Babel support for the Latin language

Claudio Beccari            Keno Wehr

v. 4.0       27th June 2021

Abstract
This manual documents the babel-latin package, which defines all language-specific macros for the babel languages latin, classiclatin, medievallatin, and ecclesiasticlatin. These languages are usable with pdfLaTeX, XeLaTeX, and LuaLaTeX. The latin language is even usable with plain \TeX (with some restrictions).

See section 2.5 on how to update from outdated modifiers and the ecclesiastic package.

Contents

1 Language variants 2

2 Modifiers 4

2.1 The letter \textit{j} 4

2.2 Case of month names 4

2.3 Shorthands for prosodic marks 5

2.4 Ecclesiastic footnotes 6

2.5 Legacy modifiers 6

3 Hyphenation 6

4 Shorthands 7

5 Incompatibilities with other packages 8

5.1 unicode-math 8

5.2 Lua\TeX 8

5.3 babel-turkish 8

5.4 babel-esperanto, babel-kurmanji, and babel-slovak 9

6 Plain \TeX 9

Latin has been the most important language of European intellectual life for a long time. Throughout the centuries, many different styles of Latin have been in use concerning wording, spelling, punctuation, and hyphenation. The typographical conventions of an edition of a Latin classic are quite different from those of a liturgical book, even if both have been printed in the 20th century. And even the same Latin text may look quite differently depending on the preferences of the editor and the typographical customs of his country. Latin is supranational, but its typography is not.

To fit all needs, the babel-latin package defines four different language variants of Latin, i.e., four different babel languages. Table 1 shows some differences between the language variants. It is no problem to use different variants of Latin within the same document. If you need classical and modern Latin, just say

\usepackage[classiclatin,latin]{babel}

and switch the language using the commands described in the babel manual.

**The latin language – modern Latin**  This language variant is intended for 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. Typically, the following alphabet is used:

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

**The classiclatin language – classical Latin**  This language variant is intended for typesetting Latin texts more or less according to the ancient usage of Latin. However, the use of lower-case letters, which are not of ancient origin, is not excluded. The following alphabet is used:

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

Note that 'V' corresponds to 'u' in lower case. This habit came up in the Middle Ages and is still in use in many text editions. It must be noted that babel-latin does not make any spelling correction in order to 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 document classes) the correct capitalization is performed and 'u' is capitalized to 'V'; the reverse takes place when transforming to lower case.

**The medievallatin language – medieval/humanist Latin**  The spelling is similar to the classical one, but the ligatures æ, Æ, œ, and Œ are used for the respective (former) diphthongs. Again, it is the typesetter's responsibility to input the text to be typeset in a correct way. The following alphabet is used:

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

As far as the current maintainer can judge it, the consequent use of 'æ' and 'œ' ligatures came up in 15th century manuscripts in Italy. So this language variant rather reflects the Latin of the humanist/Renaissance period than that of the Middle Ages. However, we stick to the medieval name chosen in earlier versions of babel-latin.

**The ecclesiasticlatin language – ecclesiastic Latin**  Ecclesiastic Latin is a spelling variety of modern Latin, which is used above all in liturgical books of the Roman Catholic Church, where the ligatures æ and œ are widely used and where acute accents are used in order to mark the tonic vowel of words with more than two syllables to make sure the correct stress. The following alphabet is used:

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

This language variant also contains a certain degree of “Frenchization” of spaces around some punctuation marks and guillemets: 1/12 of a quad is inserted before '!', '?', '.', ',',
'»', and '›' as well as after '«' and '‹'. The spacing of guillemets does not work with pdfTEX except when using the shorthands "< and "> (see section 4).

For what concerns babel and typesetting with TeX, the differences between the language variants reveal themselves in the strings used to name, for example, the "Preface", that becomes "Praefatio" or "Præfatio", respectively. Hyphenation rules are also different, cf. section 3.

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.

2 Modifiers

The four language variants described above do not cover all variations of Latin typography. Additionally there are several modifiers: usej, lowercasemonth, withprosodicmarks, and ecclesiasticfootnotes. The meaning of these modifiers is explained below.

To apply a modifier you have to append it (prefixed with a dot) to the language name when loading babel:

```
\usepackage[ecclesiasticlatin.lowercasemonth]{babel}
```

If you need two modifiers or more, just concatenate them in arbitrary order:

```
\usepackage[latin.usej.withprosodicmarks]{babel}
```

2.1 The letter j

The letter j is not of ancient origin. In early modern times, it was used to distinguish the consonant i from the vocalic i. In liturgical books j was in use until the 1960s. Nowadays, the use of j has disappeared from most Latin publications. This is why babel-latin does not use j in predefined terms by default. Use the usej modifier if you prefer Januarii and Maji to Ianuarii and Mai.

2.2 Case of month names

Traditionally, Latin month names are capitalized: Ianuarii, Februarii, Martii, ... (We state the genitive forms here as this is what we need for Latin dates.) So babel-latin capitalizes the month names for all four language variants. However, in recent liturgical books month names are written in lower case (as in Romance languages). Use the lowercasemonth modifier if you prefer not to capitalize the month names printed by the \today command: Ianuarii, februarii, martii, ...
2.3 Shorthands for prosodic marks

Textbooks, grammars, and dictionaries often use letters with prosodic marks (macrons and breves) like 'ā' and 'ă' to mark long and short vowels. On modern systems, the required characters can be input directly thanks to Unicode. For backwards compatibility and as an perhaps more comfortable alternative even today, babel-latin provides shorthands for prosodic marks if you load the language with the withprosodicmarks modifier.

Note that these shorthands may interfere with other packages. The active character used for macrons will cause problems with commands using key=value interfaces, such as the command \includegraphics[\text{scale=2}]{{...}). Therefore, the shorthands are disabled by default. You have to use dedicated commands to turn them on and off. Use \ProsodicMarksOn to enable them an \ProsodicMarksOff to disable them again. To get "Găllia ĕst omnīs divisī in pārtēs trēs", type:

\ProsodicMarksOn
G^all^i^a ^est ^omn^is d=iv=is^a ^in p^art=es tr=es
\ProsodicMarksOff

The following shorthands are available:

=a for ā (a with macron), also available for ē, ĩ, ŏ, ŭ, and ȳ

=A for Ā (A with macron), also available for Ė, Ī, Ŭ, Ģ, and Ŷ. Note that a macron above the letter V is only displayed if your font supports the Unicode character U+0304 (combining macron).

=ae for ae (ae diphthong with macron, for latin and classiclatin) or æ (ae ligature with macron, for medievallatin and ecclesiasticlatin), respectively; also available for aē, eē, and õē/œ̄. Note that macrons above diphthongs are only displayed if your font supports the Unicode character U+035E (combining double macron), which always requires X\LaTeX{} or Lua\LaTeX{}.

=Æ for AE (AE diphthong with macron, for latin and classiclatin) or Ė (AE ligature with macron, for medievallatin and ecclesiasticlatin), respectively; also available for Æu, Eũ, and Œũ/Œê.

=Å for ÅE (ÅE diphthong with macron, for latin and classiclatin) or Å (ÅE ligature with macron, for medievallatin and ecclesiasticlatin), respectively; also available for Åu, EU, and Œü/Œé.

^a for ă (a with breve), also available for ē, ĩ, ŏ, ŭ, and ȳ. Note that a breve above the letter y is only displayed if your font supports the Unicode character U+0306 (combining breve).

^Å for ÅA (A with breve), also available for E, Ŕ, Ū, V, and Ŷ. Note that breves above the letters V and Y are only displayed if your font supports the Unicode character U+0306 (combining breve).

