Latin support for polyglossia

Élie Roux

Version 1.03 dated 2016/09/10

Abstract

This file documents the three language variants for the Latin language, as they should be used with XeLaTeX and LuaLaTeX; they describe each variant as a different language so that even LuaTeX, that loads one pattern file for each language used in a specific document, can use all the three languages simultaneously in a specific document by calling them with their specific names.

Contents

1 The Latin language

This file `gloss-latin.dtx` defines all the language-specific macros for the Latin language variants that include the modern, the medieval, the ecclesiastic, the classic, and the liturgical ones.

The language description file `gloss-latin.ldf` describes what is required to typeset in modern, medieval and ecclesiastic latin; the first variety is the one commonly used in typesetting school books, dictionaries, grammars, and the like; it manages also the medieval and ecclesiastic variants, because these variants share the same pattern file; they differ mainly in the character set actually used in their scripts; modern Latin uses only the 26 letter alphabet, commonly called Latin Alphabet; the medieval variant in addition uses the ligated diphthongs 'æ' and 'œ', but does not use the capital 'U' and the lowercase 'v', as it was customary in the medieval times and later on until about the XVII century; the ecclesiastic variant uses the full 26 letter Latin alphabet, plus the ligated diphthongs 'æ' and 'œ', but uses also the accented vowels, always with an acute accent, to mark the stress on the proper syllable so as to aid clergy coming from different countries and with different mother languages to pronounce all wards with the same rhythm when that attend collective liturgical activities; see table 1. Ecclesiastic Latin is typeset with a French-like style for punctuation and footnote marks, although the space introduced before the “tall’ punctuation marks is smaller than in French typography.

Modern and medieval Latin are pronounced with a definite word stress; therefore it is possible to recognise other diphthongs beyond the traditional ‘ae’ or ‘æ’ and ‘oe’ or ‘œ’. Significantly there are the ascending and descending diphthongs present today in most romance languages and some triphthongs; modern ascending diphthongs are formed with an unstressed closed vowel ‘i’ or ‘u’ followed by another vowel; descending diphthongs are formed with a vowel followed by an unstressed closed vowel; triphthongs are formed by two (different) unstressed closed vowels followed by another vowel (for example ‘quietus’), or an open vowel sandwiched between two unstressed closed vowels (for example ‘ieu’ in ‘maieutica’); hiati may appear as diphthongs, but they differ in the sense that the closed vowel is stressed; if it is marked with an accent or with the diaeresis, the hyphenation algorithm may distinguish them, but even with this capability good typography avoids line breaks across hiati, even if the grammar allows them. This is why the hyphenation algorithm avoids line breaking not only between the vowels that form a diphthong or a triphthong, but also between the vowels that form a hiatus. Clusters of more than two vowels are possibly divided before ascending diphthongs.

This long explanation about hyphenation is to stress the difference between the next Latin variants, classic and liturgical.

The Language description file `gloss-classiclatin.ldf` describes what is necessary in order to typeset (presumably philological) documents containing stretches of classical Latin text. Of course the hyphenation that is defined by the corresponding hyphenation pattern file was not used in classical times when not only the written texts were not hyphenated, but were written with the *lectio continua*, i.e. without even a space between words. In the first century AD Quintilianus used to teach oratory skills to his students by suggesting them to separate the various phrases with some kind of marks; these were supposed to delimit the “comma”, the ‘colon” and the ‘periodus”; these names survive today in English to denote certain modern punctuation marks. But no rules were given to hyphenate words.

What is actually called today “classic Latin hyphenation” is a set of rules agreed upon by scholars from different countries with different modern languages that try to take into account the metrics of various poems, the pronunciation and the etymology of each word so that prefixes and suffixes are divided from the word stem, as well any suffixes that do not belong to the declination or the conjugation of the word itself; within the word they try to follow a phonetic syllable division, although with different phonetic bases from modern languages.

The classical Latin hyphenation is a practical necessity in order to typeset texts that are justified on both sides; its rules are totally artificial, in the sense that ancient Romans certainly used phonetic syllables while speaking, but the Latin language, well spoken by the orators of the republican and of the initial imperial era, probably did not sound as we pronounce it today, be it a scholar’s speech or a prayer during some liturgy in christian churches. Ancient Latin had a melodic rhythm based on long and short vowels reasonably without a stress accent as we use it in most modern languages. Eventually Latin evolved into the romance

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

(*a*): modern Latin; letters j and w are only for latinised foreign modern names

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

(*b*) medieval Latin

a á æ ǽ b c d e é f g h i í j k l m n o ó œ œ́ p q r s t u ú v w x y z
A Á Æ Ǽ B C D E É F G H I Í J K L M N O Ó Œ Œ́ P Q R S T U Ú V W X Y Z

(*c*) ecclesiastic Latin; j and w are used for latinised foreign names; all vowels and diphthongs may be accented.

