect declaration
1027 - rename `\baselanguagename` => `\baselanguage`
1028 - drop earlier `\baselanguage` (it was and still is
1029 available as `\@trnslt@language`)
1030 2015/07/09 v1.2b - add language 'korean'
1031 2015/08/29 v1.2c - fix bug in `\@trnslt@save@translation@for`

Bibliography

- [Bra13] Johannes BRAAMS, current maintainer: Javier BEZOS.
babel. version 3.9f, May 16, 2013.
URL: <http://mirror.ctan.org/macros/latex/required/babel/>.
- [Cha13] François CHARETTE, current maintainer: Arthur REUTENAUER.
polyglossia. version 1.33.4, June 27, 2013.
URL: <http://mirror.ctan.org/macros/latex/contrib/polyglossia/>.
- [Koh15] Markus KOHM. KOMA-Script. version 3.18, July 2, 2015.
URL: <http://mirror.ctan.org/macros/latex/contrib/koma-script/>.
- [Nie14] Clemens NIEDERBERGER. cnltx. version 0.10a, Jan. 23, 2014.
URL: <http://mirror.ctan.org/macros/latex/contrib/cnltx/>.
- [Tan10] Till TANTAU. translator. version 1.0, June 12, 2010.
URL: <http://mirror.ctan.org/macros/contrib/beamer/base/translator/>.

Index

B	cnltx (bundle) 2	32–36
babel (package) 1 f., 4–7, 14, 17	cnltx-base (package) 2	
<code>\baselanguage</code> 6, 21, 37		<code>\DeclareLanguageDialect</code> . . . 3, 20,
BEZOS, Javier 1		33–36
BRAAMS, Johannes 1	D	
	<code>\DeclareDictTranslation</code> 5, 10, 29,	<code>\DeclareTranslation</code> . 4, 7 f., 23, 26
	37	
C	<code>\DeclareLanguage</code> . . 3, 5, 9, 19, 30 ff.	<code>\DeclareTranslationFallback</code> . . 4,
CHARETTE, François 1	<code>\DeclareLanguageAlias</code> . . 3, 21, 30,	7 f., 23

INDEX

G		
\GetLCTranslation	4	
\GetLCTranslationFor	4	
\GetLCTranslationForWarn	4	
\GetLCTranslationWarn	5	
\GetTranslation .. 2, 4, 6 ff., 25, 37		
\GetTranslationFor	4, 6 f., 24	
\GetTranslationForWarn	4	
\GetTranslationWarn	4	
I		
\ifcurrentbaselanguage	6	
\ifcurrentlanguage	6	
K		
KOHM, Markus	2	
L		
KOMA-Script (bundle)	2	
L		
\LoadDictionary	5, 9, 27, 37	
\LoadDictionaryFor ...	5, 9, 27, 37	
LPPL	2	
N		
\NewDictTranslation	5, 9 f.	
\NewTranslation	3, 23	
NIEDERBERGER, Clemens	2	
P		
polyglossia (package) ..	1 f., 7, 14, 17	
\PrintDictionaryFor	5	
\ProvideDictionaryFor ..	5, 9 f., 28	
\ProvideDictTranslation	5, 9 f., 29	
\ProvideTranslation	4	
R		
\RenewDictTranslation	5, 10	
\RenewTranslation	4, 23	
REUTENAUER, Arthur	1	
S		
\SaveTranslation	5 f., 27	
\SaveTranslationFor	5 f., 26	
scrfile (package)	2	
T		
TANTAU, Till	1	
translator (package)	1 f.	