\footnote{\text{A good choice for a font supporting the combining double macron might be Libertinus Serif, the font of this manual.}}
Note the incompatibilities described in section 5.

### 2.4 Ecclesiastic footnotes

The ecclesiastic package, an outdated extension of former versions of babel-latin, typeset footnotes with ordinary instead of superior numbers and without indentation.

As many ecclesiastic documents and liturgical books use footnotes that are very similar to the ordinary \LaTeX ones, we do not use this footnote style as default even for the \texttt{ecclesiasticlatin} language variant. But you may use the \texttt{ecclesiasticfootnotes} modifier (with any variant of Latin) if you prefer that footnote style.

Note that this modifier affects the entire document. It can only be applied to the document’s main language.

### 2.5 Legacy modifiers

\texttt{babel-latin} defined only one single \texttt{babel} language up to v. 3.5. Language variants used to be accessible via modifiers. This approach has proved to be disadvantageous concerning compatibility with other language-specific packages like \texttt{biblatex}. That’s why v. 4.0 introduced the \texttt{classiclatin}, \texttt{medievallatin}, and \texttt{ecclesiasticlatin} languages.

The legacy modifiers \texttt{classic}, \texttt{medieval}, and \texttt{ecclesiastic} are still available and backwards compatibility is made sure. However, a warning is issued if you use one of these modifiers. They may be dropped from \texttt{babel-latin} in a future version.

For maximum compatibility, replace

\begin{itemize}
  \item \texttt{\usepackage[latin.classic]{babel}} by \texttt{\usepackage[classiclatin]{babel}},
  \item \texttt{\usepackage[latin.medieval]{babel}} by \texttt{\usepackage[medievallatin]{babel}},
  \item \texttt{\usepackage[latin.ecclesiastic]{babel}} by \texttt{\usepackage[ecclesiasticlatin.ecclesiasticfootnotes,activeacute]{babel}}.
\end{itemize}

The last replacement is also recommended if you have been loading the ecclesiastic package so far. This package is no longer necessary as its functionality is provided by \texttt{babel-latin} now.

### 3 Hyphenation

There are three different sets of hyphenation patterns for Latin, reflecting three different styles of hyphenation: \textit{classical}, \textit{modern}, and \textit{liturgical}. Separate documentation for these hyphenation styles is available on the Internet.\footnote{https://github.com/gregorio-project/hyphen-la/blob/master/doc/README.md#hyphenation-styles} Each of the four Latin language variants has its default hyphenation style as indicated by table 2. Use the \texttt{\babelformatter} command with the \texttt{hyphenrules} option if the default style does not fit your needs.

To typeset a liturgical book in the recent "Solesmes style" say

\begin{verbatim}
\usepackage[\textit{Solesmes style}]{babel} \texttt{\babelformatter}\texttt{\textit{Solesmes style}}
\end{verbatim}
<table>
<thead>
<tr>
<th>Language variant</th>
<th>Hyphenation style</th>
<th>Name of patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>latin</td>
<td>modern</td>
<td>latin</td>
</tr>
<tr>
<td>classiclatin</td>
<td>classical</td>
<td>classiclatin</td>
</tr>
<tr>
<td>medievallatin</td>
<td>modern</td>
<td>latin</td>
</tr>
<tr>
<td>ecclesiasticlatin</td>
<td>modern</td>
<td>latin</td>
</tr>
<tr>
<td>−</td>
<td>liturgical</td>
<td>liturgicallatin</td>
</tr>
</tbody>
</table>

Table 2: Latin hyphenation styles

\usepackage[english]{babel}
\babelprovide[hyphenrules=liturgicallatin]{english}

The typical commands for a Latin text edition in the German-speaking world will be

\usepackage[english]{babel}
\babelprovide[hyphenrules=classical]{english}

Note that the liturgical hyphenation patterns are the default of none of the language variants. To use them, you have to load them explicitly in any case.

4 Shorthands

The following shorthands are available for all variants of Latin. Note that shorthands beginning with ‘ ’ are only available if you load babel with the \texttt{activetext} option.

"< for « (left guillemet)

"> for » (right guillemet)

" If no other shorthand applies, " before any letter character defines an optional break point allowing further break points within the same word (as opposed to the \texttt{-} command).

"| the same as ", but also possible before non-letter characters

‘a for á (a with acute), also available for é, í, ó, ú, ý, æ, and ð

‘A for À (A with acute), also available for Ê, Í, Ó, Ú, Ý, Æ, and ð

The following shorthands are only available for the medievallatin and the ecclesiasticlatin languages. Again, the shorthands beginning with ‘ ‘ only work with babel’s \texttt{activetext} option.

"æ for æ (ae ligature), also available for ð

"Ae for ÀE (AE ligature), also available for Ð

"Æ for ÀE (AE ligature), also available for Ð
Furthermore, there are shorthands for prosodic marks; see section 2.3. Note the incompatibilities described in section 5.

5 Incompatibilities with other packages

5.1 unicode-math

Loading the Latin language together with the activeacute babel option may cause error messages if the unicode-math package is loaded. Do not use activeacute if you need unicode-math, even if Latin is only a secondary language of your document.⁴

5.2 LuaTeX

The " character is made active by babel-latin; its use within the \directlua command will lead to problems (except in the preamble). Switch the shorthand off for such commands:

\shorthandoff{"}
\directlua{tex.print("Salve")}
\shorthandon{"}

You may avoid the shorthand switching by using single instead of double quotes. However, note that this will not work if the activeacute option is used, as ‘ is active in this case as well.

Furthermore, beware of using \directlua commands containing the = character between \ProsodicMarksOn and \ProsodicMarksOff if you load the Latin language with the withprosodicmarks modifier.

5.3 babel-turkish

Both Turkish and Latin (when loaded with the withprosodicmarks modifier) make the = character active. However, babel-latin takes care the active behaviour of this character is only enabled between \ProsodicMarksOn and \ProsodicMarksOff to avoid conflicts with packages using key=value interfaces.

If you need Latin with prosodic shorthands and Turkish with active = character in one document, you have to say \shorthandon{=} before the first occurrence of = in each Turkish text part.

5.4 babel-esperanto, babel-kurmanji, and babel-slovak

Esperanto, Kurmanji, Slovak, and Latin (when loaded with the \withprosodicmarks modifier) make the ^ character active. However, babel-latin takes care the active behaviour of this character is only enabled between \ProsodicMarksOn and \ProsodicMarksOff to avoid conflicts with \TeX’s ^^xx convention.

If you need Latin with prosodic shorthands and Esperanto/Kurmanji/Slovak with active ^ character in one document, you have to say \shorthand{^} before the first occurrence of ^ in each Esperanto/Kurmanji/Slovak text part.

6 Plain \TeX

According to the babel manual, the recommended way to load the Latin language in plain \TeX is:

\begin{verbatim}
\input latin.sty
\begindocument
\end{verbatim}

The modifiers usej and lowercasemonth may be accessed by means of the \languageattribute command:

\begin{verbatim}
\input latin.sty
\languageattribute{latin}{usej,lowercasemonth}
\begindocument
\end{verbatim}

\TeX does not provide sty files for classiclatin, medievallatin, and ecclesiasticlatin. It should be possible to create them locally if needed.

Note that no Latin shorthands are available in plain \TeX.

7 The code

We identify the language definition file.

\ProvidesLanguage{latin}[2021-06-27 v4.0 Latin support from the babel system]

The macro \LdfInit takes care of preventing that this file is loaded more than once with the same option, checking the category code of the @ sign, etc. \CurrentOption is the language requested by the user, i.e., latin, classiclatin, medievallatin, or ecclesiasticlatin.

