

1 The Latin language

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

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

0. Modern spelling and hyphenation; this is the default setting.
1. Medieval spelling and hyphenation: in this spelling the lower and upper case alphabets become the following:
a æ b c d e f g h i k l m n o œ p q r s t u x y z
A Æ B C D E F G H I K L M N O Æ P Q R S T V X Y Z
It must be noted that this file description language does not make any spelling correction in order 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 classes) the correct capitalisation is performed and ‘u’ is capitalised to ‘V’; the reverse takes place when transforming to lowercase.
2. Classical Latin and classical hyphenation; the spelling is similar to the medieval one, except for the ligatures æ, Æ, œ and Æ that are missing; again it is the typesetter’s responsibility to input the text to be typeset in a correct way; while the hyphenation patterns for modern and medieval Latin form a non contrasting set of patterns, this is impossible with classical Latin and a different pattern set has been prepared as explained below.
3. Modern spelling with prosodic marks: this is to be used mainly in modern grammars and dictionaries. Its settings can be turned on and off according to the user needs.
4. Ecclesiastic Latin is a spelling variety of modern Latin; it is taken care by the external module `ecclesiastic.sty`, that is being loaded by this language description file at the right moment, that is when `babel` has been completely read together with all the options and modifiers. This special spelling of modern Latin is used in theological and devotional books of the Roman Catholic clergy where ligatures æ and œ are widely used and where acute accents are used in order to mark the tonic vowel of many words so as to allows clergy with different backgrounds and mother languages to recite

¹The file described in this section has version number v.3.5 and was last revised on 2015/08/20. The original author is Claudio Beccari.

their community prayers with the same rhythmic scheme. This ecclesiastic Latin spelling contains also other differences, including a certain degree of “frenchisation” of spaces around some punctuation marks and guillemots, and the typesetting style of footnotes. When this spelling is chosen, its features are irrevocable, especially those that concern spacing. And this is the main reason why we decided not to include the ecclesiastic style modifications into the language description file, thus avoiding to mess up any setting for the other Latin typesetting styles.

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

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

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

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

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

The typesetting style `withprosodicmarks` is defined in order to use special shorthands for inserting breves and macrons while typesetting grammars, dictionaries, teaching texts, and the like, where prosodic marks are important for the

complete information on the words or the verses. The shorthands, listed in table 1 and described in subsection 2.1, may interfere with other packages; therefore by default this style is turned off and no interference is introduced. If this style is used and interference is experienced, there are special commands for turning on and off its specific shorthand commands.

For what concerns `babel` and typesetting with \LaTeX , the differences between the spelling styles reveal themselves in the strings used to name, for example, the “Preface” that becomes “Praefatio” or “Præfatio” respectively. Hyphenation rules are also different, but the hyphenation pattern file `hyph-la.tex` takes care of the modern and medieval versions of the language, while `hyph-la-x-classic.tex` takes care of the classical hyphenation. The user should not attempt to modify these files, because they are not dealt with by `babel` they are used only during the creation of format files. If some errors or modifications are required or suggested, it is necessary to ask their maintainer.

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

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

- `ˆi` inserts the breve accent as \i ; valid also for the other lowercase vowels, but it does not operate on the medieval ligatures \ae and \oe because they are naturally long.
- `=a` inserts the macron accent as \a ; valid also for the other lowercase vowels, but it does not operate on the medieval ligatures \ae and \oe because they are naturally long.
- `"` 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). With the ecclesiastic spelling it eases the insertion of the ligatures \ae and \oe by simply typing `"ae` and `"oe` without the need of curly brackets or spaces after the character macros `\ae` and `\oe`.
- `"|` inserts a compound word mark, but operates also when the next token is not a letter or it is an accented character.

Table 1: Shorthands defined for the Latin language. The characters `ˆ` and `=` are active only when the language attribute `withprosodicmarks` has been declared, otherwise they are disabled; see subsection 2.1 at page 8 for more details.

The user should attentively remember that this file describes an option for the `babel` package; it works as expected when a document containing a text in part or completely in Latin is being processed with `pdfLaTeX` and 8-bit encoded fonts that contain at most 256 glyphs. `babel` can be used also with `XeLaTeX` and possibly by `LuaLaTeX`, may be with reduced functionalities; expect anomalous results if you typeset your document with this language option and `XeLaTeX`. If you need the functionalities provided by `XeLaTeX` or `LuaLaTeX`, then use `polyglossia` in place of `babel` and set the Latin desired functionalities with suitable values to the `variant` keyword to the Latin language specifications; The valid values have the same names as this `babel` module modifiers, i.e. `medieval`, `classic`, `ecclesiastic`; with `XeLaTeX` and `LuaLaTeX` that use OpenType fonts, the `withprosodicmarks` functionality is not required as the required glyphs can be directly input with a UNICODE input encoding. Actually with such an encoding this modifier is not necessary even with `pdfLaTeX`, but it eases input when one is using a vintage computer or operating system; backwards compatibility is therefore maintained.