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

(*d*) classic Latin

a á æ ǽ b c d e é f g h i í j k l m n o ó œ œ́ p q r s t u ú v w x y z
A Á Æ Ǽ B C D E É F G H I Í J K L M N O Ó Œ Œ́ P Q R S T U Ú V W X Y Z

(*e*) liturgical Latin; the character set is identical to the ecclesiastic Latin one, but hyphenation is different

Table 1: Latin character sets: (*a*) modern Latin; (*b*) medieval Latin; (*c*) ecclesiastic Latin; (*d*) classic Latin; (*e*) liturgical Latin. Notice that in this document that uses OpenType fonts the accent over œ and Œ has been set with a macro; you perceive the difference with regular accented chars, because the accent is set in a slightly higher position; the macros have not been corrected, so as to show that these glyphs are not precomposed.

languages where the new derived words did not distinguish any more between long and short vowels, but used a stress, generally on what remained of a long vowel. With this in mind, it is difficult to think about diphthongs and triphthongs in classical Latin; the hyphenation algorithm therefore divides clusters of more than two vowels before what *today* we would interpret and pronounce a couple of vowels as a diphthong, otherwise the hyphenation algorithm divides between any couple of vowels except the couples ‘ae’ and ‘oe’. These vocalic pairs, on their side, might be real diphthongs or hiati; ‘aeris’ (genitive of ‘aer’, air) and ‘aeris’ (genitive of ‘aes’, bronze) are written the same, ma in the former case they for a hiatus, while in the second they for a diphthong. The hyphenation algorithm does not divide either pair.

Classic Latin uses the full 23 letter Latin alphabet, with no ligated diphthongs, no ‘U’ and no ‘’v; latinists pronounce the ‘ae’ and ‘oe’ diphthongs as two vowels, and pronunce the ‘u’ as a vocalic /u/ or a semivowel /w/ according to its position and its adjacency to another vowel. Of course the few infix words in the language definition file must have a different spelling from modern ones, but what is more important, the hyphenation patterns are much different from the modern ones, as explained above.

The rules for classical Latin are taken from 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.

The file `liturgicallatin.ldf` handles a special variant of the Latin language; its spelling is similar to modern latin, but it uses the ligated diphthongs 'æ' and 'œ', and it uses accents even on composed letters such as 'œ'; this sets forth some problems because the precomposed character for œ́ is not defined in the UNICODE standard; therefore it is necessary to use the self combining acute accent (U+0301); unfortunately some fonts do not centre this accent over the 'œ' glyph, so it we provide suitable macros to allow hyphenation in spite of their presence.

The hyphenation rules are similar partly to modern Latin, and partly to classical Latin; therefore they involve a different set of patterns.

See `https://github.com/gregorio-project/hyphen-la` for further documentation, patterns and tests.

2 Usage

With XeLaTeX and LuaLaTeX, polyglossia is the specific package used to deal with languages.

There are two ways to handle Latin with XeLaTeX.

1. Handle three different languages, latin, classiclatin, and liturgicallatin as in this example:

   ```
   \usepackage{polyglossia}
   \setmainlanguage{...}
   \setotherlanguage[babelshorthands]{classiclatin}
   ```

2. Handle a single language, latin, with several variants, as in this example:

   ```
   \usepackage{polyglossia}
   \setmainlanguage{...}
   \setotherlanguage[variant=classic,babelshorthands]{latin}
   ```

The second approach was defined as the first attempt to Latin, then the first approach was defined and proved to be more effective.

Actually the first approach is the one that is necessary to use with LuaLaTeX, and you can manage the Latin varieties as different languages. With the secondo approach, once a certain variant has bees specified, no other variants can be selected; therefore the second approach is sort a fallback for backwards compatibility with a previous version of the language description file. At the same time there are certain options that are not implemented for LuaLaTeX, therefore

`"`	inserts a compound word mark where hyphenation is legal; the next character must not be either a non-letter token or an accented letter (for foreign names and ecclesiastic or liturgical Latin).
`"|`	inserts a compound word mark, but operates also when the next token is not a letter or it is an accented character. This shorthand is required when it is necessary to specify a break *before* and accented diphthongs ǽ and œ́.
`'a`	inserts an *accented* ǽ ligated diphthong in liturgical Latin, without the need of curly braces; hyphenation is allowed after this accented ligature by means of an implied discretionary break command.
`'o`	operates the same as the preceding shorthand, but inserts an accented œ́ diphthong.

Table 2: Shorthands defined for the Latin language.

the full `latin.ldf` with all its variant options ha some utility by itself, although some options can be used only with XeLaTeX.

For example, in a modern Latin grammar written for English students English is set as the main language; if it was necessary to make examples with citations or stretches of text in modern Latin and in liturgical Latin, with *both* XeLaTeX and LuaLaTeX you would do the following:

```
\documentclass[...]{...}
...
\usepackage{polyglossia}
\setmainlanguage{english}
\setotherlanguage[babelshorthands]{latin}
\setotherlanguage[babelshorthands]{liturgicallatin}
...
\begin{document}
...
 A modern Latin  version of Julius Caesar's \emph{De bello gallico}
 would appear as follows.
\begin{quote}\begin{latin}
 Gallia est omnis divisa in partes tres, quarum unam
 incolunt Belgae, aliam Aquitani, tertiam qui ipsorum
 lingua Celtae, nostra Galli appellantur.
\end{latin}\end{quote}

Liturgical Latin would have instead the following text.
\begin{quote}\begin{liturgiccallatin}
 Gállia est omnis divisa in partes tres, quarum unam
 íncolunt Belgæ, áliam Aquitani, tértiam qui ipsorum
 lingua Celtæ, nostra Galli appellantur.
```

	XeLaTeX	LuaLaTeX
Short hands	*babelshorthnads=false* babelshorthands babelshorthands=true	*babelshorthnads=false* babelshorthands babelshorthands=true

Variants	*variant=modern*
	variant=medieval
	variant=classic
	variant=liturgical
		variant=modern
		variant=medieval (See table 4)

Ecclesiastic	*ecclesiastic=false* ecclesiastic ecclesiastic=true	*ecclesiastic=false*

Table 3: Differences between XeLaTeX and LuaTeX behaviour when using the options of file `gloss-latin.ldf`. Items in italics are the defaults.

