

Babel

Version 26.15
2026/10/29

Javier Bezos
Current maintainer

Johannes L. Braams
Original author

Localization and
internationalization

Plain T_EX

NOTE This file is devoted to babel for Plain $\text{\TeX}$. For many years, most of the code was shared between $\text{\LaTeX}$ and Plain, but recent advances in $\text{\LaTeX}$ have made joint maintenance increasingly complex. Consequently, the code for Plain will be split from main babel for $\text{\LaTeX}$ and essentially frozen. This does not mean it is unmaintained, and the door remains open to future improvements.

Description of many macros are the same as those in babel-code.pdf, and here only some hints are provided. Please, refer to that document for details.

This file is loaded not only by Plain. $\text{\LaTeX}$ loads it with the package option base (and also plain.def) with \bbl@onlyswitch = \@empty. See sec. 5.

It needs to be cleaned up to remove unnecessary code in Plain. Note also the engine specific files are still loaded.

1. A few preliminary definitions

Just a block. To be revised.

```
1 <<version=26.15>>
2 <<date=2026/10/29>>
```

Some files identify themselves with a $\text{\LaTeX}$ macro. The following code is placed before them to define (and then undefine) if not in $\text{\LaTeX}$.

\language Just for compatibility, for not to touch hyphen.cfg.

```
3 <<*Define Plain core switching macros>> ▶
4 \ifx\language\@undefined
5   \csname newcount\endcsname\language
6 \fi
7 <</Define Plain core switching macros>>
```

\last@language Another counter is used to keep track of the allocated languages. $\text{\TeX}$ and $\text{\LaTeX}$ reserves for this purpose the count 19.

\addlanguage This macro was introduced for $\text{\TeX}$ < 2. Preserved for compatibility.

```
8 <<*Define Plain core switching macros>> ▶
9 \countdef\last@language=19
10 \def\addlanguage{\csname newlanguage\endcsname}
11 <</Define Plain core switching macros>>
```

First, exit immediately if previously loaded.

```
12 <*core>
13 \ifx\ldf@quit\@undefined\else
14 \endinput\fi % Same line!
15 \ifx\ProvidesFile\@undefined
16   \def\ProvidesFile#1[#2 #3 #4]{%
17     \wlog{File: #1 #4 #3 <#2>}%
18     \let\ProvidesFile\@undefined}
19 \fi
20 \ProvidesFile{babel.def}[<@date@> v<@version@> Babel common definitions]
```

2. Emulating some $\text{\LaTeX}$ features

The file babel.def expects some definitions made in the $\text{\LaTeX} 2_{\epsilon}$ style file. So, in Plain we must provide at least some predefined values as well some tools to set them (even if not all options are available). There are no package options, and therefore and alternative mechanism is provided. For the moment, only \babeloptionstrings and \babeloptionmath are provided, which can be defined before loading babel. \BabelModifiers can be set too (but not sure it works).

```

21 \ifx\AtBeginDocument\@undefined
22 \def\@empty{}
23 \def\loadlocalcfg#1{%
24   \openin0#1.cfg
25   \ifeof0
26     \closein0
27   \else
28     \closein0
29     {\immediate\writel6{*****}%
30     \immediate\writel6{* Local config file #1.cfg used}%
31     \immediate\writel6{*}%
32     }
33   \input #1.cfg\relax
34 \fi
35 \endoflfd}

```

2.1. General tools

A number of $\LaTeX$ macro's that are needed later on.

```

36 \long\def\@firstofone#1{#1}
37 \long\def\@firstoftwo#1#2{#1}
38 \long\def\@secondoftwo#1#2{#2}
39 \def\@nnil{\@nil}
40 \def\@gobbletwo#1#2{}
41 \def\@ifstar#1{\@ifnextchar *{\@firstoftwo{#1}}}
42 \def\@star@or@long#1{%
43   \@ifstar
44   {\let\l@ngrel@x\relax#1}%
45   {\let\l@ngrel@x\long#1}}
46 \let\l@ngrel@x\relax
47 \def\@car#1#2\@nil{#1}
48 \def\@cdr#1#2\@nil{#2}
49 \let\@typeset@protect\relax
50 \let\protected@edef\edef
51 \long\def\@gobble#1{}
52 \edef\@backslashchar{\expandafter\@gobble\string\}
53 \def\strip@prefix#1>{}
54 \def\g@addto@macro#1#2{{%
55   \toks@{\expandafter{#1#2}%
56   \xdef#1{\the\toks@}}}
57 \def\@namedef#1{\expandafter\def\csname #1\endcsname}
58 \def\@nameuse#1{\csname #1\endcsname}
59 \def\@ifundefined#1{%
60   \expandafter\ifx\csname#1\endcsname\relax
61     \expandafter\@firstoftwo
62   \else
63     \expandafter\@secondoftwo
64   \fi}
65 \def\@expandtwoargs#1#2#3{%
66   \edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}
67 \def\zap@space#1 #2{%
68   #1%
69   \ifx#2\@empty\else\expandafter\zap@space\fi
70   #2}
71 \let\bbl@trace\@gobble
72 \def\bbl@error#1{% Implicit #2#3#4
73   \begingroup
74     \catcode\=0 \catcode\==12 \catcode\`=12
75     \catcode\^M=5 \catcode\%=14
76     \input errbabel.def
77   \endgroup
78   \bbl@error{#1}}

```

```

79 \def\bbl@warning#1{%
80   \begingroup
81     \newlinechar=`^^J
82   \def\{^^J(babel) }%
83   \message{\#1}%
84   \endgroup}
85 \let\bbl@infowarn\bbl@warning
86 \def\bbl@info#1{%
87   \begingroup
88     \newlinechar=`^^J
89   \def\{^^J}%
90   \wlog{#1}%
91   \endgroup}

```

$\LaTeX 2_{\epsilon}$ has the command `\@onlypreamble` which adds commands to a list of commands that are no longer needed after `\begin{document}`.

```

92 \ifx\@preamblecmds\undefined
93   \def\@preamblecmds{}
94 \fi
95 \def\@onlypreamble#1{%
96   \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
97     \@preamblecmds\do#1}}
98 \@onlypreamble\@onlypreamble

```

Mimic $\LaTeX$'s `\AtBeginDocument`; for this to work the user needs to add `\begin{document}` to his file.

```

99 \def\begin{document}{%
100   \@begin{document}hook
101   \global\let\@begin{document}hook\@undefined
102   \def\do##1{\global\let##1\@undefined}%
103   \@preamblecmds
104   \global\let\do\noexpand}
105 \ifx\@begin{document}hook\@undefined
106   \def\@begin{document}hook{}
107 \fi
108 \@onlypreamble\@begin{document}hook
109 \def\AtBeginDocument{\g@addto@macro\@begin{document}hook}

```

We also have to mimic $\LaTeX$'s `\AtEndOfPackage`. Our replacement macro is much simpler; it stores its argument in `\@endofldf`.

```

110 \def\AtEndOfPackage#1{\g@addto@macro\@endofldf{#1}}
111 \@onlypreamble\AtEndOfPackage
112 \def\@endofldf{}
113 \@onlypreamble\@endofldf
114 \let\bbl@afterlang\empty
115 \chardef\bbl@opt@hyphenmap\z@

```

$\LaTeX$ needs to be able to switch off writing to its auxiliary files; plain doesn't have them by default. There is a trick to hide some conditional commands from the outer `\ifx`. The same trick is applied below.

```

116 \catcode`\&=\z@
117 \ifx&\if@files\@undefined
118   \expandafter\let\csname if@files\expandafter\endcsname
119   \csname iffalse\endcsname
120 \fi
121 \catcode`\&=4

```

Mimic $\LaTeX$'s commands to define control sequences.

```

122 \def\newcommand{\@star@or@long\new@command}
123 \def\new@command#1{%
124   \@testopt{\@newcommand#1}0}
125 \def\@newcommand#1[#2]{%
126   \@ifnextchar [{\@xargdef#1[#2]}%

```

```

127             {\@argdef#1[#2]}}
128 \long\def\@argdef#1[#2]#3{%
129   \@yargdef#1\@ne{#2}{#3}}
130 \long\def\@xargdef#1[#2][#3]#4{%
131   \expandafter\def\expandafter#1\expandafter{%
132     \expandafter\@protected@testopt\expandafter #1%
133     \csname\string#1\expandafter\endcsname{#3}}%
134   \expandafter\@yargdef \csname\string#1\endcsname
135   \tw@{#2}{#4}}
136 \long\def\@yargdef#1#2#3{%
137   \@tempcnta#3\relax
138   \advance \@tempcnta \@ne
139   \let\@hash@\relax
140   \edef\reserved@a{\ifx#2\tw@ [\@hash@1]\fi}%
141   \@tempcntb #2%
142   \@whilenum \@tempcntb < \@tempcnta
143   \do{%
144     \edef\reserved@a{\reserved@a\@hash@\the\@tempcntb}%
145     \advance\@tempcntb \@ne}%
146   \let\@hash@##%
147   \l@ngrelx\expandafter\def\expandafter#1\reserved@a}
148 \def\providecommand{\@star@or@long\provide@command}
149 \def\provide@command#1{%
150   \begingroup
151     \escapechar\m@ne\xdef\@gtempa{\string#1}%
152   \endgroup
153   \expandafter\ifundefined\@gtempa
154     {\def\reserved@a{\new@command#1}}%
155     {\let\reserved@a\relax
156      \def\reserved@a{\new@command\reserved@a}}%
157   \reserved@a}%
158 \def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
159 \def\declare@robustcommand#1{%
160   \edef\reserved@a{\string#1}%
161   \def\reserved@b{#1}%
162   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
163   \edef#1{%
164     \ifx\reserved@a\reserved@b
165       \noexpand\x@protect
166       \noexpand#1%
167     \fi
168     \noexpand\protect
169     \expandafter\noexpand\csname
170       \expandafter\@gobble\string#1 \endcsname
171   }%
172   \expandafter\new@command\csname
173     \expandafter\@gobble\string#1 \endcsname
174 }
175 \def\x@protect#1{%
176   \ifx\protect\@typeset@protect\else
177     \x@protect#1%
178   \fi
179 }
180 \catcode`\&=\z@ % Trick to hide conditionals
181 \def\@x@protect#1&\fi#2#3{&\fi\protect#1}

```

The following little macro `\in@` is taken from `latex.ltx`; it checks whether its first argument is part of its second argument. It uses the boolean `\in@`; allocating a new boolean inside conditionally executed code is not possible, hence the construct with the temporary definition of `\bbl@tempa`.

```

182 \def\bbl@tempa{\csname newif\endcsname&\fi\in@}
183 \catcode`\&=4
184 \ifx\in@\@undefined

```

```

185 \def\in@#1#2{%
186   \def\in@@##1#1##2##3\in@@{%
187     \ifx\in@@#2\in@false\else\in@true\fi}%
188   \in@@#2#1\in@\in@@}
189 \else
190   \let\bbl@tempa\@empty
191 \fi
192 \bbl@tempa

```

Used in some ldf files. Assume true.

```

193 \def\@ifpackagewith#1#2#3#4{#3}

```

The $\LaTeX$ macro `\@ifl@aded` checks whether a file was loaded. No-op in Plain.

```

194 \def\@ifl@aded#1#2#3#4{}

```

For the following code we need to make sure that the commands `\newcommand` and `\providecommand` exist with some sensible definition. They are not fully equivalent to their $\LaTeX 2_{\epsilon}$ versions; just enough to make things work in plain $\TeX$ environments.

```

195 \ifx\@tempcnta\undefined
196   \csname newcount\endcsname\@tempcnta\relax
197 \fi
198 \ifx\@tempcntb\undefined
199   \csname newcount\endcsname\@tempcntb\relax
200 \fi

```

To prevent wasting two counters in $\LaTeX$ (because counters with the same name are allocated later by it) we reset the counter that holds the next free counter (`\count10`).

```

201 \ifx\bye\undefined
202   \advance\count10 by -2\relax
203 \fi
204 \ifx\@ifnextchar\undefined
205   \def\@ifnextchar#1#2#3{%
206     \let\reserved@d=#1%
207     \def\reserved@a{#2}\def\reserved@b{#3}%
208     \futurelet\@let@token\@ifnch}
209 \def\@ifnch{%
210   \ifx\@let@token\@sptoken
211     \let\reserved@c\@xifnch
212   \else
213     \ifx\@let@token\reserved@d
214       \let\reserved@c\reserved@a
215     \else
216       \let\reserved@c\reserved@b
217   \fi
218 \fi
219 \reserved@c}
220 \def\:\let\@sptoken= \: % this makes \@sptoken a space token
221 \def\:\@xifnch \expandafter\def\:\{\futurelet\@let@token\@ifnch}
222 \fi
223 \def\@testopt#1#2{%
224   \@ifnextchar[#{1}{#1[#{2}}]}
225 \def\@protected@testopt#1{%
226   \ifx\protect\@typeset@protect
227     \expandafter\@testopt
228   \else
229     \@x@protect#1%
230   \fi}
231 \long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
232   #2\relax}\fi}
233 \long\def\@iwhilenum#1{\ifnum #1\relax\expandafter\@iwhilenum
234   \else\expandafter\@gobble\fi{#1}}

```

2.2. Encoding related macros

Code from `ltoutenc.dtx`, adapted for use in the plain $\TeX$ environment.

```
235 \def\DeclareTextCommand{%
236   \@dec@text@cmd\providecommand
237 }
238 \def\ProvideTextCommand{%
239   \@dec@text@cmd\providecommand
240 }
241 \def\DeclareTextSymbol#1#2#3{%
242   \@dec@text@cmd\chardef#1{#2}#3\relax
243 }
244 \def\@dec@text@cmd#1#2#3{%
245   \expandafter\def\expandafter#2%
246     \expandafter{%
247       \csname#3-cmd\expandafter\endcsname
248       \expandafter#2%
249       \csname#3\string#2\endcsname
250     }%
251 %   \let\@ifdefinable\@rc@ifdefinable
252   \expandafter#1\csname#3\string#2\endcsname
253 }
254 \def\@current@cmd#1{%
255   \ifx\protect\@typeset@protect\else
256     \noexpand#1\expandafter\@gobble
257   \fi
258 }
259 \def\@changed@cmd#1#2{%
260   \ifx\protect\@typeset@protect
261     \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
262       \expandafter\ifx\csname ?\string#1\endcsname\relax
263         \expandafter\def\csname ?\string#1\endcsname{%
264           \@changed@x@err{#1}%
265         }%
266       \fi
267       \global\expandafter\let
268         \csname\cf@encoding \string#1\expandafter\endcsname
269         \csname ?\string#1\endcsname
270     \fi
271     \csname\cf@encoding\string#1%
272       \expandafter\endcsname
273   \else
274     \noexpand#1%
275   \fi
276 }
277 \def\@changed@x@err#1{%
278   \errhelp{Your command will be ignored, type <return> to proceed}%
279   \errmessage{Command \protect#1 undefined in encoding \cf@encoding}}
280 \def\DeclareTextCommandDefault#1{%
281   \DeclareTextCommand#1?%
282 }
283 \def\ProvideTextCommandDefault#1{%
284   \ProvideTextCommand#1?%
285 }
286 \expandafter\let\csname OT1-cmd\endcsname\@current@cmd
287 \expandafter\let\csname?-cmd\endcsname\@changed@cmd
288 \def\DeclareTextAccent#1#2#3{%
289   \DeclareTextCommand#1{#2}[1]{\accent#3 #1}
290 }
291 \def\DeclareTextCompositeCommand#1#2#3#4{%
292   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
293   \edef\reserved@b{\string#1}%
294   \edef\reserved@c{%
```

```

295 \expandafter\@strip@args\meaning\reserved@a:-\@strip@args}%
296 \ifx\reserved@b\reserved@c
297 \expandafter\expandafter\expandafter\ifx
298 \expandafter\@car\reserved@a\relax\relax\@nil
299 \@text@composite
300 \else
301 \edef\reserved@b##1{%
302 \def\expandafter\noexpand
303 \csname#2\string#1\endcsname###1{%
304 \noexpand\@text@composite
305 \expandafter\noexpand\csname#2\string#1\endcsname
306 ###1\noexpand\@empty\noexpand\@text@composite
307 {##1}%
308 }%
309 }%
310 \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
311 \fi
312 \expandafter\def\csname\expandafter\string\csname
313 #2\endcsname\string#1-\string#3\endcsname{#4}
314 \else
315 \errhelp{Your command will be ignored, type <return> to proceed}%
316 \errmessage{\string\DeclareTextCompositeCommand\space used on
317 inappropriate command \protect#1}
318 \fi
319 }
320 \def\@text@composite#1#2#3\@text@composite{%
321 \expandafter\@text@composite@x
322 \csname\string#1-\string#2\endcsname
323 }
324 \def\@text@composite@x#1#2{%
325 \ifx#1\relax
326 #2%
327 \else
328 #1%
329 \fi
330 }
331 %
332 \def\@strip@args#1:#2-#3\@strip@args{#2}
333 \def\DeclareTextComposite#1#2#3#4{%
334 \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
335 \bgroup
336 \lccode`\@=#4%
337 \lowercase{%
338 \egroup
339 \reserved@a @%
340 }%
341 }
342 %
343 \def\UseTextSymbol#1#2{#2}
344 \def\UseTextAccent#1#2#3{}
345 \def\@use@text@encoding#1{}
346 \def\DeclareTextSymbolDefault#1#2{%
347 \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}%
348 }
349 \def\DeclareTextAccentDefault#1#2{%
350 \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}%
351 }
352 \def\cf@encoding{OT1}

