

TRANSLATIONS

V1.4 2016/06/02

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 . . .	10
			3.5.3	TRANSLATIONS' Basic Dictionaries	11
2	License and Requirements	2			
3	Usage	2	4	Defined Languages	14
3.1	Background	2	4.1	Base Languages	14
3.2	Available Commands	3	4.2	Language Dialects	15
3.3	A Small Example	7	4.3	Language Aliases	16
3.4	Usage in Packages	8			
3.4.1	Basic Structure	8	5	Implementation	18
3.4.2	The 'fallback' language	8			
3.5	Dictionaries	10		Bibliography	42
3.5.1	Background	10		Index	42

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 [Bra16] 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 [Cha15]) 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 15), language dialects (see table 3 on page 15), and language aliases (see table 4 on page 16). 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.

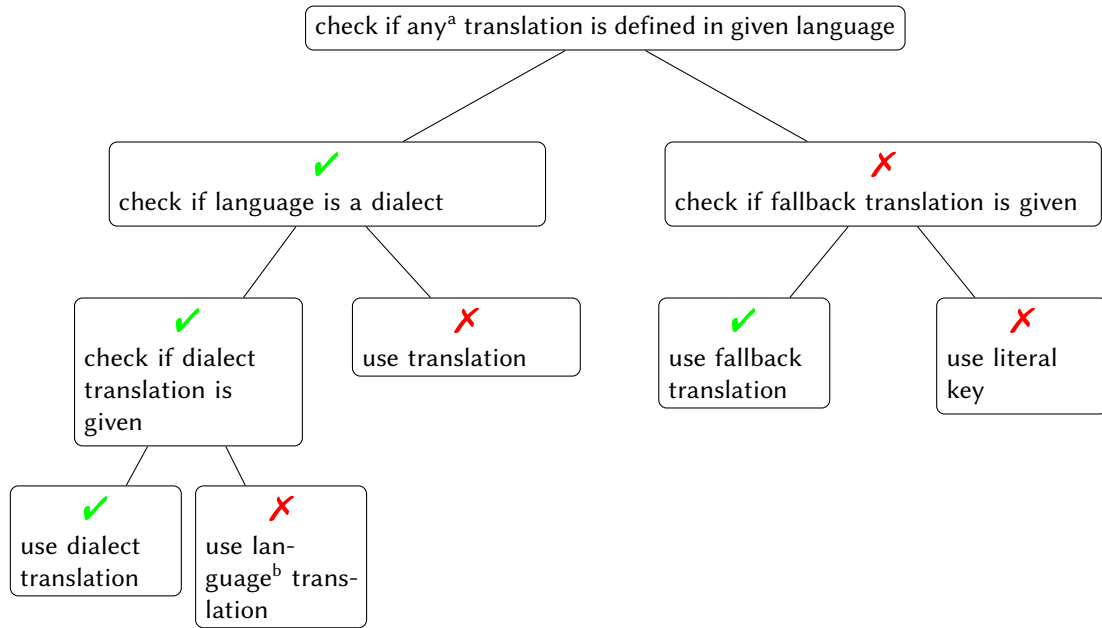


FIGURE 1: Schematic representation of **TRANSLATIONS**' translating mechanism. Notes: ^a except for a possible fallback translation. ^b *i. e.*, the base language of the dialect.

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⟩` in language `⟨lang⟩` already exists. This command can only be used in the preamble.

Introduced in
version 1.4

`\NewTranslationFallback{⟨key⟩}{⟨translation⟩}`

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.

Introduced in
version 1.4

`\RenewTranslationFallback{⟨key⟩}{⟨translation⟩}`

Renews a fallback translation. This command can only be used in the preamble.

Introduced in
version 1.2

`\ProvideTranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`

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.

Introduced in
version 1.4

`\ProvideTranslationFallback{⟨key⟩}{⟨translation⟩}`

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.

Introduced in
version 1.4

`\newtranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`

A version of `\NewTranslation` that *can* be used after begin document.

Introduced in
version 1.4

`\newtranslationfallback{⟨key⟩}{⟨translation⟩}`

A version of `\NewTranslationFallback` that *can* be used after begin document.

Introduced in
version 1.4

`\renewtranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`

A version of `\RenewTranslation` that *can* be used after begin document.

Introduced in
version 1.4

`\renewtranslationfallback{⟨key⟩}{⟨translation⟩}`

A version of `\RenewTranslationFallback` that *can* be used after begin document.

Introduced in
version 1.4

`\providetranslation{⟨lang⟩}{⟨key⟩}{⟨translation⟩}`

A version of `\ProvideTranslation` that *can* be used after begin document.

Introduced in
version 1.4

`\providetranslationfallback{⟨key⟩}{⟨translation⟩}`

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

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}` \Rightarrow english
`\baselanguage{American}` \Rightarrow english

* `\ifcurrentlanguage{<lang>}{<true>}{<false>}`

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{<lang>}{<true>}{<false>}`

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
14
15 \IfTranslation{German}{Kueche}{true}{false} \par
16 \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:

```

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 \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:


```

1 \documentclass[margin=5mm]{standalone}
2 \usepackage{translations}
3 \DeclareTranslation{German}{foo-literal}{foo-german}
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}{foo-german}
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 \DeclareTranslationFallback{foo-literal}{foo}
5 \DeclareTranslation{German}{foo-literal}{foo-german}
6 \begin{document}
7 \GetTranslation{foo-literal} % foo-german
8 \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>}`

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.

Introduced in
version 1.3

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⟩}`

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 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
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

continues

key	translation
or	oder
Outline	Gliederung
Overview	Übersicht
Pages	Seiten
pages	Seiten
Page	Seite
page	Seite
Paragraphs	Absätze
paragraphs	Absätze
Paragraph	Absatz
paragraph	Absatz
Parts	Teile
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

continues

key	translation
tables	Tabellen
Table	Tabelle
table	Tabelle
To	An
to	an
Monday	Montag
Tuesday	Dienstag
Wednesday	Mittwoch
Thursday	Donnerstag
Friday	Freitag
Saturday	Samstag
Sunday	Sonntag
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	latin	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
canadien	acadian	american	australian	british
canadian	newzealand	irish	scottish	austrian
hindi	urdu	indonesian	brazil	serbian
croatian	lowersorbian	uppersorbian	swissgerman	swissfrench
swissitalian	swissromansh			

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

continues

dialect	language	dialect	language
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	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
français	french	Français	français
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

continues

4 Defined Languages

alias	language	alias	language
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	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

continues

alias	language	alias	language
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
Isorbian	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	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

```

5 Implementation

