1 The Latin language

The file `latin.dtx`[1]; defines all the language-specific macros for the Latin language including the classical spelling and hyphenation.

Without modifiers or attributes, this file describes the modern usage of Latin; with this we mean the kind of Latin that is used as an official language in the State of Vatican City, and in the teaching of Latin in modern schools. This file description language accepts three modifiers, in order to be adapted to other situations:

0. Modern spelling and hypheenation; this is the default setting;

1. Modern spelling with prosodic marks: this is to be used mainly in modern grammars and dictionaries;

2. Medieval spelling and hyphenation: in this spelling the lower and upper case alphabets become the following:

   a æ b c d e f g h i k l m n o œ p q r s t u x y z
   A Æ B C D E F G H I K L M N O Œ P Q R S T V X Y Z

   It must be noted that this file description language does not make any spelling correction in order tu use only 'u' in lower case and only 'V' in upper case: if the input text is wrongly typed in, it remains as such; this means it's the typesetter's responsibility to correctly input the source text to be typeset; in spite of this, when the transformation from lower to upper case is performed (such as, for example, while typesetting headers with some classes) the correct capitalization is performed and 'u' is capitalized to 'V'; the reverse takes place when transforming to lowercase.

3. Classical Latin and classical hyphenation; the spelling is similar to the medieval one, except for the ligatures æ, Æ, œ and Œ that are missing; again it is the typesetter's responsibility to input the text to be typeset in a correct way; while the hyphenation patterns for modern and medieval Latin form a non contrasting set of patterns, this is impossible with classical Latin and a different pattern set has been prepared as explained below.

4. Ecclesiastic Latin is a spelling variety of modern Latin; at the moment of writing this documentation, it is taken care by an external module `ecclesiastic.sty`, but it is under study the extension of this language description file in order to include also the ecclesiastic variety. This special spelling of modern Latin is used in liturgical and devotional books of the Roman Catholic clergy where ligatures æ and œ are widely used and where acute accents are used in order to mark the tonic vowel of those words where the penultimate syllable is not tonic; this allows clergy with different backgrounds and mother languages to recite the prayers with the same rhythmic scheme.

[1] The file described in this section has version number v.3.0 and was last revised on 2014/06/01. The original author is Claudio Beccari.

`babel` is not directly concerned with hyphenation; nevertheless in this case it is necessary to switch the hyphenation rules to those of another language, `classiclatin`, that must be available while creating the format files so that this language description file can chose between the modern and medieval hyphenation rules or the classical ones.

In summary modern and medieval Latin hyphenation rules are similar to those of current Italian, except that the patterns must include also some consonant groups that do not exist in Italian; furthermore with medieval spelling there are many more vocalic groups than in Italian, because of the widespread use of the letter 'u' that often actually plays the rôle of a consonant.

The rules for classical Latin are taken form Raffaello Farina and Nino Marinone's guide *Metodologia* published by Società Editrice Internazionale, Torino, 1979. In spite of the publication date of this guide the hyphenation rules did not change in the meanwhile, since the Classical Latin Language is sort of frozen during the past, say, twenty centuries. Yes, it is true that the original writings in Classical Latin were not hyphenated, and the hyphenation rules were established at the beginning of the Renaissance, when mechanical typography required suitable justification. Therefore such rules are sort of a compromise between the procedures used by the various proto-typographers and the scholars' of these disciplines. Nevertheless the rules given by Farina and Marinone are generally shared among modern scholars and are sort of official at both universities of Torino and Vercelli; in the latter University they are adopted by the working group that founded the DigilibLT laboratory; I acknowledge their contribution in letting me become aware of the problem, and therefore to create this update by adding the classical Latin support for the `babel` package, with its special hyphenation rules.

In order to implement the above described four styles of typesetting three modifiers/attributes have to be defined: `medieval`, `withprosodicmarks` and `classic`. They can be used in any order so that many combinations are possible, although, generally speaking, just the four described ones have a real meaning.