```

Currently we only use the $\text{\LaTeX 2}_{\epsilon}$ method for accents for those that are known to be made active in *some* language definition file.

```

353 \DeclareTextAccent{"}{OT1}{127}
354 \DeclareTextAccent{'}{OT1}{19}

```

```

355 \DeclareTextAccent{\^}{0T1}{94}
356 \DeclareTextAccent{\`}{0T1}{18}
357 \DeclareTextAccent{\~}{0T1}{126}

```

The following control sequences are used in `babel.def` but are not defined for PLAIN $\TeX$.

```

358 \DeclareTextSymbol{\textquotedblleft}{0T1}{92}
359 \DeclareTextSymbol{\textquotedblright}{0T1}{`\"}
360 \DeclareTextSymbol{\textquoteleft}{0T1}{`\'}
361 \DeclareTextSymbol{\textquoteright}{0T1}{`\'}
362 \DeclareTextSymbol{\i}{0T1}{16}
363 \DeclareTextSymbol{\ss}{0T1}{25}

```

For a couple of languages we need the $\TeX$ -control sequence `\scriptsize` to be available. Because plain $\TeX$ doesn't have such a sophisticated font mechanism as $\TeX$ has, we just `\let` it to `\sevenrm`.

```

364 \ifx\scriptsize\undefined
365   \let\scriptsize\sevenrm
366 \fi

```

And a few more “dummy” definitions.

```

367 \def\language{english}%
368 \def\bbl@ifshorthand#1#2#3#2}%
369 \let\bbl@language@opts\empty
370 \let\bbl@provide@locale\relax
371 \ifx\babeloptionstrings\undefined
372   \let\bbl@opt@strings\@nnil
373 \else
374   \let\bbl@opt@strings\babeloptionstrings
375 \fi
376 \def\BabelStringsDefault{generic}
377 \def\bbl@tempa{normal}
378 \ifx\babeloptionmath\bbl@tempa
379   \def\bbl@mathnormal{\noexpand\textormath}
380 \fi
381 \def\AfterBabelLanguage#1#2{}
382 \ifx\BabelModifiers\undefined\let\BabelModifiers\relax\fi
383 \let\bbl@afterlang\relax
384 \ifx\@uclclist\undefined\let\@uclclist\empty\fi
385 \chardef\bbl@bidimode\z@
386 \fi

```

3. Basic macros

The emulation is finished and now comes the basic macros for Plain + base. See `babel-code.pdf`.

```

387 \bbl@trace{Basic macros}
388 \def\bbl@stripslash{\expandafter\@gobble\string}
389 \def\bbl@add#1#2{%
390   \bbl@ifunset{\bbl@stripslash#1}%
391     {\def#1{#2}}%
392     {\expandafter\def\expandafter#1\expandafter{#1#2}}}
393 \def\bbl@xin@{\@expandtwoargs\in@}
394 \def\bbl@carg#1#2{\expandafter#1\csname#2\endcsname}%
395 \def\bbl@ncarg#1#2#3{\expandafter#1\expandafter#2\csname#3\endcsname}%
396 \def\bbl@ccarg#1#2#3{%
397   \expandafter#1\csname#2\expandafter\endcsname\csname#3\endcsname}%
398 \def\bbl@csarg#1#2{\expandafter#1\csname bbl@#2\endcsname}%
399 \def\bbl@cs#1{\csname bbl@#1\endcsname}
400 \def\bbl@c1#1{\csname bbl@#1\language\endcsname}
401 \def\bbl@loop#1#2#3{\bbl@loop#1{#3}#2,\@nnil,}
402 \def\bbl@loopx#1#2{\expandafter\bbl@loop\expandafter#1\expandafter{#2}}

```

```

403 \def\bbl@loop#1#2#3,{%
404   \ifx\@nnil#3\relax\else
405     \def#1{#3}#2\bbl@afterfi\bbl@loop#1{#2}%
406   \fi}
407 \def\bbl@for#1#2#3{\bbl@loopx#1{#2}{\ifx#1\@empty\else#3\fi}}

```

\bbl@add@list This internal macro adds its second argument to a comma separated list in its first argument.

```

408 \def\bbl@add@list#1#2{%
409   \edef#1{%
410     \bbl@ifunset{\bbl@stripslash#1}%
411     }%
412     {\ifx#1\@empty\else#1,\fi}%
413   #2}}

```

\bbl@afterelse

\bbl@afterfi Because the code that is used in the handling of active characters may need to look ahead, we take extra care to ‘throw’ it over the \else and \fi parts of an \if-statement.

```

414 \long\def\bbl@afterelse#1\else#2\fi{\fi#1}
415 \long\def\bbl@afterfi#1\fi{\fi#1}

```

\bbl@exp Now, just syntactical sugar.

```

416 \def\bbl@exp#1{%
417   \begingroup
418     \let\\\noexpand
419     \let\<\bbl@exp@en
420     \let\[\bbl@exp@ue
421     \edef\bbl@exp@aux{\endgroup#1}%
422     \bbl@exp@aux}
423 \def\bbl@exp@en#1>{\expandafter\noexpand\csname#1\endcsname}%
424 \def\bbl@exp@ue#1]{%
425   \unexpanded\expandafter\expandafter\expandafter{\csname#1\endcsname}}%

```

\bbl@trim Define two macros: \bbl@trim and \bbl@trim@def.

```

426 \def\bbl@tempa#1{%
427   \long\def\bbl@trim##1##2{%
428     \futurelet\bbl@trim@a\bbl@trim@c##2\@nil\@nil#1\@nil\relax{##1}}%
429   \def\bbl@trim@c{%
430     \ifx\bbl@trim@a\@sptoken
431       \expandafter\bbl@trim@b
432     \else
433       \expandafter\bbl@trim@b\expandafter#1%
434     \fi}%
435   \long\def\bbl@trim@b#1##1 \@nil{\bbl@trim@i##1}}
436 \bbl@tempa{ }
437 \long\def\bbl@trim@i#1\@nil#2\relax#3{#3{#1}}
438 \long\def\bbl@trim@def#1{\bbl@trim{\def#1}}

```

\bbl@ifunset To check if a macro is defined, we create a new macro, which does the same as \@ifundefined. However, in an ϵ -tex engine, it is based on \ifcsname, which is more efficient, and does not waste memory. Defined inside a group, to avoid \ifcsname being implicitly set to \relax by the \csname test.

```

439 \begingroup
440   \gdef\bbl@ifunset#1{%
441     \expandafter\ifx\csname#1\endcsname\relax
442       \expandafter\@firstoftwo
443     \else

```

```

444     \expandafter\@secondoftwo
445     \fi}
446 \bbl@ifunset{ifcscname}%
447 {}%
448 {\gdef\bbl@ifunset#1{%
449     \ifcscname#1\endcsname
450     \expandafter\ifx\cscname#1\endcsname\relax
451     \bbl@afterelse\expandafter\@firstoftwo
452     \else
453     \bbl@afterfi\expandafter\@secondoftwo
454     \fi
455     \else
456     \expandafter\@firstoftwo
457     \fi}}
458 \endgroup

```

\bbl@ifblank A tool from url, by Donald Arseneau, which tests if a string is empty or space.

```

459 \def\bbl@ifblank#1{%
460     \bbl@ifblank@i#1\@nil\@nil\@secondoftwo\@firstoftwo\@nil}
461 \long\def\bbl@ifblank@i#1#2\@nil#3#4#5\@nil{#4}
462 \def\bbl@ifset#1#2#3{%
463     \bbl@ifunset{#1}{#3}{\bbl@exp{\bbl@ifblank{\@nameuse{#1}}}{#3}{#2}}}

```

For each element in the comma separated <key>=<value> list, execute <code> with #1 and #2 as the key and the value of current item (trimmed).

```

464 \def\bbl@forkv#1#2{%
465     \def\bbl@kvcmd##1##2##3{#2}%
466     \bbl@kvnext#1,\@nil,}
467 \def\bbl@kvnext#1,{%
468     \ifx\@nil#1\relax\else
469     \bbl@ifblank{#1}{\bbl@forkv@eq#1=\@empty=\@nil{#1}}%
470     \expandafter\bbl@kvnext
471     \fi}
472 \def\bbl@forkv@eq#1=#2=#3\@nil#4{%
473     \bbl@trim\def\bbl@forkv@a{#1}%
474     \bbl@trim{\expandafter\bbl@kvcmd\expandafter{\bbl@forkv@a}}{#2}{#4}}

```

A *for* loop. Each item (trimmed) is #1. It cannot be nested (it's doable, but we don't need it).

```

475 \def\bbl@vforeach#1#2{%
476     \def\bbl@forcmd##1{#2}%
477     \bbl@fornext#1,\@nil,}
478 \def\bbl@fornext#1,{%
479     \ifx\@nil#1\relax\else
480     \bbl@ifblank{#1}{\bbl@trim\bbl@forcmd{#1}}%
481     \expandafter\bbl@fornext
482     \fi}
483 \def\bbl@foreach#1{\expandafter\bbl@vforeach\expandafter{#1}}

```

Some code should be executed once. The first argument is a flag.

```

484 \global\let\bbl@done\@empty
485 \def\bbl@once#1#2{%
486     \bbl@xin@{, #1,}{, \bbl@done,}%
487     \ifin@ \else
488     #2%
489     \xdef\bbl@done{\bbl@done, #1,}%
490     \fi}
491 % \end{macro}
492 %
493 % \macro{\bbl@replace}
494 %
495 % Returns implicitly |\toks@| with the modified string.

```

```

496 %
497 % \begin{macrocode}
498 \def\bbl@replace#1#2#3{% in #1 -> repl #2 by #3
499 \toks@{}%
500 \def\bbl@replace@aux##1#2##2#2{%
501 \ifx\bbl@nil##2%
502 \toks@\expandafter{\the\toks@##1}%
503 \else
504 \toks@\expandafter{\the\toks@##1#3}%
505 \bbl@afterfi
506 \bbl@replace@aux##2#2%
507 \fi}%
508 \expandafter\bbl@replace@aux#1#2\bbl@nil#2%
509 \edef#1{\the\toks@}}

```

An extension to the previous macro:

```

510 \ifx\detokenize\undefined\else % Unused macros if old Plain TeX
511 \bbl@exp{\def\\bbl@parsedef##1\detokenize{macro:}}#2->#3\relax{%
512 \def\bbl@tempa{#1}%
513 \def\bbl@tempb{#2}%
514 \def\bbl@tempe{#3}}
515 \def\bbl@sreplace#1#2#3{%
516 \begingroup
517 \expandafter\bbl@parsedef\meaning#1\relax
518 \def\bbl@tempc{#2}%
519 \edef\bbl@tempc{\expandafter\strip@prefix\meaning\bbl@tempc}%
520 \def\bbl@tempd{#3}%
521 \edef\bbl@tempd{\expandafter\strip@prefix\meaning\bbl@tempd}%
522 \bbl@xin@{\bbl@tempc}{\bbl@tempe}% If not in macro, do nothing
523 \ifin@
524 \bbl@exp{\\bbl@replace\\bbl@tempe{\bbl@tempc}{\bbl@tempd}}%
525 \def\bbl@tempc{% Expanded an executed below as 'uplevel'
526 \\makeatletter % "internal" macros with @ are assumed
527 \\scantokens{%
528 \bbl@tempa\\@namedef{\bbl@stripslash#1}\bbl@tempb{\bbl@tempe}%
529 \noexpand\noexpand}%
530 \catcode64=\the\catcode64\relax}% Restore @
531 \else
532 \let\bbl@tempc\@empty % Not \relax
533 \fi
534 \bbl@exp{% For the 'uplevel' assignments
535 \endgroup
536 \bbl@tempc}} % empty or expand to set #1 with changes
537 \fi

```

Two further tools.

```

538 \def\bbl@ifsamestring#1#2{%
539 \begingroup
540 \protected@edef\bbl@tempb{#1}%
541 \edef\bbl@tempb{\expandafter\strip@prefix\meaning\bbl@tempb}%
542 \protected@edef\bbl@tempc{#2}%
543 \edef\bbl@tempc{\expandafter\strip@prefix\meaning\bbl@tempc}%
544 \ifx\bbl@tempb\bbl@tempc
545 \aftergroup\@firstoftwo
546 \else
547 \aftergroup\@secondoftwo
548 \fi
549 \endgroup}
550 \chardef\bbl@engine=%
551 \ifx\directlua\undefined
552 \ifx\XeTeXinputencoding\undefined
553 \z@
554 \else

```

```

555     \tw@
556     \fi
557 \else
558     \@ne
559 \fi

```

A somewhat hackish tool (hence its name) to avoid spurious spaces in some contexts.

```

560 \def\bbl@bsphack{%
561     \ifhmode
562         \hskip\z@skip
563     \def\bbl@esphack{\loop\ifdim\lastskip>\z@ \unskip\repeat\unskip}%
564 \else
565     \let\bbl@esphack\@empty
566 \fi}

```

Ordered key/value lists (aka associative arrays).

```

567 \def\bbl@array@end{\@empty}
568 \def\bbl@array@set#1#2#3{%
569     \def\bbl@array@aux##1\bbl@elt#2\@##2##3\bbl@array@aux##4{%
570         \ifx\bbl@noelt##2%
571             \def#1{##1\bbl@elt#2\@{##4}}%
572         \else
573             \def\bbl@array@aux####1\bbl@elt#2\@####2###3\bbl@array@aux####4{%
574                 \def#1{####1\bbl@elt#2\@{####4}###3}}%
575             \expandafter\bbl@array@aux#1\bbl@array@aux{#3}%
576         \fi}%
577     \expandafter\bbl@array@aux#1\bbl@elt#2\@ \bbl@noelt\bbl@array@aux{#3}}
578 \def\bbl@array@get#1#2#3{%
579     \def\bbl@array@aux##1\bbl@elt#2\@##2##3\bbl@array@aux{%
580         \ifx\bbl@array@end##2\@empty \let#3\relax \else \def#3{##2}\fi}%
581     \expandafter\bbl@array@aux#1\bbl@elt#2\@ \bbl@array@end\bbl@array@aux}

```

That is all for the moment. Now follows some common stuff, for both Plain and $\LaTeX$. After it, we will resume the $\LaTeX$ -only stuff.