2 The code

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

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

Never load this file as if it was a `.sty` file by means of the `\usepackage` command; this practice is deprecated. We check for the existence of `\l@latin` to see whether we have to do something here.

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

Now we declare the `medieval` language attribute.

```
5 \bbl@declare@ttribute{latin}{medieval}{%
6   \addto\captionslatin{\def\prefacename{Pr{\ae}fatio}}%
7   \expandafter\addto\expandafter\extraslatin
8   \expandafter{\extrasmedievallatin}%
9 }
```

The typesetting style `withprosodicmarks` is defined here:

```
10 \bbl@declare@ttribute{latin}{withprosodicmarks}{%
11   \expandafter\addto\expandafter\extraslatin
12   \expandafter{\extraswithprosodicmarks}%
13 }
```

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

loaded, otherwise we issue a warning and keep going with the default modern hyphenation

```

14 \bbl@declare@ttribute{latin}{classic}{%
15   \expandafter\addto\expandafter\extraslatin
16   \expandafter{\extraclassic}%
17 }
18 \ifx\l@classiclatin\undefined
19   \let\l@classiclatin\l@latin
20   \PackageWarningNoLine{babel}{%
21     Attention: hyphenation patterns for language\MessageBreak
22     classiclatin have not been loaded.\MessageBreak
23     I go on using the modern Latin hyphenation patterns.\MessageBreak
24     Please, load the suitable patterns or upgrade your TeX distribution}
25 \fi
26 \addto\extraclassic{\let\l@latin\l@classiclatin}

```

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

The next step consists in defining commands to switch to (and from) the Latin language².

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

```

27 \@namedef{captionslatin}{%
28   \def\prefacename{Praefatio}%
29   \def\refname{Conspectus librorum}%
30   \def\abstractname{Summarium}%
31   \def\bibName{Conspectus librorum}%
32   \def\chaptername{Caput}%
33   \def\appendixname{Additamentum}%
34   \def\contentsname{Index}%
35   \def\listfigurename{Conspectus descriptionum}%
36   \def\listtablename{Conspectus tabularum}%
37   \def\indexname{Index rerum notabilium}%
38   \def\figurename{Descriptio}%
39   \def\tablename{Tabula}%
40   \def\partname{Pars}%
41   \def\enclname{Adduntur}% Or " Additur" ? Or simply Add.?
42   \def\ccname{Exemplar}% Use the recipient's dative
43   \def\headtoname{\ignorespaces}% Use the recipient's dative
44   \def\pagename{Charta}%
45   \def\seename{cfr.}%
46   \def\alsoname{cfr.}% Tabacco never saw "cfr" + 'atque' or similar forms
47   \def\proofname{Demonstratio}%
48   \def\glossaryname{Glossarium}%
49 }

```

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

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

1. the `\enclname` is translated by a passive verb, that literally means “(they) are being added”; if just one enclosure is joined to the document, the plural passive is not suited any more; nevertheless a generic plural passive might be incorrect but suited for most circumstances. On the opposite “Additur”, the corresponding singular passive, might be more correct with one enclosure and less suited in general: what about the abbreviation “Add.” that works in both cases, but certainly is less elegant?
2. The `\headtoname` is empty and gobbles the possible following space; in practice the typesetter should use the dative of the recipient’s name; since nowadays not all such names can be translated into Latin, they might result indeclinable. The clever use of a dative appellative by the typesetter such as “Domino” or “Dominae” might solve the problem, but the header might get too impressive. The typesetter must make a decision on his own.
3. The same holds true for the copy recipient’s name in the “Cc” field of `\ccname`.