Language	XeLaTeX	LuaLaTeX
Modern	gloss-latin	gloss-latin
Medieval	gloss-latin + variant=medieval	gloss-latin + variant=medieval
Classic	gloss-classiclatin	gloss-classiclatin
Liturgical	gloss-liturgicallatin	gloss-liturgicallatin

Table 4: Use of `gloss-latin/ldf`, `gloss-classiclatin.ldf`, and `gloss-liturgicallatin.ldf` when using single languages

```
\end{classiclatin}\end{quote}
...
\end{document}
```

It is worth repeating that with LuaLaTeX it is necessary to use only the first procedure described for XeLaTeX. Please see table 3. The first procedure may be used also with XeLaTeX so as to be able to use in the same document several variants of Latin, as it is shown in table 4.

Each language definition file provides the babelshorthands option in order to use the double straight quote as an active character that performs special actions as described in table 2. The option withprosodicmarks, defined in the analogous language description file for babel, was intended for easing the setting of the macron and breve signs over single vowels, but since *pdfLaTeX* handles only 8-bit encoded fonts, none of which includes the ten vocalic glyphs with macrons and breves, it is necessary to build up such graphemes. Such artifices are not necessary with XeLaTeX and LuaLaTeX that use UNICODE encoded fonts, where the ten glyphs already exist.

At the same time accents on the Latin diphthongs are not generally easy to set with a national keyboard, therefore these files provide the active 'acute' accent (actually the usual ASCII single straight quote), so that it is simpler to write `'a` to set ǽ, rather than writing `\'{\ae}` or `\'\ae␣` in order to avoid attaching the 'ǽ' glyph to the following letters.

This self extracting documented TEX file produces both this documentation and the desired three language description files to be used with XeLaTeX and LuaLaTeX.

3 The code

This section defines three language description files.

1. File `gloss-latin.ldf` is for typeseting modern and medieval latin, possibly with the `ecclesiastic` settings.
2. File `gloss-classiclatin.ldf` is for typesetting classic Latin.
3. File `gloss-liturgicallatin.ldf` is for typeseting liturgical Latin .

All these three language description files accept the option `babelshorthands=true`, or simply `babelshorthands`; they have available the special features of the active double quote and, except for `gloss-classiclatin`, the active single quote (ASCII apostrophe or single straight quote).

Only `gloss-latin.ldf` accepts the `ecclesiastic=true` or simply `ecclesiastic`, option which is not available with the other two language description files.

While using XeLaTeX it is possible to specify the spelling and hyphenation variety with an option, `variant=`⟨*language*⟩, where ⟨*language*⟩ may be any of the five ones listed in table 1; if no variant is specified, the modern one is the default one.

While using LuaLaTeX this possibility is not available (except possibly) for `variant=medieval`; furthermore, even if it is possible to specify the `ecclesiastic` option, nothing particular is done, because the special punctuation and footnote styles have not been implemented for LuaLaTeX (and possibly they will never be).

The LuaLaTeX approach is interesting because it is possible to mix the various language variants in the same document by simply selecting the language. This approach is possible also with XeLaTeX by by omitting to specify the classic or the liturgic variant name, but by specifying the specific language names as "other languages".

3.1 Code for gloss-latin

We start with the polyglossia settings of the main language, in particular by selecting the pattern file with modern phonetic hyphenation rules.

```
% !TEX encoding = UTF-8 Unicode
\PolyglossiaSetup{latin}{%
      hyphennames={latin},
      hyphenmins={2,2},
      frenchspacing=true,
      fontsetup=true,
}
```

We define a macro that sets the proper values for the upper- and lower-casing of the letters ‘u’ and ‘V’; this macro will be used for the medieval and classic variants for the proper processing of the infix words and those strings that are uppercased by certain classes, for example the contents of headers by the standard classes. For the document text it is the users’ responsibility to use the proper spelling for these variants.

```
\def\classicuclccodes{\lccode`\V=`\u \uccode`\u=`\V}
\def\noclassicuclccodes{\lccode`\V=`\v \uccode`\u=`\U}
%
```

We then define some service macros containing the names of the variants in order to perform the necessary tests; we exclude such definitions when the *luatex* engine is being used, because, as explained in the documentation, *luatex* loads one pattern set per language, therefore changing pattern set according to the language variant specified becomes meaningless. It is convenient to define switches to distinguish modern form medieval or classic Latin spelling. These names are not very meaningful, but they are maintained for backwards compatibility; in practice switch `\ifmedieval` handles the treatment of ‘æ’ and ‘œ’ while `\ifclassic` handles the treatment of the ‘u’ spelling in infix words. For modern Latin they are both false; for medieval Latin they are both true’ for classic Latin `\ifmedieval` is false, while `\ifclassic` is true; for liturgical Latin `\ifmedieval` is true while `\ifclassic` is false.

```
\def\tmp@modern{modern}
\def\tmp@medieval{medieval}
\unless\ifluatex
  \def\tmp@classic{classic}
  \def\tmp@liturgical{liturgical}
\fi
\newif\ifmedieval\medievalfalse
\newif\ifclassic\classicfalse
```

For the ecclesiatic option we need to set a kvalue boolean key such that when the option is specified the `true` value is assumed. The initial setting of this key, though, is `false` so that without specifying this option no ecclesiastic typesetting parameters are set.

```
\define@boolkey{latin}[latin@]{ecclesiastic}[true]{}