Among these modifiers only `withprosodicmarks` can be turned on and off; the others are sort of global; for example, once `classic` is specified, there is no easy way to revert to modern spelling and hypehenation; the same holds true for the other modifiers, except `withprosodicmarks`.

The typesetting style `withprosodicmarks` is defined in order to use special shorthands for inserting breves and macrons when typesetting grammars, dictionaries, teaching texts, and the like, where prosodic marks are important for the complete information on the words or the verses. The shorthands, listed in table 1 and described in subsection 2.1, may interfere with other packages; therefore by default this style is turned off and no interference is introduced. If this style is used and interference is experienced, there are special commands for turning on and off its specific shorthand commands.

For what concerns `babel` and typesetting with LaTeX, the differences between the three spelling styles reveal themselves in the strings used to name, for example, the "Preface" that becomes "Praefatio" or "Præfatio" respectively. Hyphenation rules are also different, but the hyphenation pattern file `hyph-la.tex` takes care of the modern and medieval versions of the language, while `hyph-llac.tex` takes

care of the classical hyphenation. The user should not attempt to modify these files, because they are not dealt with by `babel`; they are used only during the creation of format files. If some errors or modifications are required or suggested, it is necessary to ask their maintainer.

The name strings for chapters, figures, tables, et cetera, have been suggested by prof. Raffaella Tabacco, a latinist of the University of Vercelli, Italy, to whom we address our warmest thanks. The names suggested by Krzysztof Konrad Żelechowski, when different, are used as the names for the medieval variety, since he made a word and spelling choice more suited for this variety.

For this language some shorthands are defined according to table 1; all of them are supposed to work with all spelling styles, except where the opposite is explicitly stated.

`^i`	inserts the breve accent as ĭ; valid also for the other lowercase vowels, but it does not operate on the medieval ligatures æ and œ because they are long wowels.
`=a`	inserts the macron accent as ā; valid also for the other lowercase vowels, but it does not operate on the medieval ligatures æ and œ because they are naturally long.
`"`	inserts a compound word mark where hyphenation is legal; the next character must not be either a nonletter token or an accented letter (for foreign names).
`"|`	same as above, but operates also when the next token is not a letter or it is an accented character.

Table 1: Shorthands defined for the Latin language. The characters `^` and `=` are active only when the language attribute `withprosodicmarks` has been declared, otherwise they are disabled; see subsection 2.1 at page 7 for more details.

2 The code

The macro `\LdfInit` takes care of preventing that this file is loaded more than once, checking the category code of the `@` sign, etc.

```
1 \LdfInit{latin}{captionslatin}
```

When this file is read as an option, i.e. by the `\usepackage` command, `latin` will be an 'unknown' language in which case we have to make it known. So we check for the existence of `\l@latin` to see whether we have to do something here.

```
2 \ifx\l@latin\@undefined
3     \@nopatterns{Latin}
4     \adddialect\l@latin0\fi
```

Now we declare the `medieval` language attribute.

```
\bbl@declare@ttribute{latin}{medieval}{%
  \addto\captionslatin{\def\prefacename{Pr{\ae}fatio}}%
  \expandafter\addto\expandafter\extraslatin
  \expandafter{\extrasmedievallatin}%
  }
```

The typesetting style `withprosodicmarks` is defined here:

```
\bbl@declare@ttribute{latin}{withprosodicmarks}{%
  \expandafter\addto\expandafter\extraslatin
  \expandafter{\extraswithprosodicmarks}%
  }
```