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

```
50 \addto\extraslatin{\def\november{Novembris}}
51 \addto\extrasmedievallatin{\def\november{Nouembris}}
52 \addto\extrasclassiclatin{\def\november{Nouembris}}
53 %
54 \def\datelatin{%
55   \def\today{%
56     {\check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont
57       \uppercase\expandafter{\romannumeral\day}}~\ifcase\month\or
58       Ianuarii\or Februarii\or Martii\or Aprilis\or Maii\or Iunii\or
59       Iulii\or Augusti\or Septembris\or Octobris\or \november\or
60       Decembris\fi
61       \space{\uppercase\expandafter{\romannumeral\year}}}}
```

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

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

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

`\noextraslatin`

```
63 \addto\extraslatin{%
64   \babel@savevariable\clubpenalty
```

```

65 \babel@savevariable\@clubpenalty
66 \babel@savevariable\widowpenalty
67 \clubpenalty3000\@clubpenalty3000\widowpenalty3000}

```

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

```

68 \addto\extraslatin{%
69 \babel@savevariable\finalhyphendemerits
70 \finalhyphendemerits50000000}

```

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

```

71 \addto\extrasmedievallatin{%
72 \babel@savevariable{\lccode'\V}%
73 \babel@savevariable{\uccode'\u}%
74 \lccode'\V='\'u \uccode'\u='\'V}
75 \addto\extrasclassiclatin{%
76 \babel@savevariable{\lccode'\V}%
77 \babel@savevariable{\uccode'\u}%
78 \lccode'\V='\'u \uccode'\u='\'V}

```

`\SetLatinLigatures`

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

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

```

79 \addto\extrasmedievallatin{%
80 \babel@savevariable{\lccode'\^^e6}% T1 \ae
81 \babel@savevariable{\uccode'\^^e6}% T1 \ae
82 \babel@savevariable{\lccode'\^^c6}% T1 \AE
83 \babel@savevariable{\lccode'\^^f7}% T1 \oe
84 \babel@savevariable{\uccode'\^^f7}% T1 \OE
85 \babel@savevariable{\lccode'\^^d7}% T1 \OE

```

```

86 \babel@savevariable{\lccode'\^^1a}% OT1 \ae
87 \babel@savevariable{\uccode'\^^1a}% OT1 \ae
88 \babel@savevariable{\lccode'\^^1d}% OT1 \AE
89 \babel@savevariable{\lccode'\^^1b}% OT1 \oe
90 \babel@savevariable{\uccode'\^^1b}% OT1 \OE
91 \babel@savevariable{\lccode'\^^1e}% OT1 \OE
92 \SetLatinLigatures}
93
94 \providecommand\SetLatinLigatures{%
95 \def\@tempA{T1}\ifx\@tempA\fontencoding
96 \catcode'\^^e6=11 \lccode'\^^e6='\^^e6 \uccode'\^^e6='\^^c6 % \ae
97 \catcode'\^^c6=11 \lccode'\^^c6='\^^e6 % \AE
98 \catcode'\^^f7=11 \lccode'\^^f7='\^^f7 \uccode'\^^f7='\^^d7 % \oe
99 \catcode'\^^d7=11 \lccode'\^^d7='\^^f7 % \OE
100 \else
101 \catcode'\^^1a=11 \lccode'\^^1a='\^^1a \uccode'\^^1a='\^^1d % \ae
102 \catcode'\^^1d=11 \lccode'\^^1d='\^^1a % \AE
103 \catcode'\^^1b=11 \lccode'\^^1b='\^^1b \uccode'\^^1b='\^^1e % \oe
104 \catcode'\^^1e=11 \lccode'\^^1e='\^^1b % \OE
105 \fi
106 \let\@tempA\undefined
107 }

```

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

2.1 Latin shorthands

For writing dictionaries or school texts (in modern spelling only) we defined a language attribute or typesetting style, such that a couple of other active characters are defined: `˘` for marking a vowel with the breve sign, and `=` for marking a vowel with the macron sign. Please take notice that neither the OT1 font encoding, nor the T1 one for most vowels, contain directly the marked vowels, therefore hyphenation of words containing these “accents” may become problematic; for this reason the above active characters not only introduce the required accent, but also an unbreakable zero skip that in practice does not introduce a discretionary break, but allows breaks in the rest of the word.

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

`\ProsodicMarksOn` We begin by adding to the normal typesetting style the definitions of the new
`\ProsodicMarksOff`

commands `\ProsodicMarksOn` and `\ProsodicMarksOff` that should produce error messages when this style is not declared:

```

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

```

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

```

118 \catcode'\= \active
119 \catcode'\^ \active

```

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

```

120 \addto\extraslatin{\languageshorthands{latin}}%
121 \addto\extraswithprosodicmarks{\bbl@activate{^}}%
122 \addto\extraswithprosodicmarks{\bbl@activate{=}}%
123 \addto\noextraswithprosodicmarks{\bbl@deactivate{^}}%
124 \addto\noextraswithprosodicmarks{\bbl@deactivate{=}}%
125 \addto\extraswithprosodicmarks{\ProsodicMarks}