```

Now we start examining the variant options. We start by setting the default language value to the `\latin@variant` macro; we then test if the *luatex* engine is being used and if the latin pattern set has been loaded; if so a special LuaTeX macro is used to define the `latin` language functionalities

```
\let\latin@variant\l@latin
\ifluatex
  \ifcsname l@latin\endcsname\xpg@set@language@luatex@ii{latin}\fi
\fi
\def\captionslatin{\latincaptions}%
\def\datelatin{\latindate}%
```

At this point we are ready for testing the variants vs the specified one; first we define the *keyvalue* key with the default value `modern`, then we proceed with the tests. We start with `variant=medieval` where we we set the `\ifmedieval` and `\ifclassic` switches that are used within the captions and date definitions, and we use the already defined macro `\classicuclccodes` in order to set the proper correspondence between the lower and upper case letters ‘u’ and ‘V’.

```
\define@key{latin}{variant}[modern]{%
\def\@tempa{#1}%
%************* medieval
\ifx\@tempa\tmp@medieval
  \ifluatex
    \ifcsname l@latin\endcsname\xpg@set@language@luatex@ii{latin}\fi
  \fi
  \let\latin@variant\l@latin
  \xpg@set@language@luatex@ii{latin}
  \medievaltrue \classictrue
  \classicuclccodes
  \xpg@info{Option: Medieval Latin}%
\else
```

Then we test the `classic` variant; we proceed in a similar way; but for backwards compatibility we test to check if the suitable classic Latin patterns have been loaded; for XeLaTeX this should be redundant, but in past distributions of the TeX system, such patterns were not available.

```
%************* classic
  \ifx\@tempa\tmp@classic
    \unless\ifluatex
      \unless\ifcsname l@classiclatin\endcsname
         \xpg@nopatterns{Classic Latin}%
         \adddialect\l@classiclatin\l@latin
         \let\latin@variant\l@latin
      \else
         \let\latin@variant\l@classiclatin
      \fi
    \fi
    \medievalfalse\classictrue\classicuclccodes
    \xpg@info{Option: Classic Latin}%
  \else
```

Eventually we do similar actions for the liturgical Latin variant; again we must check for backwards compatibility, since in past distributions liturgical latin patterns were not available.

```
%************** liturgical
   \ifx\@tempa\tmp@liturgical\unless\ifluatex
      \unless\ifcsname l@liturgicallatin\endcsname
         \xpg@nopatterns{Liturgical Latin}%
         \adddialect\l@liturgicallatin\l@latin
         \def\latin@variant{\l@latin}%
      \else
```

```
        \let\latin@variant\l@liturgicallatin
      \fi
        \medievaltrue\classicfalse
        \xpg@info{Option: Liturgical Latin}\fi
    \else
```

Even if the `modern` variant is the default one, we must test also for this variant in case the users specified this value to the key `variant`. Some of this code might be redundant, but it is better than nothing.

```
%************* modern
      \ifx\@tempa\tmp@modern
        \let\latin@variant\l@latin
        \ifluatex\xpg@set@language@luatex@ii{latin}\fi
        \xpg@info{Option: Modern Latin}%
      \else
        \def\latin@variant{\l@nohyphenation}%
        \PackageWarning{polyglossia}{%
          *******************\MessageBreak
          No hyphenation set for \@tempa
          *******************\MessageBreak
        }{}%
      \fi
    \fi
  \fi
\fi
}

```

After these different language variants we need to assign the correct value to the internal macro `\latin@language`.

```
\def\latin@language{\language=\latin@variant}%
```

At this point we define the option ecclesiastic; for XeLaTeX we use a set of little known commands to define character classes to be handled the same way; for LuaLaTeX we do not do the same, therefore the ecclesiastic option with LuaLaTeX remains ineffective, without even a 'Warning' message. The main reason is that liturgical Latin is used mainly for typesetting liturgical texts containing many hymns, psalms, and such musical contents, often composed with the GregorioTeX package; in such books the French like punctuation and footnote styles are often attentively avoided, therefore the ecclesiastic style is not used. Since GregorioTeX works only with LuaLaTeX, the `ecclesiastic` option appears to be superfluous. In any case there is always the possibility to typeset in the ecclesiastic style with XeLaTeX.

Let us remember that the `\iflatin@ecclesiastic` switch is set to `true` only at runtime when the Latin language is selected and the ecclesiastic option is specified. The definitions that follow are skipped if the document is typeset with Lua(La)Tex, but they are defined when XelaTeX is being used since the special internals necessary to do the job are available only with XeLaTeX.

We start defining a long named macro `\ecclesiasticlatin@punctuation`; its substitution text contains some macros that set all the spacing parameters; such macro will be used in practice to set all the necessary parameters; its opposite macro `\noecclesiasticlatin@punctuation` is used to reset all parameters.