```
10 % -----
11 % Copyright 2012-2016 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 \newcommand*\@trnslt@date{2016/06/02}
29 \newcommand*\@trnslt@version{v1.4}
30 \newcommand*\@trnslt@info{internationalization of LaTeX2e packages}
31
32 \ProvidesPackage{translations}[%
33   \@trnslt@date\space
34   \@trnslt@version\space
35   \@trnslt@info\space (CN)]
36 \RequirePackage{cnltx-base,scrlfile}
37
38 % -----
39 % message handling
40 % generic help message:
41 \newcommand*\@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{
```

5 Implementation

```

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{}
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

```

5 Implementation

```
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     \listead{\@trnslt@languages{#1}%
130   }%
131 }
132
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}%

```

5 Implementation

```

157 }%
158 \@trnslt@if@dialect{#1}
159 {% => ist schon als dialect definiert => irgendwelche weiteren checks?
160 }
161 {%
162   \@trnslt@if@alias{#2}
163   {%
164     \csedef{@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:
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

```

5 Implementation

```

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}}
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@if@language[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@if@baselanguage[1]{%
252   \@trnslt@if@dialect{\language}
253   {%
254     \cnltx@braced@expanded@fully

```

5 Implementation

```

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
280 \newrobustcmd*\declaretranslation[3]{%
281   \@trnslt@declare@translation{#2}{#1}{#3}%
282 }
283 \newrobustcmd*\newtranslation[3]{%
284   \@trnslt@new@translation{#2}{#1}{#3}%
285 }
286 \newrobustcmd*\renewtranslation[3]{%
287   \@trnslt@renew@translation{#2}{#1}{#3}%
288 }
289 \newrobustcmd*\providetranslation[3]{%
290   \@trnslt@provide@translation{#2}{#1}{#3}%
291 }
292
293 \let\DeclareTranslation\declaretranslation
294 \@onlypreamble\DeclareTranslation
295 \let\NewTranslation\newtranslation
296 \@onlypreamble\NewTranslation
297 \let\RenewTranslation\renewtranslation
298 \@onlypreamble\RenewTranslation
299 \let\ProvideTranslation\providetranslation
300 \@onlypreamble\ProvideTranslation
301
302 \newrobustcmd*\declaretranslationfallback{%
303   \declaretranslation{fallback}%

```


5 Implementation

```

304 }
305 \newrobustcmd*\newtranslationfallback{%
306   \newtranslation{fallback}%
307 }
308 \newrobustcmd*\renewtranslationfallback{%
309   \renewtranslation{fallback}%
310 }
311 \newrobustcmd*\providetranslationfallback{%
312   \providetranslation{fallback}%
313 }
314
315 \let\DeclareTranslationFallback\declaretranslationfallback
316 \@onlypreamble\DeclareTranslationFallback
317 \let\NewTranslationFallback\newtranslationfallback
318 \@onlypreamble\NewTranslationFallback
319 \let\RenewTranslationFallback\renewtranslationfallback
320 \@onlypreamble\RenewTranslationFallback
321 \let\ProvideTranslationFallback\providetranslationfallback
322 \@onlypreamble\ProvideTranslationFallback
323
324 % #1: word
325 % #2: language
326 % #3: replacement
327 \newrobustcmd*\@trnslt@declare@translation[3]{%
328   \@trnslt@if@language{#2}
329   {%
330     \@trnslt@if@dialect{#2}
331     {%
332       \csdef{@trnslt@word@detokenize{#1}@@trnslt@dialect{#2}}{#3}%
333       \@trnslt@if@word\@trnslt@dialect@of{#1}{#2}
334       {}
335       {\csdef{@trnslt@word@detokenize{#1}@@trnslt@dialect@of{#2}}{#3}}%
336     }
337     {\csdef{@trnslt@word@detokenize{#1}@@trnslt@language{#2}}{#3}}%
338     % save the <word> as <word>:
339     \csdef{@trnslt@word@detokenize{#1}@literal}{#1}%
340   }
341   {\@trnslt@err@unknown@lang{#2}}}%
342 }
343
344 \newrobustcmd*\@trnslt@new@translation[3]{%
345   \@trnslt@if@word\@trnslt@language{#1}{#2}
346   {\@trnslt@err@already@defined{#1}{#2}}
347   {\@trnslt@declare@translation{#1}{#2}{#3}}%
348 }
349
350 \newrobustcmd*\@trnslt@renew@translation[3]{%
351   \@trnslt@if@word\@trnslt@language{#1}{#2}
352   {\@trnslt@declare@translation{#1}{#2}{#3}}

```

5 Implementation

```

353     {\@trnslt@err@not@defined{#1}{#2}}%
354 }
355
356 \newrobustcmd*\@trnslt@provide@translation[3]{%
357   \@trnslt@if@word\@trnslt@language{#1}{#2}
358   {}
359   {\@trnslt@declare@translation{#1}{#2}{#3}}%
360 }
361
362 % -----
363 % let's go through some trouble to check if a translation exists:
364 \newcommand*\@trnslt@if@word[3]{%
365   \ifcsname @trnslt@word\detokenize{#2}@\#1{#3}\endcsname
366   \expandafter\@firstoftwo
367   \else
368   \expandafter\@secondoftwo
369   \fi
370 }%
371
372 % #1: word
373 % #2: language
374 \newcommand*\@trnslt@if@translation[2]{%
375   \@trnslt@if@word\@trnslt@language{#1}{#2}
376   {\expandafter\@firstoftwo}
377   {%
378     \@trnslt@if@dialect{#2}
379     {%
380       \@trnslt@if@word\@trnslt@dialect{#1}{#2}
381       {\expandafter\@firstoftwo}
382       {%
383         \@trnslt@if@word\@trnslt@dialect@of{#1}{#2}
384         {\expandafter\@firstoftwo}
385         {\expandafter\@secondoftwo}%
386       }
387     }
388     {\expandafter\@secondoftwo}%
389   }%
390 }
391
392 % #1: language
393 % #2: word
394 \newcommand*\IfTranslation[2]{\@trnslt@if@translation{#2}{#1}}
395
396 % -----
397 % We also need to be able to look up a translation:
398 % \GetTranslationFor and \GetTranslation
399 % these need to be expandable!
400 % #1: language
401 % #2: word

```

5 Implementation