4. Selectors and declaring languages

Still Plain + base.

```

582 \def\bbl@version{<@version@>}
583 \def\bbl@date{<@date@>}
584 <@Define Plain core switching macros@>

```

\adddialect The macro `\adddialect` can be used to add the name of a dialect or variant language, for which an already defined hyphenation table can be used.

```

585 \def\adddialect#1#2{%
586     \global\chardef#1#2\relax
587     \bbl@usehooks{adddialect}{\{#1\}\{#2\}}%
588     \begingroup
589         \count@#1\relax
590         \def\bbl@elt##1##2##3##4{%
591             \ifnum\count@=##2\relax
592                 \edef\bbl@tempa{\expandafter\@gobbletwo\string#1}%
593                 \bbl@info{Hyphen rules for '\expandafter\@gobble\bbl@tempa'
594                     set to \expandafter\string\csname l@##1\endcsname\%
595                     (\string\language\the\count@). Reported}%
596                 \def\bbl@elt####1####2####3####4{%
597                     \fi}%
598             \bbl@cs{languages}%
599         \endgroup}

```

`\bbl@iflanguage` executes code only if the language `l@` exists.

```
600 \def\bbl@fixname#1{%
601   \begingroup
602     \def\bbl@tempe{l@}%
603     \edef\bbl@tempd{\noexpand\@ifundefined{\noexpand\bbl@tempe#1}}%
604     \bbl@tempd
605       {\lowercase\expandafter{\bbl@tempd}%
606        {\uppercase\expandafter{\bbl@tempd}%
607         \@empty
608          {\edef\bbl@tempd{\def\noexpand#1{#1}}%
609           \uppercase\expandafter{\bbl@tempd}}}%
610        {\edef\bbl@tempd{\def\noexpand#1{#1}}%
611         \lowercase\expandafter{\bbl@tempd}}}%
612     \@empty
613   \edef\bbl@tempd{\endgroup\def\noexpand#1{#1}}%
614   \bbl@tempd
615   \bbl@exp{\bbl@usehooks{language}{\language}{#1}}
616 \def\bbl@iflanguage#1{%
617   \@ifundefined{l@#1}{\@nolanerr{#1}\@gobble}\@firstofone}
```

\iflanguage Users might want to test (in a private package for instance) which language is currently active.

```
618 \def\iflanguage#1{%
619   \bbl@iflanguage{#1}{%
620     \ifnum\cscname l@#1\endcsname=\language
621       \expandafter\@firstoftwo
622     \else
623       \expandafter\@secondoftwo
624     \fi}}
```

4.1. Selecting the language

\selectlanguage It checks whether the language is already defined before it performs its actual task, which is to update `\language` and activate language-specific definitions.

```
625 \let\bbl@select@type\z@
626 \edef\selectlanguage{%
627   \noexpand\protect
628   \expandafter\noexpand\cscname selectlanguage \endcsname}
629 \ifx\@undefined\protect\let\protect\relax\fi
630 \let\xstring\string
```

\bbl@pop@language But when the language change happens *inside* a group the end of the group doesn't write anything to the auxiliary files.

\bbl@language@stack The previous solution works for one level of nesting groups, but as soon as more levels are used it is no longer adequate.

```
631 \def\bbl@language@stack{}
```

\bbl@push@language

\bbl@pop@language The stack is simply a list of languagenames, separated with a '+' sign; the push function can be simple:

```
632 \def\bbl@push@language{%
633   \ifx\language\@undefined\else
634     \ifx\currentgrouplevel\@undefined
635       \xdef\bbl@language@stack{\language+\bbl@language@stack}%
636     \else
637       \ifnum\currentgrouplevel=\z@
638         \xdef\bbl@language@stack{\language+}%
```

```

639     \else
640     \xdef\bbl@language@stack{\language+\bbl@language@stack}%
641     \fi
642 \fi
643 \fi}

```

Retrieving information from the stack is a little bit less simple, as we need to remove the element from the stack while storing it in the macro `\language`.

\bbl@pop@lang This macro stores its first element (which is delimited by the ‘+’-sign) in `\language` and stores the rest of the string in `\bbl@language@stack`.

```

644 \def\bbl@pop@lang#1+#2\@@{%
645   \edef\language{#1}%
646   \xdef\bbl@language@stack{#2}}
647 \let\bbl@ifrestoring\@secondoftwo
648 \def\bbl@pop@language{%
649   \expandafter\bbl@pop@lang\bbl@language@stack\@@
650   \let\bbl@ifrestoring\@firstoftwo
651   \expandafter\bbl@set@language\expandafter{\language}%
652   \let\bbl@ifrestoring\@secondoftwo}

```

An alternative way to identify languages (in the babel sense) with a numerical value is introduced in 3.30.

```

653 \chardef\localeid\z@
654 \gdef\bbl@id@last{0}    % No real need for a new counter
655 \def\bbl@id@assign{%
656   \bbl@ifunset\bbl@id@\language}%
657   {\count@\bbl@id@last\relax
658   \advance\count@\@ne
659   \global\bbl@csarg\chardef{id@\language}\count@
660   \xdef\bbl@id@last{\the\count@}%
661   \ifcase\bbl@engine\or
662     \directlua{
663       Babel.locale_props[\bbl@id@last] = {}
664       Babel.locale_props[\bbl@id@last].name = '\language'
665       Babel.locale_props[\bbl@id@last].vars = {}
666     }%
667   \fi}%
668 {}%
669 \chardef\localeid\bbl@c{id@}

```

The unprotected part of `\selectlanguage`. In case it is used as environment, declare `\endselectlanguage`, just for safety.

```

670 \let\bbl@select@opts\@empty
671 \expandafter\def\csname selectlanguage \endcsname{%
672   \@ifnextchar[\bbl@select@s{\bbl@select@s[]}}
673 \def\bbl@select@s[#1]#2{%
674   \bbl@forkv{#1}{\bbl@array@set\bbl@select@opts{##1}{##2}}%
675   \ifnum\bbl@hymapsel=\@cclv\let\bbl@hymapsel\tw\fi
676   \bbl@push@language
677   \aftergroup\bbl@pop@language
678   \bbl@set@language{#2}}
679 \let\endselectlanguage\relax

```

\bbl@set@language The macro `\bbl@set@language` takes care of switching the language environment *and* of writing entries on the auxiliary files.

```

680 \def\BabelContentsFiles{toc,lof,lot}
681 \def\bbl@set@language#1{% from selectlanguage, pop@
682   % The old buggy way. Preserved for compatibility, but simplified
683   \edef\language{\expandafter\string#1\@empty}%
684   \select@language{\language}%

```

```

685 \bbl@array@get\bbl@select@opts{main}\bbl@tempa
686 \ifx\bbl@tempa\relax\else
687 \let\bbl@main@language\localename
688 \let\mainlocalename\localename
689 \fi
690 % write to aux files
691 \expandafter\ifx\csname date\language\endcsname\relax\else
692 \if@files
693 \bbl@array@get\bbl@select@opts{nofiles}\bbl@tempa
694 \ifx\bbl@tempa\relax
695 \ifx\babel@aux@gobbletwo\else % Set if single in the first, redundant
696 \bbl@savelastskip
697 \protected@write\@auxout{}\string\babel@aux{\bbl@auxname}{}}%
698 \bbl@restorelastskip
699 \fi
700 \bbl@usehooks{write}}}%
701 \fi
702 \fi
703 \fi
704 \let\bbl@select@opts\@empty}
705 %
706 \let\bbl@restorelastskip\relax
707 \let\bbl@savelastskip\relax
708 %
709 \def\select@language#1{% from set@, babel@aux, babel@toc
710 \ifx\bbl@select@name\@empty
711 \def\bbl@select@name{select}%
712 \fi
713 % set hymap
714 \ifnum\bbl@hymapsel=\@ccclv\chardef\bbl@hymapsel4\relax\fi
715 % set name (when coming from babel@aux)
716 \edef\language{#1}%
717 \bbl@fixname\language
718 % define \localename when coming from set@, with a trick
719 \ifx\scantokens\@undefined
720 \def\localename{??}%
721 \else
722 \bbl@exp{\scantokens{\def\localename{\language}\noexpand}\relax}%
723 \fi
724 \bbl@provide@locale
725 \bbl@iflanguage\language{%
726 \let\bbl@select@type\z@
727 \expandafter\bbl@switch\expandafter{\language}}%
728 %% TODO %% Plain:
729 \def\babel@aux#1#2{%
730 \select@language{#1}%
731 \bbl@foreach\BabelContentsFiles{% \relax -> don't assume vertical mode
732 \@writefile{##1}{\babel@toc{#1}{#2}\relax}}}%
733 \def\babel@toc#1#2{%
734 \select@language{#1}}

```

First, check if the user asks for a known language. If so, update the value of `\language` and call `\originalTeX` to bring $\TeX$ in a certain pre-defined state.

```

735 \newif\ifbbl@usedategroup
736 \let\bbl@savedextras\@empty
737 \def\bbl@switch#1{% from select@, foreign@
738 % restore
739 \originalTeX
740 \expandafter\def\expandafter\originalTeX\expandafter{%
741 \csname noextras#1\endcsname
742 \let\originalTeX\@empty
743 \babel@beginsave}%
744 \bbl@usehooks{afterreset}}}%

```

```

745 \languageshorthands{none}%
746 % set the locale id
747 \bbl@id@assign
748 % switch captions, date
749 \bbl@bsphack
750 \ifcase\bbl@select@type
751   \csname captions#1\endcsname\relax
752   \csname date#1\endcsname\relax
753 \else
754   \bbl@array@get\bbl@select@opts{captions}\bbl@tempa
755   \ifx\bbl@tempa\relax\else
756     \csname captions#1\endcsname\relax
757   \fi
758   \bbl@array@get\bbl@select@opts{date}\bbl@tempa
759   \ifx\bbl@tempa\relax\else % if \foreign... within \<language>date
760     \csname date#1\endcsname\relax
761   \fi
762 \fi
763 \bbl@esphack
764 % switch extras
765 \csname bbl@preextras@#1\endcsname
766 \bbl@usehooks{beforeextras}{}%
767 \csname extras#1\endcsname\relax
768 \bbl@usehooks{afterextras}{}%
769 % > babel-ensure
770 % > babel-sh-<short>
771 % > babel-bidi
772 % > babel-fontspec
773 \let\bbl@savedextras\@empty
774 % hyphenation - case mapping
775 \ifcase\bbl@opt@hyphenmap\or
776   \def\BabelLower##1##2{\lccode##1=##2\relax}%
777   \ifnum\bbl@hymap>4\else
778     \csname\language @bbl@hyphenmap\endcsname
779   \fi
780   \chardef\bbl@opt@hyphenmap\z@
781 \else
782   \ifnum\bbl@hymap>\bbl@opt@hyphenmap\else
783     \csname\language @bbl@hyphenmap\endcsname
784   \fi
785 \fi
786 \let\bbl@hymap\@cclv
787 % hyphenation - select rules
788 \ifnum\csname l@\language\endcsname=\l@unhyphenated
789   \edef\bbl@tempa{u}%
790 \else
791   \edef\bbl@tempa{\bbl@cl{\lnbrk}}%
792 \fi
793 % linebreaking - handle u, e, k (v in the future)
794 \bbl@xin@{/u}{/\bbl@tempa}%
795 \ifin@else\bbl@xin@{/e}{/\bbl@tempa}\fi % elongated forms
796 \ifin@else\bbl@xin@{/k}{/\bbl@tempa}\fi % only kashida
797 \ifin@else\bbl@xin@{/p}{/\bbl@tempa}\fi % padding (e.g., Tibetan)
798 \ifin@else\bbl@xin@{/v}{/\bbl@tempa}\fi % variable font
799 % hyphenation - save mins
800 \babel@savevariable\lefthyphenmin
801 \babel@savevariable\righthyphenmin
802 \ifnum\bbl@engine=\@ne
803   \babel@savevariable\hyphenationmin
804 \fi
805 \ifin@
806   % unhyphenated/kashida/elongated/padding = allow stretching
807   \language\l@unhyphenated

```

```

808 \babel@savevariable\emergencystretch
809 \emergencystretch\maxdimen
810 \babel@savevariable\hbadness
811 \hbadness\@M
812 \else
813 % other = select patterns
814 \bbl@patterns{#1}%
815 \fi
816 % hyphenation - set mins
817 \expandafter\ifx\csname #1hyphenmins\endcsname\relax
818 \set@hyphenmins\tw@\thr@@\relax
819 \@nameuse{bbl@hyphenmins@}%
820 \else
821 \expandafter\expandafter\expandafter\set@hyphenmins
822 \csname #1hyphenmins\endcsname\relax
823 \fi
824 \@nameuse{bbl@hyphenmins@}%
825 \@nameuse{bbl@hyphenmins@language}%
826 \@nameuse{bbl@hyphenatmin@}%
827 \@nameuse{bbl@hyphenatmin@language}%
828 \let\bbl@selectorname\empty

```

otherlanguage It can be used as an alternative to using the `\selectlanguage` declarative command.

```

829 \edef\otherlanguage{%
830 \noexpand\protect
831 \expandafter\noexpand\csname otherlanguage \endcsname}
832 \expandafter\def\csname otherlanguage \endcsname{%
833 \ifstar{\@nameuse{otherlanguage*}}\bbl@otherlanguage}
834 \def\bbl@otherlanguage#1{%
835 \def\bbl@selectorname{other}%
836 \ifnum\bbl@hymapsel=\ccclv\let\bbl@hymapsel\thr@@\fi
837 \csname selectlanguage \endcsname{#1}%
838 \ignorespaces}

```

The `\endotherlanguage` part of the environment tries to hide itself when it is called in horizontal mode.

```

839 \long\def\endotherlanguage{\@ignoretrue\ignorespaces}

```

otherlanguage* It is meant to be used when a large part of text from a different language needs to be typeset, but without changing the translation of words such as ‘figure’.

```

840 \expandafter\def\csname otherlanguage*\endcsname{%
841 \ifnextchar[\bbl@otherlanguage@s{\bbl@otherlanguage@s[]}}
842 \def\bbl@otherlanguage@s[#1]#2{%
843 \def\bbl@selectorname{other*}%
844 \ifnum\bbl@hymapsel=\ccclv\chardef\bbl@hymapsel4\relax\fi
845 \bbl@forkv{#1}{\bbl@array@set\bbl@select@opts{##1}{##2}}%
846 \foreign@language{#2}}
847 \expandafter\let\csname endotherlanguage*\endcsname\relax

```

\foreignlanguage This command takes two arguments, the first argument is the name of the language to use for typesetting the text specified in the second argument.

```

848 \providecommand\bbl@beforeforeign{}
849 \edef\foreignlanguage{%
850 \noexpand\protect
851 \expandafter\noexpand\csname foreignlanguage \endcsname}
852 \expandafter\def\csname foreignlanguage \endcsname{%
853 \ifstar\bbl@foreign@s\bbl@foreign@x}
854 \providecommand\bbl@foreign@x[3][]{%
855 \begingroup

```

```

856 \def\bbl@selectorname{foreign}%
857 \bbl@forkv{#1}{\bbl@array@set\bbl@select@opts{##1}{##2}}%
858 \let\BabelText\@firstofone
859 \bbl@beforeforeign
860 \foreign@language{#2}%
861 \bbl@usehooks{foreign}{}%
862 \BabelText{#3}% Now in horizontal mode!
863 \endgroup}
864 \def\bbl@foreign@s#1#2{%
865 \begingroup
866 {\par}%
867 \def\bbl@selectorname{foreign*}%
868 \let\bbl@select@opts\@empty
869 \let\BabelText\@firstofone
870 \foreign@language{#1}%
871 \bbl@usehooks{foreign*}{}%
872 \bbl@dirparastext
873 \BabelText{#2}% Still in vertical mode!
874 {\par}%
875 \endgroup}
876 \providecommand\BabelWrapText[1]{%
877 \def\bbl@tempa{\def\BabelText####1}%
878 \expandafter\bbl@tempa\expandafter{\BabelText{#1}}}

```

\foreign@language This macro does the work for \foreignlanguage and the otherlanguage* environment.

```

879 \def\foreign@language#1{%
880 % set name
881 \edef\languagename{#1}%
882 \ifbbl@usedategroup
883 \bbl@array@set\bbl@select@opts{date}{\@empty}%
884 \bbl@usedategroupfalse
885 \fi
886 \bbl@fixname\languagename
887 \let\localename\languagename
888 \bbl@provide@locale
889 \bbl@iflanguage\languagename{%
890 \let\bbl@select@type\@ne
891 \expandafter\bbl@switch\expandafter{\languagename}}

```

The following macro executes conditionally some code based on the selector being used.

```

892 \def\IfBabelSelectorTF#1{%
893 \bbl@xin@{,\bbl@selectorname,}{,\zap@space#1 \@empty,}%
894 \ifin@
895 \expandafter\@firstoftwo
896 \else
897 \expandafter\@secondoftwo
898 \fi}

```

\bbl@patterns This macro selects the hyphenation patterns by changing the \language register.

```

899 \let\bbl@hyphlist\@empty
900 \let\bbl@hyphenation@\relax
901 \let\bbl@pttnlist\@empty
902 \let\bbl@patterns@\relax
903 \let\bbl@hymapset=\@cclv
904 \def\bbl@patterns#1{%
905 \language=\expandafter\ifx\csname l@#1:\f@encoding\endcsname\relax
906 \csname l@#1\endcsname
907 \edef\bbl@tempa{#1}%
908 \else