Notice that three new 'intercharclasses' are defined and the various punctuations marks involved are assigned to such classes; then the tokens to be inserted between chars of different classes are set according to the classes they belong to.

```
\ifluatex
      \PackageWarning{polyglossia}{\MessageBreak\MessageBreak
      *****************\MessageBreak
      The ecclesiastic option is not active\MessageBreak
      when typesetting with LuaLaTeX\MessageBreak
      *****************\MessageBreak
      \MessageBreak}{}
  \else
% The following code is disabled for use with luatex
%**************************************************
   \def\ecclesiasticlatin@punctuation{%
      \def\xpg@unskip{\ifhmode\ifdim\lastskip>\z@\unskip\fi\fi}
      \lccode\string"2019=\string"2019
%
      \newXeTeXintercharclass\ecclesiasticlatin@punctthin
      \newXeTeXintercharclass\ecclesiasticlatin@punctguillstart
      \newXeTeXintercharclass\ecclesiasticlatin@punctguillend
      \XeTeXinterchartokenstate=1
      \XeTeXcharclass `\! \ecclesiasticlatin@punctthin
      \XeTeXcharclass `\? \ecclesiasticlatin@punctthin
      \XeTeXcharclass `\; \ecclesiasticlatin@punctthin
      \XeTeXcharclass `\: \ecclesiasticlatin@punctthin
      \XeTeXcharclass `\« \ecclesiasticlatin@punctguillstart
      \XeTeXcharclass `\» \ecclesiasticlatin@punctguillend
      \XeTeXinterchartoks \z@ \ecclesiasticlatin@punctthin = {\penalty\@M
      \hskip.2\fontdimen2\font \@plus\z@\@minus\z@}%
      \XeTeXinterchartoks 255 \ecclesiasticlatin@punctthin = {\xpg@unskip}
      \XeTeXinterchartoks \ecclesiasticlatin@punctguillstart \z@ = {\penalty\@M
      \hskip.2\fontdimen2\font \@plus\z@\@minus\z@\ignorespaces}
      \XeTeXinterchartoks \z@ \ecclesiasticlatin@punctguillend = {\xpg@unskip
      \penalty\@M\hskip.2\fontdimen2\font \@plus\z@\@minus\z@}
    }

   \def\noecclesiasticlatin@punctuation{%
      \lccode\string"2019=\z@
%
        \XeTeXcharclass `\! \z@
        \XeTeXcharclass `\? \z@
        \XeTeXcharclass `\; \z@
        \XeTeXcharclass `\: \z@
        \XeTeXcharclass `\« \z@
        \XeTeXcharclass `\» \z@
        \XeTeXinterchartokenstate=0
      }
```

Eventually we redefine certain internal macros in order to set footnotes the way that has been established for this ecclesiastic footnote style.

```
    \let\latin@original@makefntext\@makefntext
    \newcommand\latin@ecclesiastic@makefntext[1]{%
        \parindent 1em \noindent
        \latin@Makefnmark{\enspace #1}}
    \newcommand\latin@Makefnmark{\hbox{\normalfont\@thefnmark.}}
\fi
\setkeys{latin}{variant,ecclesiastic=false}
```

This is the right position to define the various captions and dates; notice that these are similar for medieval and classical Latin, and, respectively, for modern and liturgical Latin; therefore we handle these similarities by means of the switches `\ifmedieval` and `\ifclassic` without any duplication of code.

```
\def\latincaptions{%
   \def\prefacename{\ifmedieval Præfatio\else Praefatio\fi}%
   \def\refname{Conspectus librorum}%
   \def\abstractname{Summarium}%
   \def\bibname{Conspectus librorum}%
   \def\chaptername{Caput}%
   \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{Additur}%
   \def\ccname{Exemplar}%
   \def\headtoname{\ignorespaces}%
   \def\pagename{charta}%
   \def\seename{cfr.}%
   \def\alsoname{cfr.}%
   \def\proofname{Demonstratio}%
   \def\glossaryname{Glossarium}%
   }

\def\latindate{%
   \def\today{\uppercase\expandafter{\romannumeral\day}%
      \space \ifcase\month%
      \or Januarii\or Februarii\or Martii\or Aprilis\or Maji\or
      Junii\or Julii\or Augusti\or Septembris\or Octobris\or
        \ifclassic Nouembris\else Novembris\fi
      \or Decembris\fi%
      \space \uppercase\expandafter{\romannumeral\year}}}
```

We now define the settings and the definitions for the few babel-style shorthands that are used with Latin. First we need some settings useful for polyglossia. Actually it is possible that the module for the babel-style shorthands gets loaded by default, but all language `gloss-*.ldf` files that use such shorthands, make these tests.

```
%%%%%%%%% Latin shorthands

\define@boolkey{latin}[latin@]{babelshorthands}[true]{}

\ifsystem@babelshorthands
  \setkeys{latin}{babelshorthands=true}
\else
  \setkeys{latin}{babelshorthands=false}
\fi
```

After this we can initialise the active chars `"` and `'`.

```
\ifcsundef{initiate@active@char}{%
\input{babelsh.def}%
\initiate@active@char{"}%
\initiate@active@char{'}%
}{}
```

We define both `\latin@shorthands` and `\nolatin@shorthands` to be used in the latin-extras and -noextras macros in order to turn on and off such shorthands whenever a language change takes place. The service macros `\xpgla@cwm` and `xpgla@putacute` are actually the ones that do the respective jobs, but the active `"` and `'` must refer to them through other service macros subject to the math mode testing so as to behave in a different way while in math mode vs text mode.

```
\def\latin@shorthands{%
  \def\language@group{latin}%
  \bbl@activate{"}%
  \declare@shorthand{latin}{"}{\relax
    \ifmmode
      \def\xpgla@nextdq{''}%
    \else
      \def\xpgla@nextdq{\futurelet\xpgla@temp\xpgla@cwm}%
    \fi
  \xpgla@nextdq}%
%
  \bbl@activate{'}%
  \declare@shorthand{latin}{'}{\relax
    \ifmmode
      \def\xpgla@nextsq{'}%
    \else
      \def\xpgla@nextsq{\futurelet\xpgla@temp@A\xpgla@putacute}%
    \fi
  \xpgla@nextsq}%
}
```