\LdfInit{\CurrentOption}{captions\CurrentOption}

For tests, we need variables containing three possible values of the language name.

\def\babellatin@classic{classiclatin}
\def\babellatin@medieval{medievallatin}
\def\babellatin@ecclesiastic{ecclesiasticlatin}
7.1 Hyphenation patterns

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

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

We define macros for testing if the required hyphenation patterns are available.

\def\babellatin@test@modern@patterns{%\ifx\l@latin\undefined \@nopatterns{latin} \adddialect\l@latin0 \fi}%
\def\babellatin@test@classic@patterns{%\ifx\l@classiclatin\undefined \PackageWarningNoLine{babel-latin}{No hyphenation patterns were found for the\MessageBreak classiclatin language. Now I will use the\MessageBreak patterns for modern Latin instead}% \babellatin@test@modern@patterns \adddialect\l@classiclatin\l@latin \fi}%

We use the `classiclatin` hyphenation patterns for classical Latin and the (modern) `latin` hyphenation patterns for all other varieties of Latin.

\ifx\CurrentOption\babellatin@classic \babellatin@test@classic@patterns \else \ifx\CurrentOption\babellatin@ecclesiastic \babellatin@test@modern@patterns \adddialect\l@ecclesiasticlatin\l@latin \else \ifx\CurrentOption\babellatin@medieval \babellatin@test@modern@patterns \adddialect\l@medievallatin\l@latin \else \babellatin@test@modern@patterns \fi \fi \fi

7.2 Latin captions

We need a conditional governing the spelling of the captions. Medieval and ecclesiastic Latin use the ligatures æ and œ, classical and modern Latin do not.

\newif\ifbabellatin@useligatures
\addto\extrasmedievallatin{\babellatin@useligaturestrue}%
\addto\noextrasmedievallatin{\babellatin@useligaturesfalse}%
\addto\extrasecclesiasticlatin{\babellatin@useligaturestrue}%
\addto\noextrasecclesiasticlatin{\babellatin@useligaturesfalse}%
We define the Latin captions using the commands recommended by the babel manual.\(^4\)

\StartBabelCommands*{\CurrentOption}{captions}
\SetString{prefacename}{\ifbabellatin@useligatures Pr\ae fatio\else Praefatio\fi}
\SetString{refname}{Conspectus librorum}
\SetString{abstractname}{Summarium}
\SetString{bibname}{Conspectus librorum}
\SetString{chaptername}{Caput}
\SetString{appendixname}{Additamentum}
\SetString{contentsname}{Index}
\SetString{listfigurename}{Conspectus descriptionum}
\SetString{listtablename}{Conspectus tabularum}
\SetString{indexname}{Index rerum notabilium}
\SetString{figurename}{Descriptio}
\SetString{tablename}{Tabula}
\SetString{partname}{Pars}
\SetString{enclname}{Adduntur}\(^5\) Or "Additur"? Or simply Add.?\(^6\)
\SetString{ccname}{Exemplar}\(^7\) Use the recipient's dative
\SetString{headtoname}{\ignorespaces}\(^8\) Use the recipient's dative
\SetString{pagename}{Charta}
\SetString{seename}{cfr.}\(^9\)
\SetString{alsoname}{cfr.}\(^10\) Tabacco never saw "cfr" + "atque" or similar forms
\SetString{proofname}{Demonstratio}
\SetString{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 apppellative 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.

7.3 Mapping between upper and lower case

For classical and medieval Latin we need the suitable correspondence between uppercase V and lower-case u since in that spelling there is only one letter for the vowel and

\(^4\)Most of these names were kindly suggested by Raffaella Tabacco.
the consonant, and the u shape is an (uncial) variant of the capital V.

We use the commands recommended by the babel manual.

63 \StartBabelCommands{classiclatin,medievallatin}

The following command takes care for the correct behaviour of the \MakeUppercase and the \MakeLowercase command. It makes sure that \MakeUppercase{Helletia} yields "HELVETIA" and that \MakeLowercase{LVDVS} yields "ludus".

64 \SetCase{\uccode`u=`V}{\lccode`V=`u}

The following command takes care for the correct hyphenation of words written in capital letters. It makes sure that "LVDVS" is hyphenated the same way as "ludus".

65 \SetHyphenMap{\BabelLower{`V}{`u}}

For Unicode-based engines, we also have to take into account characters with diacritics. We map ú, ū, and ŭ to V because Unicode does not define a single-character V with the respective diacritic.

66 \StartBabelCommands{classiclatin,medievallatin}{unicode,fontenc=TU,charset=utf8}
67 \SetCase{\uccode`u=`V \uccode`ú=`V \uccode`ū=`V \uccode`ŭ=`V}{\lccode`V=`u}

According to the babel manual, the last \StartBabelCommands block has to be finished by the following command.

68 \EndBabelCommands

7.4 The Latin date

We need three conditionals governing the spelling of the month names. Ecclesiastic and modern Latin use the character v, classical and medieval Latin use only u. This affects the month of November. The user may demand to use the letter j where suitable or to lowercase month names using the respective modifiers.

69 \newif\ifbabellatin@usev
70 \newif\ifbabellatin@usej
71 \newif\ifbabellatin@lowercasemonth
72 \babellatin@usevtrue
73 \addto\extrasclassiclatin{\babellatin@usevfalse}\
74 \addto\noextrasclassiclatin{\babellatin@usevtrue}\
75 \addto\extrasmedievallatin{\babellatin@usevfalse}\
76 \addto\noextrasmedievallatin{\babellatin@usevtrue}\

The Latin month names are needed in the genitive case.

77 \def\babellatin@monthname{\%}
78 \ifcase\month\or\ifbabellatin@usej Januarii\else Januarii\fi
79 \or Februarii\%  
80 \or Martii\%  
81 \or Aprilis\%  
82 \or\ifbabellatin@usej Maji\else Maii\fi
83 \or\ifbabellatin@usej Juni\else Iunii\fi
84 \or\ifbabellatin@usej Julii\else Iulii\fi
85 \or Augusti\%  
86 \or Septembris\%  
87 \or Octobris\%
Depending on the chosen language, we have to define a `\latindate`, `\classiclatindate`, `\medievallatindate`, or `\ecclesiasticlatindate` command. The date format is "XXXI Decembris MMXXI".

```
\def\csname date\CurrentOption\endcsname{\
  \ifbabellatin@lowercasemonth\
  \uppercase\expandafter{\romannumeral\day}~\
  \babellatin@monthname\
  \else\
  \babellatin@monthname\
  \fi\
  \uppercase\expandafter{\romannumeral\year}\
}\%
```

### 7.5 Shorthands

We define shorthands only if the LaTeX format is used because we need commands for them that are not available in plain TeX.

```
\def\babellatin@latex{LaTeX2e}\
\ifx\fmtname\babellatin@latex\
  Every shorthand character needs an `\initiate@active@char` command, which makes the respective character active, but expanding to itself as long as no further definitions occur. The apostrophe (acute) is only made active if babel has been called with the `activeacute` option.

```
\initiate@active@char{"}\
\@ifpackagewith{babel}{activeacute}{\initiate@active@char{'}}{}\%
```

The following command is defined by the `hyperref` package. We use a dummy definition if this package is not loaded.

```
\providecommand\texorpdfstring[2]{#1}\
```

A peculiarity of the babel-latin package are shorthands of different lengths. " before a letter character defines an additional hyphenation point, but "æ is a shorthand for the ligature 'æ' in medieval and ecclesiastic Latin. So the shorthands definitions are rather complex and we need expl3 syntax for them.