```

402 \newcommand*\GetTranslationFor[2]{%
403   \@trnslt@checkandget@translation@for{#2}{#1}}
404
405 \newcommand*\GetTranslation[1]{%
406   \@trnslt@checkandget@translation@for{#1}{\@trnslt@current@language}}
407
408 % those will print the translation lowercase... they're also expandable; that
409 % is: nearly...
410 %   \DeclareTranslation{German}{foo}{Bla}
411 %   \edef\foo{\GetLCTranslationFor{German}{foo}}
412 %   \show\foo
413 %   > \foo=macro:
414 %   ->\lowercase {Bla}.
415 \newcommand*\GetLCTranslationFor[2]{%
416   \@trnslt@get@lowercase{\@trnslt@checkandget@translation@for{#2}{#1}}%
417 }
418
419 \newcommand*\GetLCTranslation[1]{%
420   \@trnslt@get@lowercase{%
421     \@trnslt@checkandget@translation@for{#1}{\@trnslt@current@language}%
422   }%
423 }
424
425 % unexpandable version of the commands that raise a warning if no translation
426 % is available:
427 \newrobustcmd*\GetTranslationForWarn[2]{%
428   \@trnslt@getandwarn@translation@for{#2}{#1}%
429 }
430
431 \newrobustcmd*\GetTranslationWarn[1]{%
432   \@trnslt@getandwarn@translation@for{#1}{\@trnslt@current@language}%
433 }
434
435 \newrobustcmd*\GetLCTranslationForWarn[2]{%
436   \@trnslt@getandwarn@lctranslation@for{#2}{#1}%
437 }
438
439 \newrobustcmd*\GetLCTranslationWarn[1]{%
440   \@trnslt@getandwarn@lctranslation@for{#1}{\@trnslt@current@language}%
441 }
442
443 % #1: word #2: language
444 \newcommand*\@trnslt@get@translation@for[2]{%
445   \@trnslt@if@dialect{#2}
446   {%
447     \ifcsdef{\@trnslt@word@\detokenize{#1}}{\@trnslt@dialect{#2}}
448     {\csuse{\@trnslt@word@\detokenize{#1}}{\@trnslt@dialect{#2}}}
449     {\csuse{\@trnslt@word@\detokenize{#1}}{\@trnslt@dialect@of{#2}}}%
450   }

```

5 Implementation

```

451     {\csuse{@trnslt@word@\detokenize{#1}@\trnslt@language{#2}}}%
452 }
453
454 \newcommand*\@trnslt@checkandget@translation@for[2]{%
455   \@trnslt@if@translation{#1}{#2}
456   {\@trnslt@get@translation@for{#1}{#2}}
457   {%
458     \@trnslt@if@translation{#1}{fallback}
459     {\csuse{@trnslt@word@\detokenize{#1}@fallback}}
460     {\csuse{@trnslt@word@\detokenize{#1}@literal}}}%
461   }%
462 }
463
464 % this is not expandable!
465 \newrobustcmd*\@trnslt@getandwarn@translation@for[2]{%
466   \@trnslt@if@translation{#1}{#2}
467   {\@trnslt@get@translation@for{#1}{#2}}
468   {%
469     \@trnslt@warning{Translation for `#1' in #2 unknown. You may try to use
470       \string\DeclareTranslation{#2}{#1}{ ... } in your preamble.}%
471     \@trnslt@if@translation{#1}{fallback}
472     {%
473       \@trnslt@info{Using fallback translation for `#1'}%
474       \csuse{@trnslt@word@\detokenize{#1}@fallback}
475     }
476     {\csuse{@trnslt@word@\detokenize{#1}@literal}}}%
477   }%
478 }
479
480 % lowercase version for translation with warnings:
481 \newrobustcmd*\@trnslt@getandwarn@lctranslation@for[2]{%
482   \@trnslt@if@translation{#1}{#2}
483   {\@trnslt@get@lowercase{\@trnslt@get@translation@for{#1}{#2}}}
484   {%
485     \@trnslt@warning{Translation for `#1' in #2 unknown. You may try to use
486       \string\DeclareTranslation{#2}{#1}{ ... } in your preamble.}%
487     \@trnslt@if@translation{#1}{fallback}
488     {%
489       \@trnslt@info{Using fallback translation for `#1'}%
490       \@trnslt@get@lowercase{\csuse{@trnslt@word@\detokenize{#1}@fallback}}%
491     }
492     {\@trnslt@get@lowercase{\csuse{@trnslt@word@\detokenize{#1}@literal}}}%
493   }%
494 }
495
496 % -----
497 % no idea if the following ones really are needed... let's define them
498 % anyway
499 % \SaveTranslationFor and \SaveTranslation

```

5 Implementation

```
500 \newrobustcmd*\SaveTranslationFor[3]{%
501   \@trnslt@save@translation@for{#1}{#3}{#2}%
502 }
503
504 \newrobustcmd*\SaveTranslation[2]{%
505   \@trnslt@save@translation@for{#1}{#2}{\@trnslt@current@language}%
506 }
507
508 \newrobustcmd*\@trnslt@save@translation@for[3]{%
509   \edef#1{%
510     \@trnslt@if@translation{#2}{#3}
511     {%
512       \unexpanded\expandafter\expandafter\expandafter
513       {\csname @trnslt@word@detokenize{#2}@@trnslt@language{#3}\endcsname}%
514     }
515     }%
516   }%
517 }
518
519 % -----
520 % The basics are set. What's missing? Right: dictionaries! Those will be files
521 % that contain translations for a specific language.
522 % -----
523 % load dictionaries and check for existing ones:
524 % \LoadDictionary and \LoadDictionaryFor
525 \newrobustcmd*\LoadDictionary[1]{%
526   \@trnslt@load@dictionary@for{#1}{\@trnslt@current@language}}
527 \onlypreamble\LoadDictionary
528
529 \newrobustcmd*\LoadDictionaryFor[2]{%
530   \@trnslt@load@dictionary@for{#2}{#1}}
531 \onlypreamble\LoadDictionaryFor
532
533 \newrobustcmd*\LoadDictionaryForDialect[2]{%
534   \@trnslt@load@dictionary@for@dialect{#2}{#1}}
535 \onlypreamble\LoadDictionaryFor
536
537 % #1: name
538 % #2: lang
539 % #3: false
540 \newrobustcmd*\@trnslt@input@if@dictionary@exists[3]{%
541   \AtBeginDocument{%
542     \InputIfFileExists{#1-\@trnslt@language{#2}.trsl}
543     {\@trnslt@check@dictionary{#1}{#2}}
544     {#3}%
545   }%
546 }
547
548 % #1: name
```

5 Implementation

