Contents

List of Tables 1

1 Introduction 2
1.1 About . . . . . . . . . . 2
1.2 License . . . . . . . . . 2
1.3 Feedback . . . . . . . 2
1.4 Acknowledgements . . 2
1.5 Prerequisites . . . . . 3

2 Database Guide 6
2.1 Entry Types . . . . . 7
2.2 Entry Fields . . . . . 13
2.3 Usage Notes . . . . . 31
2.4 Hints and Caveats . . 39

3 User Guide 43
3.1 Package Options . . . 43
3.2 Global Customization . 65
3.3 Standard Styles . . . . 65
3.4 Related Entries . . . . 70
3.5 Sorting Options . . . 72
3.6 Data Annotations . . . 73
3.7 Bibliography Commands 76
3.8 Citation Commands . . 93
3.9 Localization Commands 103
3.10 Formatting Commands 105
3.11 Language notes . . . 117
3.12 Usage Notes . . . . . 119
3.13 Hints and Caveats . . 128
3.14 Using the fallback BibTeX backend . . . 133

4 Author Guide 134
4.1 Overview . . . . . . . 134
4.2 Bibliography Styles . . 137
4.3 Citation Styles . . . . 156
4.4 Data Interface . . . . 158
4.5 Customization . . . . 163
4.6 Auxiliary Commands . 194
4.7 Punctuation . . . . . 209
4.8 Localization Strings . . 214
4.9 Localization Modules . 215
4.10 Formatting Commands 225
4.11 Hints and Caveats . . 234

Appendix 246

A Default Driver Source Mappings 246
A.1 bibtex . . . . . . . 246

B Default Inheritance Setup 247

C Default Sorting Schemes 248
C.1 Alphabetic 1 . . . . . 248
C.2 Alphabetic 2 . . . . . 249
C.3 Chronological . . . . 249

D biblatexxml 250
D.1 Header . . . . . . . . 250
D.2 Body . . . . . . . . . 250

E Option Scope 253

F Revision History 255

List of Tables

1 biblatex compatibility matrix . . . . . . . . . . . . . . 6
2 Supported Languages . . . . . . . . . . . . . . . . . . 26
3 Date Specifications . . . . . . . . . . . . . . . . . . 37
4 EDTF 5.2.2 Unspecified Date Parsing . . . . . . . . 38
5 Enhanced Date Specifications . . . . . . . . . . . . 39
6 Work Uniqueness options . . . . . . . . . . . . . . . . 60
7 Disambiguation counters . . . . . . . . . . . . . . . . 62
8 mcite-like commands . . . . . . . . . . . . . . . . . . 103
9 mcite-like syntax . . . . . . . . . . . . . . . . . . . . 104
10 Date Interface . . . . . . . . . . . . . . . . . . . . . 152
11 Valid transliteration pairs . . . . . . . . . . . . . . 188
12 Field types for \nosort . . . . . . . . . . . . . . . . . 190
13 \mkcomprange setup . . . . . . . . . . . . . . . . . 206
1 Introduction

This document is a systematic reference manual for the `biblatex` package. Look at the sample documents which ship with `biblatex` to get a first impression.\footnote{http://ctan.org/pkg/biblatex/doc/examples} For a quick start guide, browse §§ 1.1, 2.1, 2.2, 2.3, 3.1, 3.3, 3.7, 3.8, 3.12.

1.1 About `biblatex`

This package provides advanced bibliographic facilities for use with LaTeX. The package is a complete reimplementation of the bibliographic facilities provided by LaTeX. The `biblatex` package works with the "backend" (program) `biber`, which is used to process BibTeX format data files and then performs all sorting, label generation (and a great deal more). Formatting of the bibliography is entirely controlled by TeX macros. Good working knowledge in LaTeX should be sufficient to design new bibliography and citation styles. This package also supports subdivided bibliographies, multiple bibliographies within one document, and separate lists of bibliographic information such as abbreviations of various fields. Bibliographies may be subdivided into parts and/or segmented by topics. Just like the bibliography styles, all citation commands may be freely defined. Features such as full Unicode support for bibliography data, customisable sorting, multiple bibliographies with different sorting, customisable labels and dynamic data modification are available. Please refer to § 1.5.5 for information on `biber/biblatex` version compatibility. The package is completely localised and can interface with the `babel` and `polyglossia` packages. Please refer to table 2 for a list of languages currently supported by this package.

1.2 License

Copyright © 2006–2012 Philipp Lehman, 2012–2016 Philip Kime, Audrey Boruvka, Joseph Wright. Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License, version 1.3.\footnote{http://www.latex-project.org/lppl.txt}

1.3 Feedback

Please use the `biblatex` project page on GitHub to report bugs and submit feature requests.\footnote{http://github.com/plk/biblatex} Before making a feature request, please ensure that you have thoroughly studied this manual. If you do not want to report a bug or request a feature but are simply in need of assistance, you might want to consider posting your question on the comp.text.tex newsgroup or TeX-LaTeX Stack Exchange.\footnote{http://tex.stackexchange.com/questions/tagged/biblatex}

1.4 Acknowledgements

The language modules of this package are made possible thanks to the following contributors: Augusto Ritter Stoffel, Mateus Araújo (Brazilian); Sebastià Vilà-Marta (Catalan); Ivo Pletikosić (Croatian); Michal Hoftich (Czech); Jonas Nyrup (Danish); Johannes Wilm (Danish/Norwegian); Alexander van Loon, Pieter Belmans, Hendrik Maryns (Dutch); Hannu Väisänen, Janne Kujanpää (Finnish); Denis Bitouzé (French); Apostolos Syropoulos, Prokopis (Greek); Baldr Kristinsson (Icelandic);
1.5 Prerequisites

This section gives an overview of all resources required by this package and discusses compatibility issues.

1.5.1 Requirements

The resources listed in this section are strictly required for \texttt{biblatex} to function. The package will not work if they are not available.

\textbf{e-TeX} \par The \texttt{biblatex} package requires e-TeX. TeX distributions have been shipping e-TeX binaries for quite some time, the popular distributions use them by default these days. \texttt{biblatex} package checks if it is running under e-TeX. Simply try compiling your documents as you usually do, the chances are that it just works. If you get an error message, try compiling the document with \texttt{elatex} instead of \texttt{latex} or \texttt{pdfelatex} instead of \texttt{pdflatex}, respectively.

\textbf{biber} \par \texttt{biber} is the backend of \texttt{biblatex} used to transfer data from source files to the \LaTeX{} code. \texttt{biber} comes with TeX Live and is also available from SourceForge.\footnote{\url{http://biblatex-biber.sourceforge.net/}}. \texttt{biber} uses the \texttt{btparse} C library for BibTeX format file parsing which aimed to be compatible with BibTeX’s parsing rules but also aimed at correcting some of the common problems. For details, see the manual page for the Perl \texttt{Text::BibTeX} module.\footnote{\url{http://search.cpan.org/~ambs/Text-BibTeX}}.

\textbf{etoolbox} \par This \LaTeX{} package, which is loaded automatically, provides generic programming facilities required by \texttt{biblatex}. It is available from CTAN.\footnote{\url{http://ctan.org/pkg/etoolbox}}

\textbf{kvoptions} \par This \LaTeX{} package, which is also loaded automatically, is used for internal option handling. It is available with the \texttt{oberdiek} package bundle from CTAN.\footnote{\url{http://ctan.org/pkg/kvoptions}}

\textbf{logreq} \par This \LaTeX{} package, which is also loaded automatically, provides a frontend for writing machine-readable messages to an auxiliary log file. It is available from CTAN.\footnote{\url{http://ctan.org/pkg/logreq/}}

\textbf{xstring} \par This \LaTeX{} package, which is also loaded automatically, provides advanced string processing macros It is available from CTAN.\footnote{\url{http://ctan.org/pkg/xstring/}}

Apart from the above resources, \texttt{biblatex} also requires the standard \LaTeX{} packages \texttt{keyval} and \texttt{ifthen} as well as the \texttt{url} package. These package are included in all common TeX distributions and will be loaded automatically.

\footnote{\url{http://ctan.org/pkg/oberdiek}}

\footnote{\url{http://ctan.org/pkg/logreq/}}
1.5.2 Recommended Packages

The packages listed in this section are not required for biblatex to function, but they provide recommended additional functions or enhance existing features. The package loading order does not matter.

**babel/polyglossia**

The `babel` and `polyglossia` packages provide the core architecture for multilingual typesetting. If you are writing in a language other than American English, using one of these packages is strongly recommended. You should load `babel` or `polyglossia` before `biblatex` and then `biblatex` will detect `babel` or `polyglossia` automatically.

**csquotes**

If this package is available, `biblatex` will use its language sensitive quotation facilities to enclose certain titles in quotation marks. If not, `biblatex` uses quotes suitable for American English as a fallback. When writing in a language other than American English, loading `csquotes` is strongly recommended.\footnote{http://ctan.org/pkg/csquotes/}

**xpatch**

The `xpatch` package extends the patching commands of `etoolbox` to `biblatex` bibliography macros, drivers and formatting directives.\footnote{http://ctan.org/pkg/xpatch/}

1.5.3 Compatible Classes and Packages

The `biblatex` package provides dedicated compatibility code for the classes and packages listed in this section.

**hyperref**

The `hyperref` package transforms citations into hyperlinks. See the `hyperref` and `backref` package options in §3.1.2.1 for further details. When using the `hyperref` package, it is preferable to load it after `biblatex`.

**showkeys**

The `showkeys` package prints the internal keys of, among other things, citations in the text and items in the bibliography. The package loading order does not matter.

**memoir**

When using the `memoir` class, the default bibliography headings are adapted such that they blend well with the default layout of this class. See §3.13.2 for further usage hints.

**KOMA-Script**

When using any of the `scrartcl`, `scrbook`, or `scrreprt` classes, the default bibliography headings are adapted such that they blend with the default layout of these classes. See §3.13.1 for further usage hints.

1.5.4 Incompatible Packages

The packages listed in this section are not compatible with `biblatex`. Since it reimplements the bibliographic facilities of LaTeX from the ground up, `biblatex` naturally conflicts with all packages modifying the same facilities. This is not specific to `biblatex`. Some of the packages listed below are also incompatible with each other for the same reason.

**babelbib**

The `babelbib` package provides support for multilingual bibliographies. This is a standard feature of `biblatex`. Use the `langid` field and the package option `autolang` for similar functionality. Note that `biblatex` automatically adjusts to the main document language if `babel` or `polyglossia` is loaded. You only need the above mentioned features if you want to switch languages on a per-entry basis within the bibliography. See §§2.2.3 and 3.1.2.1 for details. Also see §3.9.
**backref**  The `backref` package creates back references in the bibliography. See the package options `hyperref` and `backref` in §3.1.2.1 for comparable functionality.

**bibtopic**  The `bibtopic` package provides support for bibliographies subdivided by topic, type, or other criteria. For bibliographies subdivided by topic, see the category feature in § 3.7.6 and the corresponding filters in § 3.7.2. Alternatively, you might use the `keywords` field in conjunction with the `keyword` and `notkeyword` filters for comparable functionality, see §§ 2.2.3 and 3.7.2 for details. For bibliographies subdivided by type, use the `type` and `nottype` filters. Also see § 3.12.4 for examples.

**bibunits**  The `bibunits` package provides support for multiple partial (e.g., per chapter) bibliographies. See `chapterbib`.

**chapterbib**  The `chapterbib` package provides support for multiple partial bibliographies. Use the `refsection` environment and the `section` filter for comparable functionality. Alternatively, you might also want to use the `refsegment` environment and the `segment` filter. See §§ 3.7.4, 3.7.5, 3.7.2 for details. Also see § 3.12.3 for examples.

**cite**  The `cite` package automatically sorts numeric citations and can compress a list of consecutive numbers to a range. It also makes the punctuation used in citations configurable. For sorted and compressed numeric citations, see the `sortcites` package option in § 3.1.2.1 and the `numeric-comp` citation style in § 3.3.1. For configurable punctuation, see § 3.10.

**citeref**  Another package for creating back references in the bibliography. See `backref`.

**inlinebib**  The `inlinebib` package is designed for traditional citations given in footnotes. For comparable functionality, see the verbose citation styles in § 3.3.1.

**jurabib**  Originally designed for citations in law studies and (mostly German) judicial documents, the `jurabib` package also provides features aimed at users in the humanities. In terms of the features provided, there are some similarities between `jurabib` and `biblatex` but the approaches taken by both packages are quite different. Since both `jurabib` and `biblatex` are full-featured packages, the list of similarities and differences is too long to be discussed here.

**mcite**  The `mcite` package provides support for grouped citations, i.e., multiple items can be cited as a single reference and listed as a single block in the bibliography. The citation groups are defined as the items are cited. This only works with unsorted bibliographies. The `biblatex` package also supports grouped citations, which are called ‘entry sets’ or ‘reference sets’ in this manual. See §§ 3.12.5, 3.7.11, 3.8.10 for details.

**mciteplus**  A significantly enhanced reimplementation of the `mcite` package which supports grouping in sorted bibliographies. See `mcite`.

**multibib**  The `multibib` package provides support for bibliographies subdivided by topic or other criteria. See `bibtopic`.

**natbib**  The `natbib` package supports numeric and author-year citation schemes, incorporating sorting and compression code found in the `cite` package. It also provides additional citation commands and several configuration options. See the numeric and author-year citation styles and their variants in § 3.3.1, the `sortcites` package option in § 3.1.2.1, the citation commands in § 3.8, and the facilities discussed in §§ 3.7.7, 3.7.8, 3.10 for comparable functionality. Also see § 3.8.9.
### Table 1: `biber/biblatex` compatibility matrix

<table>
<thead>
<tr>
<th><code>biber</code> version</th>
<th><code>biblatex</code> version</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6</td>
<td>3.5, 3.6</td>
</tr>
<tr>
<td>2.5</td>
<td>3.4</td>
</tr>
<tr>
<td>2.4</td>
<td>3.3</td>
</tr>
<tr>
<td>2.3</td>
<td>3.2</td>
</tr>
<tr>
<td>2.2</td>
<td>3.1</td>
</tr>
<tr>
<td>2.1</td>
<td>3.0</td>
</tr>
<tr>
<td>2.0</td>
<td>3.0</td>
</tr>
<tr>
<td>1.9</td>
<td>2.9</td>
</tr>
<tr>
<td>1.8</td>
<td>2.8</td>
</tr>
<tr>
<td>1.7</td>
<td>2.7</td>
</tr>
<tr>
<td>1.6</td>
<td>2.6</td>
</tr>
<tr>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td>1.4</td>
<td>2.4</td>
</tr>
<tr>
<td>1.3</td>
<td>2.3</td>
</tr>
<tr>
<td>1.2</td>
<td>2.1, 2.2</td>
</tr>
<tr>
<td>1.1</td>
<td>2.1</td>
</tr>
<tr>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>0.9.9</td>
<td>1.7x</td>
</tr>
<tr>
<td>0.9.8</td>
<td>1.7x</td>
</tr>
<tr>
<td>0.9.7</td>
<td>1.7x</td>
</tr>
<tr>
<td>0.9.6</td>
<td>1.7x</td>
</tr>
<tr>
<td>0.9.5</td>
<td>1.6x</td>
</tr>
<tr>
<td>0.9.4</td>
<td>1.5x</td>
</tr>
<tr>
<td>0.9.3</td>
<td>1.5x</td>
</tr>
<tr>
<td>0.9.2</td>
<td>1.4x</td>
</tr>
<tr>
<td>0.9.1</td>
<td>1.4x</td>
</tr>
<tr>
<td>0.9</td>
<td>1.4x</td>
</tr>
</tbody>
</table>

#### splitbib

The `splitbib` package provides support for bibliographies subdivided by topic. See `bibtopic`.

#### titlesec

The `titlesec` package redefines user-level document division commands such as `\chapter` or `\section`. This approach is not compatible with internal command changes applied by the `biblatex` `refsection` and `refsegment` option settings described in §3.1.2.1.

#### ucs

The `ucs` package provides support for UTF-8 encoded input. Either use `inputenc`'s standard `utf8` module or a Unicode enabled engine such as XeTeX or LuaTeX instead.

#### 1.5.5 Compatibility Matrix for biber

`biber` versions are closely coupled with `biblatex` versions. You need to have the right combination of the two. `biber` will throw a fatal error during processing if it encounters information which comes from a `biblatex` version which is incompatible. Table 1 shows a compatibility matrix for the recent versions.

#### 2 Database Guide

This section describes the default data model defined in the `blx-dm.def` file which is part of `biblatex`. The data model is defined using the macros documented in §4.5.4. It is possible to redefine the data model which both `biblatex` and `biber` use so that datasources can contain new entrytypes and fields (which of course will need style support). The data model specification also allows for constraints to be
defined so that data sources can be validated against the data model (using biber’s `--validate-datamodel` option). Users who want to customise the data model need to look at the `blx-dm.def` file and to read § 4.5.4.

### 2.1 Entry Types

This section gives an overview of the entry types supported by the default biblatex data model along with the fields supported by each type.

#### 2.1.1 Regular Types

The lists below indicate the fields supported by each entry type. Note that the mapping of fields to an entry type is ultimately at the discretion of the bibliography style. The lists below therefore serve two purposes. They indicate the fields supported by the standard styles which ship with this package and they also serve as a model for custom styles. Note that the ‘required’ fields are not strictly required in all cases, see § 2.3.2 for details. The fields marked as ‘optional’ are optional in a technical sense. Bibliographical formatting rules usually require more than just the ‘required’ fields. The default data model defined a few constraints for the format of date fields, ISBNs and some special fields like gender but the constraints are only used if validating against the data model with biber’s `--validate-datamodel` option. Generic fields like abstract and annotation or label and shorthand are not included in the lists below because they are independent of the entry type. The special fields discussed in § 2.2.3, which are also independent of the entry type, are not included in the lists either. See the default data model specification in the file `blx-dm.def` which comes with biblatex for a complete specification.

**article** An article in a journal, magazine, newspaper, or other periodical which forms a self-contained unit with its own title. The title of the periodical is given in the journaltitle field. If the issue has its own title in addition to the main title of the periodical, it goes in the issuetitle field. Note that editor and related fields refer to the journal while translator and related fields refer to the article.

**Required fields:** author, title, journaltitle, year/date

**Optional fields:** translator, annotator, commentator, subtitle, titleaddon, editor, editora, editorb, editorc, journalsubtitle, issuetitle, issuesubtitle, language, origlanguage, series, volume, number, eid, issue, month, pages, version, note, issn, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

**book** A single-volume book with one or more authors where the authors share credit for the work as a whole. This entry type also covers the function of the @inbook type of traditional BibTeX, see § 2.3.1 for details.

**Required fields:** author, title, year/date

**Optional fields:** editor, editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate
mvbook A multi-volume @book. For backwards compatibility, multi-volume books are also supported by the entry type @book. However, it is advisable to make use of the dedicated entry type @mvbook.

Required fields: author, title, year/date

Optional fields: editor, editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, language, origlanguage, edition, volumes, series, number, note, publisher, location, isbn, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

inbook A part of a book which forms a self-contained unit with its own title. Note that the profile of this entry type is different from standard BibTeX, see § 2.3.1.

Required fields: author, title, booktitle, year/date

Optional fields: bookauthor, editor, editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, maintitleaddon, booksubtitle, booktitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

bookinbook This type is similar to @inbook but intended for works originally published as a stand-alone book. A typical example are books reprinted in the collected works of an author.

suppbook Supplemental material in a @book. This type is closely related to the @inbook entry type. While @inbook is primarily intended for a part of a book with its own title (e.g., a single essay in a collection of essays by the same author), this type is provided for elements such as prefaces, introductions, forewords, afterwords, etc. which often have a generic title only. Style guides may require such items to be formatted differently from other @inbook items. The standard styles will treat this entry type as an alias for @inbook.

booklet A book-like work without a formal publisher or sponsoring institution. Use the field howpublished to supply publishing information in free format, if applicable. The field type may be useful as well.

Required fields: author/editor, title, year/date

Optional fields: subtitle, titleaddon, language, howpublished, type, note, location, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

collection A single-volume collection with multiple, self-contained contributions by distinct authors which have their own title. The work as a whole has no overall author but it will usually have an editor.

Required fields: editor, title, year/date
Optional fields: editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, language, orilanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

mvcollection A multi-volume @collection. For backwards compatibility, multi-volume collections are also supported by the entry type @collection. However, it is advisable to make use of the dedicated entry type @mvcollection.

Required fields: editor, title, year/date

Optional fields: editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, language, orilanguage, edition, volumes, series, number, note, publisher, location, isbn, pagetotal, addendum, subtitle, doi, eprint, eprintclass, eprinttype, url, urldate

incollection A contribution to a collection which forms a self-contained unit with a distinct author and title. The author refers to the title, the editor to the booktitle, i.e., the title of the collection.

Required fields: author, title, booktitle, year/date

Optional fields: editor, editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, booksuffix, booktitleaddon, language, orilanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

suppcollection Supplemental material in a @collection. This type is similar to @suppbook but related to the @collection entry type. The standard styles will treat this entry type as an alias for @incollection.

manual Technical or other documentation, not necessarily in printed form. The author or editor is omissible in terms of §2.3.2.

Required fields: author/editor, title, year/date

Optional fields: subtitle, titleaddon, language, edition, type, series, number, version, note, organization, publisher, location, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

misc A fallback type for entries which do not fit into any other category. Use the field howpublished to supply publishing information in free format, if applicable. The field type may be useful as well. author, editor, and year are omissible in terms of §2.3.2.

Required fields: author/editor, title, year/date
Optional fields: subtitle, titleaddon, language, howpublished, type, version, note, organization, location, date, month, year, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

online  An online resource. author, editor, and year are omissible in terms of § 2.3.2. This entry type is intended for sources such as web sites which are intrinsically online resources. Note that all entry types support the url field. For example, when adding an article from an online journal, it may be preferable to use the @article type and its url field.

Required fields: author/editor, title, year/date, url

Optional fields: subtitle, titleaddon, language, version, note, organization, date, month, year, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

patent  A patent or patent request. The number or record token is given in the number field. Use the type field to specify the type and the location field to indicate the scope of the patent, if different from the scope implied by the type. Note that the location field is treated as a key list with this entry type, see § 2.2.1 for details.

Required fields: author, title, number, year/date

Optional fields: holder, subtitle, titleaddon, type, version, location, note, date, month, year, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

periodical  An complete issue of a periodical, such as a special issue of a journal. The title of the periodical is given in the title field. If the issue has its own title in addition to the main title of the periodical, it goes in the issuetitle field. The editor is omissible in terms of § 2.3.2.

Required fields: editor, title, year/date

Optional fields: editora, editorb, editorc, subtitle, issuetitle, issuesubtitle, language, series, volume, number, issue, date, month, year, note, issn, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

suppperiodical  Supplemental material in a @periodical. This type is similar to @suppbook but related to the @periodical entry type. The role of this entry type may be more obvious if you bear in mind that the @article type could also be called @inperiodical. This type may be useful when referring to items such as regular columns, obituaries, letters to the editor, etc. which only have a generic title. Style guides may require such items to be formatted differently from articles in the strict sense of the word. The standard styles will treat this entry type as an alias for @article.

proceedings  A single-volume conference proceedings. This type is very similar to @collection. It supports an optional organization field which holds the sponsoring institution. The editor is omissible in terms of § 2.3.2.

Required fields: title, year/date
mvproceedings A multi-volume @proceedings entry. For backwards compatibility, multi-volume proceedings are also supported by the entry type @proceedings. However, it is advisable to make use of the dedicated entry type @mvproceedings

Required fields: title, year/date

Optional fields: editor, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, eventtitle, eventtitleaddon, eventdate, venue, language, volume, part, volumes, series, number, note, organization, publisher, location, month, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

inproceedings An article in a conference proceedings. This type is similar to @incollection. It supports an optional organization field.

Required fields: author, title, booktitle, year/date

Optional fields: editor, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, booksubtitle, booktitleaddon, eventtitle, eventtitleaddon, eventdate, venue, language, volume, part, volumes, series, number, note, organization, publisher, location, month, isbn, chapter, pages, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

reference A single-volume work of reference such as an encyclopedia or a dictionary. This is a more specific variant of the generic @collection entry type. The standard styles will treat this entry type as an alias for @collection.

mvreference A multi-volume @reference entry. The standard styles will treat this entry type as an alias for @mvcollection. For backwards compatibility, multi-volume references are also supported by the entry type @reference. However, it is advisable to make use of the dedicated entry type @mvreference.

inreference An article in a work of reference. This is a more specific variant of the generic @incollection entry type. The standard styles will treat this entry type as an alias for @incollection.

report A technical report, research report, or white paper published by a university or some other institution. Use the type field to specify the type of report. The sponsoring institution goes in the institution field.

Required fields: author, title, type, institution, year/date

Optional fields: subtitle, titleaddon, language, number, version, note, location, month, isrn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

set An entry set. This entry type is special, see § 3.12.5 for details.

thesis A thesis written for an educational institution to satisfy the requirements for a degree. Use the type field to specify the type of thesis.
unpublished A work with an author and a title which has not been formally published, such as a manuscript or the script of a talk. Use the fields howpublished and note to supply additional information in free format, if applicable.

Required fields: author, title, year/date

Optional fields: subtitle, titleaddon, language, howpublished, note, location, isbn, date, month, year, addendum, pubstate, url, urldate

xdata This entry type is special. @xdata entries hold data which may be inherited by other entries using the xdata field. Entries of this type only serve as data containers; they may not be cited or added to the bibliography. See §3.12.6 for details.

custom[a–f] Custom types for special bibliography styles. Not used by the standard styles.

2.1.2 Type Aliases

The entry types listed in this section are provided for backwards compatibility with traditional BibTeX styles. These aliases are resolved by the backend as the data is processed. Bibliography styles will see the entry type the alias points to, not the alias name. All unknown entry types are generally exported as @misc.

conference A legacy alias for @inproceedings.

electronic An alias for @online.

mastersthesis Similar to @thesis except that the type field is optional and defaults to the localised term 'Master's thesis'. You may still use the type field to override that.

phdthesis Similar to @thesis except that the type field is optional and defaults to the localised term 'PhD thesis'. You may still use the type field to override that.

techreport Similar to @report except that the type field is optional and defaults to the localised term 'technical report'. You may still use the type field to override that.

www An alias for @online, provided for jurabib compatibility.

2.1.3 Unsupported Types

The types in this section are similar to the custom types @custom[a–f], i.e., the standard bibliography styles provide no support for these types. When using the standard styles, they will be treated as @misc entries.

artwork Works of the visual arts such as paintings, sculpture, and installations.

audio Audio recordings, typically on audio CD, DVD, audio cassette, or similar media. See also @music.
This special entry type is not meant to be used in the bib file like other types. It is provided for third-party packages like notes2bib which merge notes into the bibliography. The notes should go into the note field. Be advised that the @bibnote type is not related to the \defbibnote command in any way. \defbibnote is for adding comments at the beginning or the end of the bibliography, whereas the @bibnote type is meant for packages which render endnotes as bibliography entries.

Commentaries which have a status different from regular books, such as legal commentaries.

Images, pictures, photographs, and similar media.

Court decisions, court recordings, and similar things.

Laws, bills, legislative proposals, and similar things.

Legal documents such as treaties.

Personal correspondence such as letters, emails, memoranda, etc.

Motion pictures. See also @video.

Musical recordings. This is a more specific variant of @audio.

Musical and theatrical performances as well as other works of the performing arts. This type refers to the event as opposed to a recording, a score, or a printed play.

Reviews of some other work. This is a more specific variant of the @article type. The standard styles will treat this entry type as an alias for @article.

Computer software.

National and international standards issued by a standards body such as the International Organization for Standardization.

Audiovisual recordings, typically on DVD, VHS cassette, or similar media. See also @movie.

\subsection{Entry Fields}

This section gives an overview of the fields supported by the biblatex default data model. See §\ref{section:datatypes} for an introduction to the data types used by the data model specification and §§\ref{section:fields} and \ref{section:fieldlist} for the actual field listings.

\subsubsection{Data Types}

In datasources such as a bib file, all bibliographic data is specified in fields. Some of those fields, for example author and editor, may contain a list of items. This list structure is implemented by the BibTeX file format via the keyword ‘and’, which is used to separate the individual items in the list. The biblatex package implements three distinct data types to handle bibliographic data: name lists, literal lists, and fields. There are also several list and field subtypes and a content type which can be used to semantically distinguish fields which are otherwise not distinguishable on the basis of only their datatype (see §\ref{section:content}). This section gives an overview of the data types supported by this package. See §§\ref{section:fields} and \ref{section:fieldlist} for information about the mapping of the BibTeX file format fields to biblatex’s data types.
**Name lists** are parsed and split up into the individual items at the and delimiter. Each item in the list is then dissected into the name part components: by default the given name, the name prefix (von, van, of, da, de, della, ...), the family name, and the name suffix (junior, senior, ...). The valid name parts can be customised by changing the datamodel definition described in § 4.2.3. Name lists may be truncated in the bib file with the keyword 'and others'. Typical examples of name lists are author and editor.

Name list fields automatically have an \ifuse* test created as per the name lists in the default data model (see § 4.6.2). They are also automatically have a ifuse* option created which controls labelling and sorting behaviour with the name (see § 3.1.3.1). biber supports a customisable set of name parts but currently this is defined to be the same set of parts as supported by traditional BibTeX:

- Family name (also known as 'last' part)
- Given name (also known as ‘first’ part)
- Name prefix (also known as ‘von’ part)
- Name suffix (also known as ‘Jr’ part)

The supported list of name parts is defined as a constant list in the default data model using the \DeclareDatamodelConstant command (see 4.5.4). However, it is not enough to simply add to this list in order to add support for another name part as name parts typically have to be hard coded into bibliography drivers and the backend processing. See the example file 93-nameparts.tex for details on how to define and use custom name parts. Also see \DeclareUniquenameTemplate in § 4.11.4 for information on how to customise name disambiguation using custom name parts.

**Literal lists** are parsed and split up into the individual items at the and delimiter but not dissected further. Literal lists may be truncated in the bib file with the keyword 'and others'. There are two subtypes:

**Literal lists** in the strict sense are handled as described above. The individual items are simply printed as is. Typical examples of such literal lists are publisher and location.

**Key lists** are a variant of literal lists which may hold printable data or localisation keys. For each item in the list, a test is performed to determine whether it is a known localisation key (the localisation keys defined by default are listed in § 4.9.2). If so, the localised string is printed. If not, the item is printed as is. A typical example of a key list is language.

**Fields** are usually printed as a whole. There are several subtypes:

**Literal fields** are printed as is. Typical examples of literal fields are title and note.

**Range fields** consist of one or more ranges where all dashes are normalized and replaced by the command \bibrangedash. A range is something optionally followed by one or more dashes optionally followed by some non-dash (e.g. 5--7). Any number of consecutive dashes will only yield a single range dash. A typical example of a range field is the pages field. See also the \bibrangessep command which can be used to customise the separator between multiple ranges. Range fields will be
skipped and will generate a warning if they do not consist of one or more
ranges. You can normalise messy range fields before they are parsed
using `\DeclareSourcemap` (see §4.5.3).

**Integer fields** hold integers which may be converted to ordinals or strings
as they are printed. A typical example is the `extrayear` or `volume`
field. Such fields are sorted as integers. `biber` makes a (quite serious)
effort to map non-arabic representations (roman numerals for example)
to integers for sorting purposes.

**Datepart fields** hold unformatted integers which may be converted
to ordinals or strings as they are printed. A typical example
is the `month` field. For every field `X` of datatype `date`
in the datamodel, datepart fields are automatically created with the fol-
lowing names: `<datatype>year`, `<datatype>endyear`,
`<datatype>month`, `<datatype>endmonth`,
`<datatype>day`, `<datatype>endday`, `<datatype>hour`,
`<datatype>endhour`, `<datatype>minute`,
`<datatype>endminute`, `<datatype>second`,
`<datatype>endsecond`, `<datatype>timezone`,
`<datatype>timezone`. `<datatype>` is the string preced-
ing `date` for any datamodel field of datatype `date`. For example, in
the default datamodel, `event`, `orig`, `url` and the empty string `''`
for the date field `date`.

**Date fields** hold a date specification in
`yyyy-mm-ddThh:nn[+-][hh[:nn]Z]` format or a date range in
`yyyy-mm-ddThh:nn[+-][hh[:nn]Z]/yyyy-mm-ddThh:nn[+-][hh[:nn]Z]`
format and other formats permitted by EDTF level 1, see §2.3.8. Date
fields are special in that the date is parsed and split up into its
datepart type components. The datepart components (see above) are
automatically defined and recognised when a field of datatype `date`
is defined in the datamodel. A typical example is the `date` field.

**Verbatim fields** are processed in verbatim mode and may contain special
characters. Typical examples of verbatim fields are `file` and `doi`.

**URI fields** are processed in verbatim mode and may contain special charac-
ters. They are also URL-escaped if they don’t look like they already are.
The typical example of a uri field is `url`.

**Separated value fields** A separated list of literal values. Examples are the
`keywords` and `options` fields. The separator can be configured to be
any Perl regular expression via the `xsvsep` option which defaults to the
usual BibTeX comma surrounded by optional whitespace.

**Pattern fields** A literal field which must match a particular pattern. An
example is the `gender` field from §2.2.3.

**Key fields** May hold printable data or localisation keys. A test is performed
to determine whether the value of the field is a known localisation key
(the localisation keys defined by default are listed in §4.9.2). If so, the
localised string is printed. If not, the value is printed as is. A typical
example is the `type` field.

**Code fields** Holds TeX code.
2.2.2 Data Fields

The fields listed in this section are the regular ones holding printable data in the default data model. The name on the left is the default data model name of the field as used by \texttt{biblatex} and its backend. The \texttt{biblatex} data type is given to the right of the name. See §2.2.1 for explanation of the various data types.

Some fields are marked as ‘label’ fields which means that they are often used as abbreviation labels when printing bibliography lists in the sense of section §3.7.3. \texttt{biblatex} automatically creates supporting macros for such fields. See §3.7.3.

\texttt{abstract} field (literal)

This field is intended for recording abstracts in a \texttt{bib} file, to be printed by a special bibliography style. It is not used by all standard bibliography styles.

\texttt{addendum} field (literal)

Miscellaneous bibliographic data to be printed at the end of the entry. This is similar to the \texttt{note} field except that it is printed at the end of the bibliography entry.

\texttt{afterword} list (name)

The author(s) of an afterword to the work. If the author of the afterword is identical to the \texttt{editor} and/or \texttt{translator}, the standard styles will automatically concatenate these fields in the bibliography. See also \texttt{introduction} and \texttt{foreword}.

\texttt{annotation} field (literal)

This field may be useful when implementing a style for annotated bibliographies. It is not used by all standard bibliography styles. Note that this field is completely unrelated to \texttt{annotator}. The \texttt{annotator} is the author of annotations which are part of the work cited.

\texttt{annotator} list (name)

The author(s) of annotations to the work. If the annotator is identical to the \texttt{editor} and/or \texttt{translator}, the standard styles will automatically concatenate these fields in the bibliography. See also \texttt{commentator}.

\texttt{author} list (name)

The author(s) of the \texttt{title}.

\texttt{authortype} field (key)

The type of author. This field will affect the string (if any) used to introduce the author. Not used by the standard bibliography styles.

\texttt{bookauthor} list (name)

The author(s) of the \texttt{booktitle}.