```

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

```

126 \def\ProsodicMarks{%
127 \def\ProsodicMarksOn{\catcode'\^ 13\catcode'\= 13\relax}%
128 \def\ProsodicMarksOff{\catcode'\^ 7\catcode'\= 12\relax}%
129 }

```

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

```

130 \initiate@active@char{^}%
131 \initiate@active@char{=}%
132 \declare@shorthand{latin}{^a}{%
133     \textormath{\u{a}\bbl@allowhyphens}{\hat{a}}}%
134 \declare@shorthand{latin}{^e}{%
135     \textormath{\u{e}\bbl@allowhyphens}{\hat{e}}}%
136 \declare@shorthand{latin}{^i}{%
137     \textormath{\u{i}\bbl@allowhyphens}{\hat{\imath}}}%

```

```

138 \declare@shorthand{latin}{^o}{%
139   \textormath{\u{o}\bbl@allowhyphens}{\hat{o}}}%
140 \declare@shorthand{latin}{^u}{%
141   \textormath{\u{u}\bbl@allowhyphens}{\hat{u}}}%
142 %
143 \declare@shorthand{latin}{=a}{%
144   \textormath{\={a}\bbl@allowhyphens}{\bar{a}}}%
145 \declare@shorthand{latin}{=e}{%
146   \textormath{\={e}\bbl@allowhyphens}{\bar{e}}}%
147 \declare@shorthand{latin}{=i}{%
148   \textormath{\={i}\bbl@allowhyphens}{\bar{\imath}}}%
149 \declare@shorthand{latin}{=o}{%
150   \textormath{\={o}\bbl@allowhyphens}{\bar{o}}}%
151 \declare@shorthand{latin}{=u}{%
152   \textormath{\={u}\bbl@allowhyphens}{\bar{u}}}%

```

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

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

```

153 \catcode'\= 12\relax
154 \catcode'\^ 7\relax

```

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

`\LatinMarksOn` The following commands remain defined for backwards compatibility, but they are obsolete and should not be used.

```

155 \addto\extraslatin{\def\LatinMarksOn{\shorthandon{^}\shorthandon{=}}
156 \addto\extraslatin{\def\LatinMarksOff{\shorthandoff{^}\shorthandoff{=}}

```

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

2.2 Ecclesiastic Latin style

We declare the `ecclesiastic` attribute; its purpose is to load the `ecclesiastic.sty` package, deferring this action to the moment when `babel` finishes executing; actually it sends `babel` the information to act; then `babel` defers the package loading to the `\begin{document}` execution. This implies the availability of the `\AtEndOfPackage` and `\AtBeginDocument` macros which are contained in the L^AT_EX kernel. Since this file is read by `babel` the current package at the end of which the code is deferred is `babel` itself, not simply this language description file.

Differently from other attribute settings we do not declare anything in a possible `\extrasecclesiasticlatin` macro, because once the package has been read, it is impossible to pretend to delete all what the package defined. Actually this `ecclesiastic` attribute simply mimics the actual loading of a package by means of a `\usepackage` or `\RequirePackage` command; if loading was done with one of these commands, it would be impossible to unload the loaded package.

```
157 \bbl@declare@ttribute{latin}{ecclesiastic}{%
158 \AtEndOfPackage{%
159   \AtBeginDocument{\RequirePackage{ecclesiastic}[2015/08/20]}%
160 }%
161 }
```

2.3 Etymological hyphenation

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

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

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

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

```
162 \initiate@active@char{"}%
163 \addto\extraslatin{\bbl@activate{"}%
164 }
```

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

```
165 \declare@shorthand{latin}{"}-%
166 \textormath{\def\lt@next{\futurelet\lt@temp\lt@cwmm}}%
```

```

167     {\def\lt@next{'}}%
168 \lt@next
169 }%

```

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

```

170 \def\lt@allowhyphens{\bbl@allowhyphens\discretionary{-}{ }\bbl@allowhyphens}
171
172 \newcommand*{\lt@cwm}{\let\lt@n@xt\relax
173   \ifcat\noexpand\lt@temp a%
174     \let\lt@n@xt\lt@allowhyphens
175   \else
176     \ifx\lt@temp\ae
177       \let\lt@n@xt\lt@allowhyphens
178     \else
179       \ifx\lt@temp\oe
180         \let\lt@n@xt\lt@allowhyphens
181       \else
182         \if\noexpand\lt@temp|string!%
183           \def\lt@n@xt{\lt@allowhyphens@gobble}%
184         \fi
185       \fi
186     \fi
187   \fi
188   \lt@n@xt}%

```

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

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

```

189 \ldf@finish{latin}

```

³This word does not exist in “regular” Latin, and it is used just as an example.