```

549 % #2: lang
550 % #3: false
551 \newrobustcmd*\@trnslt@load@dictionary@for@F[3]{%
552   \@trnslt@input@if@dictionary@exists{#1}{#2}{#3}%
553 }
554
555 % #1: name
556 % #2: lang
557 \newrobustcmd*\@trnslt@load@dictionary@for@dialect[2]{%
558   \@trnslt@load@dictionary@for@F{#1}{#2}
559   {%
560     \@trnslt@warning{dictionary file `#1-\@trnslt@language{#2}.trsl' not
561       found.}%
562   }%
563 }
564
565 % #1: name
566 % #2: lang
567 \newrobustcmd*\@trnslt@load@dictionary@for[2]{%
568   \@trnslt@load@dictionary@for@F{#1}{#2}
569   {%
570     \@trnslt@load@dictionary@for@F{#1}{\baselanguage{#2}}
571     {%
572       \@trnslt@warning{neither dictionary file
573         `#1-\@trnslt@language{#2}.trsl' nor `#1-\baselanguage{#2}.trsl'
574         found.}%
575     }%
576   }%
577 }
578
579 \newrobustcmd*\@trnslt@load@dictionary@silent@for[2]{%
580   \@trnslt@load@dictionary@for@F{#1}{#2}{%
581     \@trnslt@load@dictionary@for@F{#1}{\baselanguage{#2}}}%
582 }%
583 }
584
585 % #1: name
586 % #2: lang
587 \newrobustcmd*\@trnslt@check@dictionary[2]{%
588   \AfterFile{#1-\@trnslt@language{#2}.trsl}
589   {%
590     \edef\@trnslt@tmpa{\@trnslt@language{#2}}%
591     \ifcsdef\@trnslt@dictionary@#1\@trnslt@tmpa{
592       {\@trnslt@info{loading dictionary `#1' for `#2'.}}
593       {%
594         \@trnslt@warning{file `#1-\@trnslt@language{#2}.trsl' does not
595           appear to be a dictionary}%
596       }%
597     }%

```

5 Implementation

```

598 }
599
600 % \newcommand*\@trnslt@if@dictionary[2]{\IfFileExists{#1-#2.trsl}}
601
602 % -----
603 % the contents of a dictionary; let's declare that is one.
604 % #1: lang
605 % #2: name
606 \newrobustcmd*\ProvideDictionaryFor[2]{%
607   \@trnslt@provide@dictionary@for{#1}{#2}}
608 \onlypreamble\ProvideDictionaryFor
609
610 % #1: lang
611 % #2: name
612 \newrobustcmd*\@trnslt@provide@dictionary@for[2]{%
613   \def\@trnslt@dictionary@name{#2}%
614   \edef\@trnslt@dictionary@lang{\@trnslt@language{#1}}%
615   % this macro can be used to check if we have a dictionary and will also be
616   % used as a list for the dictionary entries:
617   \csdef{\@trnslt@dictionary@\@trnslt@dictionary@name \@trnslt@dictionary@lang}{}%
618   \@ifnextchar[
619     {\@trnslt@provide@dictionary@version}
620     {
621       \ProvidesFile
622         {#2-\@trnslt@dictionary@lang.trsl}%
623         [(\@trnslt@dictionary@lang\space translation file `#2')]
624     }%
625 }
626
627 \protected\def\@trnslt@provide@dictionary@version[#1]{%
628   \ProvidesFile
629     {\@trnslt@dictionary@name-\@trnslt@dictionary@lang.trsl}%
630     [(\@trnslt@dictionary@lang\space translation file \@trnslt@dictionary@name')
631     #1]}
632
633 % change this test (we can't use braces inside the item with \ifinlist):
634 \newcommand*\@trnslt@check@dictionary@entry[2]{%
635   \cnltx@ifinlistcs
636     {{#1}{#2}}
637     {\@trnslt@dictionary@\@trnslt@dictionary@name \@trnslt@dictionary@lang}%
638 }
639
640 \newrobustcmd*\@trnslt@add@dictionary@entry[2]{%
641   \listcsadd
642     {\@trnslt@dictionary@\@trnslt@dictionary@name \@trnslt@dictionary@lang}
643     {{#1}{#2}}%
644 }
645 % -----

```

5 Implementation

```
646 % now we should be able to add translations without repeatedly type the
647 % language ...
648 \newrobustcmd*\NewDictTranslation[2]{%
649   \@trnslt@check@dictionary@entry{#1}{#2}
650   {\@trnslt@err@dict@already@defined{#1}{#2}}%
651   {%
652     \@trnslt@add@dictionary@entry{#1}{#2}
653     \@trnslt@new@translation{#1}{\@trnslt@dictionary@lang}{#2}%
654   }%
655 }
656 \@onlypreamble\NewDictTranslation
657
658 \newrobustcmd*\RenewDictTranslation[2]{%
659   \@trnslt@check@dictionary@entry{#1}{#2}
660   {\@trnslt@renew@translation{#1}{\@trnslt@dictionary@lang}{#2}}
661   {\@trnslt@err@dict@not@defined{#1}{#2}}%
662 }
663 \@onlypreamble\RenewDictTranslation
664
665 \newrobustcmd*\DeclareDictTranslation[2]{%
666   \@trnslt@check@dictionary@entry{#1}{#2}
667   {}
668   {\@trnslt@add@dictionary@entry{#1}{#2}}%
669   \@trnslt@declare@translation{#1}{\@trnslt@dictionary@lang}{#2}%
670 }
671 \@onlypreamble\DeclareDictTranslation
672
673 \newrobustcmd*\ProvideDictTranslation[2]{%
674   \@trnslt@check@dictionary@entry{#1}{#2}
675   {}
676   {%
677     \@trnslt@add@dictionary@entry{#1}{#2}%
678     \@trnslt@provide@translation{#1}{\@trnslt@dictionary@lang}{#2}%
679   }%
680 }
681 \@onlypreamble\ProvideDictTranslation
682
683 % -----
684 % last not least let's inspect which entries a dictionary contains of:
685 % \PrintDictionaryFor
686 % #1: lang
687 % #2: name
688 % #3: pre
689 % #4: mid
690 % #5: post
691 \newcommand*\PrintDictionaryFor[5]{%
692   \@trnslt@print@dictionary@for{#1}{#2}{#3}{#4}{#5}%
693 }
694
```


5 Implementation