\texttt{bookpagination} field (key)

If the work is published as part of another one, this is the pagination scheme of the enclosing work, i.e., \texttt{bookpagination} relates to \texttt{pagination} like \texttt{booktitle} to \texttt{title}. The value of this field will affect the formatting of the \texttt{pages} and \texttt{pagetotal} fields. The key should be given in the singular form. Possible keys are \texttt{page}, \texttt{column}, \texttt{line}, \texttt{verse}, \texttt{section}, and \texttt{paragraph}. See also \texttt{pagination} as well as §\texttt{2.3.10}.
booksubtitle field (literal)

The subtitle related to the booktitle. If the subtitle field refers to a work which is part of a larger publication, a possible subtitle of the main work is given in this field. See also subtitle.

booktitle field (literal)

If the title field indicates the title of a work which is part of a larger publication, the title of the main work is given in this field. See also title.

booktitleaddon field (literal)

An annex to the booktitle, to be printed in a different font.

chapter field (literal)

A chapter or section or any other unit of a work.

commentator list (name)

The author(s) of a commentary to the work. Note that this field is intended for commented editions which have a commentator in addition to the author. If the work is a stand-alone commentary, the commentator should be given in the author field. If the commentator is identical to the editor and/or translator, the standard styles will automatically concatenate these fields in the bibliography. See also annotator.

date field (date)

The publication date. See also month and year as well as § 2.3.8.

doi field (verbatim)

The Digital Object Identifier of the work.

date field (integer or literal)

The edition of a printed publication. This must be an integer, not an ordinal. Don’t say edition={First} or edition={1st} but edition={1}. The bibliography style converts this to a language dependent ordinal. It is also possible to give the edition as a literal string, for example “Third, revised and expanded edition”.

editor list (name)

The editor(s) of the title, booktitle, or maintitle, depending on the entry type. Use the editortype field to specify the role if it is different from 'editor'. See § 2.3.6 for further hints.

editora list (name)

A secondary editor performing a different editorial role, such as compiling, redacting, etc. Use the editoratype field to specify the role. See § 2.3.6 for further hints.

editorb list (name)

Another secondary editor performing a different role. Use the editorbtype field to specify the role. See § 2.3.6 for further hints.
editorc list (name)

Another secondary editor performing a different role. Use the editorctype field to specify the role. See § 2.3.6 for further hints.

deditortype field (key)

The type of editorial role performed by the editor. Roles supported by default are editor, compiler, founder, continuator, redactor, reviser, collaborator. The role ‘editor’ is the default. In this case, the field is omissible. See § 2.3.6 for further hints.

deditoratype field (key)

Similar to deditortype but referring to the editora field. See § 2.3.6 for further hints.

deditorbtype field (key)

Similar to deditortype but referring to the editorb field. See § 2.3.6 for further hints.

deditorctype field (key)

Similar to deditortype but referring to the editorc field. See § 2.3.6 for further hints.

deid field (literal)

The electronic identifier of an @article.

dentriesubtype field (literal)

This field, which is not used by the standard styles, may be used to specify a subtype of an entry type. This may be useful for bibliography styles which support a finer-grained set of entry types.

deprint field (verbatim)

The electronic identifier of an online publication. This is roughly comparable to a doi but specific to a certain archive, repository, service, or system. See § 3.12.7 for details. Also see deprinttype and deprintclass.

deprintclass field (literal)

Additional information related to the resource indicated by the deprinttype field. This could be a section of an archive, a path indicating a service, a classification of some sort, etc. See § 3.12.7 for details. Also see deprint and deprinttype.

deprinttype field (literal)

The type of eprint identifier, e.g., the name of the archive, repository, service, or system the eprint field refers to. See § 3.12.7 for details. Also see eprint and eprintclass.

deventdate field (date)

The date of a conference, a symposium, or some other event in @proceedings and @inproceedings entries. This field may also be useful for the custom types listed in § 2.1.3. See also eventtitle and venue as well as § 2.3.8.
eventtitle field (literal)

The title of a conference, a symposium, or some other event in @proceedings and @inproceedings entries. This field may also be useful for the custom types listed in §2.1.3. Note that this field holds the plain title of the event. Things like “Proceedings of the Fifth XYZ Conference” go into the titleaddon or booktitleaddon field, respectively. See also eventdate and venue.

eventtitleaddon field (literal)

An annex to the eventtitle field. Can be used for known event acronyms, for example.

file field (verbatim)

A local link to a PDF or other version of the work. Not used by the standard bibliography styles.

foreword list (name)

The author(s) of a foreword to the work. If the author of the foreword is identical to the editor and/or translator, the standard styles will automatically concatenate these fields in the bibliography. See also introduction and afterword.

holder list (name)

The holder(s) of a @patent, if different from the author. Not that corporate holders need to be wrapped in an additional set of braces, see §2.3.3 for details. This list may also be useful for the custom types listed in §2.1.3.

howpublished field (literal)

A publication notice for unusual publications which do not fit into any of the common categories.

indextitle field (literal)

A title to use for indexing instead of the regular title field. This field may be useful if you have an entry with a title like “An Introduction to …” and want that indexed as “Introduction to ..., An”. Style authors should note that biblatex automatically copies the value of the title field to indextitle if the latter field is undefined.

institution list (literal)

The name of a university or some other institution, depending on the entry type. Traditional BibTeX uses the field name school for theses, which is supported as an alias. See also §§2.2.5 and 2.3.4.

introduction list (name)

The author(s) of an introduction to the work. If the author of the introduction is identical to the editor and/or translator, the standard styles will automatically concatenate these fields in the bibliography. See also foreword and afterword.

isan field (literal)

The International Standard Audiovisual Number of an audiovisual work. Not used by the standard bibliography styles.
isbn field (literal)
The International Standard Book Number of a book.

ismn field (literal)
The International Standard Music Number for printed music such as musical scores.
Not used by the standard bibliography styles.

isrn field (literal)

issn field (literal)
The International Standard Serial Number of a periodical.

issue field (literal)
The issue of a journal. This field is intended for journals whose individual issues are identified by a designation such as 'Spring' or 'Summer' rather than the month or a number. Since the placement of issue is similar to month and number, this field may also be useful with double issues and other special cases. See also month, number, and §2.3.9.

issuesubtitle field (literal)
The subtitle of a specific issue of a journal or other periodical.

issuetitle field (literal)
The title of a specific issue of a journal or other periodical.

iswc field (literal)
The International Standard Work Code of a musical work. Not used by the standard bibliography styles.

journalsubtitle field (literal)
The subtitle of a journal, a newspaper, or some other periodical.

journaltitle field (literal)
The name of a journal, a newspaper, or some other periodical.

label field (literal)
A designation to be used by the citation style as a substitute for the regular label if any data required to generate the regular label is missing. For example, when an author-year citation style is generating a citation for an entry which is missing the author or the year, it may fall back to label. See §2.3.2 for details. Note that, in contrast to shorthand, label is only used as a fallback. See also shorthand.

language list (key)
The language(s) of the work. Languages may be specified literally or as localisation keys. If localisation keys are used, the prefix lang is omissible. See also origlanguage and compare langid in §2.2.3.
library field (literal)
This field may be useful to record information such as a library name and a call number. This may be printed by a special bibliography style if desired. Not used by the standard bibliography styles.

location list (literal)
The place(s) of publication, i.e., the location of the publisher or institution, depending on the entry type. Traditional BibTeX uses the field name address, which is supported as an alias. See also §§2.2.5 and 2.3.4. With @patent entries, this list indicates the scope of a patent. This list may also be useful for the custom types listed in §2.1.3.

mainsubtitle field (literal)
The subtitle related to the maintitle. See also subtitle.

maintitle field (literal)
The main title of a multi-volume book, such as Collected Works. If the title or booktitle field indicates the title of a single volume which is part of multi-volume book, the title of the complete work is given in this field.

maintitleaddon field (literal)
An annex to the maintitle, to be printed in a different font.

month field (datepart)
The publication month. This must be an integer, not an ordinal or a string. Don’t say month={January} but month={1}. The bibliography style converts this to a language dependent string or ordinal where required. See also date as well as §§2.3.9 and 2.3.8.

nameaddon field (literal)
An addon to be printed immediately after the author name in the bibliography. Not used by the standard bibliography styles. This field may be useful to add an alias or pen name (or give the real name if the pseudonym is commonly used to refer to that author).

note field (literal)
Miscellaneous bibliographic data which does not fit into any other field. The note field may be used to record bibliographic data in a free format. Publication facts such as “Reprint of the edition London 1831” are typical candidates for the note field. See also addendum.

number field (integer)
The number of a journal or the volume/number of a book in a series. See also issue as well as §§2.3.7 and 2.3.9. With @patent entries, this is the number or record token of a patent or patent request. It is expected to be an integer, not necessarily in arabic numerals since biber will automatically from roman numerals or arabic letter to integers internally for sorting purposes.
organization list (literal)

The organization(s) that published a @manual or an @online resource, or sponsored a conference. See also § 2.3.4.

origdate field (date)

If the work is a translation, a reprint, or something similar, the publication date of the original edition. Not used by the standard bibliography styles. See also date.

origlanguage field (key)

If the work is a translation, the language of the original work. See also language.

origlocation list (literal)

If the work is a translation, a reprint, or something similar, the location of the original edition. Not used by the standard bibliography styles. See also location and § 2.3.4.

origpublisher list (literal)

If the work is a translation, a reprint, or something similar, the publisher of the original edition. Not used by the standard bibliography styles. See also publisher and § 2.3.4.

origtitle field (literal)

If the work is a translation, the title of the original work. Not used by the standard bibliography styles. See also title.

pages field (range)

One or more page numbers or page ranges. If the work is published as part of another one, such as an article in a journal or a collection, this field holds the relevant page range in that other work. It may also be used to limit the reference to a specific part of a work (a chapter in a book, for example).

pagetotal field (literal)

The total number of pages of the work.

pagination field (key)

The pagination of the work. The value of this field will affect the formatting the ⟨postnote⟩ argument to a citation command. The key should be given in the singular form. Possible keys are page, column, line, verse, section, and paragraph. See also bookpagination as well as §§ 2.3.10 and 3.13.3.

part field (literal)

The number of a partial volume. This field applies to books only, not to journals. It may be used when a logical volume consists of two or more physical ones. In this case the number of the logical volume goes in the volume field and the number of the part of that volume in the part field. See also volume.

publisher list (literal)

The name(s) of the publisher(s). See also § 2.3.4.
pubstate field (key)

The publication state of the work, e.g., ‘in press’. See §4.9.2.11 for known publication states.

reprinttitle field (literal)

The title of a reprint of the work. Not used by the standard styles.

series field (literal)

The name of a publication series, such as “Studies in ...”, or the number of a journal series. Books in a publication series are usually numbered. The number or volume of a book in a series is given in the number field. Note that the @article entry type makes use of the series field as well, but handles it in a special way. See §2.3.7 for details.

shortauthor list (name)

The author(s) of the work, given in an abbreviated form. This field is mainly intended for abbreviated forms of corporate authors, see §2.3.3 for details.

shorteditor list (name)

The editor(s) of the work, given in an abbreviated form. This field is mainly intended for abbreviated forms of corporate editors, see §2.3.3 for details.

shorthand field (literal)

A special designation to be used by the citation style instead of the usual label. If defined, it overrides the default label. See also label.

shorthandintro field (literal)

The verbose citation styles which comes with this package use a phrase like “henceforth cited as [shorthand]” to introduce shorthands on the first citation. If the shorthandintro field is defined, it overrides the standard phrase. Note that the alternative phrase must include the shorthand.

shortjournal field (literal)

A short version or an acronym of the journaltitle. Not used by the standard bibliography styles.

shortseries field (literal)

A short version or an acronym of the series field. Not used by the standard bibliography styles.

shorttitle field (literal)

The title in an abridged form. This field is usually not included in the bibliography. It is intended for citations in author-title format. If present, the author-title citation styles use this field instead of title.

subtitle field (literal)

The subtitle of the work.
**title** field (literal)

The title of the work.

**titleaddon** field (literal)

An annex to the title, to be printed in a different font.

**translator** list (name)

The translator(s) of the title or booktitle, depending on the entry type. If the translator is identical to the editor, the standard styles will automatically concatenate these fields in the bibliography.

**type** field (key)

The type of a manual, patent, report, or thesis. This field may also be useful for the custom types listed in § 2.1.3.

**url** field (uri)

The url of an online publication. If it is not URL-escaped (no '%' chars) it will be URI-escaped according to RFC 3987, that is, even Unicode chars will be correctly escaped.

**urldate** field (date)

The access date of the address specified in the url field. See also § 2.3.8.

**venue** field (literal)

The location of a conference, a symposium, or some other event in @proceedings and @inproceedings entries. This field may also be useful for the custom types listed in § 2.1.3. Note that the location list holds the place of publication. It therefore corresponds to the publisher and institution lists. The location of the event is given in the venue field. See also eventdate and eventtitle.

**version** field (literal)

The revision number of a piece of software, a manual, etc.

**volume** field (integer)

The volume of a multi-volume book or a periodical. It is expected to be an integer, not necessarily in arabic numerals since biber will automatically from roman numerals or arabic letter to integers internally for sorting purposes. See also part.

**volumes** field (integer)

The total number of volumes of a multi-volume work. Depending on the entry type, this field refers to title or maintitle. It is expected to be an integer, not necessarily in arabic numerals since biber will automatically from roman numerals or arabic letter to integers internally for sorting purposes.

**year** field (literal)

The year of publication. It is better to use the date field as this is compatible with plain years too. See § 2.3.8.
2.2.3 Special Fields

The fields listed in this section do not hold printable data but serve a different purpose. They apply to all entry types in the default data model.

**crossref** field (entry key)

This field holds an entry key for the cross-referencing feature. Child entries with a `crossref` field inherit data from the parent entry specified in the `crossref` field. If the number of child entries referencing a specific parent entry hits a certain threshold, the parent entry is automatically added to the bibliography even if it has not been cited explicitly. The threshold is settable with the `mincrossrefs` package option from §3.1.2.1. Style authors should note that whether or not the `crossref` fields of the child entries are defined on the `biblatex` level depends on the availability of the parent entry. If the parent entry is available, the `crossref` fields of the child entries will be defined. If not, the child entries still inherit the data from the parent entry but their `crossref` fields will be undefined. Whether the parent entry is added to the bibliography implicitly because of the threshold or explicitly because it has been cited does not matter. See also the `xref` field in this section as well as §2.4.1.

**entryset** field (separated values)

This field is specific to entry sets. See §3.12.5 for details. This field is consumed by the backend processing and does not appear in the `.bbl`.

**execute** field (code)

A special field which holds arbitrary TeX code to be executed whenever the data of the respective entry is accessed. This may be useful to handle special cases. Conceptually, this field is comparable to the hooks `\AtEveryBibitem`, `\AtEveryLositem`, and `\AtEveryCitekey` from §4.10.6, except that it is definable on a per-entry basis in the `bib` file. Any code in this field is executed automatically immediately after these hooks.

**gender** field (Pattern matching one of: sf, sm, sn, pf, pm, pn, pp)

The gender of the author or the gender of the editor, if there is no author. The following identifiers are supported: sf (feminine singular, a single female name), sm (masculine singular, a single male name), sn (neuter singular, a single neuter name), pf (feminine plural, a list of female names), pm (masculine plural, a list of male names), pn (neuter plural, a list of neuter names), pp (plural, a mixed gender list of names). This information is only required by special bibliography and citation styles and only in certain languages. For example, a citation style may replace recurrent author names with a term such as ‘idem’. If the Latin word is used, as is custom in English and French, there is no need to specify the gender. In German publications, however, such key terms are usually given in German and in this case they are gender-sensitive.

**langid** field (identifier)

The language id of the bibliography entry. The alias `hyphenation` is provided for backwards compatibility. The identifier must be a language name known to the `babel/polyglossia` packages. This information may be used to switch hyphenation patterns and localise strings in the bibliography. Note that the language names
<table>
<thead>
<tr>
<th>Language</th>
<th>Region/Dialect</th>
<th>Identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalan</td>
<td>Spain, France, Andorra, Italy</td>
<td>catalan</td>
</tr>
<tr>
<td>Croatian</td>
<td>Croatia, Bosnia and Herzegovina, Serbia</td>
<td>croatian</td>
</tr>
<tr>
<td>Czech</td>
<td>Czech Republic</td>
<td>czech</td>
</tr>
<tr>
<td>Danish</td>
<td>Denmark</td>
<td>danish</td>
</tr>
<tr>
<td>Dutch</td>
<td>Netherlands</td>
<td>dutch</td>
</tr>
<tr>
<td>English</td>
<td>USA</td>
<td>american, USenglish, english</td>
</tr>
<tr>
<td></td>
<td>United Kingdom</td>
<td>british, UKenglish</td>
</tr>
<tr>
<td></td>
<td>Canada</td>
<td>canadian</td>
</tr>
<tr>
<td></td>
<td>Australia</td>
<td>australian</td>
</tr>
<tr>
<td></td>
<td>New Zealand</td>
<td>newzealand</td>
</tr>
<tr>
<td>Estonian</td>
<td>Estonia</td>
<td>estonian</td>
</tr>
<tr>
<td>Finnish</td>
<td>Finland</td>
<td>finnish</td>
</tr>
<tr>
<td>French</td>
<td>France, Canada</td>
<td>french</td>
</tr>
<tr>
<td>German</td>
<td>Germany</td>
<td>german</td>
</tr>
<tr>
<td></td>
<td>Austria</td>
<td>austrian</td>
</tr>
<tr>
<td></td>
<td>Switzerland</td>
<td>swissgerman</td>
</tr>
<tr>
<td>German (new)</td>
<td>Germany</td>
<td>ngerman</td>
</tr>
<tr>
<td></td>
<td>Austria</td>
<td>nau austrian</td>
</tr>
<tr>
<td></td>
<td>Switzerland</td>
<td>nswissgerman</td>
</tr>
<tr>
<td>Greek</td>
<td>Greece</td>
<td>greek</td>
</tr>
<tr>
<td>Italian</td>
<td>Italy</td>
<td>italian</td>
</tr>
<tr>
<td>Norwegian</td>
<td>Norway</td>
<td>norwegian, norsk, nynorsk</td>
</tr>
<tr>
<td>Polish</td>
<td>Poland</td>
<td>polish</td>
</tr>
<tr>
<td>Portuguese</td>
<td>Brazil</td>
<td>brazil</td>
</tr>
<tr>
<td></td>
<td>Portugal</td>
<td>portuguese, portuges</td>
</tr>
<tr>
<td>Russian</td>
<td>Russia</td>
<td>russian</td>
</tr>
<tr>
<td>Slovak</td>
<td>Slovakia</td>
<td>slovak</td>
</tr>
<tr>
<td>Slovene</td>
<td>Slovenia</td>
<td>slovene</td>
</tr>
<tr>
<td>Spanish</td>
<td>Spain</td>
<td>spanish</td>
</tr>
<tr>
<td>Swedish</td>
<td>Sweden</td>
<td>swedish</td>
</tr>
</tbody>
</table>

Table 2: Supported Languages
are case sensitive. The languages currently supported by this package are given in table 2. Note that babel treats the identifier english as an alias for british or american, depending on the babel version. The biblatex package always treats it as an alias for american. It is preferable to use the language identifiers american and british (babel) or a language specific option to specify a language variant (polyglossia, using the langidopts field) to avoid any possible confusion. Compare language in §2.2.2.

**langidopts** field (literal)

For polyglossia users, allows per-entry language specific options. The literal value of this field is passed to polyglossia’s language switching facility when using the package option autolang=langname. For example, the fields:

<table>
<thead>
<tr>
<th>langid</th>
<th>= {english},</th>
</tr>
</thead>
<tbody>
<tr>
<td>langidopts</td>
<td>= {variant=british},</td>
</tr>
</tbody>
</table>

would wrap the bibliography entry in:

```latex
\english[variant=british]
...
\endenglish
```

**ids** field (separated list of entrykeys)

Citation key aliases for the main citation key. An entry may be cited by any of its aliases and biblatex will treat the citation as if it had used the primary citation key. This is to aid users who change their citation keys but have legacy documents which use older keys for the same entry. This field is consumed by the backend processing and does not appear in the .bbl.

**indexsorttitle** field (literal)

The title used when sorting the index. In contrast to indextitle, this field is used for sorting only. The printed title in the index is the indextitle or the title field. This field may be useful if the title contains special characters or commands which interfere with the sorting of the index. Consider this example:

<table>
<thead>
<tr>
<th>title</th>
<th>= {The \LaTeX\ Companion},</th>
</tr>
</thead>
<tbody>
<tr>
<td>indextitle</td>
<td>= {\LaTeX\ Companion, The},</td>
</tr>
<tr>
<td>indexsorttitle</td>
<td>= {LATEX Companion},</td>
</tr>
</tbody>
</table>

Style authors should note that biblatex automatically copies the value of either the indextitle or the title field to indexsorttitle if the latter field is undefined.

**keywords** field (separated values)

A separated list of keywords. These keywords are intended for the bibliography filters (see §§3.7.2 and 3.12.4), they are usually not printed. Note that with the default separator (comma), spaces around the separator are ignored.
**options** field (separated ⟨key⟩=⟨value⟩ options)

A separated list of entry options in ⟨key⟩=⟨value⟩ notation. This field is used to set options on a per-entry basis. See § 3.1.3 for details. Note that citation and bibliography styles may define additional entry options.

**presort** field (string)

A special field used to modify the sorting order of the bibliography. This field is the first item the sorting routine considers when sorting the bibliography, hence it may be used to arrange the entries in groups. This may be useful when creating subdivided bibliographies with the bibliography filters. Please refer to § 3.5 for further details. Also see § 4.5.6. This field is consumed by the backend processing and does not appear in the .bbl.

**related** field (separated values)

Citation keys of other entries which have a relationship to this entry. The relationship is specified by the relatedtype field. Please refer to § 3.4 for further details.

**relatedoptions** field (separated values)

Per-type options to set for a related entry. Note that this does not set the options on the related entry itself, only the dataonly clone which is used as a datasource for the parent entry.

**relatedtype** field (identifier)

An identifier which specified the type of relationship for the keys listed in the related field. The identifier is a localised bibliography string printed before the data from the related entry list. It is also used to identify type-specific formatting directives and bibliography macros for the related entries. Please refer to § 3.4 for further details.

**relatedstring** field (literal)

A field used to override the bibliography string specified by relatedtype. Please refer to § 3.4 for further details.

**sortkey** field (literal)

A field used to modify the sorting order of the bibliography. Think of this field as the master sort key. If present, biblatex uses this field during sorting and ignores everything else, except for the presort field. Please refer to § 3.5 for further details. This field is consumed by the backend processing and does not appear in the .bbl.

**sortname** list (name)

A name or a list of names used to modify the sorting order of the bibliography. If present, this list is used instead of author or editor when sorting the bibliography. Please refer to § 3.5 for further details. This field is consumed by the backend processing and does not appear in the .bbl.

**sortshorthand** field (literal)

Similar to sortkey but used in the list of shorthands. If present, biblatex uses this field instead of shorthand when sorting the list of shorthands. This is useful if
the shorthand field holds shorthands with formatting commands such as \textbf{or} \textit{. This field is consumed by the backend processing and does not appear in the .bbl.

**sorttitle** field (literal)

A field used to modify the sorting order of the bibliography. If present, this field is used instead of the title field when sorting the bibliography. The sorttitle field may come in hand if you have an entry with a title like “An Introduction to…” and want that alphabetized under ‘I’ rather than ‘A’. In this case, you could put “Introduction to…” in the sorttitle field. Please refer to § 3.5 for further details. This field is consumed by the backend processing and does not appear in the .bbl.

**sortyear** field (integer)

A field used to modify the sorting order of the bibliography. If present, this field is used instead of the year field when sorting the bibliography. Please refer to § 3.5 for further details. This field is consumed by the backend processing and does not appear in the .bbl.

**xdata** field (separated list of entrykeys)

This field inherits data from one or more \@xdata entries. Conceptually, the xdata field is related to crossref and xref: crossref establishes a logical parent/child relation and inherits data; xref establishes as logical parent/child relation without inheriting data; xdata inherits data without establishing a relation. The value of the xdata may be a single entry key or a separated list of keys. See § 3.12.6 for further details. This field is consumed by the backend processing and does not appear in the .bbl.

**xref** field (entry key)

This field is an alternative cross-referencing mechanism. It differs from crossref in that the child entry will not inherit any data from the parent entry specified in the xref field. If the number of child entries referencing a specific parent entry hits a certain threshold, the parent entry is automatically added to the bibliography even if it has not been cited explicitly. The threshold is settable with the minxrefs package option from § 3.1.2.1. Style authors should note that whether or not the xref fields of the child entries are defined on the biblatex level depends on the availability of the parent entry. If the parent entry is available, the xref fields of the child entries will be defined. If not, their xref fields will be undefined. Whether the parent entry is added to the bibliography implicitly because of the threshold or explicitly because it has been cited does not matter. See also the crossref field in this section as well as § 2.4.1.

### 2.2.4 Custom Fields

The fields listed in this section are intended for special bibliography styles. They are not used by the standard bibliography styles.

**name[a–c]** list (name)

Custom lists for special bibliography styles. Not used by the standard bibliography styles.
**name[a–c]type** field (key)

Similar to **author** and **editor** but referring to the fields **name[a–c]**. Not used by the standard bibliography styles.

**list[a–f]** list (literal)

Custom lists for special bibliography styles. Not used by the standard bibliography styles.

**user[a–f]** field (literal)

Custom fields for special bibliography styles. Not used by the standard bibliography styles.

**verb[a–c]** field (literal)

Similar to the custom fields above except that these are verbatim fields. Not used by the standard bibliography styles.

### 2.2.5 Field Aliases

The aliases listed in this section are provided for backwards compatibility with traditional BibTeX and other applications based on traditional BibTeX styles. Note that these aliases are immediately resolved as the bib file is processed. All bibliography and citation styles must use the names of the fields they point to, not the alias. In bib files, you may use either the alias or the field name but not both at the same time.

**address** list (literal)

An alias for **location**, provided for BibTeX compatibility. Traditional BibTeX uses the slightly misleading field name **address** for the place of publication, i.e., the location of the publisher, while biblatex uses the generic field name **location**. See §§ 2.2.2 and 2.3.4.

**annotate** field (literal)

An alias for **annotation**, provided for jurabib compatibility. See § 2.2.2.

**archiveprefix** field (literal)

An alias for **eprinttype**, provided for arXiv compatibility. See §§ 2.2.2 and 3.12.7.

**journal** field (literal)

An alias for **journaltitle**, provided for BibTeX compatibility. See § 2.2.2.

**key** field (literal)

An alias for **sortkey**, provided for BibTeX compatibility. See § 2.2.3.

**pdf** field (verbatim)

An alias for **file**, provided for JabRef compatibility. See § 2.2.2.

**primaryclass** field (literal)

An alias for **eprintclass**, provided for arXiv compatibility. See §§ 2.2.2 and 3.12.7.
An alias for institution, provided for BibTeX compatibility. The institution field is used by traditional BibTeX for technical reports whereas the school field holds the institution associated with theses. The \texttt{biblatex} package employs the generic field name institution in both cases. See §§2.2.2 and 2.3.4.

### 2.3 Usage Notes

The entry types and fields supported by this package should for the most part be intuitive to use for anyone familiar with BibTeX. However, apart from the additional types and fields provided by this package, some of the familiar ones are handled in a way which is in need of explanation. This package includes some compatibility code for \texttt{bib} files which were generated with a traditional BibTeX style in mind. Unfortunately, it is not possible to handle all legacy files automatically because \texttt{biblatex}'s data model is slightly different from traditional BibTeX. Therefore, such \texttt{bib} files will most likely require editing in order to work properly with this package. In sum, the following items are different from traditional BibTeX styles:

- The entry type \texttt{@inbook}. See §§2.1.1 and 2.3.1 for details.
- The fields institution, organization, and publisher as well as the aliases address and school. See §§2.2.2, 2.2.5, 2.3.4 for details.
- The handling of certain types of titles. See §2.3.5 for details.
- The field series. See §§2.2.2 and 2.3.7 for details.
- The fields year and month. See §§2.2.2, 2.3.8, 2.3.9 for details.
- The field edition. See §2.2.2 for details.
- The field key. See §2.3.2 for details.

Users of the \texttt{jurabib} package should note that the \texttt{shortauthor} field is treated as a name list by \texttt{biblatex}, see §2.3.3 for details.

#### 2.3.1 The Entry Type \texttt{@inbook}

Use the \texttt{@inbook} entry type for a self-contained part of a book with its own title only. It relates to \texttt{@book} just like \texttt{@incollection} relates to \texttt{@collection}. See §2.3.5 for examples. If you want to refer to a chapter or section of a book, simply use the \texttt{book} type and add a \texttt{chapter} and/or \texttt{pages} field. Whether a bibliography should at all include references to chapters or sections is controversial because a chapter is not a bibliographic entity.

#### 2.3.2 Missing and Omissible Data

The fields marked as ‘required’ in §2.1.1 are not strictly required in all cases. The bibliography styles which ship with this package can get by with as little as a title field for most entry types. A book published anonymously, a periodical without an explicit editor, or a software manual without an explicit author should pose no problem as far as the bibliography is concerned. Citation styles, however, may have different requirements. For example, an author-year citation scheme obviously requires an author/editor and a year field.

You may generally use the \texttt{label} field to provide a substitute for any missing data required for citations. How the \texttt{label} field is employed depends on the citation style.
The author-year citation styles which come with this package use the label field as a fallback if either the author/editor or the year is missing. The numeric styles, on the other hand, do not use it at all since the numeric scheme is independent of the available data. The author-title styles ignore it as well, because the bare title is usually sufficient to form a unique citation and a title is expected to be available in any case. The label field may also be used to override the non-numeric portion of the automatically generated labelalpha field used by alphabetic citation styles. See §4.2.4 for details.

Note that traditional BibTeX styles support a key field which is used for alphabetizing if both author and editor are missing. The biblatex package treats key as an alias for sortkey. In addition to that, it offers very fine-grained sorting controls, see §§2.2.3 and 3.5 for details. The natbib package employs the key field as a fallback label for citations. Use the label field instead.

2.3.3 Corporate Authors and Editors

Corporate authors and editors are given in the author or editor field, respectively. Note that they must be wrapped in an extra pair of curly braces to prevent data parsing from treating them as personal names which are to be dissected into their components. Use the shortauthor field if you want to give an abbreviated form of the name or an acronym for use in citations.

```
author = {{National Aeronautics and Space Administration}},
shortauthor = {NASA},
```

The default citation styles will use the short name in all citations while the full name is printed in the bibliography. For corporate editors, use the corresponding fields editor and shorteditor. Since all of these fields are treated as name lists, it is possible to mix personal names and corporate names, provided that the names of all corporations and institutions are wrapped in braces.

```
editor = {{National Aeronautics and Space Administration} and Doe, John},
shorteditor = {NASA and Doe, John},
```

Users switching from the jurabib package to biblatex should note that the shortauthor field is treated as a name list.

2.3.4 Literal Lists

The fields institution, organization, publisher, and location are literal lists in terms of §2.2. This also applies to origlocation, origpublisher and to the field aliases address and school. All of these fields may contain a list of items separated by the keyword ‘and’. If they contain a literal ‘and’, it must be wrapped in braces.

```
publisher = {William Reid {and} Company},
institution = {Office of Information Management {and} Communications},
organization = {American Society for Photogrammetry {and} } Remote Sensing
```
and
American Congress on Surveying \{and\}
\texttt{\rightarrow Mapping},

Note the difference between a literal '{\{and\}}' and the list separator '{\textit{and}}' in the above examples. You may also wrap the entire name in braces:

\begin{verbatim}
publisher = {{William Reid and Company}},
institution = {{Office of Information Management and Communications}},
organization = {{American Society for Photogrammetry and Remote Sensing} and
{American Congress on Surveying and Mapping}},
\end{verbatim}

Legacy files which have not been updated for use with \texttt{biblatex} will still work if these fields do not contain a literal 'and'. However, note that you will miss out on the additional features of literal lists in this case, such as configurable formatting and automatic truncation.

\subsection*{2.3.5 Titles}

The following examples demonstrate how to handle different types of titles. Let’s start with a five-volume work which is referred to as a whole:

\begin{verbatim}
@MvBook{works,  
  author = {Shakespeare, William}, 
  title = {Collected Works}, 
  volumes = {5},
  ...
\end{verbatim}

The individual volumes of a multi-volume work usually have a title of their own. Suppose the fourth volume of the \textit{Collected Works} includes Shakespeare’s sonnets and we are referring to this volume only:

\begin{verbatim}
@Book{works:4,  
  author = {Shakespeare, William}, 
  maintitle = {Collected Works}, 
  title = {Sonnets}, 
  volume = {4},
  ...
\end{verbatim}

If the individual volumes do not have a title, we put the main title in the \texttt{title} field and include a volume number:

\begin{verbatim}
@Book{works:4,  
  author = {Shakespeare, William}, 
  title = {Collected Works}, 
  volume = {4},
  ...
\end{verbatim}
In the next example, we are referring to a part of a volume, but this part is a self-contained work with its own title. The respective volume also has a title and there is still the main title of the entire edition:

```plaintext
@InBook{lear,
    author = {Shakespeare, William},
    bookauthor = {Shakespeare, William},
    maintitle = {Collected Works},
    booktitle = {Tragedies},
    title = {King Lear},
    volume = {1},
    pages = {53-159},
    ...
}
```

Suppose the first volume of the *Collected Works* includes a reprinted essay by a well-known scholar. This is not the usual introduction by the editor but a self-contained work. The *Collected Works* also have a separate editor:

```plaintext
@InBook{stage,
    author = {Expert, Edward},
    title = {Shakespeare and the Elizabethan Stage},
    bookauthor = {Shakespeare, William},
    editor = {Bookmaker, Bernard},
    maintitle = {Collected Works},
    booktitle = {Tragedies},
    volume = {1},
    pages = {7-49},
    ...
}
```

See §2.3.7 for further examples.

### 2.3.6 Editorial Roles

The type of editorial role performed by an editor in one of the `editor` fields (i.e., `editor`, `editora`, `editorb`, `editorc`) may be specified in the corresponding `editortype` field. The following roles are supported by default. The role ‘editor’ is the default. In this case, the `editortype` field is omissible.

- **editor**: The main editor. This is the most generic editorial role and the default value.
- **compiler**: Similar to editor but used if the task of the editor is mainly compiling.
- **founder**: The founding editor of a periodical or a comprehensive publication project such as a ‘Collected Works’ edition or a long-running legal commentary.
- **continuator**: An editor who continued the work of the founding editor (founder) but was subsequently replaced by the current editor (editor).
- **redactor**: A secondary editor whose task is redacting the work.
- **reviser**: A secondary editor whose task is revising the work.
- **collaborator**: A secondary editor or a consultant to the editor.

For example, if the task of the editor is compiling, you may indicate that in the corresponding `editortype` field:
There may also be secondary editors in addition to the main editor:

```latex
@Book{..., 
  author = {...}, 
  editor = {Editor, Edward}, 
  editora = {Redactor, Randolph}, 
  editoratype = {redactor}, 
  editorb = {Consultant, Conrad}, 
  editorbtype = {collaborator}, 
... 
}
```

Periodicals or long-running publication projects may see several generations of editors. For example, there may be a founding editor in addition to the current editor:

```latex
@Book{..., 
  author = {...}, 
  editor = {Editor, Edward}, 
  editora = {Founder, Frederic}, 
  editoratype = {founder},  
... 
}
```

Note that only the editor is considered in citations and when sorting the bibliography. If an entry is typically cited by the founding editor (and sorted accordingly in the bibliography), the founder goes into the editor field and the current editor moves to one of the editor... fields:

```latex
@Collection{..., 
  editor = {Founder, Frederic}, 
  editortype = {founder}, 
  editora = {Editor, Edward}, 
... 
}
```

You may add more roles by initializing and defining a new localisation key whose name corresponds to the identifier in the editor...type field. See §§ 3.9 and 4.9.1 for details.

### 2.3.7 Publication and Journal Series

The series field is used by traditional BibTeX styles both for the main title of a multi-volume work and for a publication series, i.e., a loosely related sequence of books by the same publisher which deal with the same general topic or belong to the same field of research. This may be ambiguous. This package introduces a maintitle field for multi-volume works and employs series for publication series only. The volume or number of a book in the series goes in the number field in this case:
The \texttt{@article} entry type makes use of the \texttt{series} field as well, but handles it in a special way. First, a test is performed to determine whether the value of the field is an integer. If so, it will be printed as an ordinal. If not, another test is performed to determine whether it is a localisation key. If so, the localised string is printed. If not, the value is printed as is. Consider the following example of a journal published in numbered series:

\begin{verbatim}
@Article{..., 
  journal = {Journal Name},
  series   = {3},
  volume   = {15},
  number   = {7},
  year     = {1995},
  ...}
\end{verbatim}

This entry will be printed as “\textit{Journal Name. 3rd ser. 15.7 (1995)}”. Some journals use designations such as “old series” and “new series” instead of a number. Such designations may be given in the \texttt{series} field as well, either as a literal string or as a localisation key. Consider the following example which makes use of the localisation key \texttt{newseries}:

\begin{verbatim}
@Article{..., 
  journal = {Journal Name},
  series   = {newseries},
  volume   = {9},
  year     = {1998},
  ...}
\end{verbatim}

This entry will be printed as “\textit{Journal Name. New ser. 9 (1998)}”. See §4.9.2 for a list of localisation keys defined by default.

\subsection*{2.3.8 Date and Time Specifications}

Date fields such as the default data model dates \texttt{date}, \texttt{origdate}, \texttt{eventdate}, and \texttt{urldate} adhere to \texttt{edtf}\textsuperscript{13} (Extended Date/Time Format) specification levels 0 and 1. Also supported are the open-ended range specifications in section 4.5 of the current working draft of ISO8601-2\textsuperscript{14}. \texttt{edtf} is a stricter subset of the somewhat messy formats allowed by \texttt{iso8601v2004} and is better suited to bibliographic data. In addition to the \texttt{edtf} empty date range markers, you may also specify an open ended/start date range by giving the range separator and omitting the end/start date

\textsuperscript{13}https://www.loc.gov/standards/datetime/pre-submission.html

\textsuperscript{14}http://www.loc.gov/standards/datetime/iso-tc154-wg5_n0039_iso_wd_8601-2_2016-02-16.pdf
<table>
<thead>
<tr>
<th>Date Specification</th>
<th>Formatted Date (Examples)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1850</td>
<td>1850</td>
</tr>
<tr>
<td>1997/</td>
<td>1997</td>
</tr>
<tr>
<td>/1997</td>
<td>–1997</td>
</tr>
<tr>
<td>1997/unknown</td>
<td>1997</td>
</tr>
<tr>
<td>1997/*</td>
<td>1997</td>
</tr>
<tr>
<td>unknown/1997</td>
<td>–1997</td>
</tr>
<tr>
<td>/1997</td>
<td>–1997</td>
</tr>
<tr>
<td>1997/open</td>
<td>1997</td>
</tr>
<tr>
<td>open/1997</td>
<td>–1997</td>
</tr>
<tr>
<td>1967-02</td>
<td>02/1967</td>
</tr>
<tr>
<td>2009-01-31</td>
<td>31/01/2009</td>
</tr>
<tr>
<td>2004-04-05T14:34:00</td>
<td>05/04/2004 2:34 PM</td>
</tr>
</tbody>
</table>

Table 3: Date Specifications

(e.g., YYYY/, /YYYY). See table 3 for some examples of valid date specifications and the formatted dates automatically generated by biblatex. The formatted date is language specific and will be adapted automatically. If there is no date field in an entry, biblatex will also consider the fields year and month for backwards compatibility with traditional BibTeX but this is not encouraged as explicit year and month are not parsed for date meta-information markers or times and are used as-is. Style authors should note that date fields like date or origdate are only available in the bib file. All dates are parsed and dissected into their components as the bib file is processed. The date and time components are made available to styles by way of the special fields discussed in §4.2.4.3. See this section and table 10 on page 152 for further information.