```
\ExplSyntaxOn\
\def\DeclareShorthand{\latin}{"}\
{\mode_if_math:TF { \token_to_str:N " }\%}
```

The character " is used as a shorthand unconditionally. In math mode it expands to itself. In text mode it is defined as a macro with one parameter. This makes it possible to read the following token, on which the actual meaning of the shorthand depends.
The character ' is used as a shorthand if the activeacute option is used. So we have to use a macro for the declaration, which can be called if necessary. In math mode the shorthand expands to \active@math@prime as defined in latex.ltx. In text mode it is a macro with one argument to read the following token.

\cs_set_protected:Npn \babellatin@declare@apostrophe@shorthands
\begin{verbatim}
\declare@shorthand {latin} {'}
\mode_if_math:TF { \active@math@prime }
\{
\texorpdfstring { \babellatin_put_acute:N } { ' }
\}
\end{verbatim}

The characters = and ^= are only used as shorthands if the withprosodicmarks modifier is used. So we have to use a macro for the declaration, which can be called if necessary. In math mode both shorthands expand to themselves. In text mode they are macros with one argument to read the following token.

\cs_set_protected:Npn \babellatin@declare@prosodic@shorthands
\begin{verbatim}
\declare@shorthand {latin} {=}
\mode_if_math:TF { \token_to_str:N = }
\{
\texorpdfstring { \babellatin_put_macron:N } { \= }
\}
\end{verbatim}

The following macro defines the behaviour of the active " character. The shorthands "AE, "Ae, "ae, "OE, "Oe, and "oe are used for ligatures if the current variety of Latin uses them. In other cases " before any letter character or before \AE, \ae, \OE, and \oe defines an additional hyphenation point. "| defines an additional hyphenation point as well. The shorthands "< and "> are used for guillemets. In other cases the active " character expands to itself and the token read as argument is reinserted.

If the argument is a braced group (e.g. if the user has typed "{ab}"), unexpected behaviour may occur as the conditionals \token_if_letter:NTF and \babellatin_if_ligature_command:NTF expect a single token as first argument. Therefore we need to check if the argument is a single token using the \tl_if_single_token:nTF command before using those conditionals.
\cs_set_protected:Npn \babellatin_apply_quotemark:N #1
{\str_case:nnF {#1}
 \{A\} { \babellatin_ligature_shorthand:Nnn E { \AE }
 \{ \babellatin_ligature_shorthand:Nnn e { \AE }
 \{ \babellatin_allowhyphens: A \}
 \}
 \}
 \{a\} { \babellatin_ligature_shorthand:Nnn e { \ae }
 \{ \babellatin_allowhyphens: a \}
 \}
 \{O\} { \babellatin_ligature_shorthand:Nnn E { \OE }
 \{ \babellatin_ligature_shorthand:Nnn e { \OE }
 \{ \babellatin_allowhyphens: O \}
 \}
 \}
 \{o\} { \babellatin_ligature_shorthand:Nnn e { \oe }
 \{ \babellatin_allowhyphens: o \}
 \}
 \} { \babellatin_allowhyphens: } \}
 \{<\} { \babellatin@guillemetleft } \}
 \{>\} { \babellatin@guillemetright } \}
 \}
 \{\tl_if_single_token:nTF {#1}
 \{ \token_if_letter:NTF #1 { \babellatin_allowhyphens: } \}
 \{ \babellatin_if_ligature_command:NTF #1 { \babellatin_allowhyphens: } \}
 \{ \token_to_str:N " \}
 \}
 \{ \token_to_str:N " \}
 \#1 \}
 \}

15
The following macro defines the behaviour of the active ‘ character. The shorthands ‘AE, ‘Ae, ‘ae, ‘OE, ‘Oe, and ‘oe are used for accented ligatures if the current variety of Latin uses them. In other cases ‘ before any vowel or before \AE, \ae, \OE, and \oe defines an accented character. The character V is treated as a vowel here as it may represent the vowel U, but v is not, as it is never used for a vowel. In other cases the active ‘ character expands to itself and the token read as argument is reinserted.

\cs_set_protected:Npn \babellatin_put_acute:N #1

\tl_if_single_token:nTF {#1}

\babellatin_if_ligature_command:NTF #1 { \'}

\token_to_str:N \'}

#1
The following macro defines the behaviour of the active character. The shorthands =AE, =ae, =AU, =Au, =EU, =Eu, =OE, =Oe, and =oe are used for diphthongs with a combining double macron (U+035E) or ligatures with a macron if the current variety of Latin uses them. In other cases = before any vowel puts a macron above the vowel. The character V is treated as a vowel here as it may represent the vowel U, but v is not, as it is never used for a vowel. In other cases the active character expands to itself and the token read as argument is reinserted.

\cs_set_protected:Npn \babellatin_put_macron:N #1
\str_case:nnF {#1}
{\babellatin_diphthong_macron:NNn #1}

\babellatin_diphthong_macron:NNn #1
\babellatin_diphthong_macron:NNn ae { \=ae }
{\babellatin_diphthong_macron:NNn au { \=a } }
\babellatin_diphthong_macron:NNn eu { \=e }
\babellatin_diphthong_macron:NNn oe { \=o }
\babellatin_diphthong_macron:NNn u { \=u }
\babellatin_diphthong_macron:NNn v { \=v }
\babellatin_diphthong_macron:NNn y { \=y }
The following macro defines the behaviour of the active ^ character. ^ before any vowel puts a breve above the vowel. The character V is treated as a vowel here as it may represent the vowel U, but v is not, as it is never used for a vowel. In other cases the active ^ character expands to itself and the token read as argument is reinserted.

\cs_set:Npn \babellatin_put_breve:N #1
{\str_case:nnF {#1}
{\u{A} }{\u{a} }{\u{E} }{\u{e} }{\u{I} }{\u{\i} }{\u{O} }{\u{o} }{\u{U} }{\u{u} }{\u{V} }{\u{y} }
{\token_to_str:N ^ #1}
}

We define a macro for an additional hyphenation point that does not suppress other hyphenation points within the word. This macro is used by the " and the "|" shorthand.
The conditional \ifbabellatin@useligatures cannot be used within a expl3 context. So we have to define a macro testing if ligatures are enabled outside the expl3 code part. The result is stored in the variable \babellatin@useligatures@bool. We define this variable analogously to expl3’s \c_true_bool and \c_false_bool.

\ExplSyntaxOff
\def\babellatin@test@for@ligatures{\ifbabellatin@useligatures\chardef\babellatin@useligatures@bool=1\else\chardef\babellatin@useligatures@bool=0\fi}\ExplSyntaxOn

The following macro is intended for defining a shorthand for a ligature where useful. The first argument is the expected second character after “e” (e.g. “a” if “a” has been read). The second argument is the true code, that applies if this character is found (the ligature command). The third argument is the false code (some other command).

\cs_set_protected:Npn \babellatin_ligature_shorthand:Nnn #1#2#3
{\babellatin@test@for@ligatures\bool_if:NTF \babellatin@useligatures@bool
{\peek_meaning_remove:NTF #1 {#2} {#3}}\else{#3}\fi}

The following macro is intended for defining a shorthand for a diphthong with a combining double macron (U+035E). The first argument is the first character of the diphthong, which has already been read. The second argument is the second character of the diphthong, which is expected to be read. The third argument is the false code, that applies if the second character is not found as expected.

For pdfLATEX a warning is issued if the diphthong is found as this engine does not support the combining double macron.

\cs_set_protected:Npn \babellatin_diphthong_macron:NNn #1#2#3
{\peek_meaning:NTF #2
{\bool_lazy_or:nnTF {\sys_if_engine_xetex_p:} {\sys_if_engine_luatex_p:}
{\iffontchar \font "35E \relax \char "35E \relax \else \msg_warning:nn {babel-latin} {no-double-macron-font} }\else}}
\msg_warning:nn {babel-latin} {no-double-macron-engine}
\msg_set:nnn {babel-latin} {no-double-macron-font}
\msg_set:nnn {babel-latin} {no-double-macron-engine}

The following macro is intended for defining a shorthand for a ligature with a macron where useful. The first argument is the first character of the diphthong, which has already been read. The second argument is the expected second character of the diphthong. The third argument is the code for the ligature with the macron. The fourth argument is the false code that applies if the second character is not found.

\cs_set_protected:Npn \babellatin_ligature_macron:NNnn #1#2#3#4
{ \babellatin_ligature_shorthand:Nnn #2 {#3}
  \babellatin_diphthong_macron:NNn #1 #2 {#4}
}

The following conditional tests if the argument is a ligature command (\AE, \ae, \OE, or \oe).
\prg_set_conditional:Npnn \babellatin_if_ligature_command:N #1 {TF}
{ \token_if_eq_meaning:NNTF #1 \AE { \prg_return_true: }
  \token_if_eq_meaning:NNTF #1 \ae { \prg_return_true: }
  \token_if_eq_meaning:NNTF #1 \OE { \prg_return_true: }
  \token_if_eq_meaning:NNTF #1 \oe { \prg_return_true: }
  \prg_return_false: }

The following macro is intended for defining a shorthand for a ligature with a macron where useful. The first argument is the first character of the diphthong, which has already been read. The second argument is the expected second character of the diphthong. The third argument is the code for the ligature with the macron. The fourth argument is the false code that applies if the second character is not found.

\cs_set_protected:Npn \babellatin_ligature_macron:NNnn #1#2#3#4
{ \babellatin_ligature_shorthand:Nnn #2 {#3}
  \babellatin_diphthong_macron:NNn #1 #2 {#4}
}

The following conditional tests if the argument is a ligature command (\AE, \ae, \OE, or \oe).
\prg_set_conditional:Npnn \babellatin_if_ligature_command:N #1 {TF}
{ \token_if_eq_meaning:NNTF #1 \AE { \prg_return_true: }
  \token_if_eq_meaning:NNTF #1 \ae { \prg_return_true: }