```
695 % #1: lang
696 % #2: name
697 % #3: pre
698 % #4: mid
699 % #5: post
700 \newcommand*\@trnslt@print@dictionary@for[5]{%
701   \forlistcsloop
702     {\@trnslt@print@dictionary@entry{#3}{#4}{#5}}
703     {\@trnslt@dictionary@#2@\@trnslt@language{#1}}%
704 }
705
706 % #1: pre
707 % #2: mid
708 % #3: post
709 % #4: (key)(translation)
710 \newcommand*\@trnslt@print@dictionary@entry[4]{%
711   \@trnslt@print@dictionary@entry@aux{#1}{#2}{#3}#4%
712 }
713
714 % #1: pre
715 % #2: mid
716 % #3: post
717 % #4: key
718 % #5: translation
719 \newcommand*\@trnslt@print@dictionary@entry@aux[5]{#1#4#2#5#3}
720
721 % =====
722 % Now that the package is finished let's use the above commands to provide a
723 % basis for usage; we need all languages known to 'babel' and 'polyglossia'
724 % as well as aliases to allow different spellings/names; we also need dialects
725 % where it makes sense (e.g. British as a dialect of English)
726
727 % -----
728 % dummy language 'fallback' -- it is needed for the whole mechanism provided
729 % earlier (\DeclareFallbackTranslation...)
730 \DeclareLanguage{fallback}
731 \DeclareLanguageAlias{Fallback}{fallback}
732
733 % -----
734 % predefined languages
735 \DeclareLanguage{afrikaans}
736 \DeclareLanguage{albanian}
737 \DeclareLanguage{amharic}
738 \DeclareLanguage{arabic}
739 \DeclareLanguage{armenian}
740 \DeclareLanguage{asturian}
741 \DeclareLanguage{basque}
742 \DeclareLanguage{bengali}
743 \DeclareLanguage{breton}
```

```

744 \DeclareLanguage{bulgarian}
745 \DeclareLanguage{catalan}
746 \DeclareLanguage{coptic}
747 \DeclareLanguage{czech}
748 \DeclareLanguage{danish}
749 \DeclareLanguage{dutch}
750 \DeclareLanguage{english}
751 \DeclareLanguage{esperanto}
752 \DeclareLanguage{estonian}
753 \DeclareLanguage{ethiop}
754 \DeclareLanguage{farsi}
755 \DeclareLanguage{finnish}
756 \DeclareLanguage{french}
757 \DeclareLanguage{friulan}
758 \DeclareLanguage{gaelic}
759 \DeclareLanguage{galician}
760 \DeclareLanguage{german}
761 \DeclareLanguage{greek}
762 \DeclareLanguage{hebrew}
763 \DeclareLanguage{hindustani}
764 \DeclareLanguage{hungarian}
765 \DeclareLanguage{icelandic}
766 \DeclareLanguage{interlingua}
767 \DeclareLanguage{italian}
768 \DeclareLanguage{japanese}
769 \DeclareLanguage{kannada}
770 \DeclareLanguage{korean}
771 \DeclareLanguage{ladin}
772 \DeclareLanguage{lao}
773 \DeclareLanguage{latin}
774 \DeclareLanguage{latvian}
775 \DeclareLanguage{lithuanian}
776 \DeclareLanguage{macedonian}
777 \DeclareLanguage{malay}
778 \DeclareLanguage{malayalam}
779 \DeclareLanguage{maldivian}
780 \DeclareLanguage{marathi}
781 \DeclareLanguage{mongolian}
782 % polyglossia seems to support this one but it is unclear which language is
783 % actually meant by it:
784 % \DeclareLanguage{nko}
785 \DeclareLanguage{norwegian}
786 \DeclareLanguage{occitan}
787 \DeclareLanguage{piedmontese}
788 \DeclareLanguage{pinyin}
789 \DeclareLanguage{polish}
790 \DeclareLanguage{portuges}
791 \DeclareLanguage{romanian}
792 \DeclareLanguage{romansh}

```

5 Implementation

```
793 \DeclareLanguage{russian}
794 \DeclareLanguage{samin}
795 \DeclareLanguage{sanskrit}
796 \DeclareLanguage{serbocroatian}
797 \DeclareLanguage{slovak}
798 \DeclareLanguage{slovenian}
799 \DeclareLanguage{sorbian}
800 % not sure about this: isn't it either a Spanish or English dialect?
801 \DeclareLanguage{spanglish}
802 \DeclareLanguage{spanish}
803 \DeclareLanguage{swedish}
804 % polyglossia seems to support this one but it is unclear which language is
805 % actually meant by it:
806 % \DeclareLanguage{syriac}
807 \DeclareLanguage{tamil}
808 \DeclareLanguage{telugu}
809 \DeclareLanguage{thai}
810 \DeclareLanguage{tibetan}
811 \DeclareLanguage{turkish}
812 \DeclareLanguage{turkmen}
813 \DeclareLanguage{ukrainian}
814 \DeclareLanguage{vietnamese}
815 \DeclareLanguage{welsh}
816
817 % -----
818 % aliases and dialects:
819 \DeclareLanguageAlias {Afrikaans}{afrikaans}
820 \DeclareLanguageAlias {Albanian}{albanian}
821 \DeclareLanguageAlias {Amharic}{amharic}
822 \DeclareLanguageAlias {Arabic}{arabic}
823 \DeclareLanguageAlias {Armenian}{armenian}
824 \DeclareLanguageAlias {Asturian}{asturian}
825 \DeclareLanguageAlias {astur-leonese}{asturian}
826 \DeclareLanguageAlias {Astur-Leonese}{astur-leonese}
827 \DeclareLanguageAlias {asturian-leonese}{asturian}
828 \DeclareLanguageAlias {Asturian-Leonese}{asturian-leonese}
829 \DeclareLanguageAlias {Basque}{basque}
830 \DeclareLanguageAlias {Bengali}{bengali}
831 \DeclareLanguageAlias {Breton}{breton}
832 \DeclareLanguageAlias {Bulgarian}{bulgarian}
833 \DeclareLanguageAlias {Catalan}{catalan}
834 \DeclareLanguageAlias {Coptic}{coptic}
835 \DeclareLanguageAlias {coptic egyptian}{coptic}
836 \DeclareLanguageAlias {Coptic Egyptian}{coptic egyptian}
837 \DeclareLanguageAlias {Czech}{czech}
838 \DeclareLanguageAlias {Danish}{danish}
839 \DeclareLanguageAlias {Dutch}{dutch}
840 \DeclareLanguageAlias {Farsi}{farsi}
841 \DeclareLanguageAlias {Finnish}{finnish}
```

5 Implementation