EDTF dates are astronomical dates in which year ‘0’ exists. When outputting dates in BCE or BC era (see the dateera option below), note that they will typically be one year earlier since BCE/BC era do not have a year 0 (year 0 is 1 BCE/BC). This conversion is automatic. See examples in table 5.

Date field names must end with the string ‘date’, as with the default date fields. Bear this in mind when adding new date fields to the datamodel (see §4.5.4). biblatex will check all date fields after reading the date model and will exit with an error if it finds a date field which does not adhere to this naming convention.

EDTF supports dates before common era (BCE/BC) by way of a negative date format and supports ‘approximate’ (circa) and uncertain dates. Such date formats set internal markers which can be tested for so that appropriate localised markers (such as circa or beforecommonera) can be inserted. Also supported are ‘unspecified’ dates (EDTF 5.2.2) which are automatically expanded into appropriate data ranges accompanied by a field <datatype>dateunspecified which details the granularity of the unspecified data. Styles may use this information to format such dates appropriately but the standard styles do not do this. See table 4 on page 38 for the allowed EDTF ‘unspecified’ formats, their range expansions and <datatype>dateunspecified values (see §4.2.4.1).

Table 5 shows formats which use appropriate tests and formatting. See the date meta-information tests in §4.6.2 and the localisation strings in §4.9.2.21. See also the 96-dates.tex example file for complete examples of the tests and localisation strings use.
Date Specification | Expanded Range     | Meta-information
-------------------|-------------------|----------------
199ₜu              | 1990/1999         | yearindecade
19uu               | 1900/1999         | yeartincentury
199ₚ-ₜu            | 1999-01/1999-12   | monthinyear
199ₚ-₀₁-ₜu         | 1999-01-01/1999-01-31 | dayinmonth
199ₚ-₀₁-₀₁-ₜu      | 1999-01-01/1999-12-31 | dayinyear

Table 4: EDTF 5.2.2 Unspecified Date Parsing

The output of ‘circa’, uncertainty and era information in standard styles (or custom styles not customising the internal \mkdaterange* macros) is controlled by the package options datecirca, dateuncertain, dateera and dateeraauto (see §3.1.2.1). See table 5 on page 39 for examples which assumes these options are all used.

2.3.9 Months and Journal Issues

The month field is an integer field. The bibliography style converts the month to a language-dependent string as required. For backwards compatibility, you may also use the following three-letter abbreviations in the month field: jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec. Note that these abbreviations are BibTeX strings which must be given without any braces or quotes. When using them, don’t say month={jan} or month="jan" but month=jan. It is not possible to specify a month such as month={8/9}. Use the date field for date ranges instead. Quarterly journals are typically identified by a designation such as ‘Spring’ or ‘Summer’ which should be given in the issue field. The placement of the issue field in @article entries is similar to and overrides the month field.

2.3.10 Pagination

When specifying a page or page range, either in the pages field of an entry or in the ⟨postnote⟩ argument to a citation command, it is convenient to have biblatex add prefixes like ‘p.’ or ‘pp.’ automatically and this is indeed what this package does by default. However, some works may use a different pagination scheme or may not be cited by page but rather by verse or line number. This is when the pagination and bookpagination fields come into play. As an example, consider the following entry:

```latex
@InBook{key, 
  title = {...}, 
  pagination = {verse}, 
  booktitle = {...}, 
  bookpagination = {page}, 
  pages = {53--65}, 
... 
```

The bookpagination field affects the formatting of the pages and pagetotal fields in the list of references. Since page is the default, this field is omissible in the above example. In this case, the page range will be formatted as ‘pp. 53–65’. Suppose that, when quoting from this work, it is customary to use verse numbers rather than page numbers in citations. This is reflected by the pagination field, which affects the formatting of the ⟨postnote⟩ argument to any citation command.
### Table 5: Enhanced Date Specifications

With a citation like `\cite{17}`, the postnote will be formatted as ‘v. 17’. Setting the pagination field to section would yield ‘§ 17’. See §3.13.3 for further usage instructions.

The pagination and bookpagination fields are key fields. This package will try to use their value as a localisation key, provided that the key is defined. Always use the singular form of the key name in bib files, the plural is formed automatically. The keys page, column, line, verse, section, and paragraph are predefined, with page being the default. The string ‘none’ has a special meaning when used in a pagination or bookpagination field. It suppresses the prefix for the respective entry. If there are no predefined localisation keys for the pagination scheme required by a certain entry, you can simply add them. See the commands `\NewBibliographyString` and `\DefineBibliographyStrings` in §3.9.

You need to define two localisation strings for each additional pagination scheme: the singular form (whose localisation key corresponds to the value of the pagination field) and the plural form (whose localisation key must be the singular plus the letter ‘s’). See the predefined keys in §4.9.2 for examples.

### 2.4 Hints and Caveats

This section provides some additional hints concerning the data interface of this package. It also addresses some common problems.

#### 2.4.1 Cross-referencing

biber features a highly customizable cross-referencing mechanism with flexible data inheritance rules. Duplicating certain fields in the parent entry or adding empty fields to the child entry is no longer required. Entries are specified in a natural way:

```plaintext
@Book{book,
  author   = {Author},
  title    = {Booktitle},
  subtitle = {Bookssubtitle},
  publisher = {Publisher},
}```
The title field of the parent will be copied to the booktitle field of the child, the subtitle becomes the booksSubtitle. The author of the parent becomes the bookauthor of the child and, since the child does not provide an author field, it is also duplicated as the author of the child. After data inheritance, the child entry is similar to this:

| author   | = {Author}, |
| bookauthor| = {Author}, |
| title    | = {Title},   |
| booktitle| = {Booktitle}, |
| booksSubtitle| = {BooksSubtitle}, |
| publisher | = {Publisher}, |
| location  | = {Location}, |
| date      | = {1995},   |
| pages     | = {5--25},  |

See appendix B for a list of mapping rules set up by default. Note that all of this is customizable. See § 4.5.11 on how to configure biber’s cross-referencing mechanism. See also § 2.2.3.

2.4.1.1 The xref field In addition to the crossref field, biblatex supports a simplified cross-referencing mechanism based on the xref field. This is useful if you want to establish a parent/child relation between two associated entries but prefer to keep them independent as far as the data is concerned. The xref field differs from crossref in that the child entry will not inherit any data from the parent. If the parent is referenced by a certain number of child entries, biblatex will automatically add it to the bibliography. The threshold is controlled by the minxrefs package option from § 3.1.2.1. See also § 2.2.3.

2.4.2 Sorting and Encoding Issues

biber handles Ascii, 8-bit encodings such as Latin 1, and utf-8. It features true Unicode support and is capable of reencoding the bib data on the fly in a robust way. For sorting, biber uses a Perl implementation of the Unicode Collation Algorithm (uca), as outlined in Unicode Technical Standard #10. Collation tailoring based on the Unicode Common Locale Data Repository (cldr) is also supported.

Supporting Unicode implies much more than handling utf-8 input. Unicode is a complex standard covering more than its most well-known parts, the Unicode character encoding and transport encodings such as utf-8. It also standardizes aspects such as string collation, which is required for language-sensitive sorting. For

---

15http://unicode.org/reports/tr10/
16http://cldr.unicode.org/
example, by using the Unicode Collation Algorithm, \texttt{biber} can handle the character \textquote{ß} without any manual intervention. All you need to do to get localised sorting is specify the locale:

\begin{verbatim}
\usepackage[sortlocale=de]{biblatex}
\end{verbatim}

or if you are using German as the main document language via \texttt{babel} or \texttt{polyglossia}:

\begin{verbatim}
\usepackage[sortlocale=auto]{biblatex}
\end{verbatim}

This will make \texttt{biblatex} pass the \texttt{babel/polyglossia} main document language as the locale which \texttt{biber} will map into a suitable default locale. \texttt{biber} will not try to get locale information from its environment as this makes document processing dependent on something not in the document which is against TeX’s spirit of reproducibility. This also makes sense since \texttt{babel/polyglossia} are in fact the relevant environment for a document. Note that this will also work with 8-bit encodings such as Latin 9, i.e., you can take advantage of Unicode-based sorting even though you are not using \texttt{utf-8} input. See §2.4.2.1 on how to specify input and data encodings properly.

2.4.2.1 Specifying Encodings When using a non-Ascii encoding in the \texttt{bib} file, it is important to understand what \texttt{biblatex} can do for you and what may require manual intervention. The package takes care of the LaTeX side, i.e., it ensures that the data imported from the \texttt{bbl} file is interpreted correctly, provided that the \texttt{bibencoding} package option is set properly. All of this is handled automatically and no further steps, apart from setting the \texttt{bibencoding} option in certain cases, are required. Here are a few typical usage scenarios along with the relevant lines from the document preamble:

- Ascii notation in both the \texttt{tex} and the \texttt{bib} file with pdfTeX or traditional TeX:

\begin{verbatim}
\usepackage{biblatex}
\end{verbatim}

- Latin 1 encoding (iso-8859-1) in the \texttt{tex} file, Ascii notation in the \texttt{bib} file with pdfTeX or traditional TeX:

\begin{verbatim}
\usepackage[latin1]{inputenc}
\usepackage[bibencoding=ascii]{biblatex}
\end{verbatim}

- Latin 9 encoding (iso-8859-15) in both the \texttt{tex} and the \texttt{bib} file with pdfTeX or traditional:

\begin{verbatim}
\usepackage[latin9]{inputenc}
\usepackage[bibencoding=auto]{biblatex}
\end{verbatim}

Since \texttt{bibencoding=auto} is the default setting, the option is omissible. The following setup will have the same effect:
The same scenario with \texttt{XeTeX} or \texttt{LuaTeX} in native \texttt{utf-8} mode:

\begin{verbatim}
\usepackage{biblatex}
\end{verbatim}

\texttt{biber} can handle Ascii notation, 8-bit encodings such as Latin 1, and \texttt{utf-8}. It is also capable of reencoding the \texttt{bib} data on the fly (replacing the limited macro-level reencoding feature of \texttt{biblatex}). This will happen automatically if required, provided that you specify the encoding of the \texttt{bib} files properly. In addition to the scenarios discussed above, \texttt{biber} can also handle the following cases:

- Transparent \texttt{utf-8} workflow, i.e., \texttt{utf-8} encoding in both the \texttt{tex} and the \texttt{bib} file with \texttt{pdfTeX} or traditional \TeX:

\begin{verbatim}
\usepackage{biblatex}
\end{verbatim}

\texttt{Since \texttt{bibencoding=auto} is the default setting, the option is omissible:}

\begin{verbatim}
\usepackage{biblatex}
\end{verbatim}

The same scenario with \texttt{XeTeX} or \texttt{LuaTeX} in native \texttt{utf-8} mode:

\begin{verbatim}
\usepackage{biblatex}
\end{verbatim}

- It is even possible to combine an 8-bit encoded \texttt{tex} file with \texttt{utf-8} encoding in the \texttt{bib} file, provided that all characters in the \texttt{bib} file are also covered by the selected 8-bit encoding:

\begin{verbatim}
\usepackage{biblatex}
\end{verbatim}

Some workarounds may be required when using traditional \TeX or \texttt{pdfTeX} with \texttt{utf-8} encoding because \texttt{inputenc}'s \texttt{utf8} module does not cover all of Unicode. Roughly speaking, it only covers the Western European Unicode range. When loading \texttt{inputenc} with the \texttt{utf8} option, \texttt{biblatex} will normally instruct \texttt{biber} to reencode the \texttt{bib} data to \texttt{utf-8}. This may lead to \texttt{inputenc} errors if some of the characters in the \texttt{bib} file are outside the limited Unicode range supported by \texttt{inputenc}. 
If you are affected by this problem, try setting the `safeinputenc` option:

\usepackage[utf8]{inputenc}
\usepackage[safeinputenc]{biblatex}

If this option is enabled, `biblatex` will ignore `inputenc`'s `utf8` option and use Ascii. `biber` will then try to convert the `bib` data to Ascii notation. For example, it will convert \$ to \$\k{S}. This option is similar to setting `texencoding=ascii` but will only take effect in this specific scenario (`inputenc/inputenx` with `UTF-8`). This workaround takes advantage of the fact that both Unicode and the `UTF-8` transport encoding are backwards compatible with Ascii.

This solution may be acceptable as a workaround if the data in the `bib` file is mostly Ascii anyway, with only a few strings, such as some authors' names, causing problems. However, keep in mind that it will not magically make traditional TeX or pdfTeX support Unicode. It may help if the occasional odd character is not supported by `inputenc`, but may still be processed by TeX when using an accent command (e.g., `\d{S}` instead of `\$`). If you need full Unicode support, however, switch to XeTeX or LuaTeX.

Typical errors when `inputenc` cannot handle a certain UTF-8 character are:

! Package `inputenc` Error: Unicode char `<char>` (U+<codepoint>) not set up for use with LaTeX.

but also less obvious things like:

! Argument of `\UTFviii@three@octets` has an extra \}.

## 3 User Guide

This part of the manual documents the user interface of the `biblatex` package. The user guide covers everything you need to know in order to use `biblatex` with the default styles that come with this package. You should read the user guide first in any case. If you want to write your own citation and/or bibliography styles, continue with the author guide afterwards.

### 3.1 Package Options

All package options are given in `(key)=(value)` notation. The value `true` is omissible with all boolean keys. For example, giving `sortcites` without a value is equivalent to `sortcites=true`.

#### 3.1.1 Load-time Options

The following options must be used as `biblatex` is loaded, i.e., in the optional argument to `\usepackage`.

**backend=bibtex,bibtex8,biber**

Specifies the database backend. The following backends are supported:
biber  biber, the default backend of biblatex, supports Ascii, 8-bit encodings, utf-8, on-the-fly reencoding, locale-specific sorting, and many other features. Locale-specific sorting, case-sensitive sorting, and upper/lowercase precedence are controlled by the options sortlocale, sortcase, and sortupper, respectively.

bibtex  Legacy BibTeX. Traditional BibTeX supports Ascii encoding only. Sorting is always case-insensitive.

bibtex8  bibtex8, the 8-bit implementation of BibTeX, supports Ascii and 8-bit encodings such as Latin 1.

See § 3.14 for details of using BibTeX as a backend.

\texttt{style}=⟨\texttt{file}⟩
\texttt{default: numeric}

Loads the bibliography style ⟨\texttt{file}⟩.bbx and the citation style ⟨\texttt{file}⟩.cbx. See § 3.3 for an overview of the standard styles.

\texttt{bibstyle}=⟨\texttt{file}⟩
\texttt{default: numeric}

Loads the bibliography style ⟨\texttt{file}⟩.bbx. See § 3.3.2 for an overview of the standard bibliography styles.

\texttt{citestyle}=⟨\texttt{file}⟩
\texttt{default: numeric}

Loads the citation style ⟨\texttt{file}⟩.cbx. See § 3.3.1 for an overview of the standard citation styles.

\texttt{natbib}=true, false
\texttt{default: false}

Loads compatibility module which provides aliases for the citation commands of the natbib package. See § 3.8.9 for details.

\texttt{mcite}=true, false
\texttt{default: false}

Loads a citation module which provides mcite/mciteplus-like citation commands. See § 3.8.10 for details.

\subsection{3.1.2 Preamble Options}
\subsubsection{3.1.2.1 General}

The following options may be used in the optional argument to \texttt{\usepackage} as well as in the configuration file and the document preamble. The default value listed to the right is the package default. Note that bibliography and citation styles may modify the default setting at load time, see § 3.3 for details.

\texttt{sorting}=nty, nyt, nyvt, anyt, anyvt, ynt, ydnt, none, debug, \texttt{default: nty (name)}

The sorting order of the bibliography. Unless stated otherwise, the entries are sorted in ascending order. The following choices are available by default:

\begin{itemize}
  \item \texttt{nty}  Sort by name, title, year.
  \item \texttt{nyt}  Sort by name, year, title.
  \item \texttt{nyvt}  Sort by name, year, volume, title.
  \item \texttt{anyt}  Sort by alphabetic label, name, year, title.
  \item \texttt{anyvt}  Sort by alphabetic label, name, year, volume, title.
\end{itemize}
Sort by year, name, title.
Sort by year (descending), name, title.
Do not sort at all. All entries are processed in citation order.
Sort by entry key. This is intended for debugging only.

Use \textit{(name)}, as defined with \texttt{\textbackslash DeclareSortingScheme} (§ 4.5.6)

Using any of the 'alphabetic' sorting schemes only makes sense in conjunction with a bibliography style which prints the corresponding labels. Note that some bibliography styles initialize this package option to a value different from the package default (nty). See § 3.3.2 for details. Please refer to § 3.5 for an in-depth explanation of the above sorting options as well as the fields considered in the sorting process. See also § 4.5.6 on how to adapt the predefined schemes or define new ones.

\texttt{\texttt{sortcase}}=true, false \texttt{\texttt{default: \texttt{true}}}

Whether or not to sort the bibliography and the list of shorthands case-sensitively.

\texttt{\texttt{sortupper}}=true, false \texttt{\texttt{default: \texttt{true}}}

This option corresponds to \texttt{biber's} --sortupper command-line option. If enabled, the bibliography is sorted in 'uppercase before lowercase' order. Disabling this option means 'lowercase before uppercase' order.

\texttt{\texttt{sortlocale}}=auto, (\texttt{locale})

This option sets the global sorting locale. Every sorting scheme inherits this locale if none is specified using the (\texttt{locale}) option to \texttt{\textbackslash printbibliography}. Setting this to auto requests that it be set to the babel/polyglossia main document language identifier, if these packages are used and en\_US otherwise. \texttt{biber} will map babel/polyglossia language identifiers into sensible locale identifiers (see the \texttt{biber} documentation). You can therefore specify either a normal locale identifier like de\_DE\_phonebook, es\_ES or one of the supported babel/polyglossia language identifiers if the mapping \texttt{biber} makes of this is fine for you.

\texttt{\texttt{related}}=true, false \texttt{\texttt{default: \texttt{true}}}

Whether or not to use information from related entries or not. See § 3.4.

\texttt{\texttt{sortcites}}=true, false \texttt{\texttt{default: \texttt{false}}}

Whether or not to sort citations if multiple entry keys are passed to a citation command. If this option is enabled, citations are sorted according to the current bibliography context sorting scheme (see § 3.7.10). This feature works with all citation styles.

\texttt{\texttt{maxnames}}=(\texttt{integer}) \texttt{\texttt{default: \texttt{3}}}

A threshold affecting all lists of names (\texttt{author}, editor, etc.). If a list exceeds this threshold, i.e., if it holds more than \texttt{\texttt{\texttt{integer}}} names, it is automatically truncated according to the setting of the \texttt{\texttt{minnames}} option. \texttt{maxnames} is the master option which sets both \texttt{\texttt{\texttt{maxbibnames}}} and \texttt{\texttt{\texttt{maxcitenames}}}.

\texttt{\texttt{minnames}}=(\texttt{integer}) \texttt{\texttt{default: \texttt{1}}}

A limit affecting all lists of names (\texttt{author}, editor, etc.). If a list holds more than \texttt{\texttt{\texttt{maxnames}}} names, it is automatically truncated to \texttt{\texttt{\texttt{minnames}}} names. The
\( \text{minnames} \) value must be smaller than or equal to \( \text{maxnames} \). \text{minnames} is the master option which sets both \text{minbibnames} and \text{mincitenames}.

\text{maxbibnames}=(\text{integer}) \quad \text{default: } \langle \text{maxnames} \rangle

Similar to \text{maxnames} but affects only the bibliography.

\text{minbibnames}=(\text{integer}) \quad \text{default: } \langle \text{minnames} \rangle

Similar to \text{minnames} but affects only the bibliography.

\text{maxcitenames}=(\text{integer}) \quad \text{default: } \langle \text{maxnames} \rangle

Similar to \text{maxnames} but affects only the citations in the document body.

\text{mincitenames}=(\text{integer}) \quad \text{default: } \langle \text{minnames} \rangle

Similar to \text{minnames} but affects only the citations in the document body.

\text{maxitems}=(\text{integer}) \quad \text{default: } 3

Similar to \text{maxnames}, but affecting all literal lists (publisher, location, etc.).

\text{minitems}=(\text{integer}) \quad \text{default: } 1

Similar to \text{minnames}, but affecting all literal lists (publisher, location, etc.).

\text{autocite}=\text{plain}, \text{inline}, \text{footnote}, \text{superscript}, \ldots

This option controls the behavior of the \text{\autocite} command discussed in § 3.8.4. The \text{plain} option makes \text{\autocite} behave like \text{\cite}, \text{inline} makes it behave like \text{\parencite}, \text{footnote} makes it behave like \text{\footcite}, and \text{superscript} makes it behave like \text{\supercite}. The options \text{plain}, \text{inline}, and \text{footnote} are always available, the \text{superscript} option is only provided by the numeric citation styles which come with this package. The citation style may also define additional options. The default setting of this option depends on the selected citation style, see § 3.3.1.

\text{autopunct}=\text{true}, \text{false} \quad \text{default: true}

This option controls whether the citation commands scan ahead for punctuation marks. See § 3.8 and \text{\DeclareAutoPunctuation} in § 4.7.5 for details.

\text{language}=\text{autobib}, \text{autocite}, \text{auto}, \langle \text{language} \rangle \quad \text{default: autobib}

This option controls multilingual support. When set to \text{autobib}, \text{autocite} or \text{auto}, \text{biblatex} will try to get the main document language from the \text{babel/polyglossia} package (and fall back to English if \text{babel/polyglossia} is not available). It is also possible to select the package language manually. In this case, the language chosen will override the \text{langid} of entries and you should still choose a language switching environment with the \text{autolang} option to select how the switch to the manually chosen language is handled. Please refer to table 2 for a list of supported languages and the corresponding identifiers. \text{autobib} switches the language for each entry in the bibliography using the \text{langid} field and the language environment specified by the \text{autolang} option. \text{autocite} switches the language for each citation using the \text{langid} field and the language environment specified by the \text{autolang} option. \text{auto} is a shorthand to set both \text{autobib} and \text{autocite}. 46
clearlang=true, false  

If this option is enabled, biblatex will automatically clear the language field of all entries whose language matches the babel/polyglossia language of the document (or the language specified explicitly with the language option) in order to omit redundant language specifications. The language mappings required by this feature are provided by the \DeclareRedundantLanguages command from § 4.9.1.

autolang=none, hyphen, other, other*, langname  

This option controls which babel language environment\footnote{polyglossia understands the babel language environments too and so this option controls both the babel and polyglossia language environments.} is used if the babel/polyglossia package is loaded and a bibliography entry includes a langid field (see § 2.2.3). Note that biblatex automatically adjusts to the main document language if babel/polyglossia is loaded. In multilingual documents, it will also continually adjust to the current language as far as citations and the default language of the bibliography is concerned. The effect of language adjustment depends on the language environment selected by this option. The possible choices are:

- none: Disable this feature, i.e., do not use any language environment at all.
- hyphen: Enclose the entry in a hyphenrules environment. This will load hyphenation patterns for the language specified in the hyphenation field of the entry, if available.
- other: Enclose the entry in an otherlanguage environment. This will load hyphenation patterns for the specified language, enable all extra definitions which babel/polyglossia and biblatex provide for the respective language, and translate key terms such as ‘editor’ and ‘volume’. The extra definitions include localisations of the date format, of ordinals, and similar things.
- other*: Enclose the entry in an otherlanguage* environment. Please note that biblatex treats otherlanguage* like otherlanguage but other packages may make a distinction in this case.
- langname: polyglossia only. Enclose the entry in a <languagename> environment. The benefit of this option value for polyglossia users is that it takes note of the langidopts field so that you can add per-language options to an entry (like selecting a language variant). When using babel, this option does the same as the other option value.

block=none, space, par, nbpar, ragged  

This option controls the extra spacing between blocks, i.e., larger segments of a bibliography entry. The possible choices are:

- none: Do not add anything at all.
- space: Insert additional horizontal space between blocks. This is similar to the default behavior of the standard LaTeX document classes.
par

Start a new paragraph for every block. This is similar to the `openbib` option of the standard LaTeX document classes.

nbpar

Similar to the `par` option, but disallows page breaks at block boundaries and within an entry.

ragged

Inserts a small negative penalty to encourage line breaks at block boundaries and sets the bibliography ragged right.

The `\newblockpunct` command may also be redefined directly to achieve different results, see §3.10.1. Also see §4.7.1 for additional information.

notetype=foot+end, footonly, endonly

default: foot+end

This option controls the behavior of `\mkbibfootnote`, `\mkbibendnote`, and similar wrappers from §4.10.4. The possible choices are:

foot+end

Support both footnotes and endnotes, i.e., `\mkbibfootnote` will generate footnotes and `\mkbibendnote` will generate endnotes.

footonly

Force footnotes, i.e., make `\mkbibendnote` generate footnotes.

endonly

Force endnotes, i.e., make `\mkbibfootnote` generate endnotes.

hyperref=true, false, auto

default: auto

Whether or not to transform citations and back references into clickable hyperlinks. This feature requires the `hyperref` package. It also requires support by the selected citation style. All standard styles which ship with this package support hyperlinks. `hyperref=auto` automatically detects if the `hyperref` package has been loaded.

backref=true, false

default: false

Whether or not to print back references in the bibliography. The back references are a list of page numbers indicating the pages on which the respective bibliography entry is cited. If there are `refsection` environments in the document, the back references are local to the reference sections. Strictly speaking, this option only controls whether the `biblatex` package collects the data required to print such references. This feature still has to be supported by the selected bibliography style. All standard styles which ship with this package do so.

backrefstyle=none, three, two, two+, three+, all+

default: three

This option controls how sequences of consecutive pages in the list of back references are formatted. The following styles are available:

none

Disable this feature, i.e., do not compress the page list.

three

Compress any sequence of three or more consecutive pages to a range, e.g., the list ‘1, 2, 11, 12, 13, 21, 22, 23, 24’ is compressed to ‘1–2, 11–13, 21–24’.

two

Compress any sequence of two or more consecutive pages to a range, e.g., the above list is compressed to ‘1–2, 11–13, 21–24’.

two+

Similar in concept to `two` but a sequence of exactly two consecutive pages is printed using the starting page and the localisation string `sequens`, e.g., the above list is compressed to ‘1 sq., 11–13, 21–24’.
three+ Similar in concept to two+ but a sequence of exactly three consecutive pages is printed using the starting page and the localisation string sequentes, e.g., the above list is compressed to ‘1 sq., 11 sqq., 21–24’.

all+ Similar in concept to three+ but any sequence of consecutive pages is printed as an open-ended range, e.g., the above list is compressed to ‘1 sq., 11 sqq., 21 sqq’.

All styles support both Arabic and Roman numerals. In order to avoid potentially ambiguous lists, different sets of numerals will not be mixed when generating ranges, e.g., the list ‘iii, iv, v, 6, 7, 8’ is compressed to ‘iii–v, 6–8’.

backrefsetstyle=setonly, memonly, setormem, setandmem, memandset, setplusmem

default: setonly

This option controls how back references to @set entries and their members are handled. The following options are available:

setonly All back references are added to the @set entry. The pageref lists of set members remain blank.

memonly References to set members are added to the respective member. References to the @set entry are added to all members. The pageref list of the @set entry remains blank.

setormem References to the @set entry are added to the @set entry. References to set members are added to the respective member.

setandmem References to the @set entry are added to the @set entry. References to set members are added to the respective member and to the @set entry.

memandset References to the @set entry are added to the @set entry and to all members. References to set members are added to the respective member.

setplusmem References to the @set entry are added to the @set entry and to all members. References to set members are added to the respective member and to the @set entry.

indexing=true, false, cite, bib

default: false

This option controls indexing in citations and in the bibliography. More precisely, it affects the \ifciteindex and \ifbibindex commands from §4.6.2. The option is settable on a global, a per-type, or on a per-entry basis. The possible choices are:

true Enable indexing globally.
false Disable indexing globally.
cite Enable indexing in citations only.
bib Enable indexing in the bibliography only.

This feature requires support by the selected citation style. All standard styles which ship with this package support indexing of both citations and entries in the bibliography. Note that you still need to enable indexing globally with \makeindex to get an index.
This option controls whether external files requested by way of the `printfile` command are loaded. See also § 3.12.8 and `printfile` in § 4.4.1. Note that this feature is disabled by default for performance reasons.

This option automatically starts a new reference section at a document division such as a chapter or a section. This is equivalent to the `newrefsection` command, see § 3.7.4 for details. The following choice of document divisions is available:

- **none**: Disable this feature.
- **part**: Start a reference section at every \part command.
- **chapter**: Start a reference section at every \chapter command.
- **section**: Start a reference section at every \section command.
- **subsection**: Start a reference section at every \subsection command.

The starred versions of these commands will not start a new reference section.

Similar to the `refsection` option but starts a new reference segment. This is equivalent to the `newrefsegment` command, see § 3.7.5 for details. When using both options, note that you can only apply this option to a lower-level document division than the one `refsection` is applied to and that nested reference segments will be local to the enclosing reference section.

This option automatically executes the `citereset` command from § 3.8.8 at a document division such as a chapter or a section. The following choice of document divisions is available:

- **none**: Disable this feature.
- **part**: Perform a reset at every \part command.
- **chapter**: Perform a reset at every \chapter command.
- **section**: Perform a reset at every \section command.
- **subsection**: Perform a reset at every \subsection command.

The starred versions of these commands will not trigger a reset.

Whether or not to use long or abbreviated strings in citations and in the bibliography. This option affects the localisation modules. If this option is enabled, key terms such as ‘editor’ are abbreviated. If not, they are written out.

This option controls the basic format of printed date specifications. The following choices are available:
<table>
<thead>
<tr>
<th>Date Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>year</strong></td>
<td>Use only years, for example:</td>
</tr>
<tr>
<td></td>
<td>2010</td>
</tr>
<tr>
<td></td>
<td>2010–2012</td>
</tr>
<tr>
<td><strong>short</strong></td>
<td>Use the short format with verbose ranges, for example:</td>
</tr>
<tr>
<td></td>
<td>01/01/2010</td>
</tr>
<tr>
<td></td>
<td>21/01/2010–30/01/2010</td>
</tr>
<tr>
<td></td>
<td>01/21/2010–01/30/2010</td>
</tr>
<tr>
<td><strong>long</strong></td>
<td>Use the long format with verbose ranges, for example:</td>
</tr>
<tr>
<td></td>
<td>1st January 2010</td>
</tr>
<tr>
<td></td>
<td>21st January 2010–30th January 2010</td>
</tr>
<tr>
<td></td>
<td>January 21, 2010–January 30, 2010</td>
</tr>
<tr>
<td><strong>terse</strong></td>
<td>Use the short format with compact ranges, for example:</td>
</tr>
<tr>
<td></td>
<td>21–30/01/2010</td>
</tr>
<tr>
<td></td>
<td>01/21–01/30/2010</td>
</tr>
<tr>
<td><strong>comp</strong></td>
<td>Use the long format with compact ranges, for example:</td>
</tr>
<tr>
<td></td>
<td>21st–30th January 2010</td>
</tr>
<tr>
<td></td>
<td>January 21–30, 2010</td>
</tr>
<tr>
<td><strong>edtf</strong></td>
<td>Use strict Extended Date/Time Format (yyyy-mm-dd), for example:</td>
</tr>
<tr>
<td></td>
<td>2010-01-01</td>
</tr>
<tr>
<td></td>
<td>2010-01-21/2010-01-30</td>
</tr>
<tr>
<td><strong>ymd</strong></td>
<td>A year-month-day format which can be modified by other options unlike strict EDTF, for example:</td>
</tr>
<tr>
<td></td>
<td>2010-1-1</td>
</tr>
<tr>
<td></td>
<td>2010-1-21/2010-1-30</td>
</tr>
</tbody>
</table>

Note that edtf format will enforce `dateera=astronomical`, `datezeros=true`, `timezeros=true`, `<datetype>time=24h` and `julian=false`. ymd is an ETFT-like format but which can change the various options which the strict edtf option does not allow for.

As seen in the above examples, the actual date format is language specific. Note that the month name in all long formats is responsive to the `abbreviate` package option. The leading zeros for months and days in all short formats may be controlled separately with the `datezeros` package option. The leading zeros for hours, minutes and seconds in all short formats may be controlled separately with the `timezeros` package option. If outputting times, the printing of seconds and timezones is controlled by the `seconds` and `timezones` options respectively.

The options `julian` and `gregorianstart` may be used to control when to output Julian Calendar dates.

**labeldate=year, short, long, terse, comp, ymd, edtf** default: `year`

Similar to the `date` option but controls the format of the date field selected with `\DeclareLabeldate`.  

---

18[https://www.loc.gov/standards/datetime/pre-submission.html]
<datatype>date=year, short, long, terse, comp, ymd, edtf    
        default: comp

Similar to the <datatype>date option but controls the format of the <datatype>date field in the datamodel.

<datatype>alldates=year, short, long, terse, comp, edtf

Sets the option for all dates in the datamodel to the same value. The date fields in the default data model are <datatype>date, <datatype>origdate, <datatype>eventdate and <datatype>urldate.

<datatype>julian=true, false    
        default: false

This option controls whether dates before the date specified in the <datatype>gregorianstart option will be converted automatically to the Julian Calendar. Dates so changed will return ‘true’ for the \ifdatejulian and \if<datatype>datejulian tests (see § 4.6.2). Please bear in mind that dates consisting of just a year like ‘1565’ will never be converted to a Julian Calendar date because a date without a month and day has an ambiguous Julian Calendar representation19. For example, in the case of ‘1565’, this is Julian year ‘1564’ until after the Gregorian date ‘10th January 1565’ when the Julian year becomes ‘1565’.

<datatype>gregorianstart=⟨YYYY-MM-DD⟩

This option controls the date before which dates are converted to the Julian Calendar. It is a strict format string, 4-digit year, 2-digit month and day, separated by a single dash character (any valid Unicode character with the ‘Dash’ property). The default is ‘1582-10-15’, the date of the instigation of the standard Gregorian Calendar. This option does not nothing unless julian is set to ‘true’.

<datatype>datezeros=true, false    
        default: true

This option controls whether <datatype>short and <datatype>terse date components are printed with leading zeros unless overridden by specific formatting.

<datatype>timezeros=true, false    
        default: true

This option controls whether time components are printed with leading zeros unless overridden by specific formatting.

<datatype>timezones=true, false    
        default: false

This option controls whether timezones are printed when printing times.

<datatype>seconds=true, false    
        default: false

This option controls whether seconds are printed when printing times.

<datatype>dateabbrev=true, false    
        default: true

This option controls whether <datatype>long and <datatype>comp dates are printed with long or abbreviated month/season names. The option is similar to the generic abbreviate option but specific to the date formatting.

<datatype>datecirca=true, false    
        default: false

This option controls whether to output ‘circa’ information about dates. If set to true, dates will be preceded by the expansion of the \datecircaprint macro (§ 3.10.1).

19This is potentially true for dates missing times too but this is not relevant for bibliographic work.
dateuncertain=true, false  
default: false

This option controls whether to output uncertainty information about dates. If set to true, dates will be followed by the expansion of the `\dateuncertainprint` macro and end dates will be followed by the `\enddateuncertainprint` macro (§ 3.10.1).

dateera=astronomical, secular, christian  
default: astronomical

This option controls how date era information is printed. ‘astronomical’ uses `\dateeraprint` to print era information before start/end dates. ‘secular’ and ‘christian’ uses `\dateeraprintpre` to print era information after the start/end/endates. By default ‘astronomical’ results in a minus sign before BCE/BC dates and ‘secular’/’christian’ results in the relevant localisation strings like ‘BCE’ or ‘BC’ after BCE/BC dates. See the relevant comments in § 3.10.1 and the localisation strings in § 4.9.2.21.

dateeraauto=⟨integer⟩  
default: 0

This option sets the astronomical year, below which era localisation strings are automatically added. This option does nothing without `dateera` being set to ‘secular’ or ‘christian’.

time=12h, 24h, 24hcomp  
default: 24h

This option controls the basic format of printed time specifications. The following choices are available:

- **24h** 24-hour format, for example:
  - 14:03:23
  - 14:3:23
  - 14:03:23+05:00
  - 14:03:23Z
  - 14:21–14:23:45
  - 14:23:23–14:23:45

- **24hcomp** 24-hour format with compressed ranges, for example:
  - 14:21–23 (hours are the same)
  - 14:23:23–45 (hour and minute are the same)

- **12h** 12-hour format with (localised) AM/PM markers, for example:
  - 2:34 PM
  - 2:34 PM–3:50 PM

As seen in the above examples, the actual time format is language specific. Note that the AM/PM string is responsive to the `abbreviate` package option, if this makes a difference in the specific locale. The leading zeros in the 24-hour formats may be controlled separately with the `timezeros` package option. The separator between time components (\bibtimesep and \bibtzminsep)and between the time and any timezone (\bibtimzone-Sep) are also language specific and customisable, see § 3.10.3. There are global package options which determine whether seconds and timezones are printed (seconds and timezone, respectively, see § 3.1.2.1). Timezones, if present, are either ‘Z’ or a numeric positive or negative offset. No
default styles print time information. Custom styles may print times by using the `\print\{datatype\}` `time` commands, see §4.4.1.

**labeltime=**12h, 24h, 24hcomp

default: 24h

Similar to the `time` option but controls the format of the time part fields obtained from the field selected with `\DeclareLabeldate`.

**<datatype>time=**12h, 24h, 24hcomp

default: 24h

Similar to the `time` option but controls the format of the time part fields obtained from the `<datatype>` `date` field in the datamodel.

**alltimes=**12h, 24h, 24hcomp

Sets `labeltime` and the `<datatype>` `time` option for all times in the datamodel to the same value. The date fields supporting time parts in the default data model are `date`, `origdate`, `eventdate` and `urldate`.

**dateusetime=**true, false

default: false

Specifies whether to print any time component of a date field after the date component. The separator between the date and time components is `\bibdatetimesep` from §3.10.3. This option does nothing if a compact date format is being used (see §3.1.2.1) as this would be very confusing.

**labeldateusetime=**true, false

default: false

Similar to the `dateusetime` option but controls the whether to print time components for the field selected with `\DeclareLabeldate`.

**<datatype>dateusetime=**true, false

default: false

Similar to the `dateusetime` option but controls the whether to print time components for the `<datatype>` `date` field in the datamodel.

**alldatesusetime=**true, false

default: false

Sets `labeldateusetime` and the `<datatype>` `dateusetime` option for all `<datatype>` `date` fields in the datamodel.

**defernumbers=**true, false

default: false

In contrast to standard LaTeX, the numeric labels generated by this package are normally assigned to the full list of references at the beginning of the document body. If this option is enabled, numeric labels (i.e., the `labelnumber` field discussed in §4.2.4) are assigned the first time an entry is printed in any bibliography. See §3.13.5 for further explanation. This option requires two LaTeX runs after the data has been exported to the `.bbl` file by the backend (in addition to any other runs required by page breaks changing etc.). An important thing to note is that if you change the value of this option in your document (or the value of options which depend on this like some of the options to the `\printbibliography` macro, see §3.7.2), then it is likely that you will need to delete your current `.aux` file and re-run LaTeX to obtain the correct numbering. See §4.1.
**punctfont**=true,false  
(default: false)

This option enables an alternative mechanism for dealing with unit punctuation after a field printed in a different font (for example, a title printed in italics). See \setpunctfont in §4.7.1 for details.

**arxiv**=abs,ps,pdf,format  
(default: abs)

Path selector for arXiv links. If hyperlink support is enabled, this option controls which version of the document the arXiv eprint links will point to. The following choices are available:

- **abs** Link to the abstract page.
- **ps** Link to the PostScript version.
- **pdf** Link to the PDF version.
- **format** Link to the format selector page.

See §3.12.7 for details on support for arXiv and electronic publishing information.

**texencoding**=auto,⟨encoding⟩  
(default: auto)

Specifies the encoding of the tex file. This option affects the data transferred from the backend to biblatex. This corresponds to biber's --output-encoding option. The following choices are available:

- **auto** Try to auto-detect the input encoding. If the inputenc/inputenx/luainputenc package is available, biblatex will get the main encoding from that package. If not, it assumes utf-8 encoding if XeTeX or LuaTeX has been detected, and Ascii otherwise.

- **⟨encoding⟩** Specifies the ⟨encoding⟩ explicitly. This is for odd cases in which auto-detection fails or you want to force a certain encoding for some reason.

Note that setting texencoding=⟨encoding⟩ will also affect the bibencoding option if bibencoding=auto.

**bibencoding**=auto,⟨encoding⟩  
(default: auto)

Specifies the encoding of the bib files. This corresponds to biber's --input-encoding option. The following choices are available:

- **auto** Use this option if the workflow is transparent, i.e., if the encoding of the bib file is identical to the encoding of the tex file.

- **⟨encoding⟩** If the encoding of the bib file is different from one of the tex file, you need to specify it explicitly.

By default, biblatex assumes that the tex file and the bib file use the same encoding (bibencoding=auto).

**safeinputenc**=true,false  
(default: false)

If this option is enabled, biblatex will automatically force texencoding=ascii if the inputenc/inputenx package has been loaded and the input encoding is utf-8, i.e., it will ignore any macro-based utf-8 support and use Ascii only. biber will then try to convert any non-Ascii data in the bib file to Ascii. For example, it will convert § to \d{S}. See §2.4.2.1 for an explanation of why you may want to enable this option.
**bibwarn** = true, false  
**default:** true

By default, **biblatex** will report warnings issued by the backend concerning the data in the **bib** file as LaTeX warnings. Use this option to suppress such warnings.

**mincrossrefs** = \textit{integer}  
**default:** 2

Sets the minimum number of cross references to \textit{integer} when requesting a backend run.\(^\text{20}\) This option also affects the handling of the \texttt{xref} field. See the field description in § 2.2.3 as well as § 2.4.1 for details.

**minxrefs** = \textit{integer}  
**default:** 2

As **mincrossrefs** but for \textit{xref} fields.

#### 3.1.2.2 Style-specific

The following options are provided by the standard styles (as opposed to the core package). Technically, they are preamble options like those in § 3.1.2.1.

**isbn** = true, false  
**default:** true

This option controls whether the fields **isbn/issn/isrn** are printed.

**url** = true, false  
**default:** true

This option controls whether the **url** field and the access date is printed. The option only affects entry types whose **url** information is optional. The **url** field of \@online entries is always printed.

**doi** = true, false  
**default:** true

This option controls whether the field **doi** is printed.

**eprint** = true, false  
**default:** true

This option controls whether **eprint** information is printed.

#### 3.1.2.3 Internal

The default settings of the following preamble options are controlled by bibliography and citation styles. Apart from the **pagetracker** and \texttt{<name>inits} options, which you may want to adapt, there is normally no need to set them explicitly.

**pagetracker** = true, false, page, spread  
**default:** false

This option controls the page tracker which is required by the \texttt{\ifsamepage} and \texttt{\iffirstonpage} tests from § 4.6.2. The possible choices are:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>Enable the tracker in automatic mode. This is like <strong>spread</strong> if LaTeX is in twoside mode, and like <strong>page</strong> otherwise.</td>
</tr>
<tr>
<td>false</td>
<td>Disable the tracker.</td>
</tr>
<tr>
<td>page</td>
<td>Enable the tracker in page mode. In this mode, tracking works on a per-page basis.</td>
</tr>
</tbody>
</table>

\(^{20}\)If an entry which is cross-referenced by other entries in the **bib** file hits this threshold, it is included in the bibliography even if it has not been cited explicitly. This is a standard feature of the BibTeX format and not specific to **biblatex**. See the description of the **crossref** field in § 2.2.3 for further information.
spread Enable the tracker in spread mode. In this mode, tracking works on a per-spread (double page) basis.

Note that this tracker is disabled in all floats, see § 4.11.5.

citecounter=true, false, context default: false

This option controls the citation counter which is required by citecounter from § 4.6.2. The possible choices are:

true Enable the citation counter in global mode.
false Disable the citation counter.
context Enable the citation counter in context-sensitive mode. In this mode, citations in footnotes and in the body text are counted independently.

citetracker=true, false, context, strict, constrict default: false

This option controls the citation tracker which is required by the \ifcite and \ifentry tests from § 4.6.2. The possible choices are:

true Enable the tracker in global mode.
false Disable the tracker.
context Enable the tracker in context-sensitive mode. In this mode, citations in footnotes and in the body text are tracked independently.
strict Enable the tracker in strict mode. In this mode, an item is only considered by the tracker if it appeared in a stand-alone citation, i.e., if a single entry key was passed to the citation command.
constrict This mode combines the features of context and strict.

Note that this tracker is disabled in all floats, see § 4.11.5.

ibidtracker=true, false, context, strict, constrict default: false

This option controls the ‘ibidem’ tracker which is required by the \ifcite test from § 4.6.2. The possible choices are:

true Enable the tracker in global mode.
false Disable the tracker.
context Enable the tracker in context-sensitive mode. In this mode, citations in footnotes and in the body text are tracked separately.
strict Enable the tracker in strict mode. In this mode, potentially ambiguous references are suppressed. A reference is considered ambiguous if either the current citation (the one including the ‘ibidem’) or the previous citation (the one the ‘ibidem’ refers to) consists of a list of references.  

\footnote{For example, suppose the initial citation is "Jones, Title; Williams, Title" and the following one "ibidem". From a technical point of view, it is fairly clear that the ‘ibidem’ refers to ‘Williams’ because this is the last reference processed by the previous citation command. To a human reader, however, this may not be obvious because the ‘ibidem’ may also refer to both titles. The strict mode avoids such ambiguous references.}
**constrict** This mode combines the features of **context** and **strict**. It also keeps track of footnote numbers and detects potentially ambiguous references in footnotes in a stricter way than the **strict** option. In addition to the conditions imposed by the **strict** option, a reference in a footnote will only be considered as unambiguous if the current citation and the previous citation are given in the same footnote or in immediately consecutive footnotes.

Note that this tracker is disabled in all floats, see § 4.11.5.

**opcitracker=true, false, context, strict, constrict**  

This option controls the ‘opcit’ tracker which is required by the \ifopcit test from § 4.6.2. This feature is similar to the ‘ibidem’ tracker, except that it tracks citations on a per-author/editor basis, i.e., \ifopcit will yield true if the cited item is the same as the last one by this author/editor. The possible choices are:

- **true** Enable the tracker in global mode.
- **false** Disable the tracker.
- **context** Enable the tracker in context-sensitive mode. In this mode, citations in footnotes and in the body text are tracked separately.
- **strict** Enable the tracker in strict mode. In this mode, potentially ambiguous references are suppressed. See ibidtracker=strict for details.
- **constrict** This mode combines the features of **context** and **strict**. See the explanation of ibidtracker=constrict for details.

Note that this tracker is disabled in all floats, see § 4.11.5.

**loccitracker=true, false, context, strict, constrict**  

This option controls the ‘loccit’ tracker which is required by the \ifloccit test from § 4.6.2. This feature is similar to the ‘opcit’ tracker except that it also checks whether the ⟨postnote⟩ arguments match, i.e., \ifloccit will yield true if the citation refers to the same page cited before. The possible choices are:

- **true** Enable the tracker in global mode.
- **false** Disable the tracker.
- **context** Enable the tracker in context-sensitive mode. In this mode, citations in footnotes and in the body text are tracked separately.
- **strict** Enable the tracker in strict mode. In this mode, potentially ambiguous references are suppressed. See ibidtracker=strict for details. In addition to that, this mode also checks if the ⟨postnote⟩ argument is numerical (based on \ifnumerals from § 4.6.2).
- **constrict** This mode combines the features of **context** and **strict**. See the explanation of ibidtracker=constrict for details. In addition to that, this mode also checks if the ⟨postnote⟩ argument is numerical (based on \ifnumerals from § 4.6.2).

Note that this tracker is disabled in all floats, see § 4.11.5.

**idemtracker=true, false, context, strict, constrict**  

This option controls the ‘idem’ tracker which is required by the \ifciteidem test from § 4.6.2. The possible choices are:
**true** Enable the tracker in global mode.

**false** Disable the tracker.

**context** Enable the tracker in context-sensitive mode. In this mode, citations in footnotes and in the body text are tracked separately.

**strict** This is an alias for true, provided only for consistency with the other trackers. Since ‘idem’ replacements do not get ambiguous in the same way as ‘ibidem’ or ‘op. cit.,’ the strict tracking mode does not apply to them.

**constrict** This mode is similar to context with one additional condition: a reference in a footnote will only be considered as unambiguous if the current citation and the previous citation are given in the same footnote or in immediately consecutive footnotes.

Note that this tracker is disabled in all floats, see § 4.11.5.

**parentracker=true,false**

This option controls the parenthesis tracker which keeps track of nested parentheses and brackets. This information is used by \parentext and \brackettext from § 3.8.5, \mkbibparens and \mkbibbrackets from § 4.10.4 and \bibopenparen, \bibcloseparen, \bibopenbracket, \bibclosebracket (also § 4.10.4).

**maxparens=⟨integer⟩**

The maximum permitted nesting level of parentheses and brackets. If parentheses and brackets are nested deeper than this value, biblatex will issue errors.

**<namepart>inits=true,false**

When enabled, all <namepart> name parts will be rendered as initials. The option will affect the \if<namepart>inits test from § 4.6.2. The valid name parts are defined in the data model by the \DeclareDatamodelConstant command (§ 4.2.3).

**terseinits=true,false**

This option controls the format of all initials generated by biblatex. If enabled, initials are rendered using a terse format without dots and spaces. For example, the initials of Donald Ervin Knuth would be rendered as ‘D. E.’ by default, and as ‘DE’ if this option is enabled. The option will affect the \ifterseinits test from § 4.6.2. The option works by redefining some macros which control the format of initials. See § 3.13.4 for details.

**labelalpha=true,false**

Whether or not to provide the special fields labelalpha and extraalpha, see § 4.2.4 for details. This option is also settable on a per-type basis. See also maxalphanames and minalphanames. Table 7 summarises the various extra* disambiguation counters and what they track.

**maxalphanames=⟨integer⟩**

Similar to the maxnames option but customizes the format of the labelalpha field.
Table 6: Work Uniqueness options

<table>
<thead>
<tr>
<th>Option</th>
<th>Test</th>
<th>Tracks</th>
</tr>
</thead>
<tbody>
<tr>
<td>singletitle</td>
<td>\ifsingletitle</td>
<td>labelname</td>
</tr>
<tr>
<td>uniquetitle</td>
<td>\ifuniquetitle</td>
<td>labeltitle</td>
</tr>
<tr>
<td>uniquetitleref</td>
<td>\ifuniquetitle</td>
<td>labeltitle when labelname is null</td>
</tr>
<tr>
<td>uniquetitle</td>
<td>\ifuniquetitle</td>
<td>labelname+labeltitle</td>
</tr>
</tbody>
</table>

\texttt{minalphanames=⟨integer⟩}

Similar to the \texttt{minnames} option but customizes the format of the labelalpha field.

\texttt{labelnumber=true, false}

default: false

Whether or not to provide the special field \texttt{labelnumber}, see § 4.2.4 for details. This option is also settable on a per-type basis.

\texttt{labeltitle=true, false}

default: false

Whether or not to provide the special field \texttt{extratitle}, see § 4.2.4 for details. Note that the special field \texttt{labeltitle} is always provided and this option controls rather whether \texttt{labeltitle} is used to generate \texttt{extratitle} information. This option is also settable on a per-type basis. Table 7 summarises the various extra* disambiguation counters and what they track.

\texttt{labeltitleyear=true, false}

default: false

Whether or not to provide the special field \texttt{extratitleyear}, see § 4.2.4 for details. Note that the special field \texttt{labeltitle} is always provided and this option controls rather whether \texttt{labeltitle} is used to generate \texttt{extratitleyear} information. This option is also settable on a per-type basis. Table 7 summarises the various extra* disambiguation counters and what they track.

\texttt{labeldateparts=true, false}

default: false

Whether or not to provide the special fields \texttt{labelyear}, \texttt{labelmonth}, \texttt{labeleday}, \texttt{labelendyear}, \texttt{labelendmonth}, \texttt{labeleday}, \texttt{labelhour}, \texttt{labelendhour}, \texttt{labelminute}, \texttt{labelendminute}, \texttt{labelsecond}, \texttt{labelendsecond}, \texttt{labelseason}, \texttt{labelendseason}, \texttt{labeltimezone}, \texttt{labelendtimezone} and \texttt{extrayear}, see § 4.2.4 for details. This option is also settable on a per-type basis. Table 7 summarises the various extra* disambiguation counters and what they track.

\texttt{singletitle=true, false}

default: false

Whether or not to provide the data required by the \texttt{\ifsingletitle} test, see § 4.6.2 for details. See table 6 for details on what determines the data for this test. This option is also settable on a per-type basis.

\texttt{uniquetitle=true, false}

default: false

Whether or not to provide the data required by the \texttt{\ifuniquetitle} test, see § 4.6.2 for details. See table 6 for details on what determines the data for this test. This option is also settable on a per-type basis.
Whether or not to provide the data required by the `\ifuniquebaretitle` test, see § 4.6.2 for details. See table 6 for details on what determines the data for this test. This option is also settable on a per-type basis.

Whether or not to provide the data required by the `\ifuniquework` test, see § 4.6.2 for details. See table 6 for details on what determines the data for this test. This option is also settable on a per-type basis.

Whether or not to provide the data required by the `\ifuniqueprimaryauthor` test, see § 4.6.2 for details.

Whether or not to update the `uniquename` counter, see § 4.6.2 for details. This feature will disambiguate individual names in the `labelname` list. This option is also settable on a per-type basis. The possible choices are:

- **true**: An alias for `full`.
- **false**: Disable this feature.
- **init**: Disambiguate using initials only.
- **full**: Disambiguate using initials or full names, as required.
- **allinit**: Similar to `init` but disambiguates all names in the `labelname` list, beyond `maxnames/minnames/uniquelist`.
- **allfull**: Similar to `full` but disambiguates all names in the `labelname` list, beyond `maxnames/minnames/uniquelist`.
- **mininit**: A variant of `init` which only disambiguates names in lists with identical last names.
- **minfull**: A variant of `full` which only disambiguates names in lists with identical last names.

Note that the `uniquename` option will also affect `uniquelist`, the `\ifsingletitle` test, and the `extrayear` field. See § 4.11.4 for further details and practical examples.

Whether or not to update the `uniquelist` counter, see § 4.6.2 for details. This feature will disambiguate the `labelname` list if it has become ambiguous after `maxnames/minnames` truncation. Essentially, it overrides `maxnames/minnames` on a per-field basis. This option is also settable on a per-type basis. The possible choices are:

- **true**: Disambiguate the `labelname` list.
- **false**: Disable this feature.
Option | Enabled field(s) | Enabled counter | Counter tracks |
--------|-----------------|----------------|----------------|
labelalpha | labelalpha | extraalpha | label |
labeldateparts | labelyear | extrayear | labelname+ |
 | labelmonth | | labelyear |
 | labelday | | |
 | labelendyear | | |
 | labelendmonth | | |
 | labelendday | | |
 | labelhour | | |
 | labelminute | | |
 | labelsecond | | |
 | labelendhour | | |
 | labelendminute | | |
 | labelendsecond | | |
 | labelseason | | |
 | labelendseason | | |
 | labeltimezone | | |
 | labeldendtime- | | |
 | zone | | |
 | labeltitle | – | extratitle | labelname+labeltitle |
 | labeltitleyear | – | extratitleyear | labeltitle+labelayear |

Table 7: Disambiguation counters

minyear | Disambiguate the labelname list only if the truncated list is identical to another one with the same labelyear. This mode of operation is useful for author-year styles and requires labeldateparts=true.

Note that the uniquelist option will also affect the \ifsingletitle test and the extrayear field. See §4.11.4 for further details and practical examples.

3.1.3 Entry Options

Entry options are package options which determine how bibliography data entries are handled. They may be set at various scopes defined below.

3.1.3.1 Preamble/Type/Entry Options | The following options are settable on a per-type basis or on a per-entry in the options field. In addition to that, they may also be used in the optional argument to usepackage as well as in the configuration file and the document preamble. This is useful if you want to change the default behaviour globally.

useauthor=true, false | default: true

Whether the author is used in labels and considered during sorting. This may be useful if an entry includes an author field but is usually not cited by author for some reason. Setting useauthor=false does not mean that the author is ignored completely. It means that the author is not used in labels and ignored during sorting. The entry will then be alphabetized by editor or title. With the standard styles, the author is printed after the title in this case. See also §3.5. This option is also settable on a per-type and per-entry basis.
useeditor=true, false  
**default**: true

Whether the editor replaces a missing author in labels and during sorting. This may be useful if an entry includes an editor field but is usually not cited by editor. Setting useeditor=false does not mean that the editor is ignored completely. It means that the editor does not replace a missing author in labels and during sorting. The entry will then be alphabetized by title. With the standard styles, the editor is printed after the title in this case. See also §3.5. This option is also settable on a per-type and per-entry basis.

usetranslator=true, false  
**default**: false

Whether the translator replaces a missing author/editor in labels and during sorting. Setting usetranslator=true does not mean that the translator overrides the author/editor. It means that the translator is considered as a fallback if the author/editor is missing or if useauthor and useeditor are set to false. In other words, in order to cite a book by translator rather than by author, you need to set the following options: This option is also settable on a per-type and per-entry basis.