  \token_if_eq_meaning:NNTF #1 \OE { \prg_return_true: }
  \token_if_eq_meaning:NNTF #1 \oe { \prg_return_true: }
  \prg_return_false: }

The following macro is intended for defining a shorthand for a ligature with a macron where useful. The first argument is the first character of the diphthong, which has already been read. The second argument is the expected second character of the diphthong. The third argument is the code for the ligature with the macron. The fourth argument is the false code that applies if the second character is not found.

\cs_set_protected:Npn \babellatin_ligature_macron:NNnn #1#2#3#4
{ \babellatin_ligature_shorthand:Nnn #2 {#3}
  \babellatin_diphthong_macron:NNn #1 #2 {#4}
}

The following conditional tests if the argument is a ligature command (\AE, \ae, \OE, or \oe).
For the "< and the "> shorthands we have to define the meaning of the macros used for their definition. The commands \guillemetleft and \guillemetright are provided by babel. We will have to change this definition later on for ecclesiastic\latin if pdf\TeX\ is used.

\let\babellatin@guillemetleft\guillemetleft
\let\babellatin@guillemetright\guillemetright

Finally, we have to add the shorthand definitions to the extras of the current language.

\expandafter\addto\csname extras\CurrentOption\endcsname{\babellatin@activate{"}}%
\languageshorthands{latin}%
}
\expandafter\addto\csname noextras\CurrentOption\endcsname{\babellatin@deactivate{"}}%

\@ifpackagewith{babel}{activeacute}{%
\babellatin@declare@apostrophe@shorthands
\def\babellatin@prepare@punctuation@spacing{\if\babellatin@luatex
\babellatin@declare@apostrophe@shorthands
\@ifpackagewith{babel}{activeacute}{%
\babellatin@luatexture}
\@ifpackagewith{babel}{xetex}{%
\babellatin@xetexture
\expandafter\addto\csname extras\CurrentOption\endcsname{\babellatin@activate{'}}%
\expandafter\addto\csname noextras\CurrentOption\endcsname{\babellatin@deactivate{'}}%
}
%
}
\fi

7.6 Ecclesiastic punctuation spacing

We define some conditionals concerning the engine used.

\newif\ifbabellatin@luatex
\newif\ifbabellatin@xetex
\ifnum\babellatin@engine=1
\babellatin@luatexture
\else
\ifnum\babellatin@engine=2
\babellatin@xetexture
\fi
\fi

The following command defines the preparations needed for punctuation spacing in the preamble.

\def\babellatin@prepare@punctuation@spacing{%
For Lua\TeX\ we load an additional file containing some Lua code. This file is documented in section 7.9.
The following command inserts a kern of 1/12 of a quad. This is the only amount of space used for punctuation within this package.

\def\babellatin@insert@punctuation@space{\kern0.08333\fontdimen6\font}

The following command inserts the same kern, removing any positive amount of space that precedes. This is needed if a closing guillemet is preceded by a space character erroneously input by the user.

\def\babellatin@replace@preceding@space{\ifdim\lastskip>\z@\unskip\fi \babellatin@insert@punctuation@space}

The following command inserts the same kern, removing any following space character. This is needed if an opening guillemet is followed by a space character erroneously input by the user.

\def\babellatin@replace@following@space{\babellatin@insert@punctuation@space \ignorespaces}

For X\TeX the punctuation spacing will be defined based on five different character classes: one for question and exclamation marks, one for colons and semicolons, one for opening and closing guillemets, respectively, and one for opening brackets. Concerning spacing, brackets are treated the same way as letter characters in most cases. However, in strings like "(?)", no spacing is desired before the question mark. So we need a dedicated character class for opening brackets.

\ifbabellatin@xetex \newXeTeXintercharclass\babellatin@qmark@class \newXeTeXintercharclass\babellatin@colon@class \newXeTeXintercharclass\babellatin@cguill@class \newXeTeXintercharclass\babellatin@cguill@class \newXeTeXintercharclass\babellatin@obracket@class \let\babellatin@boundary@class\e@alloc@intercharclass@top \XeTeXinterchartoks\babellatin@qmark@class\babellatin@cguill@class={\babellatin@insert@punctuation@space} \XeTeXinterchartoks\babellatin@qmark@class\babellatin@colon@class={\babellatin@insert@punctuation@space} \XeTeXinterchartoks\babellatin@qmark@class\babellatin@cguill@class={\babellatin@insert@punctuation@space} \XeTeXinterchartoks\babellatin@qmark@class\babellatin@colon@class={\babellatin@insert@punctuation@space}

Furthermore, we need a class representing the word boundary. This class has a fixed number defined in \texttt{latex.ltx}.

\let\babellatin@boundary@class\e@alloc@intercharclass@top

A space is inserted between a question or exclamation mark and a closing guillemet.

\XeTeXinterchartoks\babellatin@qmark@class\babellatin@cguill@class={\babellatin@insert@punctuation@space}

A space is inserted between a question or exclamation mark and a colon or semicolon.

\XeTeXinterchartoks\babellatin@qmark@class\babellatin@colon@class={\babellatin@insert@punctuation@space}
A space is inserted between a colon or semicolon and a closing guillemet.
\XeTeXinterchartoks\babellatin@colon@class\babellatin@cguill@class={\babellatin@insert@punctuation@space}

A space character after an opening guillemet is replaced by the correct amount of space.
\XeTeXinterchartoks\babellatin@oguill@class\babellatin@boundary@class={\babellatin@replace@following@space}

A space is inserted between two opening guillemets.
\XeTeXinterchartoks\babellatin@oguill@class\babellatin@oguill@class={\babellatin@insert@punctuation@space}

A space is inserted between an opening guillemet and any ordinary character.
\XeTeXinterchartoks\babellatin@oguill@class\z@={\babellatin@insert@punctuation@space}

A space is inserted between two closing guillemets.
\XeTeXinterchartoks\babellatin@cguill@class\babellatin@cguill@class={\babellatin@insert@punctuation@space}

A space is inserted between a closing guillemet and a question or exclamation mark.
\XeTeXinterchartoks\babellatin@cguill@class\babellatin@qmark@class={\babellatin@insert@punctuation@space}

A space is inserted between a closing guillemet and a colon or semicolon.
\XeTeXinterchartoks\babellatin@cguill@class\babellatin@colon@class={\babellatin@insert@punctuation@space}

A space character before a question or exclamation mark is replaced by the correct amount of space.
\XeTeXinterchartoks\babellatin@boundary@class\babellatin@qmark@class={\babellatin@replace@preceding@space}

A space character before a colon or semicolon is replaced by the correct amount of space.
\XeTeXinterchartoks\babellatin@boundary@class\babellatin@colon@class={\babellatin@replace@preceding@space}

A space character before a closing guillemet is replaced by the correct amount of space.
\XeTeXinterchartoks\babellatin@boundary@class\babellatin@cguill@class={\babellatin@replace@preceding@space}

A space is inserted between any ordinary character and a question or exclamation mark.
\XeTeXinterchartoks\z@\babellatin@qmark@class={\babellatin@insert@punctuation@space}

A space is inserted between any ordinary character and a colon or semicolon.
\XeTeXinterchartoks\z@\babellatin@colon@class={\babellatin@insert@punctuation@space}

A space is inserted between any ordinary character and a closing guillemet.
\XeTeXinterchartoks\z@\babellatin@cguill@class={\babellatin@insert@punctuation@space}

\else
In \texttt{pdfTeX} active characters are needed for punctuation spacing.

\input{babellatin}

\begin{verbatim}
 We call the previously defined command for ecclesiastic Latin.
\end{verbatim}

\begin{verbatim}
 The following function actually enables the spacing of punctuation.
\end{verbatim}

For \texttt{LuaTeX} we just have to call a function of the Lua module.
For \TeX{} we have to enable the character classes functionality and assign the punctuation characters to the character classes.

\ifbabellatin@xetex
\XeTeXinterchartokenstate = 1
\XeTeXcharclass `! \babellatin@qmark@class
\XeTeXcharclass `? \babellatin@qmark@class
\XeTeXcharclass `!! \babellatin@qmark@class
\XeTeXcharclass `!! \babellatin@qmark@class
\XeTeXcharclass `‼ \babellatin@qmark@class
\XeTeXcharclass `⁇ \babellatin@qmark@class
\XeTeXcharclass `⁈ \babellatin@qmark@class
\XeTeXcharclass `⁉ \babellatin@qmark@class
\XeTeXcharclass `‽ \babellatin@qmark@class
\XeTeXcharclass `; \babellatin@colon@class
\XeTeXcharclass `: \babellatin@colon@class
\XeTeXcharclass `« \babellatin@oguill@class
\XeTeXcharclass `» \babellatin@cguill@class
\XeTeXcharclass `‹ \babellatin@oguill@class
\XeTeXcharclass `› \babellatin@cguill@class
\fi
\else
For pdf\TeX{} we activate the shorthands.
\bbl@activate{;}%
\bbl@activate{:}%
\bbl@activate{!}%
\bbl@activate{?}%
\fi
The following function disables the spacing of punctuation.
\def\babellatin@no@punctuation@spacing{%
\ifbabellatin@luatex
\directlua{ecclesiasticlatin.deactivate_spacing()}%
\else
\fi
\else
\fi
\fi
7.7 Modifiers

We define some language options accessible via modifiers.

7.7.1 Using the letter j

The usej option sets the conditional \ifbabellatin@usej to true.

Punctuation is spaced in ecclesiastic Latin only.

\addto\extrasecclesiasticlatin{\babellatin@punctuation@spacing}%
\addto\noextrasecclesiasticlatin{\babellatin@no@punctuation@spacing}%
7.7.2 Typesetting months in lower case

The lowercasemonth option sets the conditional \ifbabellatin@lowercasemonth to true.