```
842 \DeclareLanguageAlias {francais}{french}
843 \DeclareLanguageAlias {Francais}{francais}
844 \DeclareLanguageDialect{canadien}{french}
845 \DeclareLanguageAlias {Canadien}{canadien}
846 \DeclareLanguageAlias {French}{french}
847 \DeclareLanguageDialect{acadian}{french}
848 \DeclareLanguageAlias {Acadian}{acadian}
849 \DeclareLanguageAlias {frenchle}{french}
850 \DeclareLanguageDialect{american}{english}
851 \DeclareLanguageAlias {American}{american}
852 \DeclareLanguageDialect{australian}{english}
853 \DeclareLanguageAlias {Australian}{australian}
854 \DeclareLanguageDialect{british}{english}
855 \DeclareLanguageAlias {British}{british}
856 \DeclareLanguageDialect{canadian}{english}
857 \DeclareLanguageAlias {Canadian}{canadian}
858 \DeclareLanguageAlias {English}{english}
859 \DeclareLanguageAlias {UKenglish}{british}
860 \DeclareLanguageAlias {USenglish}{american}
861 \DeclareLanguageDialect{newzealand}{english}
862 \DeclareLanguageAlias {Newzealand}{newzealand}
863 \DeclareLanguageAlias {Ethiop}{ethiop}
864 \DeclareLanguageAlias {Esperanto}{esperanto}
865 \DeclareLanguageAlias {Estonian}{estonian}
866 \DeclareLanguageAlias {Friulan}{friulan}
867 \DeclareLanguageAlias {Gaelic}{gaelic}
868 \DeclareLanguageDialect{irish}{gaelic}
869 \DeclareLanguageDialect{scottish}{gaelic}
870 \DeclareLanguageAlias {Irish}{irish}
871 \DeclareLanguageAlias {irish gaelic}{irish}
872 \DeclareLanguageAlias {Irish Gaelic}{irish}
873 \DeclareLanguageAlias {Scottish}{scottish}
874 \DeclareLanguageAlias {scottish gaelic}{scottish}
875 \DeclareLanguageAlias {Scottish Gaelic}{scottish}
876 \DeclareLanguageAlias {Galician}{galician}
877 \DeclareLanguageAlias {German}{german}
878 \DeclareLanguageAlias {germanb}{german}
879 \DeclareLanguageAlias {ngerman}{german}
880 \DeclareLanguageDialect{austrian}{german}
881 \DeclareLanguageAlias {Austrian}{austrian}
882 \DeclareLanguageAlias {naustrian}{austrian}
883 \DeclareLanguageAlias {Greek}{greek}
884 \DeclareLanguageAlias {polutonikogreek}{greek}
885 \DeclareLanguageAlias {ibygreek}{greek}
886 \DeclareLanguageAlias {bgreek}{greek}
887 \DeclareLanguageAlias {Hebrew}{hebrew}
888 \DeclareLanguageAlias {Hindustani}{hindustani}
889 \DeclareLanguageAlias {hindi-urdu}{hindustani}
890 \DeclareLanguageAlias {Hindi-Urdu}{hindi-urdu}
```

5 Implementation

```
891 \DeclareLanguageDialect{hindi}{hindustani}
892 \DeclareLanguageAlias {Hindi}{hindi}
893 \DeclareLanguageDialect{urdu}{hindustani}
894 \DeclareLanguageAlias {Urdu}{urdu}
895 \DeclareLanguageAlias {Hungarian}{hungarian}
896 \DeclareLanguageAlias {magyar}{hungarian}
897 \DeclareLanguageAlias {Magyar}{magyar}
898 \DeclareLanguageAlias {Icelandic}{icelandic}
899 \DeclareLanguageAlias {Interlingua}{interlingua}
900 \DeclareLanguageAlias {Italian}{italian}
901 \DeclareLanguageAlias {Japanese}{japanese}
902 \DeclareLanguageAlias {Kannada}{kannada}
903 \DeclareLanguageAlias {Korean}{korean}
904 \DeclareLanguageAlias {Ladin}{ladin}
905 \DeclareLanguageAlias {Lao}{lao}
906 \DeclareLanguageAlias {laotian}{lao}
907 \DeclareLanguageAlias {Laotian}{laotian}
908 \DeclareLanguageAlias {Latin}{latin}
909 \DeclareLanguageAlias {Latvian}{latvian}
910 \DeclareLanguageAlias {lettish}{latvian}
911 \DeclareLanguageAlias {Lettish}{lettish}
912 \DeclareLanguageAlias {Lithuanian}{lithuanian}
913 \DeclareLanguageAlias {Macedonian}{macedonian}
914 % hopefully someone who knows better than me can comment on these
915 \DeclareLanguageAlias {Malay}{malay}
916 \DeclareLanguageDialect{indonesian}{malay}
917 \DeclareLanguageAlias {Indonesian}{indonesian}
918 \DeclareLanguageAlias {indon}{indonesian}
919 \DeclareLanguageAlias {bahasa meyalu}{malay}
920 \DeclareLanguageAlias {Bahasa Meyalu}{bahasa meyalu}
921 \DeclareLanguageAlias {bahasa}{bahasa meyalu}
922 \DeclareLanguageAlias {Bahasa}{bahasa}
923 \DeclareLanguageAlias {bahasai}{bahasa}
924 \DeclareLanguageAlias {bahasam}{bahasa}
925 \DeclareLanguageAlias {Malayalam}{malayalam}
926 \DeclareLanguageAlias {Maldivian}{maldivian}
927 \DeclareLanguageAlias {divehi}{maldivian}
928 \DeclareLanguageAlias {Divehi}{divehi}
929 \DeclareLanguageAlias {Marathi}{marathi}
930 \DeclareLanguageAlias {Mongolian}{mongolian}
931 % \DeclareLanguageAlias {Nko}{nko}
932 \DeclareLanguageAlias {norsk}{norwegian}
933 \DeclareLanguageAlias {Norsk}{norsk}
934 \DeclareLanguageAlias {Norwegian}{norwegian}
935 \DeclareLanguageAlias {nynorsk}{norwegian}
936 \DeclareLanguageAlias {Nynorsk}{nynorsk}
937 \DeclareLanguageAlias {Occitan}{occitan}
938 \DeclareLanguageAlias {lenga d'oc}{occitan}
939 \DeclareLanguageAlias {langue d'oc}{occitan}
```

5 Implementation