The 'classic' typesetting style is described here: notice that the general typesetting rules are identical with those of modern Latin, the default one, but the hyphenation rules are different; therefore we just change the meaning of counter `\l@latin` so that it points to the language counter of classical Latin. In order to avoid a fatal error we test if language `classiclatin` has its patterns already loaded, otherwise we issue a warning and keep going with the default modern hyphenation

```
\bbl@declare@ttribute{latin}{classic}{%
  \expandafter\addto\expandafter\extraslatin
  \expandafter{\extrasclassic}%
  }
\ifx\l@classiclatin\undefined
  \let\l@classiclatin\l@latin
  \PackageWarningNoLine{babel}{%
  Attention: hyphenation patterns for language\MessageBreak
  classiclatin have not been loaded\MessageBreak
  I go on using the modern Latin hyphenation patterns.\MessageBreak
  Please, load the suitable patterns or upgrade your TeX distribution}
\fi
\addto\extrasclassic{\let\l@latin\l@classiclatin}
```

It must be remembered that the `medieval` and the `withprosodicmarks` styles may be used together. They are also compatible with the `classic` attribute, but it would be unusual to typeset classical Latin with some medieval features and/or with the prosodic marks that were unknown twenty centuries ago.

The next step consists in defining commands to switch to (and from) the Latin language[2].

`\captionslatin` The macro `\captionslatin` defines all strings used in the four standard document classes provided with LaTeX.

```
\@namedef{captionslatin}{%
  \def\prefacename{Praefatio}%
  \def\refname{Conspectus librorum}%
  \def\abstractname{Summarium}%
  \def\bibname{Conspectus librorum}%
  \def\chaptername{Caput}%
```

[2] Most of these names were kindly suggested by Raffaella Tabacco.

```
\def\appendixname{Additamentum}%
\def\contentsname{Index}%
\def\listfigurename{Conspectus descriptionum}%
\def\listtablename{Conspectus tabularum}%
\def\indexname{Index rerum notabilium}%
\def\figurename{Descriptio}%
\def\tablename{Tabula}%
\def\partname{Pars}%
\def\enclname{Adduntur}%   Or " Additur" ? Or simply Add.?
\def\ccname{Exemplar}%     Use the recipient's dative
\def\headtoname{\ignorespaces}% Use the recipient's dative
\def\pagename{Charta}%
\def\seename{cfr.}%
\def\alsoname{cfr.}% R.Tabacco never saw "cfr. atque" or similar forms
\def\proofname{Demonstratio}%
\def\glossaryname{Glossarium}%
}
```

In the above definitions there are some points that might change in the future or that require a minimum of attention from the typesetter.

1. the `\enclname` is translated by a passive verb, that literally means "(they) are being added"; if just one enclosure is joined to the document, the plural passive is not suited any more; nevertheless a generic plural passive might be incorrect but suited for most circumstances. On the opposite "Additur", the corresponding singular passive, might be more correct with one enclosure and less suited in general: what about the abbreviation "Add." that works in both cases, but certainly is less elegant?

2. The `\headtoname` is empty and gobbles the possible following space; in practice the typesetter should use the dative of the recipient's name; since nowadays not all such names can be translated into Latin, they might result indeclinable. The clever use of a dative appellative by the typesetter such as "Domino" or "Dominae" might solve the problem, but the header might get too impressive. The typesetter must make a decision on his own.

3. The same holds true for the copy recipient's name in the "Cc" field of `\ccname`.

`\datelatin` The macro `\datelatin` redefines the command `\today` to produce Latin dates; the choice of faked small caps Latin numerals is arbitrary and may be changed in the future. For medieval and classic Latin the spelling of 'Novembris' should be *Nouembris*. This is taken care of by using a control sequence which can be redefined when the attribute 'medieval' and/or 'classic' is selected.

```
\addto\extraslatin{\def\november{Novembris}}
\addto\extrasmedievallatin{\def\november{Nouembris}}
\addto\extrasclassiclatin{\def\november{Nouembris}}
%
\def\datelatin{%
```

```
\def\today{%
  {\check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont
    \uppercase\expandafter{\romannumeral\day}}~\ifcase\month\or
  Ianuarii\or Februarii\or Martii\or Aprilis\or Maii\or Iunii\or
  Iulii\or Augusti\or Septembris\or Octobris\or \november\or
  Decembris\fi
  \space{\uppercase\expandafter{\romannumeral\year}}}}
```