```
\bbl@declare@ttribute\CurrentOption{lowercasemonth}{%
  \expandafter\addto\csname extras\CurrentOption\endcsname{%
    \babellatin@lowercasemonthtrue%
  }%
  \expandafter\addto\csname noextras\CurrentOption\endcsname{%
    \babellatin@lowercasemonthfalse%
  }%
}%
```

7.7.3 Shorthands for prosodic marks

The withprosodicmarks option makes it possible to use shorthands like \=a or \^a for vowels with macrons and breves. We define it for all four language variants of Latin, but only if the \LaTeX{} format is used.

```
\ifx\fmtname\babellatin@latex
  \bbl@declare@ttribute\CurrentOption{withprosodicmarks}{%
    Every shorthand character needs an \initiate@active@char command, which makes the respective character active, but expanding to itself as long as no further definitions occur. Both active characters needs to be switched off at the beginning of the document to avoid problems with commands using key=value interfaces (e.g. \includegraphics) and \LaTeX{}'s ^^xx convention.

    \initiate@active@char{=}%
    \initiate@active@char{^}%
    \AtBeginDocument{%
      \shorthandoff{=}%
      \shorthandoff{^}%
    }%

    We do not use \shorthandoff{=} and \shorthandoff*{^} in the following lines because babel-french redefines the \shorthandoff command for Xe\LaTeX{} and Lua\LaTeX. Instead, we use babel's internal definition of this command.

    \bbl@shorthandoff\z@{=}%

    The following line is currently uncommented because switching ^ off and on does not work as expected.\footnote{See https://github.com/latex3/babel/issues/126.}

    \bbl@shorthandoff\tw@{^}%
  }%
\babellatin@declare@prosodic@shorthands
\expandafter\addto\csname extras\CurrentOption\endcsname{%
  \babellatin@activate{=}%
  \babellatin@activate{^}%
}
```

The active \= and \^ are normally turned off to avoid problems with commands using key=value interfaces and \LaTeX{}'s ^^xx convention. We define the commands \ProsodicMarksOn and \ProsodicMarksOff for turning them on and off within the document. We use the starred form of \shorthandoff when turning off \^ to keep it working within math formulas.