```
940 \DeclareLanguageAlias {Piedmontese}{piedmontese}
941 \DeclareLanguageAlias {piemontese}{piedmontese}
942 \DeclareLanguageAlias {Piemontese}{piemontese}
943 \DeclareLanguageAlias {piemonteis}{piedmontese}
944 \DeclareLanguageAlias {Piemonteis}{piemonteis}
945 \DeclareLanguageAlias {Pinyin}{pinyin}
946 \DeclareLanguageAlias {Polish}{polish}
947 \DeclareLanguageDialect{brazil}{portuges}
948 \DeclareLanguageAlias {Brazil}{brazil}
949 \DeclareLanguageAlias {brazilian}{brazil}
950 \DeclareLanguageAlias {Brazilian}{brazilian}
951 \DeclareLanguageAlias {Portuges}{portuges}
952 \DeclareLanguageAlias {portuguese}{portuges}
953 \DeclareLanguageAlias {Portuguese}{portuguese}
954 \DeclareLanguageAlias {Romanian}{romanian}
955 \DeclareLanguageAlias {Romansh}{romansh}
956 \DeclareLanguageAlias {Romansch}{romansh}
957 \DeclareLanguageAlias {Rumantsh}{romansh}
958 \DeclareLanguageAlias {Rumantsch}{romansh}
959 \DeclareLanguageAlias {Romanche}{romansh}
960 \DeclareLanguageAlias {Russian}{russian}
961 \DeclareLanguageAlias {Samin}{samin}
962 \DeclareLanguageAlias {north sami}{samin}
963 \DeclareLanguageAlias {North Sami}{north sami}
964 \DeclareLanguageAlias {northern sami}{north sami}
965 \DeclareLanguageAlias {Northern Sami}{northern sami}
966 \DeclareLanguageAlias {Sanskrit}{sanskrit}
967 \DeclareLanguageAlias {Serbocroatian}{serbocroatian}
968 \DeclareLanguageAlias {serbo-croatian}{serbocroatian}
969 \DeclareLanguageAlias {Serbo-Croatian}{serbocroatian}
970 \DeclareLanguageDialect{serbian}{serbocroatian}
971 \DeclareLanguageAlias {Serbian}{serbian}
972 \DeclareLanguageAlias {serbianc}{serbian}
973 \DeclareLanguageDialect{croatian}{serbocroatian}
974 \DeclareLanguageAlias {Croatian}{croatian}
975 \DeclareLanguageAlias {Slovak}{slovak}
976 \DeclareLanguageAlias {Slovenian}{slovenian}
977 \DeclareLanguageAlias {slovene}{slovenian}
978 \DeclareLanguageAlias {Slovene}{slovene}
979 \DeclareLanguageAlias {Sorbian}{sorbian}
980 \DeclareLanguageDialect{lowersorbian}{sorbian}
981 \DeclareLanguageDialect{uppersorbian}{sorbian}
982 \DeclareLanguageAlias {Lowersorbian}{lowersorbian}
983 \DeclareLanguageAlias {Uppersorbian}{uppersorbian}
984 \DeclareLanguageAlias {lsorbian}{lowersorbian}
985 \DeclareLanguageAlias {usorbian}{uppersorbian}
986 \DeclareLanguageAlias {lower sorbian}{lowersorbian}
987 \DeclareLanguageAlias {upper sorbian}{uppersorbian}
988 \DeclareLanguageAlias {lower Sorbian}{lowersorbian}
```

5 Implementation

```

989 \DeclareLanguageAlias {upper Sorbian}{uppersorbian}
990 \DeclareLanguageAlias {Lower Sorbian}{lowersorbian}
991 \DeclareLanguageAlias {Upper Sorbian}{uppersorbian}
992 \DeclareLanguageAlias {Spanglish}{spanglish}
993 \DeclareLanguageAlias {Spanish}{spanish}
994 \DeclareLanguageAlias {Swedish}{swedish}
995 \DeclareLanguageDialect{swissgerman}{german}
996 \DeclareLanguageDialect{swissfrench}{french}
997 \DeclareLanguageDialect{swissitalian}{italian}
998 \DeclareLanguageDialect{swissromansh}{romansh}
999 \DeclareLanguageAlias {Swissgerman}{swissgerman}
1000 \DeclareLanguageAlias {swiss german}{swissgerman}
1001 \DeclareLanguageAlias {Swiss German}{swissgerman}
1002 \DeclareLanguageAlias {Swissfrench}{swissfrench}
1003 \DeclareLanguageAlias {swiss french}{swissfrench}
1004 \DeclareLanguageAlias {Swiss French}{swissfrench}
1005 \DeclareLanguageAlias {Swissitalian}{swissitalian}
1006 \DeclareLanguageAlias {swiss italian}{swissitalian}
1007 \DeclareLanguageAlias {Swiss Italian}{swissitalian}
1008 \DeclareLanguageAlias {Swissromansh}{swissromansh}
1009 \DeclareLanguageAlias {swiss romansh}{swissromansh}
1010 \DeclareLanguageAlias {Swiss Romansh}{swissromansh}
1011 % this is to be discussed: swiss could also be an alias of french, italian or
1012 % romansh:
1013 \DeclareLanguageAlias {swiss}{swissgerman}
1014 \DeclareLanguageAlias {Swiss}{swiss}
1015 % \DeclareLanguageAlias {Syriac}{syriac}
1016 \DeclareLanguageAlias {Tamil}{tamil}
1017 \DeclareLanguageAlias {Telugu}{telugu}
1018 \DeclareLanguageAlias {Thai}{thai}
1019 \DeclareLanguageAlias {thaicjk}{thai}
1020 \DeclareLanguageAlias {Thaicjk}{thaicjk}
1021 \DeclareLanguageAlias {Tibetan}{tibetan}
1022 \DeclareLanguageAlias {Turkish}{turkish}
1023 \DeclareLanguageAlias {Turkmen}{turkmen}
1024 \DeclareLanguageAlias {Ukrainian}{ukrainian}
1025 \DeclareLanguageAlias {Vietnamese}{vietnamese}
1026 \DeclareLanguageAlias {Welsh}{welsh}
1027
1028 % =====
1029 % OK, we have everything, do we? No, wait: let's load the basic dictionary
1030 % that is part of this package if it is available for the document language
1031 % -----
1032 % load basic dictionary if available
1033 \AtBeginDocument{%
1034   % \@trnslt@if@dialect{\@trnslt@current@language}
1035   %   {%
1036   %     \@trnslt@if@dictionary
1037   %       {translations-basic-dictionary}

```

5 Implementation

```

1038 %      {\@trnslt@language{\@trnslt@current@language}}%
1039 %      {%
1040 %          \@trnslt@load@dictionary@silent@for
1041 %              {translations-basic-dictionary}
1042 %              {\@trnslt@language{\@trnslt@current@language}}%
1043 %      }%
1044 %      {%
1045 %          \@trnslt@load@dictionary@silent@for
1046 %              {translations-basic-dictionary}
1047 %              {\@trnslt@dialect@of{\@trnslt@current@language}}%
1048 %      }
1049 %  }
1050 %  {%
1051 %      \@trnslt@load@dictionary@silent@for
1052 %          {translations-basic-dictionary}
1053 %          {\@trnslt@language{\@trnslt@current@language}}%
1054 %      }%
1055 \@trnslt@load@dictionary@silent@for
1056     {translations-basic-dictionary}
1057     \@trnslt@current@language
1058 }
1059
1060 \endinput
1061
1062 % =====
1063 % HISTORY:
1064 2012/09/30 v0.2beta - first version (as part of the `exsheets' bundle)
1065 2012/10/05 v0.2    - \LoadDictionary and \LoadDictionaryFor added and loads of
1066                    languages defined.
1067 2013/03/10 v0.8    - basic dictionaries for English, German, French and Spanish
1068                    - new command \DeclareDictTranslation
1069 2013/04/04 v0.8a   - bug fix in \DeclareDictTranslation
1070 2013/04/07 v0.9    - slightly improved messages
1071 2013/04/08 v0.9a   - changed fallback warning into info
1072                    - synchronized version number with `exsheets' until now but
1073                    won't any more
1074 2013/06/22 v0.9b   - added Swiss
1075 2013/06/28 v0.10   - declaring aliases of dialects now works as expected
1076                    - declarings dialects of an alias now correctly declares
1077                    the dialect to the correct base language
1078                    - corrected a few erroneous language declarations
1079 2013/07/12 v0.10a  - \GetTranslation gets two-folded fallback: use
1080                    fallback-translation if no translation for the current
1081                    language has been defined; use literal string if /no/
1082                    language is used - this should never happen but /will/
1083                    happen if neither `babel' nor `polyglossia' have been
1084                    loaded, i.e., no language has been chosen /and/ the
1085                    package writer did not provide an English translation
1086 2013/07/16 v1.0    - removed from `exsheets' bundle - `translations' should

```


5 Implementation