```

```

909     \csname l@#1:\f@encoding\endcsname
910     \edef\bbl@tempa{#1:\f@encoding}%
911     \fi
912     \@expandtwoargs\bbl@usehooks{patterns}{{#1}{\bbl@tempa}}%
913     % > luatex
914     \@ifundefined{bbl@hyphenation@}{{}% Can be \relax!
915     \beginingroup
916     \bbl@xin@{,\number\language,}{,\bbl@hyphlist}%
917     \ifin@else
918     \@expandtwoargs\bbl@usehooks{hyphenation}{{#1}{\bbl@tempa}}%
919     \hyphenation{%
920     \bbl@hyphenation@
921     \@ifundefined{bbl@hyphenation@#1}%
922     \@empty
923     {\space\csname bbl@hyphenation@#1\endcsname}}%
924     \xdef\bbl@hyphlist{\bbl@hyphlist\number\language,}%
925     \fi
926     \endgroup}}

```

hyphenrules It can be used to select *just* the hyphenation rules.

```

927 \def\hyphenrules#1{%
928   \edef\bbl@tempf{#1}%
929   \bbl@fixname\bbl@tempf
930   \bbl@iflanguage\bbl@tempf{%
931     \expandafter\bbl@patterns\expandafter{\bbl@tempf}%
932     \ifx\languageshorthands\@undefined\else
933       \languageshorthands{none}%
934     \fi
935     \expandafter\ifx\csname\bbl@tempf hyphenmins\endcsname\relax
936       \set@hyphenmins\tw@\thr@@\relax
937     \else
938       \expandafter\expandafter\expandafter\set@hyphenmins
939       \csname\bbl@tempf hyphenmins\endcsname\relax
940     \fi}}
941 \let\endhyphenrules\@empty

```

\providehyphenmins The macro `\providehyphenmins` should be used in the language definition files to provide a *default* setting for the hyphenation parameters `\lefthyphenmin` and `\righthyphenmin`

```

942 \def\providehyphenmins#1#2{%
943   \expandafter\ifx\csname #1hyphenmins\endcsname\relax
944     \@namedef{#1hyphenmins}{#2}%
945   \fi}

```

\set@hyphenmins This macro sets the values of `\lefthyphenmin` and `\righthyphenmin`. It expects two values as its argument.

```

946 \def\set@hyphenmins#1#2{%
947   \lefthyphenmin#1\relax
948   \righthyphenmin#2\relax}

```

\ProvidesLanguage The identification code for each file is something that was introduced in $\text{\LaTeX 2}_{\epsilon}$.

```

949 \ifx\ProvidesFile\@undefined
950   \def\ProvidesLanguage#1[#2 #3 #4]{%
951     \wlog{Language: #1 #4 #3 <#2>}%
952   }
953 \else
954   \def\ProvidesLanguage#1{%
955     \beginingroup

```

```

956 \catcode\ 10 %
957 \@makeother\/%
958 \@ifnextchar[%
959 {\@provideslanguage{#1}}{\@provideslanguage{#1}[]}}
960 \def\@provideslanguage#1[#2]{%
961 \wlog{Language: #1 #2}%
962 \expandafter\xdef\csname ver@#1.1\endcsname{#2}%
963 \endgroup}
964 \fi
965 \ifx\originalTeX\@undefined\let\originalTeX\@empty\fi
966 \ifx\babel@beginsave\@undefined\let\babel@beginsave\relax\fi

```

A few macro names are reserved for future releases of babel, which will use the concept of ‘locale’:

```

967 \providecommand\setlocale{\bbl@error{not-yet-available}}{}{}
968 \let\uselocale\setlocale
969 \let\locale\setlocale
970 \let\selectlocale\setlocale
971 \let\textlocale\setlocale
972 \let\textlanguage\setlocale
973 \let\languagegettext\setlocale

```

4.2. Errors

\@nolanerr

\@nopatterns The babel package will signal an error when a documents tries to select a language that hasn’t been defined earlier.

\@noopterr When the package was loaded without options not everything will work as expected. An error message is issued in that case.

```

974 \edef\bbl@nulllanguage{\string\language=0}
975 \def\bbl@nocaption#1#2{\NoCaseChange{\protect\bbl@nocaption@i{#1}{#2}}}
976 \def\bbl@nocaption@i#1#2{% 1: text to be printed 2: caption macro \langXname
977 \global\@namedef{#2}{\textbf{?#1?}}}%
978 \@nameuse{#2}%
979 \edef\bbl@tempa{#1}%
980 \bbl@sreplace\bbl@tempa{name}}}%
981 \bbl@warning{%
982 \@backslashchar#1 not set for '\language'. Please,\\%
983 define it after the language has been loaded\\%
984 (typically in the preamble) with:\\%
985 \string\setlocalecaption{\language}{\bbl@tempa}{.}\\%
986 Feel free to contribute on github.com/latex3/babel.\\%
987 Reported}}
988 \def\bbl@tentative{\protect\bbl@tentative@i}
989 \def\bbl@tentative@i#1{%
990 \bbl@warning{%
991 Some functions for '#1' are tentative.\\%
992 They might not work as expected and their behavior\\%
993 could change in the future.\\%
994 Reported}}
995 \def\@nolanerr#1{\bbl@error{undefined-language}{#1}}{}
996 \def\@nopatterns#1{%
997 \bbl@warning
998 {No hyphenation patterns were preloaded for\\%
999 the language '#1' into the format.\\%
1000 Please, configure your TeX system to add them and\\%
1001 rebuild the format. Now I will use the patterns\\%
1002 preloaded for \bbl@nulllanguage\space instead}}
1003 \let\bbl@usehooks\@gobbletwo

```

5. Finish the very basic macros

Here ended the base option ($\LaTeX$).

6. Only Plain

```
1004 \ifx\bbbl@onlyswitch\@empty\endinput\fi
```

6.1. More on selection

\babelensure The user command just parses the optional argument and creates a new macro named `\bbbl@e@<language>`.

```
1005 \bbbl@trace{Defining babelensure}
1006 \newcommand\babelensure[2][]{}%
1007 \AddBabelHook{babel-ensure}{afterextras}{%
1008   \ifcase\bbbl@select@type
1009     \bbbl@ccl{e}%
1010   \fi}%
1011 \begingroup
1012   \let\bbbl@ens@include\@empty
1013   \let\bbbl@ens@exclude\@empty
1014   \def\bbbl@ens@fontenc{\relax}%
1015   \def\bbbl@tempb##1{%
1016     \ifx\@empty##1\else\noexpand##1\expandafter\bbbl@tempb\fi}%
1017   \edef\bbbl@tempa{\bbbl@tempb#1\@empty}%
1018   \def\bbbl@tempb##1=##2\@{\@namedef{\bbbl@ens@##1}{##2}}%
1019   \bbbl@foreach\bbbl@tempa{\bbbl@tempb##1\@}%
1020   \def\bbbl@tempc{\bbbl@ensure}%
1021   \expandafter\bbbl@add\expandafter\bbbl@tempc\expandafter{%
1022     \expandafter{\bbbl@ens@include}}%
1023   \expandafter\bbbl@add\expandafter\bbbl@tempc\expandafter{%
1024     \expandafter{\bbbl@ens@exclude}}%
1025   \toks@\expandafter{\bbbl@tempc}%
1026   \bbbl@exp{%
1027 \endgroup
1028 \def<\bbbl@e@#2>{\the\toks@{\bbbl@ens@fontenc}}}%
1029 \def\bbbl@ensure#1#2#3% 1: include 2: exclude 3: fontenc
1030 \def\bbbl@tempb##1{% elt for (excluding) \bbbl@captionslist list
1031   \ifx##1\@undefined % 3.32 - Don't assume the macro exists
1032     \edef##1{\noexpand\bbbl@nocaption
1033       {\bbbl@stripslash##1}{\language\bbbl@stripslash##1}}%
1034   \fi
1035   \ifx##1\@empty\else
1036     \in@{##1}{#2}%
1037     \ifin@ \else
1038       \let\bbbl@tempc\language\name
1039       \bbbl@replace\bbbl@tempc{-}{@}%
1040       \bbbl@ifunset{\bbbl@ensure@\bbbl@tempc}%
1041       {\bbbl@exp{%
1042         \\\DeclareRobustCommand\<\bbbl@ensure@\bbbl@tempc>[1]{%
1043           \\\foreignlanguage{\language\name}%
1044             {\ifx\relax#3\else
1045               \\\fontencoding{#3}\selectfont
1046               \fi
1047             #####1}}}%
1048       }%
1049       \toks@\expandafter{##1}%
1050       \edef##1{%
1051         \bbbl@csarg\noexpand{\bbbl@tempc}%
1052         {\the\toks@}}%
1053     \fi
1054   \expandafter\bbbl@tempb
```

```

1055 \fi}%
1056 \expandafter\bbbl@tempb\bbbl@captionslist\today\@empty
1057 \def\bbbl@tempa##1{% elt for include list
1058 \ifx##1\@empty\else
1059 \let\bbbl@tempc\languagename
1060 \bbbl@replace\bbbl@tempc{-}{@}%
1061 \bbbl@csarg\in@{ensure@\bbbl@tempc\expandafter}\expandafter{##1}%
1062 \ifin@\else
1063 \bbbl@tempb##1\@empty
1064 \fi
1065 \expandafter\bbbl@tempa
1066 \fi}%
1067 \bbbl@tempa#1\@empty}
1068 \def\bbbl@captionslist{%
1069 \prefacename\refname\abstractname\bibname\chaptername\appendixname
1070 \contentsname\listfigurename\listtablename\indexname\figurename
1071 \tablename\partname\enclname\ccname\headtoname\pagename\seename
1072 \alsoname\proofname\glossaryname}

```

6.2. Short tags

\babetags This macro is straightforward.

```

1073 \bbbl@trace{Short tags}
1074 \newcommand\babetags[1]{%
1075 \edef\bbbl@tempa{\zap@space#1 \@empty}%
1076 \def\bbbl@tempb##1=##2\@{;%
1077 \edef\bbbl@tempc{%
1078 \noexpand\newcommand
1079 \expandafter\noexpand\csname ##1\endcsname{%
1080 \noexpand\protect
1081 \expandafter\noexpand\csname otherlanguage*\endcsname{##2}}
1082 \noexpand\newcommand
1083 \expandafter\noexpand\csname text##1\endcsname{%
1084 \noexpand\foreignlanguage{##2}}}
1085 \bbbl@tempc}%
1086 \bbbl@for\bbbl@tempa\bbbl@tempa{%
1087 \expandafter\bbbl@tempb\bbbl@tempa\@{}}

```

6.3. Compatibility with language.def

Plain e-TeX doesn't rely on language.dat, but babel can be made compatible with this format easily.

```

1088 \bbbl@trace{Compatibility with language.def}
1089 \ifx\directlua\@undefined\else
1090 \ifx\bbbl@luapatterns\@undefined
1091 \input luababel.def
1092 \fi
1093 \fi
1094 \ifx\bbbl@languages\@undefined
1095 \ifx\directlua\@undefined
1096 \openin0 = language.def
1097 \ifeof0
1098 \closein0
1099 \message{I couldn't find the file language.def}
1100 \else
1101 \closein0
1102 \begingroup
1103 \def\addlanguage#1#2#3#4#5{%
1104 \expandafter\ifx\csname lang@#1\endcsname\relax\else
1105 \global\expandafter\let\csname l@#1\expandafter\endcsname
1106 \csname lang@#1\endcsname

```

```

1107         \fi}%
1108         \def\uselanguage#1{%
1109         \input language.def
1110         \endgroup
1111     \fi
1112 \fi
1113 \chardef\l@english\z@
1114 \fi

```

\addto It takes two arguments, a *(control sequence)* and TeX-code to be added to the *(control sequence)*.

```

1115 \def\addto#1#2{%
1116     \ifx#1\@undefined
1117         \def#1{#2}%
1118     \else
1119         \ifx#1\relax
1120             \def#1{#2}%
1121         \else
1122             {\toks@\expandafter{#1#2}%
1123             \xdef#1{\the\toks@}}%
1124         \fi
1125     \fi}

```

6.4. Hooks

Admittedly, the current implementation is a somewhat simplistic.

```

1126 \bbl@trace{Hooks}
1127 \newcommand\AddBabelHook[3][{}]{%
1128     \bbl@i funset{\bbl@hk@#2}{\EnableBabelHook{#2}}}%
1129     \def\bbl@tempa##1,##2,##3\@empty{\def\bbl@tempb{##2}}%
1130     \expandafter\bbl@tempa\bbl@evargs,##3=,\@empty
1131     \bbl@i funset{\bbl@ev@#2@#3@#1}%
1132         {\bbl@csarg\bbl@add{ev@#3@#1}{\bbl@elth{#2}}}%
1133         {\bbl@csarg\let{ev@#2@#3@#1}\relax}%
1134     \bbl@csarg\newcommand{ev@#2@#3@#1}{\bbl@tempb}}
1135 \newcommand\EnableBabelHook[1]{\bbl@csarg\let{hk@#1}\@firstofone}
1136 \newcommand\DisableBabelHook[1]{\bbl@csarg\let{hk@#1}\@gobble}
1137 \def\bbl@usehooks{\bbl@usehooks@lang\language}
1138 \def\bbl@usehooks@lang#1#2#3{% Test for Plain
1139     \ifx\UseHook\@undefined\else\UseHook{babel/*/#2}\fi
1140     \def\bbl@elth##1{%
1141         \bbl@cs{hk@##1}{\bbl@cs{ev@##1@#2@#3}}%
1142     \bbl@cs{ev@#2@}%
1143     \ifx\language\@undefined\else % Test required for Plain (?)
1144         \ifx\UseHook\@undefined\else\UseHook{babel/#1/#2}\fi
1145         \def\bbl@elth##1{%
1146             \bbl@cs{hk@##1}{\bbl@cs{ev@##1@#2@#1#3}}%
1147             \bbl@cs{ev@#2@#1}%
1148         \fi}
1149 \def\bbl@evargs{,% <- don't delete this comma
1150     everylanguage=1,loadkernel=1,loadpatterns=1,loadexceptions=1,%
1151     adddialect=2,patterns=2,defaultcommands=0,encodedcommands=2,write=0,%
1152     beforeextras=0,afterextras=0,stopcommands=0,stringprocess=0,%
1153     hyphenation=2,initiateactive=3,afterreset=0,foreign=0,foreign*=0,%
1154     beforestart=0,language=2,begindocument=1}
1155 \ifx\NewHook\@undefined\else % Test for Plain (?)
1156     \def\bbl@tempa#1=#2@@{\NewHook{babel/#1}\NewHook{babel/*/#1}}
1157     \bbl@foreach\bbl@evargs{\bbl@tempa#1@@}
1158 \fi

```

6.5. Setting up language files

\LdfInit \LdfInit macro takes two arguments. The first argument is the name of the language that will be defined in the language definition file; the second argument is either a control sequence or a string from which a control sequence should be constructed.

```
1159 \bbl@trace{Macros for setting language files up}
1160 \def\bbl@ldfinit{%
1161   \let\bbl@screset\@empty
1162   \let\BabelStrings\bbl@opt@string
1163   \let\BabelOptions\@empty
1164   \let\BabelLanguages\relax
1165   \ifx\originalTeX\@undefined
1166     \let\originalTeX\@empty
1167   \else
1168     \originalTeX
1169   \fi}
1170 \def\LdfInit#1#2{%
1171   \chardef\atcatcode=\catcode`\@
1172   \catcode`\@=11\relax
1173   \chardef\eqcatcode=\catcode`\=
1174   \catcode`\==12\relax
1175   \ifpackagewith{babel}{ensureinfo=off}{}%
1176   {\ifx\InputIfFileExists\@undefined\else
1177     \bbl@ifunset\bbl@lname@#1}%
1178     {{\let\bbl@ensuring\@empty % Flag used in babel-serbianc.tex
1179       \def\language#1}%
1180       \bbl@id@assign
1181       \bbl@load@info{#1}}}%
1182   {}%
1183   \fi}%
1184   \expandafter\if\expandafter\@backslashchar
1185     \expandafter\@car\string#2\@nil
1186     \ifx#2\@undefined\else
1187       \ldf@quit{#1}%
1188     \fi
1189   \else
1190     \expandafter\ifx\csname#2\endcsname\relax\else
1191       \ldf@quit{#1}%
1192     \fi
1193   \fi
1194   \bbl@ldfinit}
```