```
\def\ProsodicMarksOn{%
  \shorthandon{=}%
}
\def\ProsodicMarksOff{%
  \shorthandoff{=}%
}
```

\footnote{See https://github.com/latex3/babel/issues/126.}
The following line is currently uncommented because switching ^ off and on does not work as expected.

\shorthandon{^}%
\def\ProsodicMarksOff{\shorthandoff{=}%

The following line is currently uncommented because switching ^ off and on does not work as expected.
\shorthandoff{*}%

The \ProsodicMarksOn and \ProsodicMarksOff commands are useless without the withprosodicmarks modifier. They only issue warnings in this case.

\expandafter\addto\csname extras\CurrentOption\endcsname{%
\def\ProsodicMarksOn{\PackageWarning{babel-latin}{The \protect\ProsodicMarksOn\space command is only available using the withprosodicmarks\messagebreak modifier}%
\def\ProsodicMarksOff{\PackageWarning{babel-latin}{The \protect\ProsodicMarksOff\space command is only available using the withprosodicmarks\messagebreak modifier}%

7.7.4 Ecclesiastic footnotes

The ecclesiasticfootnotes option sets the footnotes globally to the style defined by the (now outdated) ecclesiastic package. The definition takes place at the end of the package to be able to check babel's main language. However, the \CurrentOption has lost its value at this moment, so we have to store it.
\bbl@declare@ttribute\CurrentOption{ecclesiasticfootnotes}{%
\let\babellatin@footnote@lang\CurrentOption
\AtEndOfPackage{%
\ifx\bbl@main@language\babellatin@footnote@lang
\let\makefntext\babellatin@variant@footnote
\else
\PackageWarningNoLine{babel-latin}{%
This is the footnote style as defined by the ecclesiastic package.

\def\babellatin@variant@footnote#1{% 
\parindent 1em% 
\noindent 
\hbox{\normalfont\@thefnmark.} \enspace #1% 
}% 

7.8 Legacy modifiers and commands

We keep the modifiers classic, medieval, and ecclesiastic for backwards compatibility. We issue a warning if they are used.

\def\babellatin@outdated@modifier#1{% 
\PackageWarningNoLine{babel-latin}{% 
The `#1' modifier is outdated. Please consult the babel-latin manual and consider to load the language `#1latin' instead of `latin.#1'}% 
}% 
\bbl@declare@tribute{latin}{classic}{% 
\babellatin@outdated@modifier{classic}% 
\addto\extraslatin{\babellatin@usevfalse}% 
\addto\noextraslatin{\babellatin@usevtrue}% 
\babellatin@test@classic@patterns 
\let\l@latin\l@classiclatin 
\StartBabelCommands*{latin}{% 
\SetCase{\uccode `u=`V}{\lccode `V=`u}% 
\EndBabelCommands 
}% 
\bbl@declare@tribute{latin}{medieval}{% 
\babellatin@outdated@modifier{medieval}% 
\addto\extraslatin{% 
\babellatin@usevfalse 
\def\prefacename{Pr\ae fatio}% 
}% 
\addto\noextraslatin{% 
\babellatin@usevtrue 
}% 
\StartBabelCommands*{latin}{% 
\SetCase{\uccode `u=`V}{\lccode `V=`u}% 
\EndBabelCommands 
}% 
\bbl@declare@tribute{latin}{ecclesiastic}{% 

The apostrophe character becomes active, even without babel’s activeacute option.

We set up the footnotes like the ecclesiastic package did.

In earlier versions of babel-latin (up to v.3.5) a \SetLatinLigatures command and a \ProsodicMarks command have been defined. We retain them for backwards compatibility, but they do nothing except issuing a warning.

We retain some legacy commands concerning guillemets from the ecclesiastic package, which is now outdated, but we deprecate them.
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\CurrentOption

babel expects ldf files for classiclatin, medievallatin and ecclesiasticlatin. These files themselves only load latin.ldf, which does the real work:

⟨classic⟩\ProvidesLanguage{classiclatin}
⟨ecclesiastic⟩\ProvidesLanguage{ecclesiasticlatin}
⟨medieval⟩\ProvidesLanguage{medievallatin}

\input latin.ldf\relax

### 7.9 The Lua module

In case Lua\TeX{} is used for compilation, the spacing of punctuation for ecclesiastic Latin requires some Lua code, which is stored in ecclesiasticlatin.lua. The original version of this code has been written for the polyglossia package by É. Roux and others.

The Lua module identifies itself using the command provided by \texttt{ltluatex}.

\texttt{luatexbase.provides_module({
  name = "ecclesiasticlatin",
  date  = "2021-06-27",
  version = "4.0",
  description = "babel-latin punctuation spacing for ecclesiastic Latin"
})}

\texttt{local add_to_callback = luatexbase.add_to_callback}
\texttt{local in_callback = luatexbase.in_callback}
\texttt{local new_attribute = luatexbase.new_attribute}
\texttt{local node = node}
\texttt{local insert_node_before = node.insert_before}
\texttt{local insert_node_after = node.insert_after}
local remove_node = node.remove
local has_attribute = node.has_attribute
local node_copy = node.copy
local new_node = node.new
local end_of_math = node.end_of_math
local get_next = node.getnext
local get_prev = node.getprev
local get_property = node.getproperty

Node types according to node.types():
local glue_code = node.id"glue"
local glyph_code = node.id"glyph"
local penalty_code = node.id"penalty"
local kern_code = node.id"kern"
local math_code = node.id"math"

We need some node subtypes:
local userkern = 1
local removable_skip = {
  [0] = true, -- userskip
  [13] = true, -- spaceskip
  [14] = true -- xspaceskip
}

We make a new node, so that we can copy it later on:
local kern_node = new_node(kern_code)
kern_node.subtype = userkern
local function get_kern_node(dim)
  local n = node_copy(kern_node)
  n.kern = dim
  return n
end

All possible space characters according to section 6.2 of the Unicode Standard (https://www.unicode.org/versions/Unicode12.0.0/ch06.pdf):
local space_chars = {
  [0x20] = true, -- space
  [0xA0] = true, -- no-break space
  [0x1680] = true, -- ogham space mark
  [0x2000] = true, -- en quad
  [0x2001] = true, -- em quad
  [0x2002] = true, -- en space
  [0x2003] = true, -- em space
  [0x2004] = true, -- three-per-em-space
  [0x2005] = true, -- four-per-em space
  [0x2006] = true, -- six-per-em space
  [0x2007] = true, -- figure space
  [0x2008] = true, -- punctuation space
  [0x2009] = true, -- thin space
  [0x200A] = true, -- hair space
  [0x202F] = true, -- narrow no-break space
}
local left_bracket_chars = {
    [0x28] = true, -- left parenthesis
    [0x5B] = true, -- left square bracket
    [0x7B] = true, -- left curly bracket
    [0x27E8] = true -- mathematical left angle bracket
}

local right_bracket_chars = {
    [0x29] = true, -- right parenthesis
    [0x5D] = true, -- right square bracket
    [0x7D] = true, -- right curly bracket
    [0x27E9] = true -- mathematical right angle bracket
}

local question_exclamation_chars = {
    [0x21] = true, -- exclamation mark !
    [0x3F] = true, -- question mark ?
    [0x203C] = true, -- double exclamation mark ‼
    [0x203D] = true, -- interrobang ‽
    [0x2047] = true, -- double question mark ⁇
    [0x2048] = true, -- question exclamation mark ⁈
    [0x2049] = true -- exclamation question mark ⁉
}

local function somespace(n)
    if n then
        local id, subtype = n.id, n.subtype
        if id == glue_code then
            return removable_skip[subtype]
        elseif id == kern_code then
            return subtype == userkern
        elseif id == glyph_code then
            return space_chars[n.char]
        end
    end
end

local function someleftbracket(n)
    if n then
        return
    end
end

local function someleftbracket(n)
    if n then
        local id, subtype = n.id, n.subtype
        if id == glue_code then
            return removable_skip[subtype]
        elseif id == kern_code then
            return subtype == userkern
        elseif id == glyph_code then
            return space_chars[n.char]
        end
    end
end
local id = n.id
if id == glyph_code then
    return left_bracket_chars[n.char]
end
end

Test for a right bracket:
local function somerightbracket(n)
    if n then
        local id = n.id
        if id == glyph_code then
            return right_bracket_chars[n.char]
        end
    end
end

Test for two question or exclamation marks:
local function question_exclamation_sequence(n1, n2)
    if n1 and n2 then
        local id1 = n1.id
        local id2 = n2.id
        if id1 == glyph_code and id2 == glyph_code then
            return question_exclamation_chars[n1.char] and question_exclamation_chars[n2.char]
        end
    end
end

Test for a penalty node:
local function somepenalty(n, value)
    if n then
        local id = n.id
        if id == penalty_code then
            if value then
                return n.penalty == value
            else
                return true
            end
        end
    end
end

LuaTeX attribute determining whether to space punctuation or not:
local punct_attr = new_attribute("ecclesiasticlatin_punct")

Tables containing the left and right space amount (in units of a quad) of every character:
local left_space = {}
local right_space = {}

Insertion of the necessary spaces to the node list:
local function process(head)
while current do
    local id = current.id
    if id == glyph_code then
        if has_attribute(current, punct_attr) then
            We try to obtain the character of the current node from its property table, which is
            the most reliable way as the same character may be rendered by different glyphs with
            different code numbers.

            local char = get_property(current) and get_property(current).glyph_info

            If the glyph_info property is not available, we use the node’s char field to obtain the
            character, which is however only possible for numbers up to 10FFFF16.

            if not char and current.char <= 0x10FFFF then
                char = utf8.char(current.char)
            end

            local leftspace, rightspace
            if char then
                leftspace = left_space[char]
                rightspace = right_space[char]
            end

            if leftspace or rightspace then
                local fontparameters = fonts.hashes.parameters[current.font]
                local spacing_node
                if leftspace and fontparameters then
                    local prev = get_prev(current)
                    local space_exception = false
                    if prev then
                        We do not add space after left (opening) brackets and between question/exclamation
                        marks:

                        space_exception = someleftbracket(prev)
                        or question_exclamation_sequence(prev, current)
                    end
                    while some_space(prev) do
                        head = remove_node(head, prev)
                        prev = get_prev(current)
                    end
                    if somepenalty(prev, 10000) then
                        head = remove_node(head, prev)
                    end
                end
                spacing_node = get_kern_node(leftspace * fontparameters.quad)
                if not space_exception then
                    head = insert_node_before(head, current, spacing_node)
                end
            end
            if rightspace and fontparameters then
                local next = get_next(current)
                local space_exception = false
                if next then
                    We do not add space before right (closing) brackets:

35
space_exception = somerightbracket(next)
local nextnext = get_next(next)
if somepenalty(next, 10000) and somespace(nextnext) then
    head, next = remove_node(head, next)
end
while somespace(next) do
    head, next = remove_node(head, next)
end
spacing_node = get_kern_node(rightspace * fontparameters.quad)
if not space_exception then
    head, current = insert_node_after(head, current, spacing_node)
end
elseif id == math_code then
    current = end_of_math(current)
end
return head

The following line does not cause an error even if current is nil.
current = get_next(current)
end

Now we define the actual amount of space for the relevant punctuation characters. For ecclesiastic Latin (and sometimes for Italian) a very small space is used for the punctuation. The ecclesiastic package, a predecessor of the current babel-latin, used a space of 0.3 \fontdimen2, where \fontdimen2 is an interword space, which is typically between 1/4 and 1/3 of a quad. We choose a half of a \thinspace here, i.e., 1/12 of a quad.

local hairspace = 0.08333 -- 1/12
local function space_left(char)
    left_space[char] = hairspace
end
local function space_right(char, kern)
    right_space[char] = hairspace
end
local function space_right(char, kern)
    right_space[char] = hairspace
end
space_left('!')
space_left('?')
space_left('‼')
space_left('⁇')
space_left('⁈')
space_left('⁉')
space_left('‽') -- U+203D (interrobang)
space_left(',:) -- U+203D (interrobang)
space_left(';')
space_left('»')
space_left('«')
space_right('«')
space_right('«')
The following functions activate and deactivate the punctuation spacing.