`\latinhyphenmins` The Latin hyphenation patterns can be used with both `\lefthyphenmin` and `\righthyphenmin` set to 2.

```
\providehyphenmins{\CurrentOption}{\tw@\tw@}
```

`\extraslatin` `\noextraslatin` For Latin the `\clubpenalty`, `\widowpenalty` are set to rather high values and `\finalhyphendemerits` is set to such a high value that hyphenation is almost prohibited between the last two lines of a paragraph.

```
\addto\extraslatin{%
  \babel@savevariable\clubpenalty
  \babel@savevariable\@clubpenalty
  \babel@savevariable\widowpenalty
  \clubpenalty3000\@clubpenalty3000\widowpenalty3000}
```

Hopefully never ever break a word between the last two lines of a paragraph in latin texts.

```
\addto\extraslatin{%
  \babel@savevariable\finalhyphendemerits
  \finalhyphendemerits50000000}
```

With medieval and classic Latin we need the suitable correspondence between upper case V and lower case u, since in that spelling there is only one sign, and the u shape is the (uncial) version of the capital V. Everything else is identical with Latin.

```
\addto\extrasmedievallatin{%
\babel@savevariable{\lccode‘\V}%
\babel@savevariable{\uccode‘\u}%
\lccode‘\V=‘\u \uccode‘\u=‘\V}
\addto\extrasclassiclatin{%
\babel@savevariable{\lccode‘\V}%
\babel@savevariable{\uccode‘\u}%
\lccode‘\V=‘\u \uccode‘\u=‘\V}
```

`\SetLatinLigatures` We need also the lccodes for æ and œ; since they occupy different positions in the OT1 TEX-fontencoding compared to the T1 one, we must save the lc- and the uccodes for both encodings, but we specify the new lc- and uccodes separately as it appears natural not to change encoding while typesetting the same language. The encoding is assumed to be set before starting to use the Latin language, so that if Latin is the default language, the font encoding must be chosen before requiring the babel package with the `latin` option, in any case before any `\selectlanguage` or `\foreignlanguage` command.

All this fuss is made in order to allow the use of the medieval ligatures æ and œ while typesetting with the medieval spelling; I have my doubts that the medieval spelling should be used at all in modern books, reports, and the like; the uncial 'u' shape of the lower case 'v' and the above ligatures were fancy styles of the copyists who were able to write faster with those rounded glyphs; with typesetting there is no question of handling a quill penn... Moreover in medieval times it was very frequent to close such ligatures to the corresponding sound, therefore instead of writing æ or œ they would often simply write 'e'. Since my (CB) opinion may be wrong, I managed to set up the instruments and it is up to the typesetter to use them or not.

```
\addto\extrasmedievallatin{%
  \babel@savevariable{\lccode‘\^^e6}% T1   \ae
  \babel@savevariable{\uccode‘\^^e6}% T1   \ae
  \babel@savevariable{\lccode‘\^^c6}% T1   \AE
  \babel@savevariable{\lccode‘\^^f7}% T1   \oe
  \babel@savevariable{\uccode‘\^^f7}% T1   \OE
  \babel@savevariable{\lccode‘\^^d7}% T1   \OE
  \babel@savevariable{\lccode‘\^^1a}% OT1  \ae
  \babel@savevariable{\uccode‘\^^1a}% OT1  \ae
  \babel@savevariable{\lccode‘\^^1d}% OT1  \AE
  \babel@savevariable{\lccode‘\^^1b}% OT1  \oe
  \babel@savevariable{\uccode‘\^^1b}% OT1  \OE
  \babel@savevariable{\lccode‘\^^1e}% OT1  \OE
  \SetLatinLigatures}

\providecommand\SetLatinLigatures{%
  \def\@tempA{T1}\ifx\@tempA\f@encoding
    \catcode‘\^^e6=11 \lccode‘\^^e6=‘\^^e6 \uccode‘\^^e6=‘\^^c6 % \ae
    \catcode‘\^^c6=11 \lccode‘\^^c6=‘\^^e6 % \AE
    \catcode‘\^^f7=11 \lccode‘\^^f7=‘\^^f7 \uccode‘\^^f7=‘\^^d7 % \oe
    \catcode‘\^^d7=11 \lccode‘\^^d7=‘\^^f7 % \OE
  \else
    \catcode‘\^^1a=11 \lccode‘\^^1a=‘\^^1a \uccode‘\^^1a=‘\^^1d % \ae
    \catcode‘\^^1d=11 \lccode‘\^^1d=‘\^^1a % \AE
    \catcode‘\^^1b=11 \lccode‘\^^1b=‘\^^1b \uccode‘\^^1b=‘\^^1e % \oe
    \catcode‘\^^1e=11 \lccode‘\^^1e=‘\^^1b % \OE
  \fi
  \let\@tempA\@undefined
  }
```