\ldf@quit This macro interrupts the processing of a language definition file.

```
1195 \def\ldf@quit#1{%
1196   \expandafter\main@language\expandafter{#1}%
1197   \catcode`\@=\atcatcode \let\atcatcode\relax
1198   \catcode`\=\eqcatcode \let\eqcatcode\relax
1199   \endinput}
```

\ldf@finish This macro takes one argument. It is the name of the language that was defined in the language definition file.

```
1200 \def\bbl@afterldf{%
1201   \bbl@afterlang
1202   \let\bbl@afterlang\relax
1203   \let\BabelModifiers\relax
1204   \let\bbl@screset\relax}%
1205 \def\ldf@finish#1{%
1206   \loadlocalcfg{#1}%
1207   \bbl@afterldf
1208   \expandafter\main@language\expandafter{#1}%
1209   \catcode`\@=\atcatcode \let\atcatcode\relax}
```

```
1210 \catcode`\==\eqcatcode \let\eqcatcode\relax}
```

After the preamble of the document the commands `\LdfInit`, `\ldf@quit` and `\ldf@finish` are no longer needed.

```
1211 \@onlypreamble\LdfInit
1212 \@onlypreamble\ldf@quit
1213 \@onlypreamble\ldf@finish
```

\main@language

\bbl@main@language This command should be used in the various language definition files.

```
1214 \def\main@language#1{%
1215   \def\bbl@main@language{#1}%
1216   \let\language\main@language
1217   \let\localename\bbl@main@language
1218   \let\mainlocalename\bbl@main@language
1219   \bbl@id@assign
1220   \ifcase\bbl@engine\or
1221     \ifx\setattribute\@undefined\else
1222       \setattribute\bbl@attr@locale\localeid
1223     \fi
1224   \fi
1225   \bbl@patterns{\language}}
```

We also have to make sure that some code gets executed at the beginning of the document.

```
1226 \AtBeginDocument{\expandafter\selectlanguage\expandafter{\bbl@main@language}}
```

6.6. Shorthands

The macro `\initiate@active@char` below takes all the necessary actions to make its argument a shorthand character.

```
1227 \bbl@trace{Shorhands}
1228 \def\bbl@withactive#1#2{%
1229   \begingroup
1230     \lccode`~=#2\relax
1231     \lowercase{\endgroup#1~}}
```

\bbl@add@special The macro `\bbl@add@special` is used to add a new character (or single character control sequence) to the macro `\dospecials` (and `\@sanitize` if *LaTeX* is used).

```
1232 \def\bbl@add@special#1{% 1:a macro like \", \?, etc.
1233   \bbl@add\dospecials{\do#1}% test @sanitize = \relax, for back. compat.
1234   \bbl@ifunset{@sanitize}{\bbl@add\@sanitize{\@makeother#1}}%
1235   \ifx\nfss@catcodes\@undefined\else
1236     \begingroup
1237       \catcode`#1\active
1238       \nfss@catcodes
1239       \ifnum\catcode`#1=\active
1240         \endgroup
1241         \bbl@add\nfss@catcodes{\@makeother#1}%
1242       \else
1243         \endgroup
1244       \fi
1245   \fi}
```

\initiate@active@char A language definition file can call this macro to make a character active.

```

1246 \def\bbl@active@def#1#2#3#4{%
1247   \@namedef{#3#1}{%
1248     \expandafter\ifx\csname#2@sh@#1\endcsname\relax
1249       \bbl@afterelse\bbl@sh@select#2#1{#3@arg#1}{#4#1}%
1250     \else
1251       \bbl@afterfi\csname#2@sh@#1\endcsname
1252     \fi}%
1253   \long\@namedef{#3@arg#1}##1{%
1254     \expandafter\ifx\csname#2@sh@#1\string##1\endcsname\relax
1255       \bbl@afterelse\csname#4#1\endcsname##1%
1256     \else
1257       \bbl@afterfi\csname#2@sh@#1\string##1\endcsname
1258     \fi}}%
1259 \def\initiate@active@char#1{%
1260   \bbl@ifunset{active@char\string#1}%
1261   {\bbl@withactive
1262     {\expandafter\@initiate@active@char\expandafter}#1\string#1#1}%
1263   {}}
1264 \def\@initiate@active@char#1#2#3{%
1265   \bbl@csarg\edef{oricat@#2}{\catcode`#2=\the\catcode`#2\relax}%
1266   \ifx#1\@undefined
1267     \bbl@csarg\def{oridef@#2}{\def#1{\active@prefix#1\@undefined}}%
1268   \else
1269     \bbl@csarg\let{oridef@#2}#1%
1270     \bbl@csarg\edef{oridef@#2}{%
1271       \let\noexpand#1%
1272       \expandafter\noexpand\csname bbl@oridef@#2\endcsname}%
1273   \fi
1274   \ifx#1#3\relax
1275     \expandafter\let\csname normal@char#2\endcsname#3%
1276   \else
1277     \bbl@info{Making #2 an active character}%
1278     \ifnum\mathcode`#2=\ifodd\bbl@engine"1000000 \else"8000 \fi
1279     \@namedef{normal@char#2}{%
1280       \textormath{#3}{\csname bbl@oridef@#2\endcsname}}%
1281   \else
1282     \@namedef{normal@char#2}{#3}%
1283   \fi
1284   \bbl@restoreactive{#2}%
1285   \AtBeginDocument{%
1286     \catcode`#2\active
1287     \if@filesw
1288       \immediate\write\@mainaux{\catcode`\string#2\active}%
1289     \fi}%
1290   \expandafter\bbl@add@special\csname#2\endcsname
1291   \catcode`#2\active
1292   \fi
1293   \let\bbl@tempa\@firstoftwo
1294   \if\string^#2%
1295     \def\bbl@tempa{\noexpand\textormath}%
1296   \else
1297     \ifx\bbl@mathnormal\@undefined\else
1298       \let\bbl@tempa\bbl@mathnormal
1299     \fi
1300   \fi
1301   \expandafter\edef\csname active@char#2\endcsname{%
1302     \bbl@tempa
1303     {\noexpand\if@safe@actives
1304       \noexpand\expandafter
1305       \expandafter\noexpand\csname normal@char#2\endcsname
1306       \noexpand\else

```

```

1307         \noexpand\expandafter
1308         \expandafter\noexpand\csname bbl@doactive#2\endcsname
1309         \noexpand\fi}%
1310         {\expandafter\noexpand\csname normal@char#2\endcsname}}}%
1311 \bbl@csarg\edef{doactive#2}{%
1312   \expandafter\noexpand\csname user@active#2\endcsname}%
1313 \bbl@csarg\edef{active@#2}{%
1314   \noexpand\active@prefix\noexpand#1%
1315   \expandafter\noexpand\csname active@char#2\endcsname}%
1316 \bbl@csarg\edef{normal@#2}{%
1317   \noexpand\active@prefix\noexpand#1%
1318   \expandafter\noexpand\csname normal@char#2\endcsname}%
1319 \bbl@ncarg\let#1\bbl@normal@#2}%
1320 \bbl@active@def#2\user@group{user@active}{language@active}%
1321 \bbl@active@def#2\language@group{language@active}{system@active}%
1322 \bbl@active@def#2\system@group{system@active}{normal@char}%
1323 \expandafter\edef\csname\user@group @sh@#2@\endcsname
1324   {\expandafter\noexpand\csname normal@char#2\endcsname}%
1325 \expandafter\edef\csname\user@group @sh@#2@\string\protect@\endcsname
1326   {\expandafter\noexpand\csname user@active#2\endcsname}%
1327 \if\string'#2%
1328   \let\prim@s\bbl@prim@s
1329   \let\active@math@prime#1%
1330 \fi
1331 \bbl@usehooks{initiateactive}{{#1}{#2}{#3}}}
1332 \@ifpackagewith{babel}{KeepShorthandsActive}%
1333 {\let\bbl@restoreactive@gobble}%
1334 {\def\bbl@restoreactive#1{%
1335   \bbl@exp{%
1336     \\\AfterBabelLanguage\\CurrentOption
1337     {\catcode`#1=\the\catcode`#1\relax}%
1338     \\\AtEndOfPackage
1339     {\catcode`#1=\the\catcode`#1\relax}}}%
1340   \AtEndOfPackage{\let\bbl@restoreactive@gobble}}

```

\bbl@sh@select This command helps the shorthand supporting macros to select how to proceed.

```

1341 \def\bbl@sh@select#1#2{%
1342   \expandafter\ifx\csname#1@sh@#2@sel\endcsname\relax
1343     \bbl@afterelse\bbl@scndcs
1344   \else
1345     \bbl@afterfi\csname#1@sh@#2@sel\endcsname
1346   \fi}

```

\active@prefix Used in the expansion of active characters has a function similar to \OT1-cmd in that it \protects the active character whenever \protect is *not* \@typeset@protect.

```

1347 \beginingroup
1348 \bbl@ifunset{ifincsname}
1349 {\gdef\active@prefix#1{%
1350   \ifx\protect\@typeset@protect
1351     \else
1352       \ifx\protect\@unexpandable@protect
1353         \noexpand#1%
1354       \else
1355         \protect#1%
1356       \fi
1357       \expandafter\@gobble
1358     \fi}}
1359 {\gdef\active@prefix#1{%
1360   \ifincsname

```

```

1361      \string#1%
1362      \expandafter\@gobble
1363    \else
1364      \ifx\protect\@typeset@protect
1365      \else
1366        \ifx\protect\@unexpandable@protect
1367          \noexpand#1%
1368        \else
1369          \protect#1%
1370        \fi
1371      \expandafter\expandafter\expandafter\@gobble
1372    \fi
1373  \fi}}
1374 \endgroup

```

if@safe@actives In some circumstances it is necessary to be able to reset the shorthand to its ‘normal’ value (usually the character with catcode ‘other’) on the fly.

```

1375 \newif\if@safe@actives
1376 \@safe@activesfalse

```

\bbl@restore@actives When the output routine kicks in while the active characters were made “safe” this must be undone in the headers to prevent unexpected typeset results.

```

1377 \def\bbl@restore@actives{\if@safe@actives\@safe@activesfalse\fi}

```

\bbl@activate

\bbl@deactivate Used to change the definition of an active character to expand to `\active@char⟨char⟩` in the case of `\bbl@activate`, or `\normal@char⟨char⟩` in the case of `\bbl@deactivate`.

```

1378 \chardef\bbl@activated\z@
1379 \def\bbl@activate#1{%
1380   \chardef\bbl@activated\@ne
1381   \bbl@withactive{\expandafter\let\expandafter}#1%
1382   \csname bbl@active@\string#1\endcsname}
1383 \def\bbl@deactivate#1{%
1384   \chardef\bbl@activated\tw@
1385   \bbl@withactive{\expandafter\let\expandafter}#1%
1386   \csname bbl@normal@\string#1\endcsname}

```

\bbl@firstcs

\bbl@scndcs These macros are used only as a trick when declaring shorthands.

```

1387 \def\bbl@firstcs#1#2{\csname#1\endcsname}
1388 \def\bbl@scndcs#1#2{\csname#2\endcsname}

```

\declare@shorthand Used to declare a shorthand on a certain level.

```

1389 \def\babel@texpdf#1#2#3#4{%
1390   \ifx\texorpdfstring\undefined
1391     \textormath{#1}{#3}%
1392   \else
1393     \texorpdfstring{\textormath{#1}{#3}}{#2}%
1394   % \texorpdfstring{\textormath{#1}{#3}}{\textormath{#2}{#4}}%
1395   \fi}
1396 %
1397 \def\declare@shorthand#1#2{\@decl@short{#1}#2\@nil}
1398 \def\@decl@short#1#2#3\@nil#4{%
1399   \def\bbl@tempa{#3}%
1400   \ifx\bbl@tempa\@empty
1401     \expandafter\let\csname #1@sh@\string#2@sel\endcsname\bbl@scndcs
1402     \bbl@ifunset{#1@sh@\string#2@}{}%

```

```

1403     {\def\bb@tempa{#4}%
1404     \expandafter\ifx\csname#1@sh@\string#2@endcsname\bb@tempa
1405     \else
1406         \bb@info
1407         {Redefining #1 shorthand \string#2\\%
1408         in language \CurrentOption}%
1409     \fi}%
1410     \@namedef{#1@sh@\string#2@}{#4}%
1411 \else
1412     \expandafter\let\csname #1@sh@\string#2@sel@endcsname\bb@firstcs
1413     \bb@ifunset{#1@sh@\string#2@\string#3@}{}%
1414     {\def\bb@tempa{#4}%
1415     \expandafter\ifx\csname#1@sh@\string#2@\string#3@endcsname\bb@tempa
1416     \else
1417         \bb@info
1418         {Redefining #1 shorthand \string#2\string#3\\%
1419         in language \CurrentOption}%
1420     \fi}%
1421     \@namedef{#1@sh@\string#2@\string#3@}{#4}%
1422 \fi}

```

\textormath Some of the shorthands that will be declared by the language definition files have to be usable in both text and mathmode.

```

1423 \def\textormath{%
1424     \ifmmode
1425         \expandafter\@secondoftwo
1426     \else
1427         \expandafter\@firstoftwo
1428     \fi}

```

\user@group

\language@group

\system@group The current concept of ‘shorthands’ supports three levels or groups of shorthands.

```

1429 \def\user@group{user}
1430 \def\language@group{english}
1431 \def\system@group{system}

```

\useshorthands This is the user level macro.

```

1432 \def\useshorthands{%
1433     \ifstar\bb@usesh@s{\bb@usesh@x{}}
1434 \def\bb@usesh@s#1{%
1435     \bb@usesh@x
1436     {\AddBabelHook{babel-sh-\string#1}{afterextras}{\bb@activate{#1}}}%
1437     {#1}}
1438 \def\bb@usesh@x#1#2{%
1439     \bb@ifshorthand{#2}%
1440     {\def\user@group{user}%
1441     \initiate@active@char{#2}%
1442     #1%
1443     \bb@activate{#2}}%
1444     {\bb@error{shorthand-is-off}{#2}}}

```

\defineshorthand Currently we only support two groups of user level shorthands, named internally user and user@(*language*) (language-dependent user shorthands).

```

1445 \def\user@language@group{user@\language@group}
1446 \def\bb@set@user@generic#1#2{%
1447     \bb@ifunset{user@generic@active#1}%
1448     {\bb@active@def#1\user@language@group{user@active}{user@generic@active}%

```

```

1449 \bbl@active@def#1\user@group{user@generic@active}{language@active}%
1450 \expandafter\edef\csname#2@sh@#1@@\endcsname{%
1451 \expandafter\noexpand\csname normal@char#1\endcsname}%
1452 \expandafter\edef\csname#2@sh@#1@string\protect@\endcsname{%
1453 \expandafter\noexpand\csname user@active#1\endcsname}}%
1454 \@empty}
1455 \newcommand\defineshorthand[3][user]{%
1456 \edef\bbl@tempa{\zap@space#1 \@empty}%
1457 \bbl@for\bbl@tempb\bbl@tempa{%
1458 \if*\expandafter\@car\bbl@tempb\@nil
1459 \edef\bbl@tempb{user@\expandafter\@gobble\bbl@tempb}%
1460 \@expandtwoargs
1461 \bbl@set@user@generic{\expandafter\string\@car#2\@nil}\bbl@tempb
1462 \fi
1463 \declare@shorthand{\bbl@tempb}{#2}{#3}}

```

\languageshorthands A user level command to change the language from which shorthands are used.

```

1464 \def\languageshorthands#1{%
1465 \bbl@ifsamestring{none}{#1}}{%
1466 \bbl@once{short-\localename-#1}{%
1467 \bbl@info{'\localename' activates '#1' shorthands.\\Reported}}}%
1468 \def\language@group{#1}}

```

\aliasshorthand *Deprecated.*

```

1469 \def\aliasshorthand#1#2{%
1470 \bbl@ifshorthand{#2}%
1471 {\expandafter\ifx\csname active@char\string#2\endcsname\relax
1472 \ifx\document\@notprerr
1473 \@notshorthand{#2}%
1474 \else
1475 \initiate@active@char{#2}%
1476 \bbl@ccarg\let{active@char\string#2}{active@char\string#1}%
1477 \bbl@ccarg\let{normal@char\string#2}{normal@char\string#1}%
1478 \bbl@activate{#2}%
1479 \fi
1480 \fi}%
1481 {\bbl@error{shorthand-is-off}{#2}}}

```

\@notshorthand

```

1482 \def\@notshorthand#1{\bbl@error{not-a-shorthand}{#1}}{}