```latex
local function activate()
  tex.setattribute(punct_attr, 1)
  for _, callback_name in ipairs{'pre_linebreak_filter', 'hpack_filter'} do
    if not in_callback(callback_name, "ecclesiasticlatin-punct.process") then
      add_to_callback(callback_name, process, "ecclesiasticlatin-punct.process", 1)
    end
  end
end

local function deactivate()
  Though it would make compilation slightly faster, it is not possible to safely remove the
  process from the callback here. Imagine the following case: you start a paragraph by
  some spaced punctuation text, then, in the same paragraph, you change the language
  to something else, and thus call this function. This means that, at the end of the para-
  graph, the function won’t be in the callback, so the beginning of the paragraph won’t
  be processed by it. So we just unset the attribute.
  tex.setattribute(punct_attr, -0x7FFFFFFF) -- this value means "unset"
end
```

For external access to the activation and deactivation of the punctuation spacing, we
define two functions with the prefix `ecclesiasticlatin`.

```latex
ecclesiasticlatin = ecclesiasticlatin or {}
ecclesiasticlatin.activate_spacing = activate
ecclesiasticlatin.deactivate_spacing = deactivate
```

### Change History

0.99  
General: Added shorthands for breve and macron  
General: Added shorthands for etymological hyphenation  
First version, from italian.dtx (CB)  

1.2  
General: Added suggestions from Krzysztof Konrad Zelechowski  

2.0a  
General: Revised by JB  

2.0b  
General: Language attribute medieval declared  
Modified breve and macro shorthands  
Simplified shorthands for etymological hyphenation  

2.0e  
General: Introduced the language attribute ‘withprosodicmarks’; modified use of breve and macron shorthands in order to avoid possible conflicts with other packages  

2.0k  
General: Inserted the various ‘November’ Latin spellings to the proper ‘extras’ macros  

3.0  
General: Added modifier for classical spelling and hyphenation  

3.5  
General: Added the modifier for the ecclesiastic Latin variety
| 4.0 | Document activation of the \texttt{liturgicallatin} hyphenation patterns | 6 |
|      | Document incompatibilities with other packages | 8 |
|      | Keep the default values of \texttt{\clubpenalty}, \texttt{@\clubpenalty}, \texttt{\widowpenalty}, and \texttt{\finalhyphenpenalties} for Latin | 9 |
|      | Make ecclesiastic Latin work with \texttt{Xe\TeX} and \texttt{LuaLa\TeX} | 1 |
|      | New babel languages \texttt{classiclatin}, \texttt{medievallatin}, and \texttt{ecclesiasticlatin}, replacing the respective modifiers | 2 |
|      | New modifiers usej, lowercasemonth, and \texttt{ecclesiasticfootnotes} | 4 |
|      | New shorthands for diphthongs with macron | 5 |
|      | Remove commands \texttt{\LatinMarksOn} and \texttt{\LatinMarksOff} | 9 |
| General: Additional shorthands for guillemets and accented letters for all language variants; additional shorthands for ligatures for medieval and ecclesiastic Latin | 7 |
| Basic support for plain \TeX | 9 |
| Complete revision by KW | 1 |
| Declare \texttt{\FrenchGuillemetsFrom}, \texttt{\ToneGuillemets}, \texttt{\og}, and \texttt{\fg} (defined by the ecclesiastic package) obsolete | 30 |
| Declare \texttt{\SetLatinLigatures} and \texttt{\ProsodicMarks} obsolete | 30 |
| Deprecate the \texttt{classic}, \texttt{medieval}, and \texttt{ecclesiastic} modifiers | 6 |
| Do not load the ecclesiastic package for the ecclesiastic modifier, use an internal implementation instead | 29 |
| Do not use small caps for the day of month | 12 |