We need also a `\xpgla@allowhyphens` resorting to the babel system macro `\bbl@allowhyphens` in order to use a breakable hyphen sign. In facts we remember the normal hyphen sign forms a legal line break point, but both word connected by the hyphen sign cannot be hyphenated; on the opposite we want a situation where both words connected by the hyphen may be hyphenated.

```
\def\xpgla@allowhyphens{\bbl@allowhyphens
        \discretionary{-}{}{}\bbl@allowhyphens}
```

Now we make a multistage set of conditionals: first we test if the next character is a letter; if it is a breakable hyphen point is inserted; otherwise if it is a ligated diphthong æ this is treated as a normal ASCII letter; otherwise the same test is made to test if the next letter is a ligated diphthong œ; otherwise eventually the next char is tested against `|` the discretionary break is inserted, but the `|` is gobbled. Through these operations we can insert into a word a discretionary break, without forbidding other legal line breaks.

```
\newcommand*{\xpgla@cwm}{\let\xpgla@@nextdq\relax
  \ifcat\noexpand\xpgla@temp a%
    \let\xpgla@@nextdq\xpgla@allowhyphens
  \else
    \ifx\xpgla@temp\ae
        \let\xpgla@@nextdq\xpgla@allowhyphens
    \else
        \ifx\xpgla@temp\oe
           \let\xpgla@@nextdq\xpgla@allowhyphens
        \else
           \if\noexpand\xpgla@temp\string|%
              \def\xpgla@@nextdq{\xpgla@allowhyphens\@gobble}%
           \fi
        \fi
    \fi
  \fi
  \xpgla@@nextdq}%
```

For the active ‘apostrophe’, to be used as an acute accent, we test the first char after the active apostrophe; if it is an `a` or an `o` or an æ or an œ, we gobble it and insert into the input reading stream the proper accented ligated diphthong; actually we insert the proper unaccented diphthong followed by the self combining acute accent with UNICODE address `0301`. Beware: this procedure is technically correct, but the result sometimes is not so good as expected, because it depends on the OpenType fonts being used. With the OpenType UCM (Unicode encoded Computer Modern) and other fonts the accent sets itself over the middle of the base diphthong; with other fonts it sets itself on the ‘e’ part of the diphthong; it is acceptable, but certainly on the middle is much better. Therefore we avoid setting the accent this way on the æ diphthong but, if it’ is available in the font, use the correct code point for the accented ǽ[1]; on the œ and Œ diphthongs this is impossible, because there is no code point in UNICODE pointing to œ́ and Œ́.

[1]You may use also the traditional TEX way by inserting `\'\ae␣` or `\'{\ae}` in the source file. With XeLaTeX and LuaLaTeX, when dealing with œ, this used to fail, because the traditional TEX accent macros are mapped to real code points, and there is no code point for œ́. Actually, when using a well updated TeXLive installation, with the lowercase œ it works the same as when the self combining acute accent `"0301` is used, possibly shifting the accent on the ‘e’ component of the ligature; but this does not work with the capital Œ ligature; this is why in table 1 we used a macro to set the acute accent on both glyphs.

```
\def\xpgla@putacute#1{\let\xpgla@nextsq\relax%
\if a\xpgla@temp@A
  æ\kern-0.175em^^^^0301\kern0.175em\xpgla@allowhyphens
\else
\if o\xpgla@temp@A
  œ\kern-0.175em^^^^0301\kern0.175em\xpgla@allowhyphens
\else
  \if æ\xpgla@temp@A
    æ^^^^0301%
  \else
    \if œ\xpgla@temp@A
      œ^^^^0301%
    \else
      \string'%
    \fi
  \fi
\fi
\fi}%

\def\nolatin@shorthands{%
  \@ifundefined{initiate@active@char}{}{\bbl@deactivate{"}}%
  \@ifundefined{initiate@active@char}{}{\bbl@deactivate{'}}%
}
```

Eventually we save the settings for Latin into the `\xpgla@savedvalues` macro, delaying the operation until the end of the preamble, so we are sure that the correct values are saved, since the default ones might have been changed by the user or by the used document class.

```

\let\xpgla@savedvalues\empty
\AtEndPreamble{%
  \edef\xpgla@savedvalues{%
    \clubpenalty=\the\clubpenalty\space
    \@clubpenalty=\the\@clubpenalty\space
    \widowpenalty=\the\widowpenalty\space
    \finalhyphendemerits=\the\finalhyphendemerits}
}
```

The final actions are to define the `\noextras@latin`, `\blockextras@latin` and `\inlineextras@latin` so as to set or reset the Latin settings when typesetting a stretch of Latin text.

```
\def\noextras@latin{%
   \lccode\string"2019=\z@
   \nolatin@shorthands
   \xpgla@savedvalues
   \noclassicuclccodes
  \iflatin@ecclesiastic
    \unless\ifluatex\noecclesiasticlatin@punctuation
    \let\@makefntext\latin@original@makefntext\fi
  \fi
}

\def\blockextras@latin{%
   \lccode\string"2019=\string"2019
   \clubpenalty=3000 \@clubpenalty=3000 \widowpenalty=3000
   \finalhyphendemerits=50000000
   \iflatin@babelshorthands\latin@shorthands\fi
   \iflatin@ecclesiastic\unless\ifluatex\ecclesiasticlatin@punctuation
   \let\@makefntext\latin@ecclesiastic@makefntext\fi
   \fi
}

\def\inlineextras@latin{%
   \lccode\string"2019=\string"2019
   \iflatin@babelshorthands\latin@shorthands\fi
   \iflatin@ecclesiastic
      \unless\ifluatex\ecclesiasticlatin@punctuation
      \let\@makefntext\latin@ecclesiastic@makefntext\fi
   \fi
}
```

3.2 Code for gloss-classiclatin

The standalone language description file for polyglossia `gloss-classiclatin.ldf` is very similar to the "plain" latin one, except for the following differences.