```
1087         be a package of it's own
1088     - load basic dictionary automatically if available
1089     - rudimentary check in \LoadDictionary if loaded file is a
1090       dictionary
1091     - new command \PrintDictionaryFor
1092     - redefined conditionals; they still seemed to make
1093       trouble in some cases
1094 2013/08/05 v1.1    - added /loads/ of languages, now the list of babel and
1095                    polyglossia languages hopefully is complete
1096     - a few languages had falsely been declared as dialect
1097       instead of an alias
1098     - added weekday names and month names to basic dictionary
1099     - new command \baselanguage
1100     - new commands \GetLCTranslation, \GetLCTranslationFor,
1101       \GetLCTranslationWarn and \GetLCTranslationForWarn
1102     - load basic dictionary also for dialects and if it
1103       doesn't exist load it for the corresponding base
1104       language instead
1105 2013/09/30 v1.1a   - Bug fix in \NewTranslation und \RenewTranslation
1106 2014/01/10 v1.2    - \ifcurrentlanguage, \ifcurrentbaselanguage
1107     - require cnltx-base
1108     - change the 'no language package' warning into an info
1109     - \ProvideTranslation
1110     - \NewDictTranslation, \RenewDictTranslation,
1111       \ProvideDictTranslation
1112     - translations in dictionaries are provided
1113     - \baselanguagename
1114 2014/01/23 v1.2a   - add \detokenize{} so that translation keys with
1115                    non-ascii chars can safely be used
1116     - fix bug in dialect declaration
1117     - rename \baselanguagename => \baselanguage
1118     - drop earlier \baselanguage (it was and still is
1119       available as \@trnslt@language)
1120 2015/07/09 v1.2b   - add language `korean'
1121 2015/08/29 v1.2c   - fix bug in \@trnslt@save@translation@for
1122 2015/09/06 v1.2d   - add alias `slovene' for `slovenian'
1123     - add user command for \@trnslt@if@translation
1124 2015/11/07 v1.2e   - Some fixes to the French translations in the basic
1125                     dictionary, thanks to Denis Bitouz\'e
1126     - add Macedonian language
1127 2016/04/19 v1.3    - \LoadDictionary first checks if a dictionary for the
1128                     dialect exists and loads it if it does; else it looks
1129                     for a dictionary of the base language and loads that
1130                     instead
1131     - New command \LoadDictionaryForDialect which only loads
1132       the dictionary for a specified dialect -- this doesn't
1133       check if a dictionary for a baselanguage exists
1134 2016/06/02 v1.4    - new commands \declaretranslation, \newtranslation,
1135                     \renewtranslation and \providetranslation which can be
```

Bibliography

- [Bra16] Johannes BRAAMS, current maintainer: Javier BEZOS.
babel. version 3.9q, Feb. 24, 2016.
URL: <http://mirror.ctan.org/macros/latex/required/babel/>.
- [Cha15] François CHARETTE, current maintainer: Arthur REUTENAUER.
polyglossia. version 1.42.0, Aug. 6, 2015.
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	G	\NewTranslation 4, 24, 41
babel (package) 1 f., 5–8, 16, 18	\GetLCTranslation 5	\newtranslation 4, 24 f., 41
\baselanguage 6 f., 22, 30, 41	\GetLCTranslationFor 5	\NewTranslationFallback 4, 25
BEZOS, Javier 1	\GetLCTranslationForWarn 5	\newtranslationfallback 4, 25
BRAAMS, Johannes 1	\GetLCTranslationWarn 5	NIEDERBERGER, Clemens 2
	\GetTranslation 2, 5, 7 ff., 27, 40	
C	\GetTranslationFor 5, 7 f., 27	P
CHARETTE, François 1	\GetTranslationForWarn 5	polyglossia (package) 1 f., 8, 16, 18
cnltx (bundle) 2	\GetTranslationWarn 5	\PrintDictionaryFor 6
cnltx-base (package) 2		\ProvideDictionaryFor 6, 10 f., 31
	I	\ProvideDictTranslation 6, 10 f., 32, 41
D	\ifcurrentbaselanguage 7	\ProvideTranslation 4, 24, 41
\DeclareDictTranslation 6, 11, 32, 40	\ifcurrentlanguage 7	\providetranslation 4, 24 f., 41
\DeclareLanguage 3, 6, 10, 21, 33 ff.	\IfTranslation 5, 7, 26	\ProvideTranslationFallback 4, 25
\DeclareLanguageAlias 3, 23, 33, 35–39	K	\providetranslationfallback 4, 25
\DeclareLanguageDialect 3, 21, 36–39	KOHM, Markus 2	
\DeclareTranslation 4 f., 8 f., 24, 28	KOMA-Script (bundle) 2	
\declaretranslation 5, 24, 41	L	R
\DeclareTranslationFallback 4 f., 8 f., 25	\LoadDictionary 6, 10, 29, 40 f.	\RenewDictTranslation 6, 11
\declaretranslationfallback 5, 24 f.	\LoadDictionaryFor 6, 10, 29, 40	\RenewTranslation 4, 24, 41
	\LoadDictionaryForDialect 10	\renewtranslation 4, 24 f., 41
	LPPL 2	\RenewTranslationFallback 4, 25
	N	\renewtranslationfallback 4, 25
	\NewDictTranslation 6, 11	REUTENAUER, Arthur 1

INDEX

S

`\SaveTranslation` 6 f., 29
`\SaveTranslationFor` 6 f., 29

`scrfile` (package) 2

T

`TANTAU`, Till 1
`translator` (package) 1 f.