With the above definitions we are sure that `\MakeUppercase` works properly and `\MakeUppercase{C{\ae}sar}` correctly yields "CÆSAR"; correspondingly `\MakeUppercase{Heluetia}` correctly yields "HELVETIA".

2.1 Latin shorthands

For writing dictionaries or school texts (in modern spelling only) we defined a language attribute or typesetting style, such that a couple of other active characters

are defined: ^ for marking a vowel with the breve sign, and = for marking a vowel with the macron sign. Please take notice that neither the OT1 font encoding, nor the T1 one for most vowels, contain directly the marked vowels, therefore hyphenation of words containing these "accents" may become problematic; for this reason the above active characters not only introduce the required accent, but also an unbreakable zero skip that in practice does not introduce a discretionary break, but allows breaks in the rest of the word.

It must be remarked that the active characters ^ and = may have other meanings in other contexts. For example, besides math, the equals sign is used by the graphic extensions for specifying keyword options for handling the graphic elements to be included in the document. At the same time, as mentioned in the previous paragraph, diacritical marking in Latin is used only for typesetting certain kinds of document, such as grammars and dictionaries. It is reasonable that the breve and macron active characters are turned on and off at will; by default they are off if the attribute `withprosodicmarks` has not been set.

`\ProsodicMarksOn` `\ProsodicMarksOff` We begin by adding to the normal typesetting style the definitions of the new commands `\ProsodicMarksOn` and `\ProsodicMarksOff` that should produce error messages when the this style is not declared:

```
\addto\extraslatin{\def\ProsodicMarksOn{%
\GenericError{(latin)\@spaces\@spaces\@spaces\@spaces}%
            {Latin language error: \string\ProsodicMarksOn\space
            is defined by setting the\MessageBreak
            language attribute to 'withprosodicmarks'\MessageBreak
            If you continue you are likely to encounter\MessageBreak
            fatal errors that I can't recover}%
            {See the Latin language description in the babel
            documentation for explanation}{\@ehd}}}
\addto\extraslatin{\let\ProsodicMarksOff\relax}
```

Next we temporarily set the caret and the equals sign to active characters so that they can receive their definitions:

```
\catcode`\= \active
\catcode`\^ \active
```

and we add the necessary declarations to the macros that are being activated when the Latin language and its typesetting styles are declared:

```
\addto\extraslatin{\languageshorthands{latin}}%
\addto\extraswithprosodicmarks{\bbl@activate{^}}%
\addto\extraswithprosodicmarks{\bbl@activate{=}}%
\addto\noextraswithprosodicmarks{\bbl@deactivate{^}}%
\addto\noextraswithprosodicmarks{\bbl@deactivate{=}}%
\addto\extraswithprosodicmarks{\ProsodicMarks}
```