```

\shorthandon

\shorthandoff The first level definition of these macros just passes the argument on to \bbl@switch@sh, adding \@nil at the end to denote the end of the list of characters.

```

1483 \newcommand*\shorthandon[1]{\bbl@switch@sh\@ne#1\@nnil}
1484 \DeclareRobustCommand*\shorthandoff{%
1485 \@ifstar{\bbl@shorthandoff\tw@}{\bbl@shorthandoff\z@}}
1486 \def\bbl@shorthandoff#1#2{\bbl@switch@sh#1#2\@nnil}

```

\bbl@switch@sh The macro \bbl@switch@sh takes the list of characters apart one by one and subsequently switches the category code of the shorthand character according to the first argument of \bbl@switch@sh.

```

1487 \def\bbl@switch@sh#1#2{%
1488 \ifx#2\@nnil\else
1489 \bbl@ifunset\bbl@active@\string#2}%
1490 {\bbl@error{not-a-shorthand-b}{#2}}}%
1491 {\ifcase#1% off, on, off*

```

```

1492     \catcode`#212\relax
1493   \or
1494     \catcode`#2\active
1495     \bbl@ifunset{bbl@shdef@\string#2}%
1496     {}%
1497     {\bbl@withactive{\expandafter\let\expandafter}#2%
1498       \csname bbl@shdef@\string#2\endcsname
1499       \bbl@csarg\let{shdef@\string#2}\relax}%
1500   \ifcase\bbl@activated\or
1501     \bbl@activate{#2}%
1502   \else
1503     \bbl@deactivate{#2}%
1504   \fi
1505   \or
1506     \bbl@ifunset{bbl@shdef@\string#2}%
1507     {\bbl@withactive{\bbl@csarg\let{shdef@\string#2}}#2}%
1508     {}%
1509     \csname bbl@oricat@\string#2\endcsname
1510     \csname bbl@oridef@\string#2\endcsname
1511   \fi}%
1512   \bbl@afterfi\bbl@switch@sh#1%
1513 \fi}

```

Note the value is that at the expansion time; e.g., in the preamble shorthands are usually deactivated.

```

1514 \def\babelshorthand{\active@prefix\babelshorthand\bbl@putsh}
1515 \def\bbl@putsh#1{%
1516   \bbl@ifunset{bbl@active@\string#1}%
1517   {\bbl@putsh@i#1@empty\@nnil}%
1518   {\csname bbl@active@\string#1\endcsname}}
1519 \def\bbl@putsh@i#1#2@nnil{%
1520   \csname\language@group @sh@\string#1@%
1521     \ifx@empty#2\else\string#2@\fi\endcsname}

```

You may want to test if a character is a shorthand. Note it does not test whether the shorthand is on or off.

```

1522 \newcommand\ifbabelshorthand[3]{\bbl@ifunset{bbl@active@\string#1}{#3}{#2}}

```

\bbl@prim@s

\bbl@pr@m@s One of the internal macros that are involved in substituting \prime for each right quote in mathmode is \prim@s.

```

1523 \def\bbl@prim@s{%
1524   \prime\futurelet\@let@token\bbl@pr@m@s}
1525 \def\bbl@if@primes#1#2{%
1526   \ifx#1\@let@token
1527     \expandafter\@firstoftwo
1528   \else\ifx#2\@let@token
1529     \bbl@afterelse\expandafter\@firstoftwo
1530   \else
1531     \bbl@afterfi\expandafter\@secondoftwo
1532   \fi\fi}
1533 \begingroup
1534   \catcode`\^=7 \catcode`\*= \active \lccode`\^=\^
1535   \catcode`\'=12 \catcode`\="= \active \lccode`\"="\^
1536   \lowercase{%
1537     \gdef\bbl@pr@m@s{%
1538       \bbl@if@primes" '%
1539       \pr@@@s
1540       {\bbl@if@primes*\^ \pr@@@t\egroup}}}
1541 \endgroup

```

Usually the ~ is active and expands to \penalty\@M_. When it is written to the aux file it is written expanded. To prevent that and to be able to use the character ~ as a start

character for a shorthand, it is redefined here as a one character shorthand on system level.

```
1542 \initiate@active@char{~}
1543 \declare@shorthand{system}{~}{\leavevmode\nobreak\ }
1544 \bbl@activate{~}
```

\OT1dqpos

\T1dqpos The position of the double quote character is different for the OT1 and T1 encodings.

```
1545 \expandafter\def\csname OT1dqpos\endcsname{127}
1546 \expandafter\def\csname T1dqpos\endcsname{4}
```

When the macro `\f@encoding` is undefined (as it is in plain $\TeX$) we define it here to expand to OT1

```
1547 \ifx\f@encoding\undefined
1548   \def\f@encoding{OT1}
1549 \fi
```

6.7. Language attributes

Language attributes provide a means to give the user control over which features of the language definition files he wants to enable.

\languageattribute The macro `\languageattribute` checks whether its arguments are valid and then activates the selected language attribute.

```
1550 \bbl@trace{Language attributes}
1551 \newcommand\languageattribute[2]{%
1552   \def\bbl@tempc{#1}%
1553   \bbl@fixname\bbl@tempc
1554   \bbl@iflanguage\bbl@tempc{%
1555     \bbl@vforeach{#2}{%
1556       \ifx\bbl@known@attrs\undefined
1557         \in@false
1558       \else
1559         \bbl@xin@{,\bbl@tempc-##1,}{,\bbl@known@attrs,}%
1560       \fi
1561       \ifin@
1562         \bbl@warning{%
1563           You have more than once selected the attribute '##1'\%
1564           for language #1. Reported}%
1565       \else
1566         \bbl@info{Activated '##1' attribute for\\%
1567           '\bbl@tempc'. Reported}%
1568         \bbl@exp{%
1569           \\ \bbl@add@list\\ \bbl@known@attrs{\bbl@tempc-##1}}%
1570         \edef\bbl@tempa{\bbl@tempc-##1}%
1571         \expandafter\bbl@ifknown@ttrib\expandafter{\bbl@tempa}\bbl@attributes
1572         {\csname\bbl@tempc @attr##1\endcsname}%
1573         {\bbl@error{unknown-attribute}{\bbl@tempc}{##1}}}%
1574       \fi}}
1575 \onlypreamble\languageattribute
```

\bbl@declare@ttribute This command adds the new language/attribute combination to the list of known attributes.

```
1576 \def\bbl@declare@ttribute#1#2#3{%
1577   \bbl@xin@{,#2,}{,\BabelModifiers,}%
1578   \ifin@
1579     \AfterBabelLanguage{#1}{\languageattribute{#1}{#2}}%
1580   \fi
```

```

1581 \bbl@add@list\bbl@attributes{#1-#2}%
1582 \expandafter\def\csname#1@attr@#2\endcsname{#3}}

```

\bbl@ifattributeset This internal macro has 4 arguments. It can be used to interpret $\TeX$ code based on whether a certain attribute was set. T

```

1583 \def\bbl@ifattributeset#1#2#3#4{%
1584 \ifx\bbl@known@attrs \@undefined
1585 \in@false
1586 \else
1587 \bbl@xin@{, #1-#2, }{, \bbl@known@attrs,}%
1588 \fi
1589 \ifin@
1590 \bbl@afterelse#3%
1591 \else
1592 \bbl@afterfi#4%
1593 \fi}

```

\bbl@ifknown@ttrib An internal macro to check whether a given language/attribute is known.

```

1594 \def\bbl@ifknown@ttrib#1#2{%
1595 \let\bbl@tempa \@secondoftwo
1596 \bbl@loopx\bbl@tempb{#2}{%
1597 \expandafter\in@\expandafter{\expandafter,\bbl@tempb,}{, #1,}%
1598 \ifin@
1599 \let\bbl@tempa \@firstoftwo
1600 \else
1601 \fi}%
1602 \bbl@tempa}

```

\bbl@clear@ttribs This macro removes all the attribute code from $\LaTeX$'s memory at `\begin{document}` time (if any is present).

```

1603 \def\bbl@clear@ttribs{%
1604 \ifx\bbl@attributes \@undefined\else
1605 \bbl@loopx\bbl@tempa{\bbl@attributes}{%
1606 \expandafter\bbl@clear@ttrib\bbl@tempa.}%
1607 \let\bbl@attributes \@undefined
1608 \fi}
1609 \def\bbl@clear@ttrib#1-#2.{%
1610 \expandafter\let\csname#1@attr@#2\endcsname \@undefined}
1611 \AtBeginDocument{\bbl@clear@ttribs}

```

6.8. Support for saving and redefining macros

To save the meaning of control sequences using `\babel@save`, we use temporary control sequences.

\babel@savecnt

\babel@beginsave The initialization of a new save cycle: reset the counter to zero.

```

1612 \bbl@trace{Macros for saving definitions}
1613 \def\babel@beginsave{\babel@savecnt\z@}

```

Before it's forgotten, allocate the counter and initialize all.

```

1614 \newcount\babel@savecnt
1615 \babel@beginsave

```

\babel@save

\babel@savevariable The macro `\babel@save{csize}` saves the current meaning of the control sequence `(csize)` to `\originalTeX` (which has to be expandable, i.e., you shouldn't let it to `\relax`).

```

1616 \def\babel@save#1{%
1617   \def\bbl@tempa{,{#1,}}% Clumsy, for Plain
1618   \expandafter\bbl@add\expandafter\bbl@tempa\expandafter{%
1619     \expandafter{\expandafter,\bbl@savedextras,}}%
1620   \expandafter\in@\bbl@tempa
1621   \ifin@%else
1622     \bbl@add\bbl@savedextras{,{#1,}}%
1623     \bbl@carg\let\babel@number\babel@savecnt}#1\relax
1624     \toks@{\expandafter{\originalTeX\let#1=}}%
1625     \bbl@exp{%
1626       \def\\originalTeX{\the\toks@<\babel@number\babel@savecnt>\relax}}%
1627     \advance\babel@savecnt@ne
1628   \fi}
1629 \def\babel@savevariable#1{%
1630   \toks@{\expandafter{\originalTeX #1=}}%
1631   \bbl@exp{\def\\originalTeX{\the\toks@the#1\relax}}}
```

\bbl@redefine To redefine a command, we save the old meaning of the macro.

```

1632 \def\bbl@redefine#1{%
1633   \edef\bbl@tempa{\bbl@stripslash#1}%
1634   \expandafter\let\csname org@\bbl@tempa\endcsname#1%
1635   \expandafter\def\csname\bbl@tempa\endcsname}
1636 \@onlypreamble\bbl@redefine
```

\bbl@redefine@long This version of `\babel@redefine` can be used to redefine `\long` commands such as `\ifthenelse`.

```

1637 \def\bbl@redefine@long#1{%
1638   \edef\bbl@tempa{\bbl@stripslash#1}%
1639   \expandafter\let\csname org@\bbl@tempa\endcsname#1%
1640   \long\expandafter\def\csname\bbl@tempa\endcsname}
1641 \@onlypreamble\bbl@redefine@long
```

\bbl@redefineroobust For commands that are redefined, but which *might* be robust we need a slightly more intelligent macro.

```

1642 \def\bbl@redefineroobust#1{%
1643   \edef\bbl@tempa{\bbl@stripslash#1}%
1644   \bbl@ifunset{\bbl@tempa\space}%
1645     {\expandafter\let\csname org@\bbl@tempa\endcsname#1%
1646       \bbl@exp{\def\\#1{\\protect<\bbl@tempa\space>}}}%
1647     {\bbl@exp{\let<org@\bbl@tempa>\<\bbl@tempa\space>}}}%
1648     \namedef{\bbl@tempa\space}}
1649 \@onlypreamble\bbl@redefineroobust
```

6.9. French spacing

\bbl@frenchspacing

\bbl@nonfrenchspacing Some languages need to have `\frenchspacing` in effect.

```

1650 \def\bbl@frenchspacing{%
1651   \ifnum\the\scode\.\=\@m
1652     \let\bbl@nonfrenchspacing\relax
1653   \else
1654     \frenchspacing
1655     \let\bbl@nonfrenchspacing\nonfrenchspacing
1656   \fi}
1657 \let\bbl@nonfrenchspacing\nonfrenchspacing
```

6.10. Hyphens

\babelhyphenation This macro saves hyphenation exceptions.

```
1658 \bbl@trace{Hyphens}
1659 \@onlypreamble\babelhyphenation
1660 \AtEndOfPackage{%
1661   \newcommand\babelhyphenation[2][\@empty]{%
1662     \ifx\bbl@hyphenation@relax
1663       \let\bbl@hyphenation@\@empty
1664     \fi
1665     \ifx\bbl@hyphlist@\@empty\else
1666       \bbl@warning{%
1667         You must not intermingle \string\selectlanguage\space and\\%
1668         \string\babelhyphenation\space or some exceptions will not\\%
1669         be taken into account. Reported}%
1670     \fi
1671     \ifx\@empty#1%
1672       \protected@edef\bbl@hyphenation@{\bbl@hyphenation@\space#2}%
1673     \else
1674       \bbl@vforeach{#1}{%
1675         \def\bbl@tempa{##1}%
1676         \bbl@fixname\bbl@tempa
1677         \bbl@iflanguage\bbl@tempa{%
1678           \bbl@csarg\protected@edef{hyphenation@\bbl@tempa}{%
1679             \bbl@ifunset{bbl@hyphenation@\bbl@tempa}%
1680             {}%
1681             {\csname bbl@hyphenation@\bbl@tempa\endcsname\space}%
1682             #2}}}%
1683       \fi}}
```

\bbl@allowhyphens This macro makes hyphenation possible.

```
1684 \def\bbl@allowhyphens{\ifvmode\else\nobreak\hskip\z@skip\fi}
1685 \def\bbl@t@one{T1}
1686 \def\allowhyphens{\ifx\cf@encoding\bbl@t@one\else\bbl@allowhyphens\fi}
```

\babelhyphen Macros to insert common hyphens.

```
1687 \newcommand\babelnullhyphen{\char\hyphenchar\font}
1688 \def\babelhyphen{\active@prefix\babelhyphen\bbl@hyphen}
1689 \def\bbl@hyphen{%
1690   \@ifstar{\bbl@hyphen@i @}{\bbl@hyphen@i \@empty}}
1691 \def\bbl@hyphen@i#1#2{%
1692   \lowercase{\bbl@ifunset{bbl@hy@#1#2\@empty}}}%
1693   {\csname bbl@#1usehyphen\endcsname{\discretionary{#2}{}{#2}}}%
1694   {\lowercase{\csname bbl@hy@#1#2\@empty\endcsname}}}
```

The following two commands are used to wrap the “hyphen” and set the behavior of the rest of the word.

```
1695 \def\bbl@usehyphen#1{%
1696   \leavevmode
1697   \ifdim\lastskip>\z@\mbox{#1}\else\nobreak#1\fi
1698   \nobreak\hskip\z@skip}
1699 \def\bbl@t@usehyphen#1{%
1700   \leavevmode\ifdim\lastskip>\z@\mbox{#1}\else#1\fi}
```

The following macro inserts the hyphen char.

```
1701 \def\bbl@hyphenchar{%
1702   \ifnum\hyphenchar\font=\m@ne
1703     \babelnullhyphen
1704   \else
1705     \char\hyphenchar\font
1706   \fi}
```

Finally, we define the hyphen “types”. Their names will not change, so you may use them in ldf’s. After a space, the `\mbox` in `\bbl@hy@nbreak` is redundant.

```

1707 \def\bbl@hy@soft{\bbl@usehyphen{\discretionary{\bbl@hyphenchar}{}}{}}
1708 \def\bbl@hy@soft{\bbl@usehyphen{\discretionary{\bbl@hyphenchar}{}}{}}
1709 \def\bbl@hy@hard{\bbl@usehyphen\bbl@hyphenchar}
1710 \def\bbl@hy@hard{\bbl@usehyphen\bbl@hyphenchar}
1711 \def\bbl@hy@nbreak{\bbl@usehyphen{\mbox{\bbl@hyphenchar}}}
1712 \def\bbl@hy@nbreak{\mbox{\bbl@hyphenchar}}
1713 \def\bbl@hy@repeat{%
1714   \bbl@usehyphen{%
1715     \discretionary{\bbl@hyphenchar}{\bbl@hyphenchar}{\bbl@hyphenchar}}}
1716 \def\bbl@hy@repeat{%
1717   \bbl@usehyphen{%
1718     \discretionary{\bbl@hyphenchar}{\bbl@hyphenchar}{\bbl@hyphenchar}}}
1719 \def\bbl@hy@empty{\hskip\z@skip}
1720 \def\bbl@hy@empty{\discretionary{}{}{}}

```

\bbl@disc For some languages the macro `\bbl@disc` is used to ease the insertion of discretionaries for letters that behave ‘abnormally’ at a breakpoint.

```

1721 \def\bbl@disc#1#2{\nobreak\discretionary{#2-}{}{#1}\bbl@allowhyphens}

```

6.11. Multiencoding strings

The aim following commands is to provide a common interface for strings in several encodings.