1. This file is a standalone one, and does not deal with other Latin varieties. This implies that no options are set, except those for the babel-style shorthands.
2. Only one pattern file is being loaded, therefore there is no need to save settings for other hyphenation patterns. And there is no need to make any tests to check the specific pattern file name; either it exists, or it is replaced by the `nohyphenation` pseudo language, as it is customary with polyglossia.
3. The uccodes are those specific for classic Latin where only 'u' is used for lower case and only 'V' is used for upper case and this has an influence on the spelling of dates; on the opposite 'captions' are equal to those of modern Latin, because the ligated diphthongs are not used at all in classical spelling.

The `\PolyglossiaSetup` differs from the "plain" Latin one because only the `classiclatin` language is called with the specific name used to associate a language to a specific pattern file as it is done in the language setting files `language.dat`, `language.def`, and `language.dat.lua`.

```
% !TEX encoding = UTF-8 Unicode
\PolyglossiaSetup{classiclatin}{%
     hyphennames={classiclatin},
     hyphenmins={2,2},
     frenchspacing=true,
     fontsetup=true,
}
```

```
\def\classicuclccodes{\lccode`\V=`\u \uccode`\u=`\V}
\def\noclassicuclccodes{\lccode`\V=`\v \uccode`\u=`\U}
```

Then the captions and date are defined.

```
\def\classiclatincaptions{%
   \def\prefacename{Praefatio}%
   \def\refname{Conspectus librorum}%
   \def\abstractname{Summarium}%
   \def\bibname{Conspectus librorum}%
   \def\chaptername{Caput}%
   \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{Additur}%
   \def\ccname{Exemplar}%
   \def\headtoname{\ignorespaces}%
   \def\pagename{charta}%
   \def\seename{cfr.}%
   \def\alsoname{cfr.}%
   \def\proofname{Demonstratio}%
   \def\glossaryname{Glossarium}%
   }

\def\classiclatindate{%
   \def\today{\uppercase\expandafter{\romannumeral\day}%
      \space \ifcase\month
      \or Januarii\or Februarii\or Martii\or Aprilis\or Maii\or Junii\or
      Julii\or Augusti\or Septembris\or Octobris\or Nouembris\or
      Decembris\fi
      \space \uppercase\expandafter{\romannumeral\year}}}
```

Then, again, the necessary tests are made for setting the babel shorthands for the proper definition of the babelshorthands option.

```

\define@boolkey{classiclatin}[classiclatin@]{babelshorthands}[true]{}

\ifsystem@babelshorthands
  \setkeys{classiclatin}{babelshorthands=true}
\else
  \setkeys{classiclatin}{babelshorthands=false}
\fi

```

Again the ASCII straight double quote " is initialised as an active character.

```
\ifcsundef{initiate@active@char}{%
    \input{babelsh.def}\initiate@active@char{"}}{}
```

```

```

And the definitions for the active " are repeated as for “plain” Latin.

```
\def\classiclatin@shorthands{%
  \def\language@group{classiclatin}%
  \bbl@activate{"}%
  \declare@shorthand{classiclatin}{"}{\relax
    \ifmmode
      \def\xpgcla@next{''}%
    \else
      \def\xpgcla@nextdq{\futurelet\xpgla@temp\xpgla@cwm}%
    \fi
  \xpgcla@nextdq}%
}

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

As in “plain” Latin the `\noclassiclatin@shorthands` is defined so as to use it within the setting and unsetting of the language parameters.

```
\def\noclassiclatin@shorthands{%
  \@ifundefined{initiate@active@char}{}{\bbl@deactivate{"}}%
}

\let\xpgcla@savedvalues\empty
\AtEndPreamble{%
  \edef\xpgcla@savedvalues{%
    \clubpenalty=\the\clubpenalty\space
    \@clubpenalty=\the\@clubpenalty\space
    \widowpenalty=\the\widowpenalty\space
    \finalhyphendemerits=\the\finalhyphendemerits}%
}

\def\noextras@classiclatin{%
   \lccode\string"2019=\z@
   \noclassiclatin@shorthands
   \noclassicuclccodes
   \xpgcla@savedvalues
}

\def\blockextras@classiclatin{%
   \lccode\string"2019=\string"2019
   \clubpenalty=3000 \@clubpenalty=3000 \widowpenalty=3000
   \finalhyphendemerits=50000000
   \classicuclccodes
   \ifclassiclatin@babelshorthands\classiclatin@shorthands\fi
}

\def\inlineextras@classiclatin{%
   \lccode\string"2019=\string"2019
   \classicuclccodes
   \ifclassiclatin@babelshorthands\classiclatin@shorthands\fi
}
```

3.3 Code for gloss-liturgicallatin

For liturgical Latin we do similar things as we did for "plain" Latin in the previous section; the main differences are the following.

1. This file is a standalone one, and does not deal with other Latin varieties. This implies that no options are set, except those for the babel-style shorthands.
2. Only one pattern file is being loaded, therefore there is no need to save settings for other hyphenation patterns. And there is no need to make any tests to check the specific pattern file name; either it exists, or it is replaced by the `nohyphenation` pseudo language, as it is customary with polyglossia.
3. The uccodes are identical with the modern Latin ones, while the 'captions' are equal to those of medieval Latin, because of the widespread use of the ligated diphthongs.

The `\PolyglossiaSetup` differs from the "plain" Latin one because only the `liturgicallatin` language is called with the specific name used to associate a language to a specific pattern file as it is done in the language setting files `language.dat`, `language.def`, and `language.dat.lua`.