`\ProsodicMarks` Next we define the macros for the active characters

```
\def\ProsodicMarks{%
\def\ProsodicMarksOn{\catcode`\^ 13\catcode`\= 13\relax}%
\def\ProsodicMarksOff{\catcode`\^ 7\catcode`\= 12\relax}%
}
```

Notice that with the above redefinitions of the commands `\ProsodicMarksOn` and `\ProsodicMarksOff`, the operation of the newly defined shorthands may be switched on and off at will, so that even if a picture has to be inserted in the document by means of the commands and keyword options of the `graphicx` package, it suffices to switch them off before invoking the picture including command, and switched on again afterwards; or, even better, since the picture very likely is being inserted within a `figure` environment, it suffices to switch them off within the environment, being conscious that their deactivation remains local to the environment.

```
\initiate@active@char{^}%
\initiate@active@char{=}%
\declare@shorthand{latin}{^a}{%
    \textormath{\u{a}\bbl@allowhyphens}{\hat{a}}}%
\declare@shorthand{latin}{^e}{%
    \textormath{\u{e}\bbl@allowhyphens}{\hat{e}}}%
\declare@shorthand{latin}{^i}{%
    \textormath{\u{\i}\bbl@allowhyphens}{\hat{\imath}}}%
\declare@shorthand{latin}{^o}{%
    \textormath{\u{o}\bbl@allowhyphens}{\hat{o}}}%
\declare@shorthand{latin}{^u}{%
    \textormath{\u{u}\bbl@allowhyphens}{\hat{u}}}%
%
\declare@shorthand{latin}{=a}{%
    \textormath{\={a}\bbl@allowhyphens}{\bar{a}}}%
\declare@shorthand{latin}{=e}{%
    \textormath{\={e}\bbl@allowhyphens}{\bar{e}}}%
\declare@shorthand{latin}{=i}{%
    \textormath{\={\i}\bbl@allowhyphens}{\bar{\imath}}}%
\declare@shorthand{latin}{=o}{%
    \textormath{\={o}\bbl@allowhyphens}{\bar{o}}}%
\declare@shorthand{latin}{=u}{%
    \textormath{\={u}\bbl@allowhyphens}{\bar{u}}}%
```

Notice that the short hand definitions are given only for lower case letters; it would not be difficult to extend the set of definitions to upper case letters, but it appears of very little use in view of the kinds of document where prosodic marks are supposed to be used. Nevertheless in those rare cases when it's required to set some uppercase letters with their prosodic marks, it is always possible to use the standard LaTeX commands such as `\u{I}` for typesetting Ĭ, or `\={A}` for typesetting Ā.

Finally we restore the caret and equals sign initial default category codes

```
\catcode`\= 12\relax
\catcode`\^ 7\relax
```

so as to avoid conflicts with other packages or other babel options.

`\LatinMarksOn` `\LatinMarksOff`

The following commands remain defined for backwards compatibility, but hey are obsolete and should not be used.

```
\addto\extraslatin{\def\LatinMarksOn{\shorthandon{^}\shorthandon{=}}}
```

```
\addto\extraslatin{\def\LatinMarksOff{\shorthandoff{^}\shorthandoff{=}}}
%\addto\extraslatin{\LatinMarksOff}
```

It must be understood that by using the above prosodic marks, line breaking is somewhat impeached; since such prosodic marks are used almost exclusively in dictionaries, grammars, and poems (only in school textbooks), this shouldn't be of any importance for what concerns the quality of typesetting.

2.2 Etymological hyphenation

In order to deal in a clean way with prefixes and compound words to be divided etymologically, the active character " is given a special definition so as to behave as a discretionary break with hyphenation allowed after it. You may see this sign as a substitute for a "compound word mark".