Tools But first, a tool. It makes global a local variable. This is not the best solution, but it works.

```

1722 \bbl@trace{Multiencoding strings}
1723 \def\bbl@tglobal#1{\global\let#1#1}

```

Main command This is the main command.

```

1724 \@onlypreamble\StartBabelCommands
1725 \def\StartBabelCommands{%
1726   \begingroup
1727   \@tempcnta="7F
1728   \def\bbl@tempa{%
1729     \ifnum\@tempcnta>"FF\else
1730       \catcode\@tempcnta=11
1731       \advance\@tempcnta\@ne
1732       \expandafter\bbl@tempa
1733     \fi}%
1734   \bbl@tempa
1735   <@Macros local to Plain BabelCommands>
1736   \def\bbl@provstring##1##2{%
1737     \providecommand##1{##2}%
1738     \bbl@tglobal##1}%
1739   \global\let\bbl@scafter\@empty
1740   \let\StartBabelCommands\bbl@startcmds
1741   \ifx\BabelLanguages\relax
1742     \let\BabelLanguages\CurrentOption
1743   \fi
1744   \begingroup
1745   \let\bbl@screset\@nnil % local flag - disable 1st stopcommands
1746   \StartBabelCommands}
1747 \def\bbl@startcmds{%
1748   \ifx\bbl@screset\@nnil\else
1749     \bbl@usehooks{stopcommands}{}%
1750   \fi

```

```

1751 \endgroup
1752 \begingroup
1753 \ifstar
1754   {\ifx\bbbl@opt@strings\@nnil
1755     \let\bbbl@opt@strings\BabelStringsDefault
1756     \fi
1757     \bbbl@startcmds@i}%
1758   \bbbl@startcmds@i}
1759 \def\bbbl@startcmds@i#1#2{%
1760   \edef\bbbl@L{\zap@space#1 \@empty}%
1761   \edef\bbbl@G{\zap@space#2 \@empty}%
1762   \bbbl@startcmds@ii}
1763 \let\bbbl@startcommands\StartBabelCommands

```

Parse the encoding info to get the label, input, and font parts.

```

1764 \newcommand\bbbl@startcmds@ii[1][\@empty]{%
1765   \let\SetString\@gobbletwo
1766   \let\bbbl@stringdef\@gobbletwo
1767   \let\AfterBabelCommands\@gobble
1768   \ifx\@empty#1%
1769     \def\bbbl@sc@label{generic}%
1770     \def\bbbl@encstring##1##2{%
1771       \ProvideTextCommandDefault##1{##2}%
1772       \bbbl@toglobal##1%
1773       \expandafter\bbbl@toglobal\csname\string?\string##1\endcsname}%
1774     \let\bbbl@sctest\in@true
1775   \else
1776     \let\bbbl@sc@charset\space % <- zapped below
1777     \let\bbbl@sc@fontenc\space % <- " "
1778     \def\bbbl@tempa##1=##2\@nil{%
1779       \bbbl@csarg\edef{sc\zap@space##1 \@empty}{##2 }}%
1780     \bbbl@vforeach{label=#1}{\bbbl@tempa##1\@nil}%
1781     \def\bbbl@tempa##1 ##2{% space -> comma
1782       ##1%
1783       \ifx\@empty##2\else\ifx,##1\else,\fi\bbbl@afterfi\bbbl@tempa##2\fi}%
1784     \edef\bbbl@sc@fontenc{\expandafter\bbbl@tempa\bbbl@sc@fontenc\@empty}%
1785     \edef\bbbl@sc@label{\expandafter\zap@space\bbbl@sc@label\@empty}%
1786     \edef\bbbl@sc@charset{\expandafter\zap@space\bbbl@sc@charset\@empty}%
1787     \def\bbbl@encstring##1##2{%
1788       \bbbl@foreach\bbbl@sc@fontenc{%
1789         \bbbl@ifunset{T@###1}%
1790         }%
1791         {\ProvideTextCommand##1{####1}{##2}%
1792         \bbbl@toglobal##1%
1793         \expandafter
1794         \bbbl@toglobal\csname####1\string##1\endcsname}}}%
1795     \def\bbbl@sctest{%
1796       \bbbl@xin@{\bbbl@opt@strings,}{,\bbbl@sc@label,\bbbl@sc@fontenc,}}%
1797   \fi
1798   \ifx\bbbl@opt@strings\@nnil % i.e., no strings key -> defaults
1799   \else\ifx\bbbl@opt@strings\relax % i.e., strings=encoded
1800     \let\AfterBabelCommands\bbbl@aftercmds
1801     \let\SetString\bbbl@setstring
1802     \let\bbbl@stringdef\bbbl@encstring
1803   \else % i.e., strings=value
1804     \bbbl@sctest
1805   \fin@
1806     \let\AfterBabelCommands\bbbl@aftercmds
1807     \let\SetString\bbbl@setstring
1808     \let\bbbl@stringdef\bbbl@provstring
1809   \fi\fi\fi
1810   \bbbl@scswitch
1811   \ifx\bbbl@G\@empty%

```

```

1812 \def\SetString##1##2{%
1813 \bbl@error{missing-group}{##1}{}}}%
1814 \fi
1815 \ifx\@empty#1%
1816 \bbl@usehooks{defaultcommands}{}%
1817 \else
1818 \@expandtwoargs
1819 \bbl@usehooks{encodedcommands}{\bbl@sc@charset}{\bbl@sc@fontenc}}%
1820 \fi}

```

There are two versions of \bbl@scswitch.

```

1821 \def\bbl@forlang#1#2{%
1822 \bbl@for#1\bbl@L{%
1823 \bbl@xin@{, #1,}{, \BabelLanguages,}%
1824 \ifin#2\relax\fi}}
1825 \def\bbl@scswitch{%
1826 \bbl@forlang\bbl@tempa{%
1827 \ifx\bbl@G\@empty\else
1828 \ifx\SetString@gobbletwo\else
1829 \edef\bbl@GL{\bbl@G\bbl@tempa}%
1830 \bbl@xin@{\bbl@GL,}{, \bbl@screset,}%
1831 \ifin\else
1832 \global\expandafter\let\csname\bbl@GL\endcsname\@undefined
1833 \xdef\bbl@screset{\bbl@screset, \bbl@GL}%
1834 \fi
1835 \fi
1836 \fi}}
1837 \AtEndOfPackage{%
1838 \def\bbl@forlang#1#2{\bbl@for#1\bbl@L{\bbl@ifunset{date#1}{}{#2}}}%
1839 \let\bbl@scswitch\relax}
1840 \@onlypreamble\EndBabelCommands
1841 \def\EndBabelCommands{%
1842 \bbl@usehooks{stopcommands}{}%
1843 \endgroup
1844 \endgroup
1845 \bbl@scafter}
1846 \let\bbl@endcommands\EndBabelCommands

```

Now we define commands to be used inside \StartBabelCommands.

Strings The following macro is the actual definition of \SetString when it is “active”,

```

1847 \def\bbl@setstring#1#2{% e.g., \prefacename{<string>}
1848 \bbl@forlang\bbl@tempa{%
1849 \edef\bbl@LC{\bbl@tempa\bbl@stripslash#1}%
1850 \bbl@ifunset{\bbl@LC}% e.g., \germanchaptername
1851 {\bbl@exp{%
1852 \global\@bbl@add\<\bbl@G\bbl@tempa>{\@bbl@scset\@#1\<\bbl@LC>}}}%
1853 }%
1854 \def\BabelString{#2}%
1855 \bbl@usehooks{stringprocess}{}%
1856 \expandafter\bbl@stringdef
1857 \csname\bbl@LC\expandafter\endcsname\expandafter{\BabelString}}

```

A little auxiliary command sets the string. Formerly used with casing. Very likely no longer necessary, although it's used in \setlocalecaption.

```

1858 \def\bbl@scset#1#2{\def#1{#2}}

```

Define \SetStringLoop, which is actually set inside \StartBabelCommands.

```

1859 ({*Macros local to Plain BabelCommands}) ▶
1860 \def\SetStringLoop##1##2{%
1861 \def\bbl@templ####1{\expandafter\noexpand\csname##1\endcsname}%
1862 \count@\z@
1863 \bbl@loop\bbl@tempa{##2}{% empty items and spaces are ok

```

```

1864 \advance\count@\@ne
1865 \toks@\expandafter{\bbl@tempa}%
1866 \bbl@exp{%
1867   \\\SetString\bbl@templ{\romannumeral\count@}{\the\toks@}%
1868   \count@=\the\count@\relax}}}%
1869 <</Macros local to Plain BabelCommands>>

```

Delaying code Now the definition of \AfterBabelCommands when it is activated.

```

1870 \def\bbl@aftercmds#1{%
1871   \toks@\expandafter{\bbl@scafter#1}%
1872   \xdef\bbl@scafter{\the\toks@}}

```

Case mapping The command \SetCase is deprecated (at least in $\text{\LaTeX}$). This is the original definition.

```

1873 \def\bbl@patchuclc{%
1874   \global\let\bbl@patchuclc\relax
1875   \g@addto@macro\@uclclist{\reserved@b{\reserved@b\bbl@uclc}}}%
1876   \gdef\bbl@uclc##1{%
1877     \let\bbl@encoded\bbl@encoded@uclc
1878     \bbl@ifunset{\language @bbl@uclc}% and resumes it
1879     {##1}%
1880     {\let\bbl@tempa##1\relax % Used by LANG@bbl@uclc
1881      \csname\language @bbl@uclc\endcsname}%
1882     {\bbl@tolower\@empty}{\bbl@toupper\@empty}}}%
1883   \gdef\bbl@tolower{\csname\language @bbl@lc\endcsname}%
1884   \gdef\bbl@toupper{\csname\language @bbl@uc\endcsname}}
1885 <<*Macros local to Plain BabelCommands>> ▶
1886   \newcommand\SetCase[3][]{%
1887     \bbl@patchuclc
1888     \bbl@forlang\bbl@tempa{%
1889       \bbl@carg\bbl@encstring{\bbl@tempa @bbl@uclc}{\bbl@tempa##1}%
1890       \bbl@carg\bbl@encstring{\bbl@tempa @bbl@uc}{##2}%
1891       \bbl@carg\bbl@encstring{\bbl@tempa @bbl@lc}{##3}}}%
1892 <</Macros local to Plain BabelCommands>>

```

Macros to deal with case mapping for hyphenation.

```

1893 <<*Macros local to Plain BabelCommands>> ▶
1894   \newcommand\SetHyphenMap[1]{%
1895     \bbl@forlang\bbl@tempa{%
1896       \expandafter\bbl@stringdef
1897       \csname\bbl@tempa @bbl@hyphenmap\endcsname{##1}}}%
1898 <</Macros local to Plain BabelCommands>>

```

There are 3 helper macros which do most of the work for you.

```

1899 \newcommand\BabelLower[2]{% one to one.
1900   \ifnum\lccode#1=#2\else
1901     \babel@savevariable{\lccode#1}%
1902     \lccode#1=#2\relax
1903   \fi}
1904 \newcommand\BabelLowerMM[4]{% many-to-many
1905   \@tempcnta=#1\relax
1906   \@tempcntb=#4\relax
1907   \def\bbl@tempa{%
1908     \ifnum\@tempcnta>#2\else
1909       \@expandtwoargs\BabelLower{\the\@tempcnta}{\the\@tempcntb}%
1910       \advance\@tempcnta#3\relax
1911       \advance\@tempcntb#3\relax
1912       \expandafter\bbl@tempa
1913     \fi}%
1914   \bbl@tempa}
1915 \newcommand\BabelLowerM0[4]{% many-to-one

```

```

1916 \@tempcnta=#1\relax
1917 \def\bbl@tempa{%
1918   \ifnum\@tempcnta>#2\else
1919     \@expandtwoargs\BabelLower{\the\@tempcnta}{#4}%
1920     \advance\@tempcnta#3
1921     \expandafter\bbl@tempa
1922   \fi}%
1923 \bbl@tempa}

```

Initial setup to provide a default behavior if hyphenmap is not set.

```

1924 \AtEndOfPackage{%
1925   \ifx\bbl@opt@hyphenmap\undefined
1926     \bbl@xin@{,}\bbl@language@opts}%
1927     \chardef\bbl@opt@hyphenmap\ifin@4\else\@ne\fi
1928   \fi}

```

6.12. Tailor captions

A general tool for resetting the caption names with a unique interface.

```

1929 \newcommand\setlocalecaption{%
1930   \@ifstar\bbl@setcaption@s\bbl@setcaption@x}
1931 \def\bbl@setcaption@x#1#2#3{% language caption-name string
1932   \bbl@trim@def\bbl@tempa{#2}%
1933   \bbl@xin@{.template}\bbl@tempa}%
1934   \ifin@
1935     \bbl@ini@captions@template{#3}{#1}%
1936   \else
1937     \edef\bbl@tempd{%
1938       \expandafter\expandafter\expandafter
1939       \strip@prefix\expandafter\meaning\csname captions#1\endcsname}%
1940     \bbl@xin@
1941       {\expandafter\string\csname #2name\endcsname}%
1942       {\bbl@tempd}%
1943     \ifin@ % Renew caption
1944       \bbl@xin@{\string\bbl@scset}\bbl@tempd}%
1945     \ifin@
1946       \bbl@exp{%
1947         \\bbl@ifsamestring{\bbl@tempa}{\language name}%
1948         {\bbl@scset\<#2name>\<#1#2name>}%
1949         {}}%
1950       \else % Old way converts to new way
1951         \bbl@ifunset{#1#2name}%
1952         {\bbl@exp{%
1953           \\bbl@add\<captions#1>\def\<#2name>\<#1#2name>}}%
1954           \\bbl@ifsamestring{\bbl@tempa}{\language name}%
1955           {\def\<#2name>\<#1#2name>}}%
1956           {}}}%
1957       {}%
1958     \fi
1959   \else
1960     \bbl@xin@{\string\bbl@scset}\bbl@tempd}% New
1961     \ifin@ % New way
1962       \bbl@exp{%
1963         \\bbl@add\<captions#1>\bbl@scset\<#2name>\<#1#2name>}%
1964         \\bbl@ifsamestring{\bbl@tempa}{\language name}%
1965         {\bbl@scset\<#2name>\<#1#2name>}%
1966         {}}%
1967       \else % Old way, but defined in the new way
1968         \bbl@exp{%
1969           \\bbl@add\<captions#1>\def\<#2name>\<#1#2name>}}%
1970           \\bbl@ifsamestring{\bbl@tempa}{\language name}%
1971           {\def\<#2name>\<#1#2name>}}%

```

```

1972         {}}%
1973     \fi%
1974 \fi
1975 \@namedef{#1#2name}{#3}%
1976 \toks@{\expandafter{\bbl@captionslist}%
1977 \bbl@exp{\in@{<#2name>}{\the\toks@}}}%
1978 \ifin@else
1979     \bbl@exp{\bbl@add{\bbl@captionslist{<#2name>}}%
1980     \bbl@tglobal\bbl@captionslist
1981 \fi
1982 \fi}

```

6.13. Making glyphs available

This section makes a number of glyphs available that either do not exist in the OT1 encoding and have to be ‘faked’, or that are not accessible through `Tlenc.def`.