```
% !TEX encoding = UTF-8 Unicode
\PolyglossiaSetup{liturgicallatin}{%
     hyphennames={liturgicallatin},
     hyphenmins={2,2},
     frenchspacing=true,
     fontsetup=true,
}
```

Then the captions and date are defined.

```
\def\liturgicallatincaptions{%
```

```
  \def\prefacename{Præfatio}%
  \def\refname{Conspectus librorum}%
  \def\abstractname{Summarium}%
  \def\bibname{Conspectus librorum}%
  \def\chaptername{Caput}%
  \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{Additur}%
  \def\ccname{Exemplar}%
  \def\headtoname{\ignorespaces}%
  \def\pagename{charta}%
  \def\seename{cfr.}%
  \def\alsoname{cfr.}%
  \def\proofname{Demonstratio}%
  \def\glossaryname{Glossarium}%
  }

\def\liturgicallatindate{%
  \def\today{\uppercase\expandafter{\romannumeral\day}%
     \space \ifcase\month%
     \or Januarii\or Februarii\or Martii\or Aprilis\or Maji\or Junii\or%
     Julii\or Augusti\or Septembris\or Octobris\or Novembris\or%
     Decembris\fi%
     \space \uppercase\expandafter{\romannumeral\year}}}

```

Then, again, the necessary tests are made for setting the babel shorthands for the proper definition of the babelshorthands option.

```
\define@boolkey{liturgicallatin}[liturgicallatin@]{babelshorthands}[true]{}

\ifsystem@babelshorthands
  \setkeys{liturgicallatin}{babelshorthands=true}
\else
  \setkeys{liturgicallatin}{babelshorthands=false}
\fi

```

Again the " and ' characters are initialised as active ones.

```
\ifcsundef{initiate@active@char}{%
    \input{babelsh.def}%
    \initiate@active@char{"}%
    \initiate@active@char{'}%
}{}
```

And the definitions for the active " and ' characters are repeated as for “plain”

Latin.

```
\def\liturgicallatin@shorthands{%
  \def\language@group{liturgicallatin}%
  \bbl@activate{"}%
  \declare@shorthand{liturgicallatin}{"}{\relax
    \ifmmode
      \def\xpglla@next{''}%
    \else
      \def\xpglla@nextdq{\futurelet\xpglla@temp\xpglla@cwm}%
    \fi
  \xpglla@nextdq}%
  \bbl@activate{'}%
  \declare@shorthand{liturgicallatin}{'}{\relax
    \ifmmode
      \def\xpglla@nextsq{'}%
    \else
      \def\xpglla@nextsq{\futurelet\temp@A\xpglla@putacute}%
    \fi
  \xpgla@nextsq}%
}

\def\xpglla@allowhyphens{\bbl@allowhyphens
     \discretionary{-}{}{}\bbl@allowhyphens}

\newcommand*{\xpglla@cwm}{\let\xpglla@@nextdq\relax
  \ifcat\noexpand\xpglla@temp a%
    \let\xpglla@@nextdq\xpglla@allowhyphens
  \else
    \ifx\xpglla@temp\ae
        \let\xpglla@@nextdq\xpglla@allowhyphens
    \else
        \ifx\xpglla@temp\oe
           \let\xpglla@@nextdq\xpglla@allowhyphens
        \else
           \if\noexpand\xpglla@temp\string|%
              \def\xpglla@@nextdq{\xpglla@allowhyphens\@gobble}%
           \fi
        \fi
    \fi
  \fi
  \xpglla@@nextdq}%

\def\xpglla@putacute#1{\let\xpglla@nextsq\relax%
\if a\xpglla@temp@A
  æ\kern-0.175em^^^^0301\kern0.175em\xpglla@allowhyphens
\else
\if o\xpglla@temp@A
  œ\kern-0.175em^^^^0301\kern0.175em\xpglla@allowhyphens
\else
  \if æ\xpglla@temp@A
    æ^^^^0301%
  \else
    \if œ\xpglla@temp@A
      œ^^^^0301%
    \else
      \string'%
    \fi
  \fi
\fi
\fi}%
```

As in “plain” Latin the `\noliturgicallatin@shorthands` is defined so as to use it within the setting and unsetting of the language parameters.

```
\def\noliturgicallatin@shorthands{%
  \@ifundefined{initiate@active@char}{}{\bbl@deactivate{"}}%
  \@ifundefined{initiate@active@char}{}{\bbl@deactivate{'}}%
}

\let\xpglla@savedvalues\empty
\AtEndPreamble{%
  \edef\xpglla@savedvalues{%
    \clubpenalty=\the\clubpenalty\space
    \@clubpenalty=\the\@clubpenalty\space
    \widowpenalty=\the\widowpenalty\space
    \finalhyphendemerits=\the\finalhyphendemerits}%
}

\def\noextras@liturgicallatin{%
   \lccode\string"2019=\z@
   \noliturgicallatin@shorthands
   \xpglla@savedvalues
}

\def\blockextras@liturgicallatin{%
   \lccode\string"2019=\string"2019
   \clubpenalty=3000 \@clubpenalty=3000 \widowpenalty=3000
   \finalhyphendemerits=50000000
   \ifliturgicallatin@babelshorthands\liturgicallatin@shorthands\fi
}

\def\inlineextras@liturgicallatin{%
   \lccode\string"2019=\string"2019
   \ifliturgicallatin@babelshorthands\liturgicallatin@shorthands\fi
}
```