This is particularly useful with classical Latin because this language requires etymological hyphenation; patterns were created with etymological hyphenation in mind, but even if for certain prefixes or suffixes it works pretty well, it does not for certain other ones. For example the word 'redire' should be hyphenated as 'red-i-re'; but there are plenty of other words starting with the string 'red' that does not play the rôle of a prefix; therefore it should be necessary to extract all the Latin words where 'red' is a prefix, and enter the suitable patterns to hyphenate correctly only those words; in the simple case of the verb 'redire' it would be necessary to enter all the words that belong to the conjugation of this verb and the declination of the present, future and past participles in masculine, feminine and neutral forms; the same problem takes place with prefix 'trans' where sometimes is a real prefix, such as in 'transire' (hyphenation trans-i-re), and sometimes is modified with the absorption of an initial 's' of the suffix, such as in 'transubstantialis' (classical hyphenation: tran-subs-tan-tia-lis).

Obviously this task would render the pattern file enormous, and mostly useless. The active character " solves the problem for isolated instances, while the hyphenations exception lists of the only forms actually used in a specific document, would do the rest.

Most of the code for dealing with the active " is already contained in the core of babel, but we are going to use it as a single character shorthand for Latin.

```
\initiate@active@char{"}%
\addto\extraslatin{\bbl@activate{"}%
}
```

A temporary macro is defined so as to take different actions in math mode and text mode: specifically in the former case the macro inserts a double quote as it should in math mode, otherwise another delayed macro comes into action.

```
\declare@shorthand{latin}{"}{%
  \ifmmode
    \def\lt@@next{''}%
  \else
    \def\lt@@next{\futurelet\lt@temp\lt@cwm}%
  \fi
```

```
  \lt@@next
}%
```

In text mode the `\lt@@next` control sequence is such that upon its execution a temporary variable `\lt@temp` is made equivalent to the next token in the input list without actually removing it. Such temporary token is then tested by the macro `\lt@cwm` and if it is found to be a letter token, then it introduces a compound word separator control sequence `\lt@allowhyphens` whose expansion introduces a discretionary hyphen and an unbreakable space; in case the token is not a letter, it is tested against the definitions of `\ae` and `\oe`, and if the test is true than such definitions are treated as letters (as they actually are), otherwise the token is tested again to find if it is the character `|`, in which case it is gobbled and a discretionary break is introduced.

```
\def\lt@allowhyphens{\bbl@allowhyphens\discretionary{-}{}{}\bbl@allowhyphens}
\newcommand*{\lt@cwm}{\let\lt@n@xt\relax
  \ifcat\noexpand\lt@temp a%
    \let\lt@n@xt\lt@allowhyphens
  \else
    \ifx\lt@temp\ae
        \let\lt@n@xt\lt@allowhyphens
    \else
        \ifx\lt@temp\oe
           \let\lt@n@xt\lt@allowhyphens
        \else
           \if\noexpand\lt@temp\string|%
              \def\lt@n@xt{\lt@allowhyphens\@gobble}%
           \fi
        \fi
    \fi
  \fi
  \lt@n@xt}%
```

Attention: the ligature commands `\ae` and `\oe` are detected correctly if they are not included within a group. In facts an input such as `super"{\ae}quitas`[3] gets wrongly hyphenated as `su-pe-ræ-qui-tas` while `super"\ae{}quitas`, that uses an empty group to terminate the `\ae` control sequence, gets correctly hyphenated as `su-per-æ-qui-tas`. If one prefers to close the `\ae` or `\oe` ligature commands within a group, then it is necessary to use the alternate etymological hyphenation command `"|` as in `super"|{\ae}quitas` in order to get `su-per-æ-qui-tas`. .

The macro `\ldf@finish` takes care of looking for a configuration file, setting the main language to be switched on at `\begin{document}` and resetting the category code of `@` to its original value.

```
\ldf@finish{latin}
```

[3] This word does not exist in "regular" Latin, and it is used just as an example.