\set@low@box The following macro is used to lower quotes to the same level as the comma.

```

1983 \bbl@trace{Macros related to glyphs}
1984 \def\set@low@box#1{\setbox\tw@\hbox{,}\setbox\z@\hbox{#1}%
1985     \dimen\z@\ht\z@ \advance\dimen\z@ -\ht\tw@%
1986     \setbox\z@\hbox{\lower\dimen\z@ \box\z@}\ht\z@\ht\tw@ \dp\z@\dp\tw@}

```

\save@s f@q The macro `\save@s f@q` is used to save and reset the current space factor.

```

1987 \def\save@s f@q#1{\leavevmode
1988     \begingroup
1989     \edef\@SF{\spacefactor\the\spacefactor}#1\@SF
1990     \endgroup}

```

6.13.1. Quotation marks

\quotedblbase In the OT1 encoding it is not available, therefore we make it available by lowering the normal open quote character to the baseline.

```

1991 \ProvideTextCommand{\quotedblbase}{OT1}{%
1992     \save@s f@q{\set@low@box{\textquotedblright\}}%
1993     \box\z@\kern-.04em\bbl@allowhyphens}}

```

Make sure that when an encoding other than OT1 or T1 is used this glyph can still be typeset.

```

1994 \ProvideTextCommandDefault{\quotedblbase}{%
1995     \UseTextSymbol{OT1}{\quotedblbase}}

```

\quotesinglbase We also need the single quote character at the baseline.

```

1996 \ProvideTextCommand{\quotesinglbase}{OT1}{%
1997     \save@s f@q{\set@low@box{\textquoteright\}}%
1998     \box\z@\kern-.04em\bbl@allowhyphens}}

```

Make sure that when an encoding other than OT1 or T1 is used this glyph can still be typeset.

```

1999 \ProvideTextCommandDefault{\quotesinglbase}{%
2000     \UseTextSymbol{OT1}{\quotesinglbase}}

```

\guillemetleft

\guillemetright The guillemet characters are not available in OT1 encoding.

```
2001 \ProvideTextCommand{\guillemetleft}{OT1}{%
2002   \ifmmode
2003     \ll
2004   \else
2005     \save@sf@q{\nobreak
2006       \raise.2ex\hbox{$\scriptscriptstyle\ll$}\bbl@allowhyphens}%
2007   \fi}
2008 \ProvideTextCommand{\guillemetright}{OT1}{%
2009   \ifmmode
2010     \gg
2011   \else
2012     \save@sf@q{\nobreak
2013       \raise.2ex\hbox{$\scriptscriptstyle\gg$}\bbl@allowhyphens}%
2014   \fi}
2015 \ProvideTextCommand{\guillemotleft}{OT1}{%
2016   \ifmmode
2017     \ll
2018   \else
2019     \save@sf@q{\nobreak
2020       \raise.2ex\hbox{$\scriptscriptstyle\ll$}\bbl@allowhyphens}%
2021   \fi}
2022 \ProvideTextCommand{\guillemotright}{OT1}{%
2023   \ifmmode
2024     \gg
2025   \else
2026     \save@sf@q{\nobreak
2027       \raise.2ex\hbox{$\scriptscriptstyle\gg$}\bbl@allowhyphens}%
2028   \fi}
```

Make sure that when an encoding other than OT1 or T1 is used these glyphs can still be typeset.

```
2029 \ProvideTextCommandDefault{\guillemetleft}{%
2030   \UseTextSymbol{OT1}{\guillemetleft}}
2031 \ProvideTextCommandDefault{\guillemetright}{%
2032   \UseTextSymbol{OT1}{\guillemetright}}
2033 \ProvideTextCommandDefault{\guillemotleft}{%
2034   \UseTextSymbol{OT1}{\guillemotleft}}
2035 \ProvideTextCommandDefault{\guillemotright}{%
2036   \UseTextSymbol{OT1}{\guillemotright}}
```

\guilsinglleft

\guilsinglright The single guillemets are not available in OT1 encoding. They are faked.

```
2037 \ProvideTextCommand{\guilsinglleft}{OT1}{%
2038   \ifmmode
2039     <%
2040   \else
2041     \save@sf@q{\nobreak
2042       \raise.2ex\hbox{$\scriptscriptstyle<$}\bbl@allowhyphens}%
2043   \fi}
2044 \ProvideTextCommand{\guilsinglright}{OT1}{%
2045   \ifmmode
2046     >%
2047   \else
2048     \save@sf@q{\nobreak
2049       \raise.2ex\hbox{$\scriptscriptstyle>$}\bbl@allowhyphens}%
2050   \fi}
```

Make sure that when an encoding other than OT1 or T1 is used these glyphs can still be typeset.

```
2051 \ProvideTextCommandDefault{\guilsinglleft}{%
2052   \UseTextSymbol{OT1}{\guilsinglleft}}
```

```

2053 \ProvideTextCommandDefault{\guilsinglright}{%
2054 \UseTextSymbol{OT1}{\guilsinglright}}

```

6.13.2. Letters

\ij

\IJ The dutch language uses the letter ‘ij’.

```

2055 \DeclareTextCommand{\ij}{OT1}{%
2056 i\kern-0.02em\bbl@allowhyphens j}
2057 \DeclareTextCommand{\IJ}{OT1}{%
2058 I\kern-0.02em\bbl@allowhyphens J}
2059 \DeclareTextCommand{\ij}{T1}{\char188}
2060 \DeclareTextCommand{\IJ}{T1}{\char156}

```

Make sure that when an encoding other than OT1 or T1 is used these glyphs can still be typeset.

```

2061 \ProvideTextCommandDefault{\ij}{%
2062 \UseTextSymbol{OT1}{\ij}}
2063 \ProvideTextCommandDefault{\IJ}{%
2064 \UseTextSymbol{OT1}{\IJ}}

```

\dj

\DJ The croatian language needs the letters \dj and \DJ,

```

2065 \def\crrtic@{\hrule height0.1ex width0.3em}
2066 \def\crttic@{\hrule height0.1ex width0.33em}
2067 \def\ddj@{%
2068 \setbox0\hbox{d}\dimen@=\ht0
2069 \advance\dimen@lex
2070 \dimen@.45\dimen@
2071 \dimen@ii\expandafter\rem@pt\the\fontdimen\@ne\font\dimen@
2072 \advance\dimen@ii.5ex
2073 \leavevmode\rlap{\raise\dimen@\hbox{\kern\dimen@ii\vbox{\crrtic@}}}}
2074 \def\DDJ@{%
2075 \setbox0\hbox{D}\dimen@=.55\ht0
2076 \dimen@ii\expandafter\rem@pt\the\fontdimen\@ne\font\dimen@
2077 \advance\dimen@ii.15ex % correction for the dash position
2078 \advance\dimen@ii-.15\fontdimen7\font % correction for cmtt font
2079 \dimen\thr@@\expandafter\rem@pt\the\fontdimen7\font\dimen@
2080 \leavevmode\rlap{\raise\dimen@\hbox{\kern\dimen@ii\vbox{\crttic@}}}}
2081 %
2082 \DeclareTextCommand{\dj}{OT1}{\ddj@ d}
2083 \DeclareTextCommand{\DJ}{OT1}{\DDJ@ D}

```

Make sure that when an encoding other than OT1 or T1 is used these glyphs can still be typeset.

```

2084 \ProvideTextCommandDefault{\dj}{%
2085 \UseTextSymbol{OT1}{\dj}}
2086 \ProvideTextCommandDefault{\DJ}{%
2087 \UseTextSymbol{OT1}{\DJ}}

```

\SS For the T1 encoding \SS is defined and selects a specific glyph from the font, but for other encodings it is not available.

```

2088 \DeclareTextCommand{\SS}{OT1}{SS}
2089 \ProvideTextCommandDefault{\SS}{\UseTextSymbol{OT1}{\SS}}

```

6.13.3. Shorthands for quotation marks

Shorthands are provided for a number of different quotation marks, which make them usable both outside and inside mathmode.

\glq

\grq The ‘german’ single quotes.

```
2090 \ProvideTextCommandDefault{\glq}{%
2091 \textormath{\quotesinglbase}{\mbox{\quotesinglbase}}}
```

The definition of \grq depends on the fontencoding. With T1 encoding no extra kerning is needed.

```
2092 \ProvideTextCommand{\grq}{T1}{%
2093 \textormath{\kern\z@\textquoteleft}{\mbox{\textquoteleft}}}
2094 \ProvideTextCommand{\grq}{TU}{%
2095 \textormath{\textquoteleft}{\mbox{\textquoteleft}}}
2096 \ProvideTextCommand{\grq}{OT1}{%
2097 \save@sf@q{\kern-.0125em
2098 \textormath{\textquoteleft}{\mbox{\textquoteleft}}}%
2099 \kern.07em\relax}}
2100 \ProvideTextCommandDefault{\grq}{\UseTextSymbol{OT1}\grq}
```

\glqq

\grqq The ‘german’ double quotes.

```
2101 \ProvideTextCommandDefault{\glqq}{%
2102 \textormath{\quotedblbase}{\mbox{\quotedblbase}}}
```

The definition of \grqq depends on the fontencoding. With T1 encoding no extra kerning is needed.

```
2103 \ProvideTextCommand{\grqq}{T1}{%
2104 \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
2105 \ProvideTextCommand{\grqq}{TU}{%
2106 \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}
2107 \ProvideTextCommand{\grqq}{OT1}{%
2108 \save@sf@q{\kern-.07em
2109 \textormath{\textquotedblleft}{\mbox{\textquotedblleft}}}%
2110 \kern.07em\relax}}
2111 \ProvideTextCommandDefault{\grqq}{\UseTextSymbol{OT1}\grqq}
```

\flq

\frq The ‘french’ single guillemets.

```
2112 \ProvideTextCommandDefault{\flq}{%
2113 \textormath{\guilsinglleft}{\mbox{\guilsinglleft}}}
2114 \ProvideTextCommandDefault{\frq}{%
2115 \textormath{\guilsinglright}{\mbox{\guilsinglright}}}
```

\flqq

\frqq The ‘french’ double guillemets.

```
2116 \ProvideTextCommandDefault{\flqq}{%
2117 \textormath{\guillemetleft}{\mbox{\guillemetleft}}}
2118 \ProvideTextCommandDefault{\frqq}{%
2119 \textormath{\guillemetright}{\mbox{\guillemetright}}}
```

6.13.4. Umlauts and tremas

The command \" needs to have a different effect for different languages.

\umlauthigh

\umlauthigh To be able to provide both positions of " we provide two commands to switch the positioning, the default will be \umlauthigh (the normal positioning).

```

2120 \def\umlauthigh{%
2121   \def\bbl@umlauta##1{\leavevmode\bgroup%
2122     \accent\csname\f@encoding dqpos\endcsname
2123     ##1\bbl@allowhyphens\egroup}%
2124   \let\bbl@umlaute\bbl@umlauta}
2125 \def\umlautlow{%
2126   \def\bbl@umlauta{\protect\lower@umlaut}}
2127 \def\umlautelow{%
2128   \def\bbl@umlaute{\protect\lower@umlaut}}
2129 \umlauthigh

```

\lower@umlaut Used to position the " closer to the letter.

```

2130 \expandafter\ifx\csname U@D\endcsname\relax
2131   \csname newdimen\endcsname\U@D
2132 \fi
2133 \def\lower@umlaut#1{%
2134   \leavevmode\bgroup
2135   \U@D lex%
2136   {\setbox\z@\hbox{%
2137     \char\csname\f@encoding dqpos\endcsname}%
2138     \dimen@ -.45ex\advance\dimen@ \ht\z@
2139     \ifdim lex<\dimen@ \fontdimen5\font\dimen@ \fi}%
2140   \accent\csname\f@encoding dqpos\endcsname
2141   \fontdimen5\font\U@D #1%
2142   \egroup}
2143 \AtBeginDocument{%
2144   \DeclareTextCompositeCommand{\}{OT1}{a}{\bbl@umlauta{a}}%
2145   \DeclareTextCompositeCommand{\}{OT1}{e}{\bbl@umlaute{e}}%
2146   \DeclareTextCompositeCommand{\}{OT1}{i}{\bbl@umlaute{i}}%
2147   \DeclareTextCompositeCommand{\}{OT1}{\i}{\bbl@umlaute{i}}%
2148   \DeclareTextCompositeCommand{\}{OT1}{o}{\bbl@umlauta{o}}%
2149   \DeclareTextCompositeCommand{\}{OT1}{u}{\bbl@umlauta{u}}%
2150   \DeclareTextCompositeCommand{\}{OT1}{A}{\bbl@umlauta{A}}%
2151   \DeclareTextCompositeCommand{\}{OT1}{E}{\bbl@umlaute{E}}%
2152   \DeclareTextCompositeCommand{\}{OT1}{I}{\bbl@umlaute{I}}%
2153   \DeclareTextCompositeCommand{\}{OT1}{O}{\bbl@umlauta{O}}%
2154   \DeclareTextCompositeCommand{\}{OT1}{U}{\bbl@umlauta{U}}

```

Finally, make sure the default hyphenrules are defined (even if empty).

```

2155 \ifx\l@english\@undefined
2156   \chardef\l@english\z@
2157 \fi
2158 \ifx\l@unhyphenated\@undefined
2159   \newlanguage\l@unhyphenated
2160 \fi

```

6.14. Layout

Layout is mainly intended to set bidi documents, but there is at least a tool useful in general.

```

2161 \bbl@trace{Bidi layout}
2162 \providecommand\IfBabelLayout[3]{#3}%

```

6.15. Load engine specific macros

Some macros are not defined in all engines, so, after loading the files define them if necessary to raise an error.

```

2163 \bbl@trace{Input engine specific macros}

```

```

2164 \ifcase\bbl@engine
2165   \input txtbabel.def
2166 \or
2167   \input luababel.def
2168 \or
2169   \input xebabel.def
2170 \fi
2171 \providecommand\babelfont{\bbl@error{only-lua-xe}{}}{}{}
2172 \providecommand\babelprehyphenation{\bbl@error{only-lua}{}}{}{}
2173 \ifx\babelposthyphenation\undefined
2174   \let\babelposthyphenation\babelprehyphenation
2175   \let\babelpatterns\babelprehyphenation
2176   \let\babelcharproperty\babelprehyphenation
2177 \fi
2178 (/core)

```

7. Additional support for Plain T_EX

7.1. Not renaming hyphen.tex

As Don Knuth has declared that the filename `hyphen.tex` may only be used to designate *his* version of the american English hyphenation patterns, a new solution has to be found in order to be able to load hyphenation patterns for other languages in a plain-based T_EX-format. When asked he responded:

That file name is “sacred”, and if anybody changes it they will cause severe upward/downward compatibility headaches.

People can have a file `locallyhyphen.tex` or whatever they like, but they mustn’t diddle with `hyphen.tex` (or `plain.tex` except to preload additional fonts).

The files `bplain.tex` and `lplain.tex` can be used as replacement wrappers around `plain.tex` and `lplain.tex` to achieve the desired effect, based on the `babel` package. If you load each of them with `iniTEX`, you will get a file called either `bplain.fmt` or `lplain.fmt`, which you can use as replacements for `plain.fmt` and `lplain.fmt`.

As these files are going to be read as the first thing `iniTEX` sees, we need to set some category codes just to be able to change the definition of `\input`.

```

2179 (*bplain|lplain)
2180 \catcode`\{=1 % left brace is begin-group character
2181 \catcode`\}=2 % right brace is end-group character
2182 \catcode`\#=6 % hash mark is macro parameter character

```

If a file called `hyphen.cfg` can be found, we make sure that *it* will be read instead of the file `hyphen.tex`. We do this by first saving the original meaning of `\input` (and I use a one letter control sequence for that so as not to waste multi-letter control sequence on this in the format).

```

2183 \openin 0 hyphen.cfg
2184 \ifeof0
2185 \else
2186   \let\input

```

Then `\input` is defined to forget about its argument and load `hyphen.cfg` instead. Once that’s done the original meaning of `\input` can be restored and the definition of `\a` can be forgotten.

```

2187 \def\input #1 {%
2188   \let\input\a
2189   \a hyphen.cfg
2190   \let\a\undefined
2191 }
2192 \fi
2193 (/bplain|lplain)

```

Now that we have made sure that `hyphen.cfg` will be loaded at the right moment it is time to load `plain.tex`.

```
2194 {\bplain}\a plain.tex
2195 {\bplain}\a lplain.tex
```

Finally we change the contents of `\fmtname` to indicate that this is *not* the plain format, but a format based on plain with the `babel` package preloaded.

```
2196 {\bplain}\def\fmtname{babel-plain}
2197 {\bplain}\def\fmtname{babel-lplain}
```

When you are using a different format, based on `plain.tex` you can make a copy of `blplain.tex`, rename it and replace `plain.tex` with the name of your format file.

8. Compatibility

A couple of proxy files, for compatibility (`plain.def` and `switch.def`).

```
2198 {\*plain}
2199 \input babel.def
2200 {/plain}
2201 {\*kernel}
2202 \let\bbl@onlyswitch\@empty
2203 \input babel.def
2204 \let\bbl@onlyswitch\@undefined
2205 {/kernel}
```