```latex
@Book{...,  
opt = {useauthor=false, usetranslator=true},  
author = {...},  
translator = {...},  
...  
```

With the standard styles, the translator is printed after the title by default. See also §3.5.

usename=true, false  
**default**: true

As per useauthor, useeditor and usetranslator, all name lists defined in the data model have an option controlling their behaviour in sorting and labelling automatically defined. Global, per-type and per-entry options called ‘usename’ are automatically created.

useprefix=true, false  
**default**: false

Whether the default date model name part ‘prefix’ (von, van, of, da, de, della, etc.) is considered when:

- Printing the last name in citations  
- Sorting  
- Generation of certain types of labels  
- Generating name uniqueness information  
- Formatting aspects of the bibliography

For example, if this option is enabled, biblatex precedes the last name with the prefix—Ludwig van Beethoven would be cited as “van Beethoven” and alphabetized as “Van Beethoven, Ludwig”. If this option is disabled (the default), he is cited as “Beethoven” and alphabetized as “Beethoven, Ludwig van” instead. This option is also settable on a per-type scope. With biblatexml datasources and the BibTeX extended name format supported by biber, this is also settable on per-namelist and per-name scopes.
The indexing option is also settable per-type or per-entry basis. See § 3.1.2.1 for details.

### 3.1.3.2 Type/Entry Options

The following options are settable on a per-type basis or on a per-entry in the options field. They are not available globally.

- **skipbib**
  
  default: false

  If this option is enabled, the entry is excluded from the bibliography but it may still be cited. This option is also settable on a per-type basis.

- **skipbiblist**
  
  default: false

  If this option is enabled, the entry is excluded from and bibliography lists. It is still included in the bibliography and it may also be cited by shorthand etc. This option is also settable on a per-type basis.

- **skiplab**
  
  default: false

  If this option is enabled, biblatex will not assign any labels to the entry. It is not required for normal operation. Use it with care. If enabled, biblatex can not guarantee unique citations for the respective entry and citations styles which require labels may fail to create valid citations for the entry. This option is also settable on a per-type basis.

- **dataonly**
  
  default: false

  Setting this option is equivalent to uniquename=false, uniquelist=false, skipbib, skipbiblist, and skiplab. It is not required for normal operation. Use it with care. This option is also settable on a per-type basis.

### 3.1.3.3 Entry Only Options

The following options are settable only on a per-entry in the options field. They are not available globally or per-type.

- **labelnamefield**
  
  Specifies the field to consider first when looking for a labelname candidate. It is essentially prepended to the search list created by \DeclareLabelname for just this entry.

- **labeltitlefield**
  
  Specifies the field to consider first when looking for a labeltitle candidate. It is essentially prepended to the search list created by \DeclareLabeltitle for just this entry.

### 3.1.4 Legacy Options

The following legacy option may be used globally in the optional argument to \documentclass or locally in the optional argument to \usepackage:

- **openbib**
  
  This option is provided for backwards compatibility with the standard LaTeX document classes. openbib is similar to block=par.  

  Deprecated
3.2 Global Customization

Apart from writing new citation and bibliography styles, there are numerous ways to customize the styles which ship with this package. Customization will usually take place in the preamble, but there is also a configuration file for permanent adaptations. The configuration file may also be used to initialize the package options to a value different from the package default.

3.2.1 Configuration File

If available, this package will load the configuration file \texttt{biblatex.cfg}. This file is read at the end of the package, immediately after the citation and bibliography styles have been loaded.

3.2.2 Setting Package Options

The load-time package options in §3.1.1 must be given in the optional argument to \usepackage. The package options in §3.1.2 may also be given in the preamble. The options are executed with the following command:

\ExecuteBibliographyOptions[\langle entrytype, \ldots \rangle]{\langle key=value, \ldots \rangle}

This command may also be used in the configuration file to modify the default setting of a package option. Certain options are also settable on a per-type basis. In this case, the optional \langle entrytype \rangle argument specifies the entry type. The \langle entrytype \rangle argument may be a comma-separated list of values.

3.3 Standard Styles

This section provides a short description of all bibliography and citation styles which ship with the \texttt{biblatex} package. If you want to write your own styles, see §4.

3.3.1 Citation Styles

The citation styles which come with this package implement several common citation schemes. All standard styles cater for the shorthand field and support hyperlinks as well as indexing.

\texttt{numeric} This style implements a numeric citation scheme similar to the standard bibliographic facilities of LaTeX. It should be employed in conjunction with a numeric bibliography style which prints the corresponding labels in the bibliography. It is intended for in-text citations. The style will set the following package options at load time: autocite=inline, labelnumber=true. This style also provides an additional preamble option called subentry which affects the handling of entry sets. If this option is disabled, citations referring to a member of a set will point to the entire set. If it is enabled, the style supports citations like “[5c]” which point to a subentry in a set (the third one in this example). See the style example for details.

\texttt{numeric-comp} A compact variant of the \texttt{numeric} style which prints a list of more than two consecutive numbers as a range. This style is similar to the \cite package and the sort\&compress option of the natbib package in numerical mode. For example, instead of “[8, 3, 1, 7, 2]” this style would print “[1–3, 7, 8]”. It is intended for in-text citations. The style will set the following package options at load time: autocite=inline, sortcites=true, labelnumber=true. It also provides the subentry option.
numeric-verb  A verbose variant of the numeric style. The difference affects the handling of a list of citations and is only apparent when multiple entry keys are passed to a single citation command. For example, instead of “[2, 5, 6]” this style would print “[2]; [5]; [6]”. It is intended for in-text citations. The style will set the following package options at load time: autocite=inline, labelnumber=true. It also provides the subentry option.

alphanumeric  This style implements an alphanumeric citation scheme similar to the alpha.bst style of traditional BibTeX. The alphanumeric labels resemble a compact author-year style to some extent, but the way they are employed is similar to a numeric citation scheme. For example, instead of “Jones 1995” this style would use the label “[Jon95]”. “Jones and Williams 1986” would be rendered as “[JW86]”. This style should be employed in conjunction with an alphabetic bibliography style which prints the corresponding labels in the bibliography. It is intended for in-text citations. The style will set the following package options at load time: autocite=inline, labelalpha=true.

alphanumeric-verb  A verbose variant of the alphanumeric style. The difference affects the handling of a list of citations and is only apparent when multiple entry keys are passed to a single citation command. For example, instead of “[Doe92; Doe95; Jon98]” this style would print “[Doe92]: [Doe95]: [Jon98]”. It is intended for in-text citations. The style will set the following package options at load time: autocite=inline, labelalpha=true.

authoryear  This style implements an author-year citation scheme. If the bibliography contains two or more works by the same author which were all published in the same year, a letter is appended to the year. For example, this style would print citations such as “Doe 1995a; Doe 1995b; Jones 1998”. This style should be employed in conjunction with an author-year bibliography style which prints the corresponding labels in the bibliography. It is primarily intended for in-text citations, but it could also be used with citations given in footnotes. The style will set the following package options at load time: autocite=inline, labeldateparts=true, uniquename=full, uniquelist=true.

authoryear-comp  A compact variant of the authoryear style which prints the author only once if subsequent references passed to a single citation command share the same author. If they share the same year as well, the year is also printed only once. For example, instead of “Doe 1995b; Doe 1992; Jones 1998; Doe 1995a” this style would print “Doe 1992, 1995a,b; Jones 1998”. It is primarily intended for in-text citations, but it could also be used with citations given in footnotes. The style will set the following package options at load time: autocite=inline, sortcites=true, labeldateparts=true, uniquename=full, uniquelist=true.

authoryear-ibid  A variant of the authoryear style which replaces repeated citations by the abbreviation ibidem unless the citation is the first one on the current page or double-page spread, or the ibidem would be ambiguous in the sense of the package option ibidtracker=constrict. The style will set the following package options at load time: autocite=inline, labeldateparts=true, uniquename=full, uniquelist=true, ibidem=ibidem, ibidtracker=constrict, pagetracker=true. This style also provides an additional preamble option called ibidpage. See the style example for details.

authoryear-icomp  A style combining authoryear-comp and authoryear-ibid. The style will set the following package options at load time: autocite=inline,
This style implements a simple author-title citation scheme. It will make use of the shorttitle field, if available. It is intended for citations given in footnotes. The style will set the following package options at load time: autocite=footnote, uniquename=full, uniquelist=true.

A compact variant of the authortitle style which prints the author only once if subsequent references passed to a single citation command share the same author. For example, instead of “Doe, First title; Doe, Second title” this style would print “Doe, First title, Second title”. It is intended for citations given in footnotes. The style will set the following package options at load time: autocite=footnote, sortcites=true, uniquename=full, uniquelist=true.

A variant of the authortitle style which replaces repeated citations by the abbreviation ibidem unless the citation is the first one on the current page or double-page spread, or the ibidem would be ambiguous in the sense of the package option ibidtracker=constrict. It is intended for citations given in footnotes. The style will set the following package options at load time: autocite=footnote, uniquename=full, uniquelist=true, ibidtracker=constrict, pagetracker=true. This style also provides an additional preamble option called ibidpage. See the style example for details.

A style combining the features of authortitle-comp and authortitle-ibid. The style will set the following package options at load time: autocite=footnote, uniquename=full, uniquelist=true, ibidtracker=constrict, pagetracker=true, sortcites=true. This style also provides an additional preamble option called ibidpage. See the style example for details.

A terse variant of the authortitle style which only prints the title if the bibliography contains more than one work by the respective author/editor. This style will make use of the shorttitle field, if available. It is suitable for in-text citations as well as citations given in footnotes. The style will set the following package options at load time: autocite=inline, singletitle=true, uniquename=full, uniquelist=true.

A style combining the features of authortitle-comp and authortitle-terse. This style will make use of the shorttitle field, if available. It is suitable for in-text citations as well as citations given in footnotes. The style will set the following package options at load time: autocite=inline, sortcites=true, singletitle=true, uniquename=full, uniquelist=true.

A style combining the features of authortitle-icomp and authortitle-terse. In other words: a variant of the authortitle-tcomp style with an ibidem feature. This style is suitable for in-text citations as well as citations given in footnotes. It will set the following package options at load time: autocite=inline, ibidtracker=constrict, pagetracker=true, sortcites = true, singletitle = true, uniquename = full,
uniquelist=true. This style also provides an additional preamble option called ibidpage. See the style example for details.

verbose
A verbose citation style which prints a full citation similar to a bibliography entry when an entry is cited for the first time, and a short citation afterwards. If available, the shorttitle field is used in all short citations. If the shorthand field is defined, the shorthand is introduced on the first citation and used as the short citation thereafter. This style may be used without a list of references and shorthands since all bibliographic data is provided on the first citation. It is intended for citations given in footnotes. The style will set the following package options at load time: autocite=footnote, citetracker=context. This style also provides an additional preamble option called citepages. See the style example for details.

verbose-ibid
A variant of theverbose style which replaces repeated citations by the abbreviation ibidem unless the citation is the first one on the current page or double-page spread, or the ibidem would be ambiguous in the sense of ibidtracker=strict. This style is intended for citations given in footnotes. The style will set the following package options at load time: autocite=footnote, citetracker=context, ibidtracker=constrict, pagetracker=true. This style also provides additional preamble options called ibidpage and citepages. See the style example for details.

verbose-note
This style is similar to the verbose style in that it prints a full citation similar to a bibliography entry when an entry is cited for the first time, and a short citation afterwards. In contrast to the verbose style, the short citation is a pointer to the footnote with the full citation. If the bibliography contains more than one work by the respective author/editor, the pointer also includes the title. If available, the shorttitle field is used in all short citations. If the shorthand field is defined, it is handled as with the verbose style. This style may be used without a list of references and shorthands since all bibliographic data is provided on the first citation. It is exclusively intended for citations given in footnotes. The style will set the following package options at load time: autocite=footnote, citetracker=context, singletitle=true. This style also provides additional preamble options called pageref and citepages. See the style example for details.

verbose-inote
A variant of the verbose-note style which replaces repeated citations by the abbreviation ibidem unless the citation is the first one on the current page or double-page spread, or the ibidem would be ambiguous in the sense of ibidtracker=strict. This style is exclusively intended for citations given in footnotes. It will set the following package options at load time: autocite=footnote, citetracker=context, ibidtracker=constrict, singletitle=true, pagetracker=true. This style also provides additional preamble options called ibidpage, pageref, and citepages. See the style example for details.

verbose-trad1
This style implements a traditional citation scheme. It is similar to the verbose style in that it prints a full citation similar to a bibliography entry when an item is cited for the first time, and a short citation afterwards. Apart from that, it uses the scholarly abbreviations ibidem, idem, op. cit., and loc. cit. to replace recurrent authors, titles, and page numbers in repeated citations in a special way. If the shorthand field is defined, the shorthand is introduced on the first citation and used as the short citation thereafter. This style may be used without a list of references and shorthands since all bibliographic data is provided
on the first citation. It is intended for citations given in footnotes. The style will set the following package options at load time: autocite=footnote, citetracker=context, ibidtracker=constrict, idemtracker=constrict, opcittracker=context, loccittracker=context. This style also provides additional preamble options called ibidpage, strict, and citepages. See the style example for details.

**verbose-trad2**  Another traditional citation scheme. It is also similar to the *verbose* style but uses scholarly abbreviations like *ibidem* and *idem* in repeated citations. In contrast to the *verbose-trad1* style, the logic of the *op. cit.* abbreviations is different in this style and *loc. cit.* is not used at all. It is in fact more similar to *verbose-ibid* and *verbose-inote* than to *verbose-trad1*. The style will set the following package options at load time: autocite=footnote, citetracker=context, ibidtracker=constrict, idemtracker=constrict. This style also provides additional preamble options called ibidpage, strict, and citepages. See the style example for details.

**verbose-trad3**  Yet another traditional citation scheme. It is similar to the *verbose-trad2* style but uses the scholarly abbreviations *ibidem* and *op. cit.* in a slightly different way. The style will set the following package options at load time: autocite=footnote, citetracker=context, ibidtracker=constrict, loccittracker=constrict. This style also provides additional preamble options called strict and citepages. See the style example for details.

**reading**  A citation style which goes with the bibliography style by the same name. It simply loads the *authortitle* style.

The following citation styles are special purpose styles. They are not intended for the final version of a document:

**draft**  A draft style which uses the entry keys in citations. The style will set the following package options at load time: autocite=plain.

**debug** This style prints the entry key rather than some kind of label. It is intended for debugging only and will set the following package options at load time: autocite=plain.

### 3.3.2 Bibliography Styles

All bibliography styles which come with this package use the same basic format for the individual bibliography entries. They only differ in the kind of label printed in the bibliography and the overall formatting of the list of references. There is a matching bibliography style for every citation style. Note that some bibliography styles are not mentioned below because they simply load a more generic style. For example, the bibliography style *authortitle-comp* will load the *authortitle* style.

**numeric** This style prints a numeric label similar to the standard bibliographic facilities of LaTeX. It is intended for use in conjunction with a numeric citation style. Note that the shorthand field overrides the default label. The style will set the following package options at load time: labelnumber=true. This style also provides an additional preamble option called subentry which affects the formatting of entry sets. If this option is enabled, all members of a set are marked with a letter which may be used in citations referring to a set member rather than the entire set. See the style example for details.
The following bibliography styles are special purpose styles. They are not intended for the final version of a document:

**draft** This draft style includes the entry keys in the bibliography. The bibliography will be sorted by entry key. The style will set the following package options at load time: `sorting=debug`.

**debug** This style prints all bibliographic data in tabular format. It is intended for debugging only and will set the following package options at load time: `sorting=debug`.

### 3.4 Related Entries

Almost all bibliography styles require authors to specify certain types of relationship between entries such as “Reprint of”, “Reprinted in” etc. It is impossible to provide data fields to cover all of these relationships and so biblatex provides a general mechanism for this using the entry fields `related`, `relatedtype` and `relatedstring`. A related entry does not need to be cited and does not appear in
the bibliography itself (unless of course it is also cited itself independently) as a clone is taken of the related entry to be used as a data source. The relatedtype field should specify a localisation string which will be printed before the information from the related entries is printed, for example “Orig. Pub. as”. The relatedstring field can be used to override the string determined via relatedtype. Some examples:

```latex
@Book{key1,
   ...
   related   = {key2},
   relatedtype = {reprintof},
   ...
}
@Book{key2,
   ...
}
```

Here we specify that entry key1 is a reprint of entry key2. In the bibliography driver for Book entries, when `\usebibmacro{related}` is called for entry key1:

- If the localisation string “reprintof” is defined, it is printed in the relatedstring:reprintof format. If this formatting directive is undefined, the string is printed in the relatedstring:default format.

- If the related:reprintof macro is defined, it is used to format the information contained in entry key2, otherwise the related:default macro is used.

- If the related:reprintof format is defined, it is used to format both the localisation string and data. If this format is not defined, then the related format is used instead.

It is also supported to have cascading and/or circular relations:

```latex
@Book{key1,
   ...
   related   = {key2},
   relatedtype = {reprintof},
   ...
}
@Book{key2,
   ...
   related   = {key3},
   relatedtype = {translationof},
   ...
}
@Book{key3,
   ...
   related   = {key2},
   ...
}
```
Multiple relations to the same entry are also possible:

```latex
@MVBook{key1,
  ...
  related = {key2,key3},
  relatedtype = {multivolume},
  ...
}
@Book{key2,
  ...
}
@Book{key3,
  ...
}
```

Note the the order of the keys in lists of multiple related entries is important. The data from multiple related entries is printed in the order of the keys listed in this field. See §4.5.1 for a more details on the mechanisms behind this feature. You can turn this feature off using the package option related from §3.1.2.1.

You can use the relatedoptions to set options on the related entry data clone. This is useful if you need to override the dataonly option which is set by default on all related entry clones. For example, if you will expose some of the names in the related clone in your document, you may want to have them disambiguated from names in other entries but normally this won’t happen as related clones have the per-entry dataonly option set and this in turn sets uniquename=false and uniquelist=false. In such a case, you can set relatedoptions to just skiplab, skipbib, skipbiblist.

### 3.5 Sorting Options

This package supports various sorting schemes for the bibliography. The sorting scheme is selected with the sorting package option from §3.1.2.1. Apart from the regular data fields there are also some special fields which may be used to optimize the sorting of the bibliography. Appendices C.1 and C.2 give an outline of the alphabetic sorting schemes supported by biblatex. Chronological sorting schemes are listed in appendix C.3. A few explanations concerning these schemes are in order.

The first item considered in the sorting process is always the presort field of the entry. If this field is undefined, biblatex will use the default value ‘mm’ as a presort string. The next item considered is the sortkey field. If this field is defined, it serves as the master sort key. Apart from the presort field, no further data is considered in this case. If the sortkey field is undefined, sorting continues with the name. The package will try using the sortname, author, editor, and translator fields, in this order. Which fields are considered also depends on the setting of the use<name> options. If all such options are disabled, the sortname field is ignored as well. Note that all name fields are responsive to maxnames and minnames. If no name field is available, either because all of them are undefined
or because all use\texttt{name} options are disabled, \texttt{biblatex} will fall back to the \texttt{sorttitle} and \texttt{title} fields as a last resort. The remaining items are, in various order: the \texttt{sortyear} field, if defined, or the first four digits of the \texttt{year} field otherwise; the \texttt{sorttitle} field, if defined, or the \texttt{title} field otherwise; the \texttt{volume} field. Note that the sorting schemes shown in appendix C.2 include an additional item: \texttt{labelalpha} is the label used by ‘alphabetic’ bibliography styles. Strictly speaking, the string used for sorting is \texttt{labelalpha} + \texttt{extraalpha}. The sorting schemes in appendix C.2 are intended to be used in conjunction with alphabetic styles only.

The chronological sorting schemes presented in appendix C.3 also make use of the \texttt{presort} and \texttt{sortkey} fields, if defined. The next item considered is the \texttt{sortyear} or the \texttt{year} field, depending on availability. The \texttt{ynt} scheme extracts the first four Arabic figures from the field. If both fields are undefined, the string 9999 is used as a fallback value. This means that all entries without a year will be moved to the end of the list. The \texttt{ydnt} scheme is similar in concept but sorts the year in descending order. As with the \texttt{ynt} scheme, the string 9999 is used as a fallback value. The remaining items are similar to the alphabetic sorting schemes discussed above. Note that the \texttt{ydnt} sorting scheme will only sort the date in descending order. All other items are sorted in ascending order as usual.

Using special fields such as \texttt{sortkey}, \texttt{sortname}, or \texttt{sorttitle} is usually not required. The \texttt{biblatex} package is quite capable of working out the desired sorting order by using the data found in the regular fields of an entry. You will only need them if you want to manually modify the sorting order of the bibliography or if any data required for sorting is missing. Please refer to the field descriptions in § 2.2.3 for details on possible uses of the special fields.

### 3.6 Data Annotations

Ideally, there should be no formatting information in a bibliography data file, however, sometimes such questionable practice seems to be the only way in which the desired results can be achieved. Data annotations are a way of addressing this by allowing users to attach semantic information (rather than typographical markup) to information in a bibliography data source so that the information can be used at markup time by a style. For example, if you wanted to highlight certain names in a work depending on whether they were a student author (indicated by a superscript asterisk in the references) or a corresponding author (indicated by bold face), then you might be tempted to try:

```latex
@MISC{Article1,  
   AUTHOR = {Last1\textsuperscript{*}, First1 and \textbf{Last2}, \textbf{First2} and Last3, First3}
}
```

There are several problems with this. Firstly, it will break BibTeX’s fragile name parsing routines and probably won’t compile at all. Secondly, it is not only mixing up data with markup, it does so in a hard-coded way: this data can’t easily be shared and used with other styles. While it is possible to achieve this formatting using \texttt{biblatex} internals in a style or document, this is a complex and unreliable method which many users will not wish to use.

In order to address these issues, \texttt{biblatex} has a general data annotation facility which allows you to attach a comma-separated list of annotations to data fields, items within data field lists (like names) and even parts of specific items such as parts
of names (given name, family name etc.). There are macros provided to check for annotations which can be used in formatting directives.

There are three “scopes” for data annotations, in order of increasing specificity:

- **field**—applied to top-level fields in a data source entry
- **item**—applied to items within a list field in a data source entry
- **part**—applied to parts within items within a list field in a data source entry

Data annotations are supported for BibTeX and biblatexml data sources.

Attaching annotations to data is easy in biblatexml data sources as they are specified by simple XML attributes. Continuing with the example above, we would have:

```
<bltx:entries xmlns:bltx="http://biblatex-biber.sourceforge.net/biblatexml">
<bltx:entry id="test" entrytype="misc">
<bltx:names type="author">
<bltx:name>
<bltx:namepart type="given" initial="F">First1</bltx:namepart>
<bltx:namepart type="family" initial="L" annotation="student">Last1</bltx:namepart>
</bltx:name>
<bltx:name annotation="corresponding">
<bltx:namepart type="given" initial="F">First2</bltx:namepart>
<bltx:namepart type="family" initial="L">Last2</bltx:namepart>
</bltx:name>
<bltx:name>
<bltx:namepart type="given" initial="F">First3</bltx:namepart>
<bltx:namepart type="family" initial="L">Last3</bltx:namepart>
</bltx:name>
</bltx:names>
</bltx:entry>
</bltx:entries>
```

Here the annotations are attached in an obvious way to the data items. With BibTeX data sources, the format for annotations is not quite as intuitive:

```
@MISC{ann1, 
  AUTHOR = {Last1, First1 and Last2, First2 and Last3, First3},
  AUTHOR+an = {1:family=student;2=corresponding},
}
```

Here the field name suffix +an is a user-definable suffix which marks a data field as an annotation of the unsuffixed field. The format of annotation fields in BibTeX data sources is as follows:

22See biber’s --annotation-marker option.

74
That is, one or more specifications separated by semi-colons. Each specification is an equals sign followed by a comma-separated list of annotation keywords. To annotation a specific item in a list, put the number of the list item before the equals sign (lists start at 1). If you need to annotate a specific part of the list item, give its name after the list item number, preceded by a colon. Name part names are defined in the data model, see §4.2.3. Some examples:

```
AUTHOR = {Last1, First1 and Last2, First2 and Last3, First3},
AUTHOR+an = {3:given=annotation1, annotation2},
TITLE = {A title},
TITLE+an = {=a title annotation, another title annotation},
LANGUAGE = {english and french},
LANGUAGE+an = {1=annotation3; 2=annotation4}
```

To access the annotation information when formatting bibliography data, three macros are provided, corresponding to the three annotation scopes:

```
\iffieldannotation{⟨annotation⟩}{⟨true⟩}{⟨false⟩}
```

Executes ⟨true⟩ if the current data field has an annotation ⟨annotation⟩ and false otherwise.

```
\ifitemannotation{⟨annotation⟩}{⟨true⟩}{⟨false⟩}
```

Executes ⟨true⟩ if the current item in the current data field has an annotation ⟨annotation⟩ and false otherwise.

```
\ifpartannotation{⟨part⟩}{⟨annotation⟩}{⟨true⟩}{⟨false⟩}
```

Executes ⟨true⟩ if the part named ⟨part⟩ in current item in the current data field has an annotation ⟨annotation⟩ and false otherwise.

These macros are available in the same places as \currentfield, \currentlist and \currentname (see §4.4.2), that is, inside formatting directives. They automatically determine the name of the current data field being processed and also the current listcount value which determines the current item in list fields. Parts such as name parts need to be named explicitly. As an example of how to use the annotation information to solve the problem originally presented in this section, this could be used in the name formatting directives to put an asterisk after all family names annotated as "student":

```
\ifpartannotation{family}{student}
  {\textsuperscript{*}}\%
```

75
To put the given and family names of name list items annotated as “corresponding” in boldface:

```
\renewcommand*{\mkbibnamegiven}{[1]{% 
\ifitemannotation{corresponding}
  \textbf{#1}}
{#1}}
\renewcommand*{\mkbibnamefamily}{[1]{% 
\ifitemannotation{corresponding}
  \textbf{#1}}
{#1}}
```

### 3.7 Bibliography Commands

#### 3.7.1 Resources

\addbibresource{⟨options⟩}{⟨resource⟩}

Adds a ⟨resource⟩, such as a .bib file, to the default resource list. This command is only available in the preamble. It replaces the \bibliography legacy command. Note that files must be specified with their full name, including the extension. Do not omit the .bib extension from the filename. Also note that the ⟨resource⟩ is a single resource. Invoke \addbibresource multiple times to add more resources, for example:

```
\addbibresource{bibfile1.bib}
\addbibresource{bibfile2.bib}
\addbibresource[location=remote]{http://www.citeulike.org/bibtex/group/9517}
\addbibresource[location=remote, label=lan]{ftp://192.168.1.57/~user/file.bib}
```

Since the ⟨resource⟩ string is read in a verbatim-like mode, it may contain arbitrary characters. The only restriction is that any curly braces must be balanced. The following ⟨options⟩ are available:

- **label=⟨identifier⟩**
  
  Assigns a label to a resource. The ⟨identifier⟩ may be used in place of the full resource name in the optional argument of refsection (see § 3.7.4).

- **location=⟨location⟩**
  
  The location of the resource. The ⟨location⟩ may be either local for local resources or remote for urls. Remote resources require biber. The protocols HTTP and FTP are supported. The remote URL must be a fully qualified path to a bib file or a URL which returns a bib file.

- **type=⟨type⟩**
  
  default: file
  
  The type of resource. Currently, the only supported type is file.

- **datatype=⟨datatype⟩**
  
  default: bibtex
  
  The data type (format) of the resource. The following formats are currently supported:
\textbf{bibtex} \quad \textit{BibTeX} format.

\textbf{biblatexml} \quad \textit{Experimental XML format for \texttt{biblatex}. See § D.}

\addglobalbib\[\langle\textit{options}\rangle\]{\langle\textit{resource}\rangle}

This command differs from \texttt{\addbibresource} in that the \langle\textit{resource}\rangle is added to the global resource list. The difference between default resources and global resources is only relevant if there are reference sections in the document and the optional argument of \texttt{refsection} (§ 3.7.4) is used to specify alternative resources which replace the default resource list. Any global resources are added to all reference sections.

\addsectionbib\[\langle\textit{options}\rangle\]{\langle\textit{resource}\rangle}

This command differs from \texttt{\addbibresource} in that the resource \langle\textit{options}\rangle are registered but the \langle\textit{resource}\rangle not added to any resource list. This is only required for resources which 1) are given exclusively in the optional argument of \texttt{refsection} (§ 3.7.4) and 2) require options different from the default settings. In this case, \texttt{\addsectionbib} is employed to qualify the \langle\textit{resource}\rangle prior to using it by setting the appropriate \langle\textit{options}\rangle in the preamble. The \texttt{label} option may be useful to assign a short name to the resource.

\texttt{\bibliography}\{\langle\textit{bibfile, ...}\rangle\}

The legacy command for adding bibliographic resources, supported for backwards compatibility. Like \texttt{\addbibresource}, this command is only available in the preamble and adds resources to the default resource list. Its argument is a comma-separated list of \texttt{bib} files. The \texttt{.bib} extension may be omitted from the filename. Invoking this command multiple times to add more files is permissible. This command is deprecated. Please consider using \texttt{\addbibresource} instead.

\textbf{3.7.2 The Bibliography}

\texttt{\printbibliography}\[\langle\textit{key=value, ...}\rangle\]

This command prints the bibliography. It takes one optional argument, which is a list of options given in \langle\textit{key}=\langle\textit{value}\rangle\rangle notation. The following options are available:

\texttt{\texttt{env}=\langle\textit{name}\rangle}\quad \text{default: bibliography/shorthands}

The ‘high-level’ layout of the bibliography and the list of shorthands is controlled by environments defined with \texttt{\defbibenvironment}. This option selects an environment. The \langle\textit{name}\rangle corresponds to the identifier used when defining the environment with \texttt{\defbibenvironment}. By default, the \texttt{\printbibliography} command uses the identifier \texttt{bibliography}; \texttt{\printbiblist} uses shorthands. See also §§ 3.7.3 and 3.7.7.

\texttt{\texttt{heading}=\langle\textit{name}\rangle}\quad \text{default: bibliography/shorthands}

The bibliography and the list of shorthands typically have a chapter or section heading. This option selects the heading \langle\textit{name}\rangle, as defined with \texttt{\defbibheading}. By default, the \texttt{\printbibliography} command uses the heading \texttt{bibliography}; \texttt{\printbiblist} uses shorthands. See also §§ 3.7.3 and 3.7.7.
title=(text)

This option overrides the default title provided by the heading selected with the heading option, if supported by the heading definition. See § 3.7.7 for details.

prenote=(name)

The prenote is an arbitrary piece of text to be printed after the heading but before the list of references. This option selects the prenote (name), as defined with \defbibnote. By default, no prenote is printed. The note is printed in the standard text font. It is not affected by \bibsetup and \bibfont but it may contain its own font declarations. See § 3.7.8 for details.

postnote=(name)

The postnote is an arbitrary piece of text to be printed after the list of references. This option selects the postnote (name), as defined with \defbibnote. By default, no postnote is printed. The note is printed in the standard text font. It is not affected by \bibsetup and \bibfont but it may contain its own font declarations. See § 3.7.8 for details.

section=(integer)  

default: current section

Print only entries cited in reference section (integer). The reference sections are numbered starting at 1. All citations given outside a refsection environment are assigned to section 0. See § 3.7.4 for details and § 3.12.3 for usage examples.

segment=(integer)  

default: 0

Print only entries cited in reference segment (integer). The reference segments are numbered starting at 1. All citations given outside a resegment environment are assigned to segment 0. See § 3.7.5 for details and § 3.12.3 for usage examples. Remember that segments within a section are numbered local to the section so the segment you request will be the nth segment in the requested (or currently active enclosing) section.

type=(entrytype)

Print only entries whose entry type is (entrytype).

nottype=(entrytype)

Print only entries whose entry type is not (entrytype). This option may be used multiple times.

subtype=(subtype)

Print only entries whose entrysubtype is defined and (subtype).

notsubtype=(subtype)

Print only entries whose entrysubtype is undefined or not (subtype). This option may be used multiple times.

keyword=(keyword)

Print only entries whose keywords field includes (keyword). This option may be used multiple times.
\texttt{notkeyword}=(\textit{keyword})

Print only entries whose \texttt{keywords} field does not include \textit{(keyword)}. This option may be used multiple times.

\texttt{category}=(\textit{category})

Print only entries assigned to category \textit{(category)}. This option may be used multiple times.

\texttt{notcategory}=(\textit{category})

Print only entries not assigned to category \textit{(category)}. This option may be used multiple times.

\texttt{filter}=(\textit{name})

Filter the entries with filter \textit{(name)}, as defined with \texttt{\defbibfilter}. See §3.7.9 for details.

\texttt{check}=(\textit{name})

Filter the entries with check \textit{(name)}, as defined with \texttt{\defbibcheck}. See §3.7.9 for details.

\texttt{resetnumbers}=(\textit{true},\textit{false},\textit{number})

This option applies to numerical citation/bibliography styles only and requires that the \texttt{deffernumbers} option from §3.1.2.1 be enabled globally. If enabled, it will reset the numerical labels assigned to the entries in the respective bibliography, i.e., the numbering will restart at 1. You can also pass a number to this option, for example: \texttt{resetnumbers=10} to reset numbering to the specified number to aid numbering continuity across documents. Use this option with care as \texttt{biblatex} can not guarantee unique labels globally if they are reset manually.

\texttt{omitnumbers=true,false}

This option applies to numerical citation/bibliography styles only and requires that the \texttt{deffernumbers} option from §3.1.2.1 be enabled globally. If enabled, \texttt{biblatex} will not assign a numerical label to the entries in the respective bibliography. This is useful when mixing a numerical subbibliography with one or more subbibliographies using a different scheme (e.g., author-title or author-year).

\texttt{\bibbysection[\textit{key=value, ...}]}

This command automatically loops over all reference sections. This is equivalent to giving one \texttt{\printbibliography} command for every section but has the additional benefit of automatically skipping sections without references. Note that \texttt{\bibbysection} starts looking for references in section 1. It will ignore references given outside of \texttt{refsection} environments since they are assigned to section 0. See §3.12.3 for usage examples. The options are a subset of those supported by \texttt{\printbibliography}. Valid options are \texttt{env}, \texttt{heading}, \texttt{prenote}, \texttt{postnote}. The current bibliography context sorting scheme is used for all sections (see §3.7.10).
\bibbysegment[\{key=value, ...\}]

This command automatically loops over all reference segments. This is equivalent to giving one \printbibliography command for every segment in the current refsection but has the additional benefit of automatically skipping segments without references. Note that \bibbysegment starts looking for references in segment 1. It will ignore references given outside of refsegment environments since they are assigned to segment 0. See § 3.12.3 for usage examples. The options are a subset of those supported by \printbibliography. Valid options are env, heading, prenote, postnote. The current bibliography context sorting scheme is used for all segments (see § 3.7.10).

\bibbycategory[\{key=value, ...\}]

This command loops over all bibliography categories. This is equivalent to giving one \printbibliography command for every category but has the additional benefit of automatically skipping empty categories. The categories are processed in the order in which they were declared. See § 3.12.3 for usage examples. The options are a subset of those supported by \printbibliography. Valid options are env, prenote, postnote, section. Note that heading is not available with this command. The name of the current category is automatically used as the heading name. This is equivalent to passing heading=(category) to \printbibliography and implies that there must be a matching heading definition for every category. The current bibliography context sorting scheme is used for all categories (see § 3.7.10).

\printbibheading[\{key=value, ...\}]

This command prints a bibliography heading defined with \defbibheading. It takes one optional argument, which is a list of options given in \{key\}=(value) notation. The options are a small subset of those supported by \printbibliography. Valid options are heading and title. By default, this command uses the heading bibliography. See § 3.7.7 for details. Also see §§ 3.12.3 and 3.12.4 for usage examples.

To print a bibliography with a different sorting scheme than the global sorting scheme, use the bibliography context switching commands from § 3.7.10.

3.7.3 Bibliography Lists

\biblatex can, in addition to printing normal bibliographies, also print arbitrary lists of information derived from the bibliography data such as a list of shorthand abbreviations for particular entries or a list of abbreviations of journal titles.

A bibliography list differs from a normal bibliography in that the same bibliography driver is used to print all entries rather than a specific driver being used for each entry depending on the entry type.

\printbiblist[\{key=value, ...\} \{<biblistname>\}]

This command prints a bibliography list. It takes an optional argument, which is a list of options given in \{key\}=(value) notation. Valid options are all options supported by \printbibliography (§ 3.7.2) except resetnumbers and omitnumbers. If there are any refsection environments in the document, the bibliography list will be local to these environments; see § 3.7.4 for details. By default, this command uses the heading biblist. See § 3.7.7 for details.
The `<biblistname>` is a mandatory argument which names the bibliography list. This name is used to identify:

- The default bibliography driver used to print the list entries
- A default filter declared with \DeclareBiblistFilter (see §4.5.7) used to filter the entries returned from `biber`
- A default check declared with \defbibcheck (see §3.7.9) used to post-process the list entries
- The default bib environment to use
- The default sorting scheme name to use

In terms of sorting the list, the default is to sort use the sorting scheme named after the bibliography list (if it exists) and only then to fall back to the current context sorting scheme is this is not defined (see §3.7.10).

The most common bibliography list is a list of shorthand abbreviations for certain entries and so this has a convenience alias \printshorthands[...] for backwards compatibility which is defined as:

```
\printbiblist[...]{shorthand}
```

`biblatex` provides automatic support for data source fields in the default data model marked as ‘Label fields’ (See §2.2.2). Such fields automatically have defined for them:

- A default bib environment (See §3.7.7)
- A bibliography list filter (See §4.5.7)
- Some supporting formats and lengths (See §4.10.5 and §4.10.4)

Therefore only a minimal setup is required to print bibliography lists with such fields. For example, to print a list of journal title abbreviations, you can minimally put this in your preamble:

```
\DeclareBibliographyDriver{shortjournal}{%
    \printfield{journaltitle}}
```

Then you can put this in your document where you want to print the list:

```
\printbiblist[title={Journal Shorthands}]{shortjournal}
```

Since `shortjournal` is defined in the default data model as a ‘Label field’, this example:

- Uses the automatically created ‘shortjournal’ bib environment
- Uses the automatically created ‘shortjournal’ bibliography list filter to return only entries with a `shortjournal` field in the `.bbl`
- Uses the defined ‘shortjournal’ bibliography driver to print the entries
- Uses the default ‘biblist’ heading but overrides the title with ‘Journal Short-hands’
• Uses the current bibliography context sorting scheme if no scheme exists with the name `shortjournal`

Often, you will want to sort on the label field of the list and since a sorting scheme is automatically picked up if it is named after the list, in this case you could simply do:

\begin{verbatim}
\DeclareSortingScheme{shortjournal}{
  \sort{
    \field{shortjournal}
  }
}
\end{verbatim}

Naturally all defaults can be overridden by options to `\printbiblist` and definitions of the environments, filters etc. and in this way arbitrary types of bibliography lists can be printed containing a variety of information from the bibliography data.

Bibliography lists are often used to print lists of various kinds of shorthands and this can result in duplicate entries if more than one bibliography entry has the same shorthand. For example, several journal articles in the same journal would result in duplicate entries in a list of journal shorthands. You can use the fact that such lists automatically pick up a `\bibcheck` with the same name as the list to define a check to remove duplicates. If you are defining a list to print all of the journal shorthands using the `shortjournal` field, you could define a `\bibcheck` like this:

\begin{verbatim}
\defbibcheck{shortjournal}{% 
  \iffieldundef{shortjournal} 
  \iffieldundef{journal} 
  \iffieldundef{\strfield{shortjournal}→\strfield{journal}} 
    \skipentry 
  \fi}
\end{verbatim}

3.7.4 Bibliography Sections

The `\refsection` environment is used in the document body to mark a reference section. This environment is useful if you want separate, independent bibliographies and bibliography lists in each chapter, section, or any other part of a document. Within a reference section, all cited works are assigned labels which are local to the environment. Technically, reference sections are completely independent from document divisions such as `\chapter` and `\section` even though they will most likely be used per chapter or section. See the `\refsection` package option in § 3.1.2.1 for a way to automate this. Also see § 3.12.3 for usage examples.

\begin{verbatim}
\begin{refsection}[\resource, ...]
\end{refsection}
\end{verbatim}

The optional argument is a comma-separated list of resources specific to the reference section. If the argument is omitted, the reference section will use the default re-
source list, as specified with `\addbibresource` in the preamble. If the argument is provided, it replaces the default resource list. Global resources specified with `\addglobalbib` are always considered. `refsection` environments may not be nested, but you may use `refsegment` environments within a `refsection` to subdivide it into segments. Use the `section` option of `\printbibliography` to select a section when printing the bibliography, and the corresponding option of `\printbiblist` when printing bibliography lists. Bibliography sections are numbered starting at 1. The number of the current section is also written to the transcript file. All citations given outside a `refsection` environment are assigned to section 0. If `\printbibliography` is used within a `refsection`, it will automatically select the current section. The `section` option is not required in this case. This also applies to `\printbiblist`.

\newrefsection{⟨resource, …⟩}

This command is similar to the `refsection` environment except that it is a stand-alone command rather than an environment. It automatically ends the previous reference section (if any) and immediately starts a new one. Note that the reference section started by the last `\newrefsection` command in the document will extend to the very end of the document. Use `\endrefsection` if you want to terminate it earlier.

### 3.7.5 Bibliography Segments

The `refsegment` environment is used in the document body to mark a reference segment. This environment is useful if you want one global bibliography which is subdivided by chapter, section, or any other part of the document. Technically, reference segments are completely independent from document divisions such as `\chapter` and `\section` even though they will typically be used per chapter or section. See the `refsegment` package option in §3.1.2.1 for a way to automate this. Also see §3.12.3 for usage examples.

\begin{refsegment}
\end{refsegment}

The difference between a `refsection` and a `refsegment` environment is that the former creates labels which are local to the environment whereas the latter provides a target for the `segment` filter of `\printbibliography` without affecting the labels. They will be unique across the entire document. `refsegment` environments may not be nested, but you may use them in conjunction with `refsection` to subdivide a reference section into segments. In this case, the segments are local to the enclosing `refsection` environment. Use the `segment` option of `\printbibliography` to select a segment when printing the bibliography. Within a section, the reference segments are numbered starting at 1 and the number of the current segment will be written to the transcript file. All citations given outside a `refsegment` environment are assigned to segment 0. In contrast to the `refsection` environment, the current segment is not selected automatically if `\printbibliography` is used within a `refsegment` environment.

\newrefsegment

This command is similar to the `refsegment` environment except that it is a stand-alone command rather than an environment. It automatically ends the previous reference segment (if any) and immediately starts a new one. Note that the reference segment started by the last `\newrefsegment` command will extend to the end of the document. Use `\endrefsegment` if you want to terminate it earlier.
3.7.6 Bibliography Categories

Bibliography categories allow you to split the bibliography into multiple parts dedicated to different topics or different types of references, for example primary and secondary sources. See § 3.12.4 for usage examples.

\DeclareBibliographyCategory{⟨category⟩}

Declares a new ⟨category⟩, to be used in conjunction with \addtocategory and the category and notcategory filters of \printbibliography. This command is used in the document preamble.

\addtocategory{⟨category⟩}{⟨key⟩}

Assigns a ⟨key⟩ to a ⟨category⟩, to be used in conjunction with the category and notcategory filters of \printbibliography. This command may be used in the preamble and in the document body. The ⟨key⟩ may be a single entry key or a comma-separated list of keys. The assignment is global.

3.7.7 Bibliography Headings and Environments

\defbibenvironment{⟨name⟩}{⟨begin code⟩}{⟨end code⟩}{⟨item code⟩}

This command defines bibliography environments. The ⟨name⟩ is an identifier passed to the env option of \printbibliography and \printbiblist when selecting the environment. The ⟨begin code⟩ is LaTeX code to be executed at the beginning of the environment; the ⟨end code⟩ is executed at the end of the environment; the ⟨item code⟩ is code to be executed at the beginning of each entry in the bibliography or a bibliography list. Here is an example of a definition based on the standard LaTeX list environment:

\defbibenvironment{bibliography}{\list{}{\setlength{\leftmargin}{\bibhang}\setlength{\itemindent}{-\leftmargin}\setlength{\itemsep}{\bibitemsep}\setlength{\parsep}{\bibparsep}}}{\endlist}{\item}

As seen in the above example, usage of \defbibenvironment is roughly similar to \newenvironment except that there is an additional mandatory argument for the ⟨item code⟩.

\defbibheading{⟨name⟩}{⟨title⟩}{⟨code⟩}

This command defines bibliography headings. The ⟨name⟩ is an identifier to be passed to the heading option of \printbibliography or \printbiblist when selecting the heading. The ⟨code⟩ should be LaTeX code generating a fully-fledged heading, including page headers and an entry in the table of contents, if desired. If \printbibliography or \printbiblist are invoked with a title option, the title will be passed to the heading definition as #1. If not, the default title specified by the optional ⟨title⟩ argument is passed as #1 instead. The ⟨title⟩ argument will typically be \bibname, \refname, or \biblistname.
(see § 4.9.2.1). This command is often needed after changes to document headers in the preamble. Here is an example of a simple heading definition:

```
\defbibheading{bibliography}{\bibname}{\chapter*{#1}\markboth{#1}{#1}}
```

The following headings, which are intended for use with \printbibliography and \printbibheading, are predefined:

**bibliography**

This is the default heading used by \printbibliography if the heading option is not given. Its default definition depends on the document class. If the class provides a \chapter command, the heading is similar to the bibliography heading of the standard LaTeX book class, i.e., it uses \chapter* to create an unnumbered chapter heading which is not included in the table of contents. If there is no \chapter command, it is similar to the bibliography heading of the standard LaTeX article class, i.e., it uses \section* to create an unnumbered section heading which is not included in the table of contents. The string used in the heading also depends on the document class. With book-like classes the localisation string bibliography is used, with other classes it is references (see § 4.9.2). See also §§ 3.13.1 and 3.13.2 for class-specific hints.

**subbibliography**

Similar to bibliography but one sectioning level lower. This heading definition uses \section* instead of \chapter* with a book-like class and \subsection* instead of \section* otherwise.

**bibintoc**

Similar to bibliography above but adds an entry to the table of contents.

**subbibintoc**

Similar to subbibliography above but adds an entry to the table of contents.

**bibnumbered**

Similar to bibliography above but uses \chapter or \section to create a numbered heading which is also added to the table of contents.

**subbibnumbered**

Similar to subbibliography above but uses \section or \subsection to create a numbered heading which is also added to the table of contents.

**none**

A blank heading definition. Use this to suppress the heading.

The following headings intended for use with \printbiblist are predefined:
biblist

This is the default heading used by \printbiblist if the heading option is not given. It is similar to bibliography above except that it uses the localisation string shorthands instead of bibliography or references (see § 4.9.2). See also §§ 3.13.1 and 3.13.2 for class-specific hints.

biblistintoc

Similar to biblist above but adds an entry to the table of contents.

biblistnumbered

Similar to biblist above but uses \chapter or \section to create a numbered heading which is also added to the table of contents.

3.7.8 Bibliography Notes

\defbibnote\{\langle name\rangle\}\{\langle text\rangle\}

Defines the bibliography note \langle name\rangle, to be used via the prenote and postnote options of \printbibliography and \printbiblist. The \langle text\rangle may be any arbitrary piece of text, possibly spanning several paragraphs and containing font declarations. Also see § 3.13.6.

3.7.9 Bibliography Filters and Checks

\defbibfilter\{\langle name\rangle\}\{\langle expression\rangle\}

Defines the custom bibliography filter \langle name\rangle, to be used via the filter option of \printbibliography. The \langle expression\rangle is a complex test based on the logical operators and, or, not, the group separator (\ldots), and the following atomic tests:

segment=\langle integer\rangle

Matches all entries cited in reference segment \langle integer\rangle.

type=\langle entrytype\rangle

Matches all entries whose entry type is \langle entrytype\rangle.

subtype=\langle subtype\rangle

Matches all entries whose entrysubtype is \langle subtype\rangle.

keyword=\langle keyword\rangle

Matches all entries whose keywords field includes \langle keyword\rangle. If the \langle keyword\rangle contains spaces, it needs to be wrapped in braces.

category=\langle category\rangle

Matches all entries assigned to \langle category\rangle with \addtocategory.

Here is an example of a filter expression:
This filter will match all entries whose entry type is either @book or @inbook and whose keywords field includes the keyword 'abc' but not 'x y z'. As seen in the above example, all elements are separated by whitespace (spaces, tabs, or line endings). There is no spacing around the equal sign. The logical operators are evaluated with the \ifboolexpr command from the etoolbox package. See the etoolbox manual for details about the syntax. The syntax of the \ifthenelse command from the ifthen package, which has been employed in older versions of biblatex, is still supported. This is the same test using ifthen-like syntax:

\defbibfilter{example}{% 
  ( \type{book} or \type{inbook} ) 
  and \keyword{abc} 
  and not \keyword{x y z} } 

Note that custom filters are local to the reference section in which they are used. Use the section filter of \printbibliography to select a different section. This is not possible from within a custom filter.

\defbibcheck{⟨name⟩}{⟨code⟩}

Defines the custom bibliography filter ⟨name⟩, to be used via the check option of \printbibliography. \defbibcheck is similar in concept to \defbibfilter but much more low-level. Rather than a high-level expression, the ⟨code⟩ is LaTeX code, much like the code used in driver definitions, which may perform arbitrary tests to decide whether or not a given entry is to be printed. The bibliographic data of the respective entry is available when the ⟨code⟩ is executed. Issuing the command \skipentry in the ⟨code⟩ will cause the current entry to be skipped. For example, the following filter will only output entries with an abstract field:

\defbibcheck{abstract}{% \iffieldundef{abstract}{\skipentry}{}} ... \printbibliography[check=abstract]

The following check will exclude all entries published before the year 2000:

\defbibcheck{recent}{% \iffieldint{year} \{\ifnumless{\thefield{year}}{2000} \{\skipentry\} \} \{\skipentry\} }
See the author guide, in particular §§ 4.6.2 and 4.6.3, for further details.

3.7.10 Reference Contexts

References in a bibliography are cited and printed in a ‘context’. The context determines the data which is actually used to cite or provide bibliographic data for an entry. A context consists of the following information (the ‘context’ concept is designed for future extensibility):

- A sorting scheme
- A scheme for constructing the sorting keys for names
- A string prefix for citation schemes which use alphabetic or numeric labels

The point of bibliography contexts is twofold. Firstly, they are used to set options which influence a printed bibliography and secondly to influence the data printed by citation commands. The former use is the most common when one needs to print more than one bibliography list with different, for example, sorting.

\usepackage[sorting=nyt]{biblatex}
\begin{document}
\cite{one}
\cite{two}
\printbibliography
\newrefcontext[sorting=ydnt]
\printbibliography

Here we print two bibliographies, one with the default ‘nyt’ sorting scheme and one with the ‘ydnt’ sorting scheme.

To demonstrate the second type of use of bibliography contexts, we have to understand that the actual data for an entry can vary depending on the context. This is most obvious in the case of the extra fields like extrayear which are generated by the backend according to the order of entries after sorting so that they come out in the expected ‘a, b, c’ order. This clearly shows that the data in an entry can be different between sorting schemes. If a document contains more than one bibliography list with different sorting schemes, it can happen then that the .bbl contains sorting lists with the same entry but containing different data (a different value for extrayear, for example). The purpose of bibliography contexts is to encapsulate things inside a context so that biblatex can use the correct entry data. An example is printing a bibliography list with a different sorting order to the global sorting order where the extra fields are different for the same entry between sorting lists:

\usepackage[sorting=nyt,style=authoryear]{biblatex}
\DeclareSortingScheme{yntd}{
  \sort{
    \field[strside=left,strwidth=4]{sortyear}
    \field[strside=left,strwidth=4]{year}
    \literal{9999}
  }
  \sort{

88
Here, the second use of the citations, along with the `\printbibliography` command will use data from the context of the custom ‘yntd’ sorting scheme which may well be different from the data associated with the default ‘nyt’ scheme. That is, the citation labels (in an author/year style which uses `extrayear`) may be different for the exact same entries between different bibliography contexts and so the citations themselves may look different.

Reference contexts can be declared with `\DeclareRefcontext` and referred to by name, see below.

By default, data for a citation is drawn from the reference context of the last bibliography in which it was printed. For example:

```
\DeclareRefcontext{ap}{labelprefix=A}
\begin{document}
\cite{book, article, misc}
\printbibliography[type=book]
\newrefcontext{ap}
\printbibliography[type=article]
\newrefcontext[sorting=ydnt]
\printbibliography[type=misc]
\end{document}
```

This example also shows the declaration and use of a named reference context. Assuming the entrykeys are indicative of their entrytypes, this is the default situation for the citations which corresponds to what users normally expect:

- The citation of entry `book` would draw its data from the global reference context, because the last bibliography in which it was printed was the one in the global reference context.
- The citation of entry article would draw its data from reference context with labelprefix=A and would therefore have a 'A' prefix when cited.

- The citation of entry misc would draw its data from the reference context with sorting=ydnt

In cases where the user has entries which occur in multiple bibliographies in different forms or with potentially different labels (in a numeric scheme with different labelprefix values for example), it may be necessary to tell biblatex from which reference context you wish to draw the citation information. As shown above this can be done by explicitly putting citations inside reference contexts. This can be onerous in a large document and so there is specific functionality for assigning citations to reference contexts programatically, see the \assignrefcontext* macros below.

\DeclareRefcontext\{⟨name⟩\}{⟨key=value, …⟩}

Declares a named reference context with name ⟨name⟩. The ⟨key=value⟩ options define the context attributes. All context attributes are optional and default to the global settings if absent. The valid options are:

- **sorting=⟨name⟩**
  Specify a sorting scheme defined previously with \DeclareSortingScheme. This scheme is used to determine which data to retrieve and/or print for an entry in the commands inside the context.

- **sortingnamekeyscheme=⟨name⟩**
  Specify a sorting name key scheme defined previously with \DeclareSortingNamekeyScheme. This scheme is used to construct sorting keys for names inside the context.

- **labelprefix=⟨string⟩**
  This option applies to numerical citation/bibliography styles only and requires that the defernumbers option from § 3.1.2.1 be enabled globally. Setting this option will implicitly enable resetnumbers for the any \printbibliography in the scope of the context (unless overridden by a user-specified value for resetnumbers). The option assigns the ⟨string⟩ as a prefix to all entries in the reference context. For example, if the ⟨string⟩ is A, the numerical labels printed will be [A1], [A2], [A3], etc. This is useful for subdivided numerical bibliographies where each subbibliography uses a different prefix. The ⟨string⟩ is available to styles in the labelprefix field of all affected entries. See § 4.2.4.2 for details.

\begin{refcontext}\{⟨key=value, …⟩\}\{⟨name⟩\}\end{refcontext}

Wraps a reference context environment. The possible ⟨key=value⟩ optional arguments are as for \DeclareRefcontext and override options given for the named reference context ⟨name⟩. ⟨name⟩ can also be omitted as {} or by omitting even the empty braces\textsuperscript{23}.

The refcontext environment cannot be nested and biblatex will generate an error if you try to do so.

\textsuperscript{23}This slightly odd syntax possibility is a result of backwards compatibility with biblatex < 3.5
This command is similar to the \texttt{refcontext} environment except that it is a stand-alone command rather than an environment. It automatically ends any previous reference context section begun with \texttt{newrefcontext} (if any) and immediately starts a new one. Note that the context section started by the last \texttt{newrefcontext} command in the document will extend to the very end of the document. Use \texttt{endrefcontext} if you want to terminate it earlier.

At the beginning of the document, there is always a global context containing global settings for each of the reference context options. Here is an example summarising the reference contexts with various settings:

\begin{verbatim}
\usepackage[sorting=nty]{biblatex}
\DeclareRefcontext{testrc}{sorting=nyt}
% Global reference context:
% sorting=nty
% sortingnamekeyscheme=global
% labelprefix=

\begin{document}
\begin{refcontext}{testrc}
% reference context:
% sorting=nyt
% sortingnamekeyscheme=global
% labelprefix=
\end{refcontext}

\begin{refcontext}[labelprefix=A]{testrc}
% reference context:
% sorting=nyt
% sortingnamekeyscheme=global
% labelprefix=A
\end{refcontext}

\begin{refcontext}[sorting=ydnt,labelprefix=A]
% reference context:
% sorting=ydnt
% sortingnamekeyscheme=global
% labelprefix=A
\end{refcontext}

\newrefcontext[\textit{labelprefix=B}]
% reference context:
% sorting=nty
% sortingnamekeyscheme=global
% labelprefix=B
\end{refcontext}
\end{document}
\end{verbatim}
These commands automate putting citations into refcontexts when the default behaviour is not sufficient. The default behaviour is that the data for a citation is drawn from the refcontext of the last bibliography in which it was printed. For citations that are used in some way but not printed in a bibliography or bibliography list, they default to drawing their data from the global refcontext established at the beginning of the document. To override this behaviour, instead of manually wrapping citation commands in \refcontext environments, which might be error-prone and tedious, you can register a comma-separated list of \langle keywords\rangle, \langle categories\rangle or \langle entrykeys\rangle which, respectively, make the entries with any of the specified keywords, entries in any of the specified categories (see §3.12.4) or entries with any of the specified citation keys draw their data from a particular refcontext specified by the \langle refcontext key/values\rangle which are parsed as the per the corresponding refcontext options. Such refcontext auto-assignments are specific to the current refsection. You may specify the same citation key in any of these commands but be aware that assignment is done in the order \langle keywords\rangle, \langle categories\rangle, \langle entrykeys\rangle with the later specifications overriding the earlier. \assignrefcontextentries accepts a single asterisk instead of a list of entrykeys which allows the assignment of all keys in a refsection to a refcontext with having to explicitly list them. An example:

\assignrefcontextentries\[labelprefix=A\]{key2}
\cite{key1}
\begin{refcontext}\[labelprefix=B\]
\cite{key2}
\end{refcontext}

Here, the data for the citation of key2 will be drawn from refcontext labelprefix=A and not labelprefix=B (resulting in a label with prefix 'A' and not 'B'). The starred versions do not override a local refcontext and so with:
the data for the citation of key2 will be drawn from refcontext labelprefix=B. Note that these commands are rarely necessary unless you have multiple bibliographies in which the same citations occur and biblatex cannot by default tell which bibliography list a citation should refer to. See the example file 94-labelprefix.tex for more details.

### 3.7.11 Dynamic Entry Sets

In addition to the @set entry type, biblatex also supports dynamic entry sets defined on a per-document/per-refsection basis. The following command, which may be used in the document preamble or the document body, defines the set ⟨key⟩:

```latex
\defbibentryset{⟨key⟩}{⟨key1,key2,key3, …⟩}
```

The ⟨key⟩ is the entry key of the set, which is used like any other entry key when referring to the set. The ⟨key⟩ must be unique and it must not conflict with any other entry key. The second argument is a comma-separated list of the entry keys which make up the set. \defbibentryset implies the equivalent of a \nocite command, i.e., all sets which are declared are also added to the bibliography. When declaring the same set more than once, only the first invocation of \defbibentryset will define the set. Subsequent definitions of the same ⟨key⟩ are ignored and work like \nocite(⟨key⟩). Dynamic entry sets defined in the document body are local to the enclosing refsection environment, if any. Otherwise, they are assigned to reference section 0. Those defined in the preamble are assigned to reference section 0. See §3.12.5 for further details.

### 3.8 Citation Commands

All citation commands generally take one mandatory and two optional arguments. The ⟨prenote⟩ is text to be printed at the beginning of the citation. This is usually a notice such as ‘see’ or ‘compare’. The ⟨postnote⟩ is text to be printed at the very end of the citation. This is usually a page number. If only one of these arguments is given, it is taken as a postnote. If you want to specify a prenote but no postnote, you need to leave the second optional argument empty, as in \cite[see][]{key}. The ⟨key⟩ argument to all citation commands is mandatory. This is the entry key or a comma-separated list of keys corresponding to the entry keys in the bib file. In sum, all basic citations commands listed further down have the following syntax:

```latex
\command[⟨prenote⟩][⟨postnote⟩]⟨(keys)⟩⟨(punctuation)⟩
```

If the autopunct package option from §3.1.2.1 is enabled, they will scan ahead for any ⟨punctuation⟩ immediately following their last argument. This is useful to avoid spurious punctuation marks after citations. This feature is configured with \DeclareAutoPunctuation, see §4.7.5 for details.

#### 3.8.1 Standard Commands

The following commands are defined by the citation style. Citation styles may provide any arbitrary number of specialized commands, but these are the standard commands typically provided by general-purpose styles.
These are the bare citation commands. They print the citation without any additions such as parentheses. The numeric and alphabetic styles still wrap the label in square brackets since the reference may be ambiguous otherwise. \Cite{} is similar to \cite{} but capitalizes the name prefix of the first name in the citation if the useprefix option is enabled, provided that there is a name prefix and the citation style prints any name at all.

\parencite{} and \Parencite{} use a format similar to \cite{} but enclose the entire citation in parentheses. The numeric and alphabetic styles use square brackets instead. \Parencite{} is similar to \parencite{} but capitalizes the name prefix of the first name in the citation if the useprefix option is enabled, provided that there is a name prefix and the citation style prints any name at all.

\footcite{} and \footcitetext{} use a format similar to \cite{} but put the entire citation in a footnote and add a period at the end. In the footnote, they automatically capitalize the name prefix of the first name if the useprefix option is enabled, provided that there is a name prefix and the citation style prints any name at all. \footcitetext{} differs from \footcite{} in that it uses \footnotetext{} instead of \footnote{}.

### 3.8.2 Style-specific Commands

The following additional citation commands are only provided by some of the citation styles which ship with this package.

\textcite{} and \Textcite{} use a format similar to \cite{} but are intended for use in the flow of text, replacing the subject of a sentence. They print the authors or editors followed by a citation label which is enclosed in parentheses. Depending on the citation style, the label may be a number, the year of publication, an abridged version of the title, or something else. The numeric and alphabetic styles use square brackets instead of parentheses. In the verbose styles, the label is provided in a footnote. Trailing punctuation is moved between the author or editor names and the footnote mark. \Textcite{} is similar to \textcite{} but capitalizes the name prefix of the first name in the citation if the useprefix option is enabled, provided that there is a name prefix.

\smartcite{} and \Smartcite{} are similar to \textcite{} but capitalizes the name prefix of the first name in the citation if the useprefix option is enabled, provided that there is a name prefix.

Like \parencite{} in a footnote and like \footcite{} in the body.

\cite{} is similar to the regular \cite{} command but merely prints the year or the title, respectively.
\parencite*[⟨prenote⟩][⟨postnote⟩]⟨key⟩

This command is provided by all author-year and author-title styles. It is similar to the regular \parencite command but merely prints the year or the title, respectively.

\supercite⟨key⟩

This command, which is only provided by the numeric styles, prints numeric citations as superscripts without brackets. It uses \supercitedelim instead of \multicitedelim as citation delimiter. Note that any ⟨prenote⟩ and ⟨postnote⟩ arguments are ignored. If they are given, \supercite will discard them and issue a warning message.

3.8.3 Qualified Citation Lists

This package supports a class of special citation commands called ‘multicite’ commands. The point of these commands is that their argument is a list of citations where each item forms a fully qualified citation with a pre- and/or postnote. This is particularly useful with parenthetical citations and citations given in footnotes. It is also possible to assign a pre- and/or postnote to the entire list. The multicite commands are built on top of backend commands like \parencite and \footcite. The citation style provides a multicite definition with \DeclareMultiCiteCommand (see §4.3.1). The following example illustrates the syntax of multicite commands:

\parencites[35]{key1}[88--120]{key2}[23]{key3}

The format of the arguments is similar to that of the regular citation commands, except that only one citation command is given. If only one optional argument is given for an item in the list, it is taken as a postnote. If you want to specify a prenote but no postnote, you need to leave the second optional argument of the respective item empty:

\parencites[35]{key1}[chapter 2 in][]{key2}[23]{key3}

In addition to that, the entire citation list may also have a pre- and/or postnote. The syntax of these global notes differs from other optional arguments in that they are given in parentheses rather than the usual brackets:

\parencites(and chapter 3)[35]{key1}[78]{key2}[23]{key3}
\parencites(Compare)()[35]{key1}[78]{key2}[23]{key3}
\parencites(See)(and the introduction)[35]{key1}[78]{key2}[23]{key3}

Note that the multicite commands keep on scanning for arguments until they encounter a token that is not the start of an optional or mandatory argument. If a left brace or bracket follows a multicite command, you need to mask it by adding \relax or a control space (a backslash followed by a space) after the last valid argument. This will cause the scanner to stop.

\parencites[35]{key1}[78]{key2}\relax [...]
\parencites[35]{key1}[78]{key2}\␣ {...}
By default, this package provides the following multicite commands which correspond to regular commands from §§3.8.1 and 3.8.2:

\cites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}
\Cites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}

The multicite version of \cite and \Cite, respectively.

\parencites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}
\Parenccites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}

The multicite version of \parencite and \Parenccite, respectively.

\footcites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}
\Footcites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}

The multicite version of \footcite and \Footcite, respectively.

\smartcites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}
\Smartcites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}

The multicite version of \smartcite and \Smartcite, respectively.

\textcites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}
\Textcites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}

The multicite version of \textcite and \Textcite, respectively.

\supercites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}
\Supercites{morren}{prenote}{postnote}{key}\ldots{prenote}{postnote}{key}

The multicite version of \supercite. This command is only provided by the numeric styles.

### 3.8.4 Style-independent Commands

Sometimes it is desirable to give the citations in the source file in a format that is not tied to a specific citation style and can be modified globally in the preamble. The format of the citations is easily changed by loading a different citation style. However, when using commands such as \parencite or \footcite, the way the citations are integrated with the text is still effectively hard-coded. The idea behind the \autocite command is to provide higher-level citation markup which makes global switching from inline citations to citations given in footnotes (or as superscripts) possible. The \autocite command is built on top of backend commands like \parencite and \footcite. The citation style provides an autocite definition with \DeclareAutoCiteCommand (see §4.3.1). This definition may be activated with the autocite package option from §3.1.2.1. The citation style will usually initialize this package option to a value which is suitable for the style, see §3.3.1 for details. Note that there are certain limits to high-level citation markup. For example, inline author-year citation schemes often integrate citations so tightly with the text that it is virtually impossible to automatically convert them to footnotes. The \autocite command is only applicable in cases in which you would normally use \parencite or \footcite (or \supercite, with a numeric style). The citations should be given at the end of a sentence or a partial sentence, immediately preceding the terminal punctuation mark, and they should not be a part of the sentence in a grammatical sense (like \textcite, for example).
In contrast to other citation commands, the \autocite command does not only scan ahead for punctuation marks following its last argument to avoid double punctuation marks, it actually moves them around if required. For example, with autocite=footnote, a trailing punctuation mark will be moved such that the footnote mark is printed after the punctuation. \Autocite is similar to \autocite but capitalizes the name prefix of the first name in the citation if the useprefix option is enabled, provided that there is a name prefix and the citation style prints any name at all.

The starred variants of \autocite do not behave differently from the regular ones. The asterisk is simply passed on to the backend command. For example, if \autocite is configured to use \parencite, then \autocite* will execute \parencite*.

This is the multicite version of \autocite. It also detects and moves punctuation if required. Note that there is no starred variant. \Autocites is similar to \autocites but capitalizes the name prefix of the first name in the citation if the useprefix option is enabled, provided that there is a name prefix and the citation style prints any name at all.

\textbf{3.8.5 Text Commands}

The following commands are provided by the core of \biblatex. They are intended for use in the flow of text. Note that all text commands are excluded from citation tracking.

These commands print the authors. Strictly speaking, it prints the \labelname list, which may be the author, the editor, or the translator. \Citeauthor is similar to \citeauthor but capitalizes the name prefix of the first name in the citation if the useprefix option is enabled, provided that there is a name prefix. The starred variants effectively force maxcitenames to 1 for just this command on so only print the first name in the labelname list (potentially followed by the “et al” string if there are more names). This allows more natural textual flow when referring to a paper in the singular when otherwise \citeauthor would generate a (naturally plural) list of names.
This command prints the title. It will use the abridged title in the \texttt{shorttitle} field, if available. Otherwise it falls back to the full title found in the \texttt{title} field. The starred variant always prints the full title.

This command prints the year (\texttt{year} field or year component of \texttt{date}). The starred variant includes the \texttt{extrayear} information, if any.

This command prints the full date (\texttt{date} or \texttt{year}). The starred variant includes the \texttt{extrayear} information, if any.

This command prints the \texttt{url} field.

This command wraps the \texttt{text} in context sensitive parentheses.

This command wraps the \texttt{text} in context sensitive brackets.

\subsection*{3.8.6 Special Commands}

The following special commands are also provided by the core of \texttt{biblatex}.

This command is similar to the standard LaTeX \texttt{nocite} command. It adds the \texttt{key} to the bibliography without printing a citation. If the \texttt{key} is an asterisk, all entries available in the in-scope bibliography datasource(s) are added to the bibliography. Like all other citation commands, \texttt{nocite} commands in the document body are local to the enclosing \texttt{refsection} environment, if any. In contrast to standard LaTeX, \texttt{nocite} may also be used in the document preamble. In this case, the references are assigned to reference section 0.

This command uses the bibliography driver for the respective entry type to create a full citation similar to the bibliography entry. It is thus related to the bibliography style rather than the citation style.

Similar to \texttt{fullcite} but puts the entire citation in a footnote and adds a period at the end.
\volcite\{\prenote\} \{\volume\} \{\page\} \{\key\}
\Volcite\{\prenote\} \{\volume\} \{\page\} \{\key\}

These commands are similar to \cite and \Cite but intended for references to multi-volume works which are cited by volume and page number. Instead of the \{postnote\}, they take a mandatory \{volume\} and an optional \{page\} argument. Since they merely compose the postnote and pass it to the \cite command provided by the citation style as a \{postnote\} argument, these commands are style independent. The format of the volume portion is controlled by the field formatting directive \volcitevolume, the format of the page/text portion is controlled by the field formatting directive \volcitepages (§ 4.10.4). The delimiter printed between the volume portion and the page/text portion may be modified by redefining the macro \volcite delim (§ 4.10.1).

\volcites\{\multiprenote\} \{\multipostnote\} \{\prenote\} \{\volume\} \{\page\} \{\key\}
\Volcites\{\multiprenote\} \{\multipostnote\} \{\prenote\} \{\volume\} \{\page\} \{\key\}

... \{\prenote\} \{\volume\} \{\page\} \{\key\}
\Volcites\{\multiprenote\} \{\multipostnote\} \{\prenote\} \{\volume\} \{\page\} \{\key\}

The multicite version of \volcite and \Volcite, respectively.

\pvolcite\{\prenote\} \{\volume\} \{\page\} \{\key\}
\Pvolcite\{\prenote\} \{\volume\} \{\page\} \{\key\}

Similar to \volcite but based on \parencite.

\pvolcites\{\multiprenote\} \{\multipostnote\} \{\prenote\} \{\volume\} \{\page\} \{\key\}
\Pvolcites\{\multiprenote\} \{\multipostnote\} \{\prenote\} \{\volume\} \{\page\} \{\key\}

... \{\prenote\} \{\volume\} \{\page\} \{\key\}
\Pvolcites\{\multiprenote\} \{\multipostnote\} \{\prenote\} \{\volume\} \{\page\} \{\key\}

... \{\prenote\} \{\volume\} \{\page\} \{\key\}

The multicite version of \pvolcite and \Pvolcite, respectively.

\fvolcite\{\prenote\} \{\volume\} \{\page\} \{\key\}
\ftvolcite\{\prenote\} \{\volume\} \{\page\} \{\key\}

Similar to \volcite but based on \footcite and \footcitetext, respectively.

\fvolcites\{\multiprenote\} \{\multipostnote\} \{\prenote\} \{\volume\} \{\page\} \{\key\}
\Fvolcites\{\multiprenote\} \{\multipostnote\} \{\prenote\} \{\volume\} \{\page\} \{\key\}

... \{\prenote\} \{\volume\} \{\page\} \{\key\}
\Fvolcites\{\multiprenote\} \{\multipostnote\} \{\prenote\} \{\volume\} \{\page\} \{\key\}

... \{\prenote\} \{\volume\} \{\page\} \{\key\}

The multicite version of \fvolcite and \Fvolcite, respectively.

\svolcite\{\prenote\} \{\volume\} \{\page\} \{\key\}
\Svolcite\{\prenote\} \{\volume\} \{\page\} \{\key\}

Similar to \volcite but based on \smartcite.
Similar to \textcite but based on \textcite.

\textcites{\text reachable format}
\textcites{\text reachable format}
\textcites{\text reachable format}
\textcites{\text reachable format}
\textcites{\text reachable format}
\textcites{\text reachable format}

The multicite version of \textcite and \textcite, respectively.

\autocite{\text reachable format}
\autocite{\text reachable format}
\autocite{\text reachable format}
\autocite{\text reachable format}

The multicite version of \autocite and \autocite, respectively.

\notecite{\text reachable format}
\notecite{\text reachable format}
\notecite{\text reachable format}
\notecite{\text reachable format}

These commands print the \text cite and \text cite arguments but no citation. Instead, a \text cite command is issued for every \text cite. This may be useful for authors who incorporate implicit citations in their writing, only giving information not mentioned before in the running text, but who still want to take advantage of the automatic \text cite formatting and the implicit \text cite function. This is a generic, style-independent citation command. Special citation styles may provide smarter facilities for the same purpose. The capitalized version forces capitalization (note that this is only applicable if the note starts with a command which is sensitive to \text cite's punctuation tracker).

\notecite{\text reachable format}
\notecite{\text reachable format}
\notecite{\text reachable format}

Similar to \notecite but the notes are printed in parentheses.

\footnotecite{\text reachable format}

Similar to \notecite but the notes are printed in a footnote.

3.8.7 Low-level Commands

The following commands are also provided by the core of \text cite. They grant access to all lists and fields at a lower level.

\typename{\text reachable format}

The \text format is a formatting directive defined with \text cite. Formatting directives are discussed in § 4.4.2. If this optional argument is omitted, this command falls back to the format \text cite. The last argument is the name of a \text cite, in the sense explained in § 2.2.
\citelist{⟨prenote⟩}{⟨postnote⟩}{⟨key⟩}{⟨format⟩}{⟨literal list⟩}

The ⟨format⟩ is a formatting directive defined with \DeclareListFormat. Formatting directives are discussed in §4.4.2. If this optional argument is omitted, this command falls back to the format citelist. The last argument is the name of a ⟨literal list⟩, in the sense explained in §2.2.

\citefield{⟨prenote⟩}{⟨postnote⟩}{⟨key⟩}{⟨format⟩}{⟨field⟩}

The ⟨format⟩ is a formatting directive defined with \DeclareFieldFormat. Formatting directives are discussed in §4.4.2. If this optional argument is omitted, this command falls back to the format citefield. The last argument is the name of a ⟨field⟩, in the sense explained in §2.2.

3.8.8 Miscellaneous Commands

The commands in this section are little helpers related to citations.

\citereset This command resets the citation style. This may be useful if the style replaces repeated citations with abbreviations like ibidem, idem, op. cit., etc. and you want to force a full citation at the beginning of a new chapter, section, or some other location. The command executes a style specific initialization hook defined with the \InitializeCitationStyle command from §4.3.1. It also resets the internal citation trackers of this package. The reset will affect the \ifciteseen, \ifentryseen, \ifciterebid, and \ifciteidem tests discussed in §4.6.2. When used inside a refsection environment, the reset of the citation tracker is local to the current refsection environment. Also see the citereset package option in §3.1.2.1.

\citereset* Similar to \citereset but only executes the style’s initialization hook, without resetting the internal citation trackers.

\mancite Use this command to mark manually inserted citations if you mix automatically generated and manual citations. This is particularly useful if the citation style replaces repeated citations by an abbreviation like ibidem which may get ambiguous or misleading otherwise. Always use \mancite in the same context as the manual citation, e.g., if the citation is given in a footnote, include \mancite in the footnote. The \mancite command executes a style specific reset hook defined with the \OnManualCitation command from §4.3.1. It also resets the internal ‘ibidem’ and ‘idem’ trackers of this package. The reset will affect the \ifciterebid and \ifciteidem tests discussed in §4.6.2.

\pno This command forces a single page prefix in the ⟨postnote⟩ argument to a citation command. See §3.13.3 for further details and usage instructions. Note that this command is only available locally in citations and the bibliography.

\ppno Similar to \pno but forces a range prefix. See §3.13.3 for further details and usage instructions. Note that this command is only available locally in citations and the bibliography.

\nopp Similar to \pno but suppresses all prefixes. See §3.13.3 for further details and usage instructions. Note that this command is only available locally in citations and the bibliography.
In the \textit{postnote} argument to a citation command, this command indicates a range of two pages where only the starting page is given. See §3.13.3 for further details and usage instructions. The suffix printed is the localisation string \textit{sequens}, see §4.9.2. The spacing inserted between the suffix and the page number may be modified by redefining the macro \texttt{\sqspace}. The default is an unbreakable interword space. Note that this command is only available locally in citations and the bibliography.

Similar to \sq but indicates an open-ended page range. See §3.13.3 for further details and usage instructions. The suffix printed is the localisation string \textit{sequentes}, see §4.9.2. This command is only available locally in citations and the bibliography.

This command prints an integer as an uppercase Roman numeral. The formatting applied to the numeral may be modified by redefining the macro \texttt{\RNfont}.

Similar to \RN but prints a lowercase Roman numeral. The formatting applied to the numeral may be modified by redefining the macro \texttt{\Rnfont}.

### 3.8.9 \texttt{natbib} Compatibility Commands

The \texttt{natbib} package option loads a \texttt{natbib} compatibility module. The module defines aliases for the citation commands provided by the \texttt{natbib} package. This includes aliases for the core citation commands \texttt{\cite} and \texttt{\citep} as well as the variants \texttt{\citealt} and \texttt{\citealp}. The starred variants of these commands, which print the full author list, are also supported. The \texttt{\cite} command, which is handled in a particular way by \texttt{natbib}, is not treated in a special way. The text commands (\texttt{\citeauthor}, \texttt{\citeyear}, etc.) are also supported, as are all commands which capitalize the name prefix (\texttt{\Citet}, \texttt{\Citep}, \texttt{\Citeauthor}, etc.). Aliasing with \texttt{\defcitealias}, \texttt{\citealtnote}, and \texttt{\citepalias} is possible as well. Note that the compatibility commands will not emulate the citation format of the \texttt{natbib} package. They merely alias \texttt{natbib}'s commands to functionally equivalent facilities of the \texttt{biblatex} package. The citation format depends on the main citation style. However, the compatibility style will adapt \texttt{nameyeardelim} to match the default style of the \texttt{natbib} package.

### 3.8.10 \texttt{mcite}-like Citation Commands

The \texttt{mcite} package option loads a special citation module which provides \texttt{mcite/mciteplus}-like citation commands. Strictly speaking, what the module provides are wrappers for the commands of the main citation style. For example, the following command:

\begin{verbatim}
\mcite{key1,setA,*keyA1,*keyA2,*keyA3,key2,setB,*keyB1 ↪ ,*keyB2,*keyB3}
\end{verbatim}

is essentially equivalent to this:

\begin{verbatim}
\defbibentryset{setA}{keyA1,keyA2,keyA3}\
\defbibentryset{setB}{keyB1,keyB2,keyB3}\
\cite{key1,setA,key2,setB}
\end{verbatim}


<table>
<thead>
<tr>
<th><strong>Standard Command</strong></th>
<th><strong>mcite-like Command</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>\cite</td>
<td>\mcite</td>
</tr>
<tr>
<td>\Cite</td>
<td>\Mcite</td>
</tr>
<tr>
<td>\parencite</td>
<td>\mparencite</td>
</tr>
<tr>
<td>\Parencite</td>
<td>\Mparencite</td>
</tr>
<tr>
<td>\footcite</td>
<td>\mfootcite</td>
</tr>
<tr>
<td>\footcitetext</td>
<td>\mfootcitetext</td>
</tr>
<tr>
<td>\textcite</td>
<td>\mtextcite</td>
</tr>
<tr>
<td>\Textcite</td>
<td>\Mtextcite</td>
</tr>
<tr>
<td>\supercite</td>
<td>\msupercite</td>
</tr>
</tbody>
</table>

Table 8: mcite-like commands

The \mcite command will work with any style since the \cite backend command is controlled by the main citation style as usual. The mcite module provides wrappers for the standard commands in §§3.8.1 and 3.8.2. See table 9 for an overview. Pre and postnotes as well as starred variants of all commands are also supported. The parameters will be passed to the backend command. For example:

\mcite*[pre][post]{setA,*keyA1,*keyA2,*keyA3}

will execute:

\defbibentryset{setA}{keyA1,keyA2,keyA3}\cite*[pre][post]{setA}

Note that the mcite module is not a compatibility module. It provides commands which are very similar but not identical in syntax and function to mcite’s commands. When migrating from mcite/mciteplus to biblatex, legacy files must be updated. With mcite, the first member of the citation group is also the identifier of the group as a whole. Borrowing an example from the mcite manual, this group:

\cite{glashow,*salam,*weinberg}

consists of three entries and the entry key of the first one also serves as identifier of the entire group. In contrast to that, a biblatex entry set is an entity in its own right. Therefore, it requires a unique entry key which is assigned to the set as it is defined:

\mcite{set1,*glashow,*salam,*weinberg}

Once defined, an entry set is handled like any regular entry in a bib file. When using one of the numeric styles which ship with biblatex and activating its subentry option, it is even possible to refer to set members. See table 9 for some examples. Restating the original definition of the set is redundant, but permissible. In contrast to mciteplus, however, restating a part of the original definition is invalid. Use the entry key of the set instead.

3.9 Localization Commands

The biblatex package provides translations for key terms such as ‘edition’ or ‘volume’ as well as definitions for language specific features such as the date format
and ordinals. These definitions, which are loaded automatically, may be modified or extended in the document preamble or the configuration file with the commands introduced in this section.

\DefineBibliographyStrings{⟨language⟩}{⟨definitions⟩}

This command is used to define localisation strings. The ⟨language⟩ must be a language name known to the babel/polyglossia packages, i.e., one of the identifiers listed in table 2 on page 26. The ⟨definitions⟩ are ⟨key⟩=⟨value⟩ pairs which assign an expression to an identifier:

\DefineBibliographyStrings{american}{
  bibliography = {Bibliography},
  shorthands = {Abbreviations},
  editor = {editor},
  editors = {editors},
}

A complete list of all keys supported by default is given in §4.9.2. Note that all expressions should be capitalized as they usually are when used in the middle of a sentence. The \biblatex package will automatically capitalize the first word when required at the beginning of a sentence. Expressions intended for use in headings should be capitalized in a way that is suitable for titling. In contrast to \DeclareBibliographyStrings, \DefineBibliographyStrings overrides both the full and the abbreviated version of the string. See §4.9.1 for further details.

\DefineBibliographyExtras{⟨language⟩}{⟨code⟩}

This command is used to adapt language specific features such as the date format and ordinals. The ⟨language⟩ must be a language name known to the babel/polyglossia packages. The ⟨code⟩, which may be arbitrary LaTeX code, will usually consist of redefinitions of the formatting commands from §3.10.3.

\UndefineBibliographyExtras{⟨language⟩}{⟨code⟩}

This command is used to restore the original definition of any commands modified with \DefineBibliographyExtras. If a redefined command is included in §3.10.3, there is no need to restore its previous definition since these commands are adapted by all language modules anyway.
\DefineHyphenationExceptions{(language)}{(text)}

This is a LaTeX frontend to TeX’s \hyphenation command which defines hyphenation exceptions. The \textit{language} must be a language name known to the babel/polyglossia packages. The \textit{text} is a whitespace-separated list of words. Hyphenation points are marked with a dash:

\DefineHyphenationExceptions{american}{%
hyphenation exception
}\NewBibliographyString{(key)}

This command declares new localisation strings, i.e., it initializes a new \textit{key} to be used in the \textit{definitions} of \DefineBibliographyStrings. The \textit{key} argument may also be a comma-separated list of key names. The keys listed in §4.9.2 are defined by default.

### 3.10 Formatting Commands

The commands and facilities presented in this section may be used to adapt the format of citations and the bibliography.

#### 3.10.1 Generic Commands and Hooks

The commands in this section may be redefined with \renewcommand in the document preamble. Those marked as ‘Context Sensitive’ in the margin can also be customised with \DeclareDelimFormat and are printed with \printdelim (§3.10.2). Note that all commands starting with \mk... take one argument. All of these commands are defined in biblatex.def.

\bibsetup

Arbitrary code to be executed at the beginning of the bibliography, intended for commands which affect the layout of the bibliography.

\bibfont

Arbitrary code setting the font used in the bibliography. This is very similar to \bibsetup but intended for switching fonts.

\citesetup

Arbitrary code to be executed at the beginning of each citation command.

\newblockpunct

The separator inserted between ‘blocks’ in the sense explained in §4.7.1. The default definition is controlled by the package option block (see §3.1.2.1).

\newunitpunct

The separator inserted between ‘units’ in the sense explained in §4.7.1. This will usually be a period or a comma plus an interword space. The default definition is a period and a space.

\finentrypunct

The punctuation printed at the very end of every bibliography entry, usually a period. The default definition is a period.

\entrysetpunct

The punctuation printed between bibliography subentries of an entry set. The default definition is a semicolon and a space.
This delimiter controls the spacing between the elements which make up a name part. It is inserted automatically after the first name element if the element is less than three characters long and before the last element. The default definition is an interword space penalized by the value of the highnamepenalty counter (§ 3.10.4). Please refer to § 3.13.4 for further details.

This delimiter is inserted between the elements which make up a name part where \bibnamedelima does not apply. The default definition is an interword space penalized by the value of the lownamepenalty counter (§ 3.10.4). Please refer to § 3.13.4 for further details.

This delimiter controls the spacing between name parts. It is inserted between the name prefix and the last name if useprefix=true. The default definition is an interword space penalized by the value of the highnamepenalty counter (§ 3.10.4). Please refer to § 3.13.4 for further details.

This delimiter is inserted between all name parts where \bibnamedelimc does not apply. The default definition is an interword space penalized by the value of the lownamepenalty counter (§ 3.10.4). Please refer to § 3.13.4 for further details.

This delimiter replaces \bibnamedelima/b after initials. Note that this only applies to initials given as such in the bib file, not to the initials automatically generated by biblatex which use their own set of delimiters.

The punctuation inserted after initials unless \bibinithyphendelim applies. The default definition is a period (\adddot). Please refer to § 3.13.4 for further details.

The spacing inserted between multiple initials unless \bibinithyphendelim applies. The default definition is an unbreakable interword space. Please refer to § 3.13.4 for further details.

The punctuation inserted between the initials of hyphenated name parts, replacing \bibinitperiod and \bibinitdelim. The default definition is a period followed by an unbreakable hyphen. Please refer to § 3.13.4 for further details.

Replaces \bibnamedelima in the index.

Replaces \bibnamedelimb in the index.

Replaces \bibnamedelimc in the index.

Replaces \bibnamedelimd in the index.

Replaces \bibnamedelimi in the index.

Replaces \bibinitperiod in the index.

Replaces \bibinitdelim in the index.

Replaces \bibinithyphendelim in the index.
\revsdnamepunct The punctuation to be printed between the first and last name parts when a name is reversed. Here is an example showing a name with the default comma as \revsdnamedelim:

Jones, Edward

This command should be used with \bibnamedelim as a reversed-name separator in formatting directives for name lists. Please refer to § 3.13.4 for further details.

\bibnamedash The dash to be used as a replacement for recurrent authors or editors in the bibliography. The default is an ‘em’ or an ‘en’ dash, depending on the indentation of the list of references.

\labelnamepunct The separator printed after the name used for alphabetizing in the bibliography (author or editor, if the author field is undefined). With the default styles, this separator replaces \newunitpunct at this location. The default definition is \newunitpunct, i.e., it is not handled differently from regular unit punctuation.

\subtitlepunct The separator printed between the fields title and subtitle, booktitle and bookssubtitle, as well as maintitle and mainsubtitle. With the default styles, this separator replaces \newunitpunct at this location. The default definition is \newunitpunct, i.e., it is not handled differently from regular unit punctuation.

\intitlepunct The separator between the word “in” and the following title in entry types such as @article, @inbook, @incollection, etc. The default definition is a colon plus an interword space (e.g., “Article, in: Journal” or “Title, in: Book”). Note that this is the separator string, not only the punctuation mark. If you don’t want a colon after “in”, \intitlepunct should still insert a space.

\bibpagespunct The separator printed before the pages field. The default is a comma plus an interword space.

\bibpagerefpunct The separator printed before the pageref field. The default is an interword space.

\multinamedelim The delimiter printed between multiple items in a name list like author or editor if there are more than two names in the list. The default is a comma plus an interword space. See \finalnamedelim for an example.\footnote{Note that \multinamedelim is not used at all if there are only two names in the list. In this case, the default styles use the \finalnamedelim.}

\finalnamedelim The delimiter printed instead of \multinamedelim before the final name in a name list. The default is the localised term ‘and’, separated by interword spaces. Here is an example:

Michel Goossens, Frank Mittelbach and Alexander Samarin Edward Jones and Joe Williams

The comma in the first example is the \multinamedelim whereas the string ‘and’ in both examples is the \finalnamedelim. See also \finalandcomma in § 3.10.3.
An extra delimiter printed after the first name in a name list if the first name is reversed (only in lists with two names). The default is an empty string, i.e., no extra delimiter will be printed. Here is an example showing a name list with a comma as \revsdnamedelim:

Jones, Edward, and Joe Williams

In this example, the comma after ‘Edward’ is the \revsdnamedelim whereas the string ‘and’ is the \finalnamedelim, printed in addition to the former.

The delimiter printed before the localisation string ‘andmore’ if a name list like author or editor is truncated. The default is an interword space.

The delimiter printed between multiple items in a literal list like publisher or location if there are more than two items in the list. The default is a comma plus an interword space. See \multinamedelim for further explanation.

The delimiter printed instead of \multilistdelim before the final item in a literal list. The default is the localised term ‘and’, separated by interword spaces. See \finalnamedelim for further explanation.

The delimiter printed before the localisation string ‘andmore’ if a literal list like publisher or location is truncated. The default is an interword space.

The delimiter printed between citations if multiple entry keys are passed to a single citation command. The default is a semicolon plus an interword space.

Similar to \multicitedelim, but used by the \supercite command only. The default is a comma.

Similar to \multicitedelim, but used by certain citation styles when ‘compressing’ multiple citations. The default definition is a comma plus an interword space.

When formatting dates with the global option datecirca enabled, the delimiter printed after any localised ‘circa’ term. Defaults to interword space.

When formatting dates with the global option dateera set, the delimiter printed before the localisation era term. Defaults to interword space.

Prints date uncertainty information when the global option dateuncertain is enabled and the \ifdateuncertain test is true. By default, prints the language specific \bibdateuncertain string (§ 3.10.3).

Prints date uncertainty information when the global option dateuncertain is enabled and the \ifenddateuncertain test is true. By default, prints the language specific \bibdateuncertain string (§ 3.10.3).

Prints date circa information when the global option datecirca is enabled and the \ifdatecirca test is true. By default, prints the ‘circa’ localised term (§ 4.9.2.21) and the datecircadelim delimiter.

Prints date circa information when the global option datecirca is enabled and the \ifenddatecirca test is true. By default, prints the ‘circa’ localised term (§ 4.9.2.21) and the datecircadelim delimiter.
\datecircaprintedtf Prints EDTF format date circa information when the global option datecirca is enabled and the \ifdatecirca test is true. Prints ~.

\enddatecircaprintedtf Prints EDTF format date circa information when the global option datecirca is enabled and the \ifenddatecirca test is true. Prints ~.

\dateeraprint yearfield Prints date era information when the global option dateera is set to ‘secular’ or ‘christian’. By default, prints the dateeradelim delimiter and the appropriate localised era term (§ 4.9.2.21). If the dateeraauto option is set, then the passed ⟨yearfield⟩ (which is the name of a year field such as ‘year’, ‘origyear’, ‘endeventyear’ etc.) is tested to see if its value is earlier than the dateeraauto threshold and if so, then the BCE/CE localisation will be output too. The default setting for dateeraauto is 0 and so only BCE/BC localisation strings are candidates for output. Detects whether the start or end year era information is to be printed by looking at the ⟨yearfield⟩ name passed to it.

\dateeraprintpre Prints date era information when the global option dateera is set to ‘astronomical’. By default, prints bibdataeraprefix. Detects whether the start or end year era information is to be printed by looking at the ⟨yearfield⟩ name passed to it.

\textcitedelim Similar to \multicitedelim, but used by \textcite and related commands (§ 3.8.2). The default is a comma plus an interword space. The standard styles modify this provisional definition to ensure that the delimiter before the final citation is the localised term ‘and’, separated by interword spaces. See also \finalandcomma and \finalandsemicolon in §3.10.3.

\nametitledelim The delimiter printed between the author/editor and the title by author-title and some verbose citation styles. The default definition inside bibliographies is \labelnamepunct and is a comma plus an interword space otherwise.

\nameyeardelim The delimiter printed between the author/editor and the year by author-year citation styles. The default definition is an interword space.

\namelabeldelim The delimiter printed between the name/title and the label by alphabetic and numeric citation styles. The default definition is an interword space.

\nonameyeardelim The delimiter printed between the substitute for the labelname when it does not exist (usually the label or title in standard styles) and the year in author-year citation styles. This is only used when there is no labelname since when the labelname exists, \nameyeardelim is used. The default definition is an interword space.

\labelalphaothers A string to be appended to the non-numeric portion of the labelalpha field (i.e., the field holding the citation label used by alphabetic citation styles) if the number of authors/editors exceeds the maxalpha names threshold or the author/editor list was truncated in the bib file with the keyword ‘and others’. This will typically be a single character such as a plus sign or an asterisk. The default is a plus sign. This command may also be redefined to an empty string to disable this feature. In any case, it must be redefined in the preamble.
\texttt{\textbackslash sortalphaothers}  Similar to \texttt{\textbackslash labelalphaothers} but used in the sorting process. Setting it to a different value is advisable if the latter contains formatting commands, for example:

\begin{verbatim}
\renewcommand*{\labelalphaothers}{\textbf{+}}
\renewcommand*{\sortalphaothers}{+}
\end{verbatim}

If \texttt{\textbackslash sortalphaothers} is not redefined, it defaults to \texttt{\textbackslash labelalphaothers}.

\texttt{\textbackslash prenotedelim}  The delimiter printed after the \texttt{\langle prenote\rangle} argument of a citation command. See § 3.8 for details. The default is an interword space.

\texttt{\textbackslash postnotedelim}  The delimiter printed before the \texttt{\langle postnote\rangle} argument of a citation command. See § 3.8 for details. The default is a comma plus an interword space.

\texttt{\textbackslash extpostnotedelim}  The delimiter printed between the citation and the parenthetical \texttt{\langle postnote\rangle} argument of a citation command when the postnote occurs outside of the citation parentheses. In the standard styles, this occurs when the citation uses the shorthand field of the entry. See § 3.8 for details. The default is an interword space.

\texttt{\textbackslash mkbibname\textquotesingle namepart\textquoteright \{(text)\}}  This command, which takes one argument, is used to format the name part ‘namepart’ of name list fields. The default datamodel defines the name parts ‘family’, ‘given’, ‘prefix’ and ‘suffix’ and therefore the following macros are automatically defined:

\begin{verbatim}
\mkbibnamefamily
\mkbibnamegiven
\mkbibnameprefix
\mkbibnamesuffix
\end{verbatim}

For backwards compatibility with the legacy BibTeX name parts, the following are also defined, will generate warnings and will set the correct macro:

\begin{verbatim}
\mkbibnamelast
\mkbibnamefirst
\mkbibnameaffix
\end{verbatim}

\texttt{\textbackslash relatedpunct}  The separator between the \texttt{\textbackslash relatedtype} bibliography localisation string and the data from the first related entry. Here is an example with \texttt{\textbackslash relatedpunct} set to a dash:

\texttt{A. Smith. Title. 2000, (Orig. pub. as-Origtitle)}

\texttt{\textbackslash relateddelim}  The generic separator between the data of multiple related entries. The default definition is an optional dot plus linebreak. Here is an example where volumes A-E are related entries of the 5 volume main work:

\texttt{Donald E. Knuth. Computers & Typesetting. 5 vols. → Reading, Mass.: Addison-}
\relateddelim<relatedtype>  The separator between the data of multiple related entries inside related entries of type ‘relatedtype’. There is no default, if such a type-specific delimiter does not exist, \relateddelim is used.

3.10.2 Context-sensitive Delimiters

The delimiters described in §3.10.1 are globally defined. That is, no matter where you use them, they print the same thing. This is not necessarily desirable for delimiters which you might want to print different things in different contexts. Here ‘context’ means things like ‘inside a text citation’ or ‘inside a bibliography item’. For this reason, \biblatex provides a more sophisticated delimiter specification and user interface alongside the standard one based on normal macros defined with \newcommand.

\DeclareDelimFormat{⟨context, ...⟩}{⟨name, ...⟩}{⟨code⟩}
\DeclareDelimFormat*{⟨context, ...⟩}{⟨name, ...⟩}{⟨code⟩}

Declares the delimiter macros in the comma-separated list ⟨names⟩ with the replacement test ⟨code⟩. If the optional comma-separated list of ⟨contexts⟩ is given, declare the ⟨names⟩ only for those contexts. ⟨names⟩ defined without any ⟨contexts⟩ behave just like the global delimiter definitions which \newcommand gives—just a plain macro with a replacement which can be used as \name. However, you can also call delimiter macros defined in this way by using \printdelim, which is context-aware. The starred version clears all ⟨context⟩ specific declarations for all ⟨names⟩ first.

\printdelim{⟨context⟩}{⟨name⟩}

Prints a delimiter with name ⟨name⟩, locally establishing a optional ⟨context⟩ first. Without the optional ⟨context⟩, \printdelim uses the currently active delimiter context.

Delimiter contexts are simply a string, the value of the internal macro \blx@delimcontext which can be set manually by the command \delimcontext

\delimcontext{⟨context⟩}

Set the delimiter context to ⟨context⟩. This setting is not global so that delimiter contexts can be nested using the usual LaTeX group method.

\DeclareDelimcontextAlias{⟨alias⟩}{⟨name⟩}

The context-sensitive delimiter system creates delimiter contexts based on the name of citation commands (‘parencite’, ‘textcite’ etc.) passed to \DeclareCiteCommand. In certain cases where there are nested definitions of citation commands where \DeclareCiteCommand calls itself (see the definition of \textcite in authoryear-icomp for example), The delimiter context
is then usually incorrect and the delimiter specifications do not work. For example, the definition of \textcite in fact defines and uses \cbx@textcite and so the context is automatically set to \cbx@textcite when printing the citation. Delimiter definitions expecting to see the context textcite therefore do not work. Therefore this command is provided for style authors which aliases the context \textit{⟨alias⟩} to the context \textit{⟨name⟩}. For example:

\begin{verbatim}
\DeclareDelimcontextAlias{cbx@textcite}{textcite}
\end{verbatim}

This (which is a default setting), makes sure that when inside the \cbx@textcite citation command, the context is in fact textcite as expected.

\textbf{biblatex} has several default contexts which are established automatically in various places:

\textbf{none} At begin document

\textbf{bib} Inside a bibliography begun with \texttt{\printbibliography} or inside a \texttt{\usedriver}

\textbf{biblist} Inside a bibliography list begun with \texttt{\printbiblist}

\textbf{‘citecommand’} Inside a citation command \texttt{\citecommand} defined with \texttt{\DeclareCiteCommand}

For example, the defaults for \texttt{\nametitledelim} are:

\begin{verbatim}
\DeclareDelimFormat{nametitledelim}{\addcomma\space}
\DeclareDelimFormat[\textcite]{nametitledelim}{\addspace →}
\end{verbatim}

This means that \texttt{\nametitledelim} is defined globally as ‘\addcomma\space’ as per the standard delimiter interface. However, in addition, the delimiter can be printed using \texttt{\printdelim} which would print the same as \texttt{\nametitledelim} apart from inside a \texttt{\textcite}, in which it would print \texttt{\addspace which is more suitable for running text}. If desired, a context can be forced with the optional argument to \texttt{\printdelim}, so

\begin{verbatim}
\printdelim[\textcite]{nametitledelim}
\end{verbatim}

Would print \texttt{\addspace regardless} of the surrounding context of the \texttt{\printdelim}. Contexts are just arbitrary strings and so you can establish them at any time, using \texttt{\delimcontext}. If \texttt{\printdelim} finds no special value for the delimiter \texttt{⟨name⟩} in the current context, it simply prints \texttt{⟨name⟩}. This means that style authors can use \texttt{\printdelim} and users expecting to be able to use \texttt{\renewcommand} to redefine delimiters can do so with one caveat—such a definition won’t change any context-specific delimiters which are defined:

\begin{verbatim}
\DeclareDelimFormat{delima}{X}
\DeclareDelimFormat[\textcite]{delima}{Y}
\renewcommand*{\delima}{Z}
\end{verbatim}
Here, \delima always prints ‘Z’. \printdelim{delima} in any context other than ‘textcite’ also prints \delima and hence ‘Z’ but in a ‘textcite’ context prints ‘Y’. See the 04-delimiters.tex example file that comes with \biblatex for more information.

3.10.3 Language-specific Commands

The commands in this section are language specific. When redefining them, you need to wrap the new definition in a \DeclareBibliographyExtras command (in an .lbx file) or a \DefineBibliographyExtras command (user documents), see § 3.9 for details. Note that all commands starting with \mk… take one or more arguments.

\bibrangedash The language specific dash to be used for ranges of numbers. Defaults to \textendash.

\bibrangessep The language specific separator to be used between multiple ranges. Defaults to a comma followed by a space.

\bibdatesep The language specific separator used between date components in terse date formats. Defaults to \hyphen.

\bibdaterangesep The language specific separator to be used for date ranges. Defaults to \textendash for all date formats apart from ymd which defaults to a \slash. The date format option \texttt{edtf} is hard-coded to \slash since this is a standards compliant format.

\mkbibdatelong Takes the names of three field as arguments which correspond to three date components (in the order year/month/day) and uses their values to print the date in the language specific long date format.

\mkbibdateshort Similar to \mkbibdatelong but using the language specific short date format.

\mkbibtimezone Modifies a timezone string passed in as the only argument. By default this changes ‘Z’ to the value of \bibtimezone.

\bibdateuncertain The language specific marker to be used after uncertain dates when the global option \texttt{dateuncertain} is enabled. Defaults to a space followed by a question mark.

\bibdateeraprefix The language specific marker which is printed as a prefix to beginning BCE/BC dates in a date range when the option \texttt{dateera} is set to ‘astronomical’. Defaults to \textminus, if defined and \textendash otherwise.

\bibdateeraprefix The language specific marker which is printed as a prefix to end BCE/BC dates in a date range when the option \texttt{dateera} is set to ‘astronomical’. Defaults to a thin space followed by \bibdateeraprefix when \bibdaterangesep is set to a dash and to \bibdateeraprefix otherwise. This is a separate macro so that you may add extra space before a negative date marker which, for example follows a dash date range marker as this can look a little odd.

\bibtimesep The language specific marker which separates time components. Defaults to a colon.

\bibtimezonesep The language specific marker which separates an optional time zone component from a time. Empty by default.
\bibtzminsep  The language specific marker which separates hour and minute component of offset timezones. Defaults to a \bibtimesep.

\bibdatetimesep  The language specific separator printed between date and time components when printing time components along with date components (see the \texttt{<datetype>dateusetime} option in §3.1.2.1). Defaults to a space for non-EDTF output formats, and 'T' for EDTF output format.

\finalandcomma  Prints the comma to be inserted before the final 'and' in a list, if applicable in the respective language. Here is an example:

\begin{quote}
Michel Goossens, Frank Mittelbach, and Alexander Samarin
\end{quote}

\finalandcomma  is the comma before the word 'and'. See also \texttt{\multinamedelim}, \texttt{\finalnamedelim}, \texttt{\textcitedelim}, and \texttt{\revsdnamedelim} in §3.10.1.

\finalandsemicolon  Prints the semicolon to be inserted before the final 'and' in a list of lists, if applicable in the respective language. Here is an example:

\begin{quote}
Goossens, Mittelbach, and Samarin; Bertram and Wenworth; and Knuth
\end{quote}

\finalandsemicolon  is the semicolon before the word 'and'. See also \texttt{\textcitedelim} in §3.10.1.

\mkbibordinal{⟨integer⟩}

This command, which takes an integer as its argument, prints an ordinal number.

\mkbibmascord{⟨integer⟩}

Similar to \mkbibordinal, but prints a masculine ordinal, if applicable in the respective language.

\mkbibfemord{⟨integer⟩}

Similar to \mkbibordinal, but prints a feminine ordinal, if applicable in the respective language.

\mkbibneutord{⟨integer⟩}

Similar to \mkbibordinal, but prints a neuter ordinal, if applicable in the respective language.

\mkbibordedition{⟨integer⟩}

Similar to \mkbibordinal, but intended for use with the term 'edition'.
Similar to `\mkbibordinal`, but intended for use with the term ‘series’.

### 3.10.4 Lengths and Counters

The length registers and counters in this section may be changed in the document preamble with `\setlength` and `\setcounter`, respectively.

- \bibhang: The hanging indentation of the bibliography, if applicable. This length is initialized to `\parindent` at load-time.

- \biblabelsep: The horizontal space between entries and their corresponding labels in the bibliography. This only applies to bibliography styles which print labels, such as the numeric and alphabetic styles. This length is initialized to twice the value of `\labelsep` at load-time.

- \bibitemsep: The vertical space between the individual entries in the bibliography. This length is initialized to `\itemsep` at load-time. Note that `\bibitemsep`, `\bibnamesep`, and `\bibinitsep` obey the rules for `\addvspace`, that is, when vertical space introduced by any of these commands immediately follows on from space introduced by another of them, the resulting total space is equal to the largest of them.

- \bibnamesep: Vertical space to be inserted between two entries in the bibliography whenever an entry starts with a name which is different from the initial name of the previous entry. The default value is zero. Setting this length to a positive value greater than `\bibitemsep` will group the bibliography by author/editor name. Note that `\bibitemsep`, `\bibnamesep`, and `\bibinitsep` obey the rules for `\addvspace`, that is, when vertical space introduced by any of these commands immediately follows on from space introduced by another of them, the resulting total space is equal to the largest of them.

- \bibinitsep: Vertical space to be inserted between two entries in the bibliography whenever an entry starts with a letter which is different from the initial letter of the previous entry. The default value is zero. Setting this length to a positive value greater than `\bibitemsep` will group the bibliography alphabetically. Note that `\bibitemsep`, `\bibnamesep`, and `\bibinitsep` obey the rules for `\addvspace`, that is, when vertical space introduced by any of these commands immediately follows on from space introduced by another of them, the resulting total space is equal to the largest of them.

- \bibparsep: The vertical space between paragraphs within an entry in the bibliography. The default value is zero.

- abbrvpenalty: This counter, which is used by the localisation modules, holds the penalty used in short or abbreviated localisation strings. For example, a linebreak in expressions such as “et al.” or “ed. by” is unfortunate, but should still be possible to prevent overfull boxes. This counter is initialized to `\hyphenpenalty` at load-time. The idea is making TeX treat the whole expression as if it were a single, hyphenatable word as far as line-breaking is concerned. If you dislike such linebreaks, use a higher value. If you do not mind them at all, set this counter to zero. If you want to suppress them unconditionally, set it to ‘infinite’ (10 000 or higher).

---

Footnote: The default values assigned to `abbrvpenalty`, `lownamepenalty`, and `highnamepenalty` are deliberately very low to prevent overfull boxes. This implies that you will hardly notice any...
\texttt{highnamepenalty} This counter holds a penalty affecting line-breaking in names. Please refer to §§ 3.13.4 and 3.10.1 for explanation. The counter is initialized to \texttt{\hyphenpenalty} at load-time. Use a higher value if you dislike the respective linebreaks. If you do not mind them at all, set this counter to zero. If you prefer the traditional BibTeX behavior (no linebreaks at \texttt{highnamepenalty} breakpoints), set it to ‘infinite’ (10 000 or higher).

\texttt{lownamepenalty} Similar to \texttt{highnamepenalty}. Please refer to §§ 3.13.4 and 3.10.1 for explanation. The counter is initialized to half the \texttt{\hyphenpenalty} at load-time. Use a higher value if you dislike the respective linebreaks. If you do not mind them at all, set this counter to zero.

### 3.10.5 All-purpose Commands

The commands in this section are all-purpose text commands which are generally available, not only in citations and the bibliography.

\texttt{\bibellipsis} An ellipsis symbol with brackets: ‘[...]’.

\texttt{\noligature} Disables ligatures at this position and adds some space. Use this command to break up standard ligatures like ‘fi’ and ‘fl’. It is similar to the "| shorthand provided by some language modules of the \texttt{babel/polyglossia} packages.

\texttt{\hyphenate} A conditional hyphen. In contrast to the standard \texttt{\-} command, this one allows hyphenation in the rest of the word. It is similar to the "- shorthand provided by some language modules of the \texttt{babel/polyglossia} packages.

\texttt{\hyphen} An explicit, breakable hyphen intended for compound words. In contrast to a literal ‘-’, this command allows hyphenation in the rest of the word. It is similar to the "= shorthand provided by some language modules of the \texttt{babel/polyglossia} packages.

\texttt{\nbhyphen} An explicit, non-breakable hyphen intended for compound words. In contrast to a literal ‘-’, this command does not permit line breaks at the hyphen but still allows hyphenation in the rest of the word. It is similar to the "~ shorthand provided by some language modules of the \texttt{babel/polyglossia} packages.

\texttt{\nohyphenation{⟨text⟩}}

Similar to \texttt{\nohyphenation} but restricted to the ⟨text⟩ argument.

\texttt{\textnohyphenation{⟨text⟩}}

\texttt{\mknualph{⟨integer⟩}}

Takes an integer in the range 1–702 as its argument and converts it to a string as follows: 1=a, ..., 26=z, 27=aa, ..., 702=zz. This is intended for use in formatting directives for the \texttt{extrayear} and \texttt{extraalpha} fields.

---

Effect on line-breaking if the text is set justified. If you set these counters to 10 000 to suppress the respective breakpoints, you will notice their effect but you may also be confronted with overfull boxes. Keep in mind that line-breaking in the bibliography is often more difficult than in the body text and that you can not resort to rephrasing a sentence. In some cases it may be preferable to set the entire bibliography \texttt{\raggedright} to prevent suboptimal linebreaks. In this case, even the fairly low default penalties will make a visible difference.
\mkbibacro{\text}

Generic command which typesets an acronym using the small caps variant of the current font, if available, and as-is otherwise. The acronym should be given in uppercase letters.

\autocap{\text}

Automatically converts the \text{character} to its uppercase form if \text{biblatex}'s punctuation tracker would capitalize a localisation string at the current location. This command is robust. It is useful for conditional capitalization of certain strings in an entry. Note that the \text{character} argument is a single character given in lowercase. For example:

\autocap{s}pecial issue

will yield ‘Special issue’ or ‘special issue’, as appropriate. If the string to be capitalized starts with an inflected character given in Ascii notation, include the accent command in the \text{character} argument as follows:

\autocap{\'e}dition sp\'eciale

This will yield ‘Édition spéciale’ or ‘édition spéciale’. If the string to be capitalized starts with a command which prints a character, such as \ae or \oe, simply put the command in the \text{character} argument:

\autocap{\oe}uvres

This will yield ‘Œuvres’ or ‘œuvres’.

3.11 Language-specific Notes

The facilities discussed in this section are specific to certain localisation modules.

3.11.1 American

The American localisation module uses \uspunctuation from §4.7.5 to enable ‘American-style’ punctuation. If this feature is enabled, all trailing commas and periods after \mkbibquote will be moved inside the quotes. If you want to disable this feature, use \stdpunctuation as follows:

\DefineBibliographyExtras{american}{% \stdpunctuation }

By default, the ‘American punctuation’ feature is enabled by the american localisation module only. The above code is only required if you want American localisation without American punctuation. Since standard punctuation is the package default, it would be redundant with any other language.

It is highly advisable to always specify \text{american, british, australian, etc.} rather than \text{english} when loading the \text{babel/polyglossia} packages to avoid
any possible confusion. Older versions of the babel package used to treat english as an alias for british; more recent ones treat it as an alias for american. The biblatex package essentially treats english as an alias for american, except for the above feature which is only enabled if american is requested explicitly.

3.11.2 Spanish

Handling the word ‘and’ is more difficult in Spanish than in the other languages supported by this package because it may be ‘y’ or ‘e’, depending on the initial sound of the following word. Therefore, the Spanish localisation module does not use the localisation string ‘and’ but a special internal ‘smart and’ command. The behavior of this command is controlled by the smartand counter.

\texttt{smartand} This counter controls the behavior of the internal ‘smart and’ command. When set to 1, it prints ‘y’ or ‘e’, depending on the context. When set to 2, it always prints ‘y’. When set to 3, it always prints ‘e’. When set to 0, the ‘smart and’ feature is disabled. This counter is initialized to 1 at load-time and may be changed in the preamble. Note that setting this counter to a positive value implies that the Spanish localisation module ignores \texttt{\finalnamedelim} and \texttt{\finallistdelim}.

\texttt{\forceE} Use this command in \texttt{bib} files if biblatex gets the ‘and’ before a certain name wrong. As its name suggests, it will enforce ‘e’. This command must be used in a special way to be correct BibTeX datafile format. Here is an example:

\begin{verbatim}
author = {Edward Jones and Eoin Maguire},
author = {Edward Jones and \forceE{E}oin Maguire},
\end{verbatim}

Note that the initial letter of the respective name component is given as an argument to \texttt{\forceE} and that the entire construct is wrapped in an additional pair of curly braces.

\texttt{\forceY} Similar to \texttt{\forceE} but enforces ‘y’.

3.11.3 Greek

The Greek localisation module requires utf-8 support. It will not work with any other encoding. Generally speaking, the biblatex package is compatible with the inputenc package and with XeLaTeX. The ucs package will not work. Since inputenc's standard utf8 module is missing glyph mappings for Greek, this leaves Greek users with XeLaTeX. Note that you may need to load additional packages which set up Greek fonts. As a rule of thumb, a setup which works for regular Greek documents should also work with biblatex. However, there is one fundamental limitation. As of this writing, biblatex has no support for switching scripts. Greek titles in the bibliography should work fine, but English and other titles in the bibliography may be rendered in Greek letters. If you need multi-script bibliographies, using XeLaTeX is the only sensible choice.

3.11.4 Russian

Like the Greek localisation module, the Russian module also requires utf-8 support. It will not work with any other encoding.
3.12 Usage Notes

The following sections give a basic overview of the biblatex package and discuss some typical usage scenarios.

3.12.1 Overview

Using the biblatex package is slightly different from using traditional BibTeX styles and related packages. Before we get to specific usage scenarios, we will therefore have a look at the structure of a typical document first:

```latex
\documentclass{...}
\usepackage[...]{biblatex}
\addbibresource{bibfile.bib}
\begin{document}
\cite{...}
...
\printbibliography
\end{document}
```

With traditional BibTeX, the \texttt{\bibliography} command serves two purposes. It marks the location of the bibliography and it also specifies the \texttt{bib} file(s). The file extension is omitted. With biblatex, resources are specified in the preamble with \texttt{\addbibresource} using the full name with \texttt{.bib} suffix. The bibliography is printed using the \texttt{\printbibliography} command which may be used multiple times (see §3.7 for details). The document body may contain any number of citation commands (§3.8). Processing this example file requires that a certain procedure be followed. Suppose our example file is called example.tex and our bibliographic data is in bibfile.bib. The procedure, then, is as follows:

1. Run \texttt{latex} on \texttt{example.tex}. If the file contains any citations, biblatex will request the respective data from biber by writing commands to the auxiliary file example.bcf.

2. Run \texttt{biber} on \texttt{example.bcf}. biber will retrieve the data from bibfile.bib and write it to the auxiliary file example.bbl in a format which can be processed by biblatex.

3. Run \texttt{latex} on \texttt{example.tex}. biblatex will read the data from example.bbl and print all citations as well as the bibliography.

3.12.2 Auxiliary Files

The biblatex package uses one auxiliary \texttt{bcf} file only. Even if there are citation commands in a file included via \texttt{\include}, you only need to run \texttt{biber} on the main \texttt{bcf} file. All information \texttt{biber} needs is in the \texttt{bcf} file, including information about all \texttt{refsection} environments (see §3.12.3).

3.12.3 Multiple Bibliographies

In a collection of articles by different authors, such as a conference proceedings volume for example, it is very common to have one bibliography for each article rather than a global one for the entire book. In the example below, each article would be presented as a separate \texttt{\chapter} with its own bibliography.
If `\printbibliography` is used inside a `refsection` environment, it automatically restricts the scope of the list of references to the enclosing `refsection` environment. For a cumulative bibliography which is subdivided by chapter but printed at the end of the book, use the `section` option of `\printbibliography` to select a reference section, as shown in the next example.

Note the definition of the bibliography heading in the above example. This is the definition taking care of the subheadings in the bibliography. The main heading is generated with a plain `\chapter` command in this case. The `biblatex` package automatically sets a label at the beginning of every `refsection` environment, using the standard `\label` command. The identifier used is the string `refsection:` followed by the number of the respective `refsection` environment. The number of the current section is accessible via the `refsection` counter. When using the `section` option of `\printbibliography`, this counter is also set locally. This
means that you may use the counter in heading definitions to print subheadings like “References for Chapter 3”, as shown above. You could also use the title of the respective chapter as a subheading by loading the nameref package and using \nameref instead of \ref:

\usepackage{nameref}
defbibheading{subbibliography}{% section*{\nameref{refsection:\therefsection}}} Since giving one \printbibliography command for each part of a subdivided bibliography is tedious, biblatex provides a shorthand. The \bibbysection command automatically loops over all reference sections. This is equivalent to giving one \printbibliography command for every section but has the additional benefit of automatically skipping sections without references. In the example above, the bibliography would then be generated as follows:

\printbibheading \bibbysection[heading=subbibliography]

When using a format with one cumulative bibliography subdivided by chapter (or any other document division) it may be more appropriate to use refsegment rather than refsection environments. The difference is that the refsection environment generates labels local to the environment while refsegment does not affect the generation of labels, hence they will be unique across the entire document. The next example could also be given in §3.12.4 because, visually, it creates one global bibliography subdivided into multiple segments.

\documentclass{...} \usepackage{biblatex} \defbibheading{subbibliography}{% \section*{References for Chapter \ref{refsegment:\therefsection\therefsegment}}} \addbibresource{...} \begin{document} \chapter{...} \begin{refsegment} ...
\end{refsegment} \chapter{...} \begin{refsegment} ...
\end{refsegment} \chapter{...} \begin{refsegment} ...
\end{refsegment} \printbibheading \printbibliography[segment=1,heading=subbibliography] \printbibliography[segment=2,heading=subbibliography] \end{document}

The use of refsegment is similar to refsection and there is also a corresponding segment option for \printbibliography. The biblatex package automatically sets a label at the beginning of every refsegment environment using
the string \texttt{refsegment}: followed by the number of the respective \texttt{refsegment} environment as an identifier. There is a matching \texttt{refsegment} counter which may be used in heading definitions, as shown above. As with reference sections, there is also a shorthand command which automatically loops over all reference segments:

\begin{verbatim}
\printbibheading
\bibbysegment[heading=subbibliography]
\end{verbatim}

This is equivalent to giving one \texttt{\printbibliography} command for every segment in the current \texttt{refsection}.

### 3.12.4 Subdivided Bibliographies

It is very common to subdivide a bibliography by certain criteria. For example, you may want to list printed and online resources separately or divide a bibliography into primary and secondary sources. The former case is straightforward because you can use the entry type as a criterion for the \texttt{type} and \texttt{notype} filters of \texttt{\printbibliography}. The next example also demonstrates how to generate matching subheadings for the two parts of the bibliography.

\begin{verbatim}
\documentclass{...}
\usepackage{biblatex}
\addbibresource{...}
\begin{document}
...
\printbibheading
\printbibliography[notype=online,heading=subbibliography, title={Printed Sources}]
\printbibliography[type=online,heading=subbibliography, title={Online Sources}]
\end{document}
\end{verbatim}

You may also use more than two subdivisions:

\begin{verbatim}
\printbibliography[type=article,...]
\printbibliography[type=book,...]
\printbibliography[notype=article,notype=book,...]
\end{verbatim}

It is even possible to give a chain of different types of filters:

\begin{verbatim}
\printbibliography[section=2,type=book,keyword=abc, notkeyword=xyz]
\end{verbatim}

This would print all works cited in reference section 2 whose entry type is \texttt{@book} and whose \texttt{keywords} field includes the keyword ‘abc’ but not ‘xyz’. When using bibliography filters in conjunction with a numeric style, see §3.13.5. If you need complex filters with conditional expressions, use the \texttt{filter} option in conjunction with a custom filter defined with \texttt{\defbibfilter}. See §3.7.9 for details on custom filters.
Dividing a bibliography into primary and secondary sources is possible with a keyword filter, as shown in the above example. In this case, with only two subdivisions, it would be sufficient to use one keyword as filter criterion:

\begin{document}
...\end{document}

\begin{verbatim}
\printbibliography[keyword=primary,...]
\printbibliography[notkeyword=primary,...]
\end{verbatim}

Since \texttt{biblatex} has no way of knowing if an item in the bibliography is considered to be primary or secondary literature, we need to supply the bibliography filter with the required data by adding a \texttt{keywords} field to each entry in the \texttt{bib} file. These keywords may then be used as targets for the \texttt{keyword} and \texttt{notkeyword} filters, as shown above. It may be a good idea to add such keywords right away while building a \texttt{bib} file.

An alternative way of subdividing the list of references are bibliography categories. They differ from the keywords-based approach shown in the example above in that they work on the document level and do not require any changes to the \texttt{bib} file.
In this case it would also be sufficient to use one category only:

\printbibliography[category=primary,...]
\printbibliography[notcategory=primary,...]

It is still a good idea to declare all categories used in the bibliography explicitly because there is a \bibbycategory command which automatically loops over all categories. This is equivalent to giving one \printbibliography command for every category, in the order in which they were declared.

The handling of the headings is different from \bibbysection and \bibbysegment in this case. \bibbycategory uses the name of the current category as a heading name. This is equivalent to passing heading=\langle category \rangle to \printbibliography and implies that you need to provide a matching heading for every category.

3.12.5 Entry Sets

An entry set is a group of entries which are cited as a single reference and listed as a single item in the bibliography. The individual entries in the set are separated by \entrysetpunct (§ 4.10.1). The biblatex package supports two types of entry sets. Static entry sets are defined in the bib file like any other entry. Dynamic entry sets are defined with \defbibentryset (§ 3.7.11) on a per-document/per-refsection basis in the document preamble or the document body. This section deals with the definition of entry sets; style authors should also see § 4.11.1 for further information.

3.12.5.1 Static entry sets

Static entry sets are defined in the bib file like any other entry. Defining an entry set is as simple as adding an entry of type @set. The entry has an entryset field defining the members of the set as a separated list of entry keys:
Entries may be part of a set in one document/refsection and stand-alone references in another one, depending on the presence of the \texttt{@set} entry. If the \texttt{@set} entry is cited, the set members are grouped automatically. If not, they will work like any regular entry.

\subsection{Dynamic entry sets} Dynamic entry sets are set up and work much like static ones. The main difference is that they are defined in the document preamble or on the fly in the document body using the \texttt{\defbibentryset} command from §3.7.11:

\begin{verbatim}
defbibentryset{set1}{key1,key2,key3}
\end{verbatim}

Dynamic entry sets in the document body are local to the enclosing \texttt{refsection} environment, if any. Otherwise, they are assigned to reference section 0. Those defined in the preamble are assigned to reference section 0.

\subsection{Data Containers} The \texttt{@xdata} entry type serves as a data container holding one or more fields. These fields may be inherited by other entries using the \texttt{xdata} field. \texttt{@xdata} entries may not be cited or added to the bibliography, they only serve as a data source for other entries. This data inheritance mechanism is useful for fixed field combinations such as publisher/location and for other frequently used data:

\begin{verbatim}
@XData{hup,
   publisher = {Harvard University Press},
   location = {Cambridge, Mass.},
}
@Book{..., 
   author   = {...},
   title    = {...},
   date     = {...},
   xdata    = {hup},
}
\end{verbatim}

Using a separated list of keys in its \texttt{xdata} field, an entry may inherit data from several \texttt{@xdata} entries. Cascading \texttt{@xdata} entries are supported as well, i.e., an \texttt{@xdata} entry may reference one or more other \texttt{@xdata} entries:

\begin{verbatim}
@XData{macmillan:name,
   publisher = {Macmillan},
}
@XData{macmillan:place,
   location = {New York and London},
}
@XData{macmillan,
   xdata    = {macmillan:name,macmillan:place},
\end{verbatim}
3.12.7 Electronic Publishing Information

The \texttt{biblatex} package provides three fields for electronic publishing information: \texttt{eprint}, \texttt{eprinttype}, and \texttt{eprintclass}. The \texttt{eprint} field is a verbatim field similar to \texttt{doi} which holds the identifier of the item. The \texttt{eprinttype} field holds the resource name, i.e., the name of the site or electronic archive. The optional \texttt{eprintclass} field is intended for additional information specific to the resource indicated by the \texttt{eprinttype} field. This could be a section, a path, classification information, etc. If the \texttt{eprinttype} field is available, the standard styles will use it as a literal label. In the following example, they would print "Resource: identifier" rather than the generic "eprint: identifier":

\begin{verbatim}
eprint = {identifier},
eprinttype = {Resource},
\end{verbatim}

The standard styles feature dedicated support for a few online archives. For arXiv references, put the identifier in the \texttt{eprint} field and the string \texttt{arxiv} in the \texttt{eprinttype} field:

\begin{verbatim}
eprint = {math/0307200v3},
eprinttype = {arxiv},
\end{verbatim}

For papers which use the new identifier scheme (April 2007 and later) add the primary classification in the \texttt{eprintclass} field:

\begin{verbatim}
eprint = {1008.2849v1},
eprinttype = {arxiv},
eprintclass = {cs.DS},
\end{verbatim}

There are two aliases which ease the integration of arXiv entries. \texttt{archiveprefix} is treated as an alias for \texttt{eprinttype}; \texttt{primaryclass} is an alias for \texttt{eprintclass}. If hyperlinks are enabled, the \texttt{eprint} identifier will be transformed into a link to \texttt{arxiv.org}. See the package option \texttt{arxiv} in §3.1.2.1 for further details.

For \texttt{jstor} references, put the stable \texttt{jstor} number in the \texttt{eprint} field and the string \texttt{jstor} in the \texttt{eprinttype} field:

\begin{verbatim}
eprint = {number},
eprinttype = {jstor},
\end{verbatim}

When using \texttt{jstor}'s export feature to export citations in BibTeX format, \texttt{jstor} uses the \texttt{url} field by default (where the \texttt{(number)} is a unique and stable identifier):
While this will work as expected, full URLs tend to clutter the bibliography. With the eprint fields, the standard styles will use the more readable \texttt{\textasciitilde jstor: \langle number \rangle}” format which also supports hyperlinks. The \langle number \rangle becomes a clickable link if hyperref support is enabled.

For PubMed references, put the stable PubMed identifier in the \texttt{eprint} field and the string \texttt{pubmed} in the \texttt{eprinttype} field. This means that:

\begin{verbatim}
\end{verbatim}

becomes:

\begin{verbatim}
\texttt{eprint} = {pmid},
\texttt{eprinttype} = {pubmed},
\end{verbatim}

and the standard styles will print “\texttt{PMID: \langle pmid \rangle}” instead of the lengthy \texttt{url}. If hyperref support is enabled, the \langle pmid \rangle will be a clickable link to PubMed.

For handles (HDLs), put the handle in the \texttt{eprint} field and the string \texttt{hdl} in the \texttt{eprinttype} field:

\begin{verbatim}
\texttt{eprint} = {handle},
\texttt{eprinttype} = {hdl},
\end{verbatim}

For Google Books references, put Google’s identifier in the \texttt{eprint} field and the string \texttt{googlebooks} in the \texttt{eprinttype} field. This means that, for example:

\begin{verbatim}
url = {http://books.google.com/books?id=XXu4AkRVBoC},
\end{verbatim}

would become:

\begin{verbatim}
\texttt{eprint} = {XXu4AkRVBoC},
\texttt{eprinttype} = {googlebooks},
\end{verbatim}

and the standard styles would print “Google Books: XXu4AkRVBoC” instead of the full \texttt{url}. If hyperref support is enabled, the identifier will be a clickable link to Google Books.\footnote{Note that the Google Books \texttt{id} seems to be a bit of an ‘internal’ value. As of this writing, there does not seem to be any way to search for an \texttt{id} on Google Books. You may prefer to use the \texttt{url} in this case.}

Note that \texttt{eprint} is a verbatim field. Always give the identifier in its unmodified form. For example, there is no need to replace \texttt{_} with \texttt{\_}. Also see §4.11.2 on how to add dedicated support for other eprint resources.

\section{External Abstracts and Annotations}

Styles which print the fields \texttt{abstract} and/or \texttt{annotation} may support an alternative way of adding abstracts or annotations to the bibliography. Instead of including the text in the \texttt{bib} file, it may also be stored in an external \LaTeX{} file. For example, instead of saying
The name of the external file must be the entry key prefixed with \bibabstract- or \bibannotation-, respectively. You can change these prefixes by redefining \bibabstractprefix and \bibannotationprefix. Note that this feature needs to be enabled explicitly by setting the package option loadfiles from § 3.1.2.1. The option is disabled by default for performance reasons. Also note that any abstract and annotation fields in the bib file take precedence over the external files. Using external files is strongly recommended if you have long abstracts or a lot of annotations since this may increase memory requirements significantly. It is also more convenient to edit the text in a dedicated \LaTeX file. Style authors should see § 4.11.3 for further information.

3.13 Hints and Caveats

This section provides additional usage hints and addresses some common problems and potential misconceptions.

3.13.1 Usage with KOMA-Script Classes

When using biblatex in conjunction with one of the \texttt{scrbook}, \texttt{scrreprt}, or \texttt{scrartcl} classes, the headings \texttt{bibliography} and \texttt{biblist} from § 3.7.7 are responsive to the bibliography-related options of these classes.\footnote{This applies to the traditional syntax of these options (\texttt{bibtotoc} and \texttt{bibtotonumbered}) as well as to the \texttt{(key)=(value)} syntax introduced in KOMA-Script 3.x, i.e., to \texttt{bibliography=nottotoc}, \texttt{bibliography=totoc}, and \texttt{bibliography=totocnumbered}. The global \texttt{toc=bibliography} and \texttt{toc=bibliographynumbered} options as well as their aliases are detected as well. In any case, the options must be set globally in the optional argument to \texttt{\documentclass}.} You can override the default headings by using the \texttt{heading} option of \texttt{\printbibliography}, \texttt{\printbibheading} and \texttt{\printbiblist}. See §§ 3.7.2, 3.7.3, 3.7.7 for details.

All default headings are adapted at load-time such that they blend with the behavior of these classes. If one of the above classes is detected, biblatex will also provide the following additional tests which may be useful in custom heading definitions:

\texttt{\ifkomabibtotoc{(\texttt{true})}{(\texttt{false})}}

Expands to \texttt{(true)} if the class would add the bibliography to the table of contents, and to \texttt{(false)} otherwise.
Expands to \(\text{true}\) if the class would add the bibliography to the table of contents as a numbered section, and to \(\text{false}\) otherwise. If this test yields \(\text{true}\), \ifkomabibtotoc \ will always yield \(\text{true}\) as well, but not vice versa.

### 3.13.2 Usage with the Memoir Class

When using \texttt{biblatex} with the \texttt{memoir} class, most class facilities for adapting the bibliography have no effect. Use the corresponding facilities of this package instead (§§3.7.2, 3.7.7, 3.7.8). Instead of redefining \texttt{memoir}'s \texttt{\bibsection}, use the heading option of \texttt{\printbibliography} and \texttt{\defbibheading} (§§3.7.2 and 3.7.7). Instead of \texttt{\prebibhook} and \texttt{\postbibhook}, use the \texttt{\prenote} and \texttt{\postnote} options of \texttt{\printbibliography} and \texttt{\defbibnote} (§§3.7.2 and 3.7.8). All default headings are adapted at load-time such that they blend well with the default layout of this class. The default headings \texttt{\bibliography} and \texttt{\biblist} (§3.7.7) are also responsive to \texttt{\bibintoc} and \texttt{\nobibintoc} switches. The length register \texttt{\bibitemsep} is used by \texttt{biblatex} in a way similar to \texttt{\bibliographystyle} (§ 3.10.4). This section also introduces some additional length registers which correspond to \texttt{\bibliographystyle}.'s \texttt{\biblistextra} switches. Lastly, \texttt{\setbiblabel} does not map to a single facility of the \texttt{biblatex} package since the style of all labels in the bibliography is controlled by the bibliography style. See §4.2.2 in the author section of this manual for details. If the \texttt{\memoir} class is detected, \texttt{biblatex} will also provide the following additional test which may be useful in custom heading definitions:

\\ifmemoirbibintoc\{\text{true}\}\{\text{false}\}

Expands to \(\text{true}\) or \(\text{false}\), depending on \texttt{\memoir}'s \texttt{\bibintoc} and \texttt{\nobibintoc} switches. This is a \LaTeX\ frontend to \texttt{\memoir}'s \texttt{\ifnobibintoc} test. Note that the logic of the test is reversed.

### 3.13.3 Page Numbers in Citations

If the \(\text{postnote}\) argument to a citation command is a page number or page range, \texttt{biblatex} will automatically prefix it with ‘p.’ or ‘pp.’ by default. This works reliably in typical cases, but sometimes manual intervention may be required. In this case, it is important to understand how this argument is handled in detail. First, \texttt{biblatex} checks if the postnote is an Arabic or Roman numeral (case insensitive). If this test succeeds, the postnote is considered as a single page or other number which will be prefixed with ‘p.’ or some other string which depends on the \texttt{\pagination} field (see §2.3.10). If it fails, a second test is performed to find out if the postnote is a range or a list of Arabic or Roman numerals. If this test succeeds, the postnote will be prefixed with ‘pp.’ or some other string in the plural form. If it fails as well, the postnote is printed as is. Note that both tests expand the \(\text{postnote}\). All commands used in this argument must therefore be robust or prefixed with \texttt{\protect}. Here are a few examples of \(\text{postnote}\) arguments which will be correctly recognized as a single number, a range of numbers, or a list of numbers, respectively:

\begin{verbatim}
\cite[25]{key}
\cite[vii]{key}
\cite[XIV]{key}
\cite[34--38]{key}
\end{verbatim}
In some other cases, however, the tests may get it wrong and you need to resort to the auxiliary commands \pno, \ppno, and \nopp from § 3.8.8. For example, suppose a work is cited by a special pagination scheme consisting of numbers and letters. In this scheme, the string ‘27a’ would mean ‘page 27, part a’. Since this string does not look like a number or a range to \biblatex, you need to force the prefix for a single number manually:

\cite{\pno~27a}{key}

There is also a \ppno command which forces a range prefix as well as a \nopp command which suppresses all prefixes:

\cite[\ppno~27a--28c]{key}
\cite[\nopp 25]{key}

These commands may be used anywhere in the \langle postnote \rangle argument. They may also be used multiple times. For example, when citing by volume and page number, you may want to suppress the prefix at the beginning of the postnote and add it in the middle of the string:

\cite[VII, \pno~5]{key}
\cite[VII, \pno~3, \ppno~40--45]{key}
\cite[see]\ppno~37--46, in particular \pno~40{key}

There are also two auxiliary command for suffixes like ‘the following page(s)’. Instead of inserting such suffixes literally (which would require \ppno to force a prefix):

\cite[\ppno~27~sq.]{key}
\cite[\ppno~55~sqq.]{key}

use the auxiliary commands \psq and \psqq. Note that there is no space between the number and the command. This space will be inserted automatically and may be modified by redefining the macro \sqspace.

\cite[27\psq]{key}
\cite[55\psqq]{key}

Since the postnote is printed without any prefix if it includes any character which is not an Arabic or Roman numeral, you may also type the prefix manually:

\cite[p.~5]{key}
It is possible to suppress the prefix on a per-entry basis by setting the pagination field of an entry to 'none', see § 2.3.10 for details. If you do not want any prefixes at all or prefer to type them manually, you can also disable the entire mechanism in the document preamble or the configuration file as follows:

\DeclareFieldFormat{postnote}{#1}

The \textit{postnote} argument is handled as a field and the formatting of this field is controlled by a field formatting directive which may be freely redefined. The above definition will simply print the postnote as is. See §§ 4.3.2 and 4.4.2 in the author guide for further details.

### 3.13.4 Name Parts and Name Spacing

The \texttt{biblatex} package gives users and style authors very fine-grained control of name spacing and the line-breaking behavior of names. The commands discussed in the following are documented in §§ 3.10.1 and 4.10.1. This section is meant to give an overview of how they are put together. A note on terminology: a name \textit{part} is a basic part of the name, for example the given or the family name. Each part of a name may be a single name or it may be composed of multiple names. For example, the name part ‘given name’ may be composed of a given and a middle name. The latter are referred to as name \textit{elements} in this section. Let’s consider a simple name first: “John Edward Doe”. This name is composed of the following parts:

- **Given**: John
- **Prefix**: Edward
- **Family**: Doe
- **Suffix**: —

The spacing, punctuation and line-breaking behavior of names is controlled by six macros:

- \texttt{\bibnamedelima}: Inserted by the backend after the first element of every name part if that element is less than three characters long and before the last element of every name part.
- \texttt{\bibnamedelimb}: Inserted by the backend between all elements of a name part where \texttt{\bibnamedelima} does not apply.
- \texttt{\bibnamedelimc}: Inserted by a formatting directive between the name prefix and the family name if \texttt{useprefix=true}. If \texttt{useprefix=false}, \texttt{\bibnamedelimd} is used instead.
- \texttt{\bibnamedelimd}: Inserted by a formatting directive between name parts where \texttt{\bibnamedelimc} does not apply.
- \texttt{\bibnamedelimi}: Replaces \texttt{\bibnamedelima/\bibnamedelimb} after initials
- \texttt{\revsdnamepunct}: Inserted by a formatting directive after the family name when the name parts are reversed.

This is how the delimiters are employed:

```
John|\bibnamedelima|Edward|\bibnamedelimb|Doe
Doe|\bibnamedelima|\bibnamedelimb|John|\bibnamedelima|Edward
```
Initials in the bib file get a special delimiter:

\texttt{J.\_\_|\_\_Edward\_\_|\_\_Doe}

Let’s consider a more complex name: “Charles-Jean Étienne Gustave Nicolas de La Vallée Poussin”. This name is composed of the following parts:

- **Given**: Charles-Jean Étienne Gustave Nicolas
- **Prefix**: de
- **Family**: La Vallée Poussin
- **Suffix**: —

The delimiters:

\texttt{Charles-Jean\_\_|\_\_Étienne\_\_|\_\_Gustave\_\_|\_\_Nicolas\_\_|\_\_de\_\_|\_\_La\_\_|\_\_Vallée\_\_|\_\_Poussin}

Note that \texttt{\bibnamedelim/b/i} are inserted by the backend. The backend processes the name parts and takes care of the delimiters between the elements that make up a name part, processing each part individually. In contrast to that, the delimiters between the parts of the complete name (\texttt{\bibnamedelimc/d}) are added by name formatting directives at a later point in the processing chain.

The spacing and punctuation of initials is also handled by the backend and may be customized by redefining the following three macros:

- \texttt{a=\bibinitperiod} Inserted by the backend after initials.
- \texttt{b=\bibinitdelim} Inserted by the backend between multiple initials.
- \texttt{c=\bibinithyphendelim} Inserted by the backend between the initials of hyphenated name parts, replacing \texttt{\bibinitperiod} and \texttt{\bibinitdelim}.

This is how they are employed:

\texttt{J.\_\_|\_\_E.\_\_|\_\_Doe}

\texttt{K.\_\_H.\_\_Mustermann}

### 3.13.5 Bibliography Filters and Citation Labels

The citation labels generated by this package are assigned to the full list of references before it is split up by any bibliography filters. They are guaranteed to be unique across the entire document (or a \texttt{refsection} environment), no matter how many bibliography filters you are using. When using a numeric citation scheme, however, this will most likely lead to discontinuous numbering in split bibliographies. Use the \texttt{defernumpers} package option to avoid this problem. If this option is enabled, numeric labels are assigned the first time an entry is printed in any bibliography.

### 3.13.6 Active Characters in Bibliography Headings

Packages using active characters, such as \texttt{babel}, \texttt{polyglossia}, \texttt{csquotes}, or \texttt{underscore}, usually do not make them active until the beginning of the document body to avoid interference with other packages. A typical example of such an active character is the Ascii quote ", which is used by various language modules of the \texttt{babel/polyglossia} packages. If shorthands such as ",< and ",a are used in the argument to \texttt{\defbibheading} and the headings are defined in the
document preamble, the non-active form of the characters is saved in the heading
definition. When the heading is typeset, they do not function as a command but
are simply printed literally. The most straightforward solution consists in mov-
ing \defbibheading after \begin{document}. Alternatively, you may use
babel’s \shorthandon and \shorthandoff commands to temporarily make
the shorthands active in the preamble. The above also applies to bibliography notes
and the \defbibnote command.

3.13.7 Grouping in Reference Sections and Segments

All LaTeX environments enclosed in \begin and \end form a group. This may have
undesirable side effects if the environment contains anything that does not expect to
be used within a group. This issue is not specific to refsection and refsegment
environments, but it obviously applies to them as well. Since these environments
will usually enclose much larger portions of the document than a typical itemize
or similar environment, they are simply more likely to trigger problems related to
grouping. If you observe any malfunctions after adding refsection environments
to a document (for example, if anything seems to be ‘trapped’ inside the environment),
try the following syntax instead:

\chapter{...}
\refsection
...
\endrefsection

This will not from a group, but otherwise works as usual. As far as biblatex
is concerned, it does not matter which syntax you use. The alternative syntax
is also supported by the refsegment environment. Note that the commands
\newrefsection and \newrefsegment do not form a group. See §§3.7.4 and
3.7.5 for details.

3.14 Using the fallback BibTeX backend

To utilise all of the features described here, biblatex must be used with the biber
program as a backend. Indeed, the documentation in general assumes this. However,
for a limited subset of use cases it is possible to use the long-established BibTeX
program (either the 7-bit bibtex or 8-bit bibtex8) as the supporting backend.
This works in much the same way as for biber with the only proviso being that
BibTeX is much more limited as a backend.

Using BibTeX as the backend requires that the option backend=bibtex or
backend=bibtex8 is given at load time. The biblatex package will then write
appropriate data for use by BibTeX into the auxiliary file(s) and a special data file
(automatically included in those to be read by BibTeX). The BibTeX (8) program
should then be run on each auxiliary file: biblatex will list all of the required
files in the log.

Key limitations of the BibTeX backend are:

- Sorting is global and is limited to Ascii ordering
- No re-encoding is possible and thus database entries must be in LICR form to
  work reliably
- The data model is fixed
• Cross-referencing is more limited and entry sets must be written into the .bib file

• Fixed memory capacity (using the --wolfgang option with bibtex8 is strongly recommended to minimize the likelihood of an issue here)

4 Author Guide

This part of the manual documents the author interface of the biblatex package. The author guide covers everything you need to know in order to write new citation and bibliography styles or localisation modules. You should read the user guide first before continuing with this part of the manual.

4.1 Overview

Before we get to the commands and facilities provided by biblatex, we will have a look at some of its fundamental concepts. The biblatex package uses auxiliary files in a special way. Most notably, the bbl file is used differently and there is no concept of a style-dependent bst file. With LaTeX’s standard bibliographic facilities, a document includes any number of citation commands in the document body plus \bibliographystyle and \bibliography, usually towards the end of the document. The location of the former is arbitrary, the latter marks the spot where the list of references is to be printed:

\documentclass{...}
\begin{document}
\cite{...}
...
\bibliographystyle{...}
\bibliography{...}
\end{document}

Processing this files requires that a certain procedure be followed. This procedure is as follows:

1. Run latex: On the first run, \bibstyle and \bibdata commands are written to the aux file, along with \citation commands for all citations. At this point, the references are undefined because LaTeX is waiting for BibTeX to supply the required data. There is also no bibliography yet.

2. Run bibtex: BibTeX writes a thebibliography environment to the bbl file, supplying all entries from the bib file which were requested by the \citation commands in the aux file.

3. Run latex: Starting with the second run, the \bibitem commands in the thebibliography environment write one \bibcite command for each bibliography entry to the aux file. These \bibcite commands define the citation labels used by \cite. However, the references are still undefined because the labels are not available until the end of this run.

4. Run latex: Starting with the third run, the citation labels are defined as the aux file is read in at the end of the preamble. All citations can now be printed.
Note that all bibliographic data is written to the \texttt{bbl} file in the final format. The \texttt{bbl} file is read in and processed like any printable section of the document. For example, consider the following entry in a \texttt{bib} file:

\begin{verbatim}
@Book{companion,
    author   = {Michel Goossens and Frank Mittelbach and
                 Alexander Samarin},
    title    = {The \LaTeX\ Companion},
    publisher = {Addison-Wesley},
    address  = {Reading, Mass.},
    year     = {1994},
}
\end{verbatim}

With the \texttt{plain.bst} style, \LaTeX{} exports this entry to the \texttt{bbl} file as follows:

\begin{verbatim}
\bibitem{companion}
Michel Goossens, Frank Mittelbach, and Alexander
Samarin.
\newblock \textit{The \LaTeX\ Companion}.
\end{verbatim}

By default, \LaTeX{} generates numeric citation labels, hence \texttt{\bibitem} writes lines such as the following to the \texttt{aux} file:

\begin{verbatim}
\bibcite{companion}{1}
\end{verbatim}

Implementing a different citation style implies that more data has to be transferred via the \texttt{aux} file. With the \texttt{natbib} package, for example, the \texttt{aux} file contains lines like this one:

\begin{verbatim}
\bibcite{companion}{{1}{1994}{{Goossens et\text{-}al.}}{{Goossens, Mittelbach, and
Samarin}}}
\end{verbatim}

The \texttt{biblatex} package supports citations in any arbitrary format, hence citation commands need access to all bibliographic data. What this would mean within the scope of the procedure outlined above becomes obvious when looking at the output of the \texttt{jurabib} package which also makes all bibliographic data available in citations:

\begin{verbatim}
\bibcite{companion}{{Goossens\jbbfsasep Mittelbach\jbbstasep Samarin}{0}{book}{1994}{Reading, Mass.\bpubaddr Addison-Wesley}{The \LaTeX\ Companion}{Goossens}{Michel}{M.}{\bibbfsasep\bibnf Mittelbach}{Frank}{F.}{Samarin}{Alexander}{A.}{\bibftfont{The}
\end{verbatim}
In this case, the contents of the entire \texttt{thebibliography} environment are effectively transferred via the \texttt{aux} file. The data is read from the \texttt{bbl} file, written to the \texttt{aux} file, read back from the \texttt{aux} file and then kept in memory. The bibliography itself is still generated as the \texttt{bbl} file is read in. The \texttt{biblatex} package would also be forced to cycle all data through the \texttt{aux} file. This implies processing overhead and is also redundant because the data has to be kept in memory anyway.

The traditional procedure is based on the assumption that the full bibliographic data of an entry is only required in the bibliography and that all citations use short labels. This makes it very effective in terms of memory requirements, but it also implies that it does not scale well. That is why \texttt{biblatex} takes a different approach. First of all, the document structure is slightly different. Instead of using \texttt{\bibliography} in the document body, database files are specified in the preamble with \texttt{\addbibresource}, \texttt{\bibliographystyle} is omitted entirely (all features are controlled by package options), and the bibliography is printed using \texttt{\printbibliography}:

\begin{verbatim}
\documentclass{...}
\usepackage[...]{biblatex}
\addbibresource{...}
\begin{document}
\cite{...}
...
\printbibliography
\end{document}
\end{verbatim}

In order to streamline the whole procedure, \texttt{biblatex} essentially employs the \texttt{bbl} file like an \texttt{aux} file, rendering \texttt{\bibcite} obsolete. We then get the following procedure:

1. Run \texttt{latex}: The first step is similar to the traditional procedure described above: \texttt{\bibstye} and \texttt{\bibdata} commands are written to the \texttt{bcf} file, along with \texttt{\citation} commands for all citations. We then wait for the backend to supply the required data.

2. Run \texttt{biber}: The backend supplies those entries from the \texttt{bib} file which were requested by the \texttt{\citation} commands in the auxiliary file. However, it does not write a printable bibliography to the \texttt{bbl} file, but rather a structured representation of the bibliographic data. Just like an \texttt{aux} file, this \texttt{bbl} file does not print anything when read in. It merely puts data in memory.

3. Run \texttt{latex}: Starting with the second run, the \texttt{bbl} file is processed right at the beginning of the document body, just like an \texttt{aux} file. From this point on, all bibliographic data is available in memory so that all citations can be printed right away.\footnote{If the \texttt{defernumbers} package option is enabled \texttt{biblatex} uses an algorithm similar to the traditional procedure to generate numeric labels. In this case, the numbers are assigned as the bibliography is printed and then cycled through the backend auxiliary file. It will take an additional LaTeX run for them to be picked up in citations.} The citation commands have access to the complete bibliographic
data, not only to a predefined label. The bibliography is generated from memory using the same data and may be filtered or split as required.

Let’s consider the sample entry given above once more:

```latex
@Book{companion,
  author = {Michel Goossens and Frank Mittelbach and Alexander Samarin},
  title = {The LaTeX Companion},
  publisher = {Addison-Wesley},
  address = {Reading, Mass.},
  year = {1994},
}
```

This entry is essentially exported in the following format:

```latex
\entry{companion}{book}{
\labelname{author}{3}{}{%
  {{uniquename=0,hash=...}{Goossens}{G.}{Michel}{M ↝ .}{}}{}}%
  {{uniquename=0,hash=...}{Mittelbach}{M.}{Frank}{F ↝ .}{}}{}}%
  {{uniquename=0,hash=...}{Samarin}{S.}{Alexander}{A ↝ .}{}}{}}%
}\name{author}{3}{}{%
  {{uniquename=0,hash=...}{Goossens}{G.}{Michel}{M ↝ .}{}}{}}%
  {{uniquename=0,hash=...}{Mittelbach}{M.}{Frank}{F ↝ .}{}}{}}%
  {{uniquename=0,hash=...}{Samarin}{S.}{Alexander}{A ↝ .}{}}{}}%
}\list{publisher}{1}{% {Addison-Wesley}\%
}\list{location}{1}{% {Reading, Mass.}\%
}\field{title}{The LaTeX Companion}\field{year}{1994}\endentry
```

As seen in this example, the data is presented in a structured format that resembles the structure of a `.bib` file to some extent. At this point, no decision concerning the final format of the bibliography entry has been made. The formatting of the bibliography and all citations is controlled by LaTeX macros, which are defined in bibliography and citation style files.

### 4.2 Bibliography Styles

A bibliography style is a set of macros which print the entries in the bibliography. Such styles are defined in files with the suffix `.bbx`. The `biblatex` package loads
the selected bibliography style file at the end of the package. Note that a small repertory of frequently used macros shared by several of the standard bibliography styles is included in `biblatex.def`. This file is loaded at the end of the package as well, prior to the selected bibliography style.

### 4.2.1 Bibliography Style Files

Before we go over the individual components of a bibliography style, consider this example of the overall structure of a typical `bbx` file:

```latex
\ProvidesFile{example.bbx}[2006/03/15 v1.0 biblatex ↪ bibliography style]
\defbibenvironment{bibliography}
  {...}
  {...}
  {...}
\defbibenvironment{shorthand}
  {...}
  {...}
  {...}
\InitializeBibliographyStyle{...}
\DeclareBibliographyDriver{article}{...}
\DeclareBibliographyDriver{book}{...}
\DeclareBibliographyDriver{inbook}{...}
...
\DeclareBibliographyDriver{shorthand}{...}
\endinput
```

The main structure of a bibliography style file consists of the following commands:

\\texttt{\textbackslash RequireBibliographyStyle\{\langle style\rangle\}}

This command is optional and intended for specialized bibliography styles built on top of a more generic style. It loads the bibliography style `style.bbx`.

\\texttt{\textbackslash InitializeBibliographyStyle\{\langle code\rangle\}}

Specifies arbitrary \langle code\rangle to be inserted at the beginning of the bibliography, but inside the group formed by the bibliography. This command is optional. It may be useful for definitions which are shared by several bibliography drivers but not used outside the bibliography. Keep in mind that there may be several bibliographies in a document. If the bibliography drivers make any global assignments, they should be reset at the beginning of the next bibliography.

\\texttt{\textbackslash DeclareBibliographyDriver\{\langle entrytype\rangle\}{\langle code\rangle}}

Defines a bibliography driver. A `driver` is a macro which handles a specific entry type (when printing bibliography lists) or a specific named bibliography list (when printing bibliography lists). The \langle entrytype\rangle corresponds to the entry type used in `bib` files, specified in lowercase letters (see §2.1). The \langle entrytype\rangle argument may also be an asterisk. In this case, the driver serves as a fallback which is used if no specific driver for the entry type has been defined. The \langle code\rangle is arbitrary code which typesets all bibliography entries of the respective \langle entrytype\rangle. This command is mandatory. Every bibliography style should provide a driver for each entry type.
\DeclareBibliographyAlias{⟨alias⟩}{⟨entrytype⟩}

If a bibliography driver covers more than one entry type, this command may be used to define an alias where ⟨entrytype⟩ is the name of a defined driver. This command is optional. The ⟨alias⟩ argument may also be an asterisk. In this case, the ⟨entrytype⟩ driver serves as a fallback which is used if no specific driver for an entry has been defined.

\DeclareBibliographyOption[⟨datatype⟩]{⟨key⟩}{⟨value⟩}{⟨code⟩}

This command defines additional preamble options in ⟨key⟩=⟨value⟩ format. The ⟨key⟩ is the option key. The ⟨code⟩ is arbitrary TeX code to be executed whenever the option is used. The value passed to the option is passed on to the ⟨code⟩ as #1. The optional ⟨value⟩ is a default value to be used if the bare key is given without any value. This is useful for boolean switches. The ⟨datatype⟩ is a the datatype for the option. If omitted, it defaults to ‘boolean’. For example, with a definition like the following:

\DeclareBibliographyOption[boolean]{somekey}[true]{...}

giving ‘somekey’ without a value is equivalent to ‘somekey=true’. Valid ⟨datatype⟩ values are defined in the default biber Datamodel as:

\DeclareDatamodelConstant[type=list]{optiondatatypes}{↪ boolean,integer,string,xml}

\DeclareEntryOption[⟨datatype⟩]{⟨key⟩}{⟨value⟩}{⟨code⟩}

Similar to \ DeclareBibliographyOption but defines options which are settable on a per-entry basis in the options field from §2.2.3. The ⟨code⟩ is executed whenever biblatex prepares the data of the entry for use by a citation command or a bibliography driver.

4.2.2 Bibliography Environments

Apart from defining bibliography drivers, the bibliography style is also responsible for the environments which control the layout of the bibliography and bibliography lists. These environments are defined with \defbibenvironment. By default, \printbibliography uses the environment bibliography. Here is a definition suitable for a bibliography style which does not print any labels in the bibliography:

\defbibenvironment{bibliography}
{
\list
{}{
\setlength{\leftmargin}{\bibhang}\% 
\setlength{\itemindent}{-\leftmargin}\%
\setlength{\itemsep}{\bibitemsep}\%
\setlength{\parsep}{\bibparsep}\
}{\endlist}
}\item
This definition employs a list environment with hanging indentation, using the \bibhang length register provided by biblatex. It allows for a certain degree of configurability by using \bibitemsep and \bibparsep, two length registers provided by biblatex for this very purpose (see §4.10.3). The authoryear and authortitle bibliography styles use a definition similar to this example.

\defbibenvironment{bibliography}
\begin{list}
\setlength{\labelwidth}{\labelnumberwidth}%
\setlength{\leftmargin}{\labelwidth}%
\setlength{\labelsep}{\biblabelsep}%
\addtolength{\leftmargin}{\labelsep}%
\setlength{\itemsep}{\bibitemsep}%
\setlength{\parsep}{\bibparsep}%
\renewcommand*{\makelabel}[1]{\hss##1}}
{\endlist}
{\item}

Some bibliography styles print labels in the bibliography. For example, a bibliography style designed for a numeric citation scheme will print the number of every entry such that the bibliography looks like a numbered list. In the first example, the first argument to \list was empty. In this example, we need it to insert the number, which is provided by biblatex in the labelnumber field. We also employ several length registers and other facilities provided by biblatex, see §§4.10.4 and 4.10.5 for details. The numeric bibliography style uses the definition given above. The alphabetic style is similar, except that labelnumber is replaced by labelalpha and labelnumberwidth by labelalphawidth.

Bibliography lists are handled in a similar way. \printbiblist uses the environment named after the bibliography list by default. A typical example is given below. See §§4.10.4 and 4.10.5 for details on the length registers and facilities used in this example.

\defbibenvironment{shorthand}
\begin{list}
\setlength{\labelwidth}{\shorthandwidth}%
\setlength{\leftmargin}{\labelwidth}%
\setlength{\labelsep}{\biblabelsep}%
\addtolength{\leftmargin}{\labelsep}%
\setlength{\itemsep}{\bibitemsep}%
\setlength{\parsep}{\bibparsep}%
\renewcommand*{\makelabel}[1]{##1\hss}}
{\endlist}
{\item}

4.2.3 Bibliography Drivers

Before we go over the commands which form the data interface of the biblatex package, it may be instructive to have a look at the structure of a bibliography driver.
Note that the example given below is greatly simplified, but still functional. For the sake of readability, we omit several fields which may be part of a @book entry and also simplify the handling of those which are considered. The main point is to give you an idea of how a driver is structured. For information about the mapping of the BibTeX file format fields to \texttt{biblatex}'s data types, see §2.2.

\begin{verbatim}
\DeclareBibliographyDriver{book}{% 
  \printnames{author}% 
  \newunit\newblock 
  \printfield{title}% 
  \newunit\newblock 
  \printlist{publisher}% 
  \newunit 
  \printlist{location}% 
  \newunit 
  \printfield{year}% 
  \finentry}
\end{verbatim}

The standard bibliography styles employ two bibliography macros \texttt{begentry} and \texttt{finentry}:

\begin{verbatim}
\DeclareBibliographyDriver{entrytype}{% 
  \usebibmacro{begentry} ... 
  \usebibmacro{finentry}}
\end{verbatim}

with the default definitions

\begin{verbatim}
\newbibmacro*{begentry}{}
\newbibmacro*{finentry}{\finentry}
\end{verbatim}

Use of these macros is recommended for easy hooks into the beginning and end of the driver.

Returning to the driver for the \texttt{book} entrytype above, there is still one piece missing: the formatting directives used by \texttt{\printnames, \printlist, and \printfield}. To give you an idea of what a formatting directive looks like, here are some fictional ones used by our sample driver. Field formats are straightforward, the value of the field is passed to the formatting directive as an argument which may be formatted as desired. The following directive will simply wrap its argument in an \texttt{\emph} command:

\begin{verbatim}
\DeclareFieldFormat{title}{\emph{#1}}
\end{verbatim}

List formats are slightly more complex. After splitting up the list into individual items, \texttt{biblatex} will execute the formatting directive once for every item in the list. The item is passed to the directive as an argument. The separator to be inserted between the individual items in the list is also handled by the corresponding directive, hence we have to check whether we are in the middle of the list or at the end when inserting it.
Formatting directives for names are similar to those for literal lists. Names depend on the datamodel constant ‘nameparts’ which has the default definition:

\DeclareDatamodelConstant[type=list]{nameparts}{prefix,family,suffix,given}

This can be customised to add more name parts to deal with things like patronyms (see the example file 93-nameparts.tex). Naturally this needs an extended name format for data sources. \biblatexml\ (§ D) handles this natively and there is an extended name format which can handle custom nameparts available when using \biber\ (see \biber\ documentation).

Inside name formats, the nameparts constant declaration makes available two macros for each name part defined in the datamodel:

\namepart<namepart>
\namepart<namepart>i

The name formatting directive is executed once for each name in the name list. Here is an example:

\DeclareNameFormat{author}{%
  \ifthenelse{\value{listcount}=1}
  {\namepartfamily%
    \ifblank{\namepartgiven}{}{
      → \addcomma\space\namepartgiven}
    \ifblank{\namepartgiven}{}{\namepartgiven\space}
    \namepartfamily%}
  \ifthenelse{\value{listcount}<\value{liststop}}{\addcomma\space}{}}

The above directive reverses the name of the first author ("Last, First") and prints the remaining names in their regular sequence ("First Last"). Note that the only component which is guaranteed to be available is the last name, hence we have to check which parts of the name are actually present. If a certain name part is not available, the corresponding macro will be empty. As with directives for literal lists, the separator to be inserted between the individual items in the name list is also handled by the formatting directive, hence we have to check whether we are in the middle of the list or at the end when inserting it. This is what the second \ifthenelse test does.
4.2.4 Special Fields

The following lists and fields are used by \texttt{biblatex} to pass data to bibliography drivers and citation commands. They are not used in \texttt{bib} files but defined automatically by the package. From the perspective of a bibliography or citation style, they are not different from the fields in a \texttt{bib} file.

4.2.4.1 Generic Fields

\begin{verbatim}
<datetype>dateunspecified field (string)

If \texttt{<datetype>date} held an EDTF 5.2.2 ‘unspecified’, this field will be set to one of \texttt{yearindecade}, \texttt{yearincentury}, \texttt{monthinyear}, \texttt{dayinmonth} or \texttt{dayinyear} which specifies the granularity of the unspecified information. These strings can be tested for and along with the date ranges which are automatically created for such ‘unspecified’ dates, a style may choose to format the date in a special way. See §2.3.8. For example, an entry with dates such as:

\begin{verbatim}
@book{key,  
  date = {19uu},  
  origdate = {199u}
}
\end{verbatim}

would result in the same information in the \texttt{.bbl} as:

\begin{verbatim}
@book{key,  
  date = {1900/1999},  
  origdate = {1990/1999}
}
\end{verbatim}

but would additionally have the field \texttt{dateunspecified} set to ‘yearincentury’ and \texttt{origdateunspecified} set to ‘yearindecade’. This information could be used to render the \texttt{date} as perhaps ‘20th century’ and \texttt{origdate} as ‘The 1990s’, information which cannot be derived from the date ranges alone. Since such auto-generated ranges have a know values, given the ‘unspecified’ meta-information, it is relatively easy to use the range values to format special cases. While the standard styles do not do this, examples are given in the file \texttt{96-dates.tex}.
\end{verbatim}

entrykey field (string)

The entry key of an item in the \texttt{bib} file. This is the string used by \texttt{biblatex} and the backend to identify an entry in the \texttt{bib} file.

childentrykey field (string)

When citing a subentry of an entry set, \texttt{biblatex} provides the data of the parent \texttt{@set} entry to citation commands. This implies that the \texttt{entrykey} field holds the entry key of the parent. The entry key of the child entry being cited is provided in the \texttt{childentrykey} field. This field is only available when citing a subentry of an entry set.

labelnamesource field (literal)

Holds the name of the field used to populate \texttt{labelname}, determined by \texttt{\DeclareLabelname}.\end{verbatim}
**labeltitlesource** field (literal)

Holds the name of the field used to populate `labeltitle`, determined by `\DeclareLabeltitle`.

**labeldatesource** field (literal)

Holds one of:

- The prefix coming before ‘date’ of the date field name chosen by `\DeclareLabeldate`
- The name of a field
- A literal or localisation string

Normally holds the prefix coming before ‘date’ of the date field name chosen by `\DeclareLabeldate`. For example, if the labeldate field is `eventdate`, then `labeldatesource` will be ‘event’. In case `\DeclareLabeldate` selects the date field, then `labeldatesource` will be defined but will be an empty string as the prefix coming before ‘date’ in the date label name is empty. This is so that the contents of `labeldatesource` can be used in constructing references to the field which `\DeclareLabeldate` chooses. Since `\DeclareLabeldate` can also select literal strings for fallbacks, `labeldatesource` may not refer to a field or may be undefined. Bear in mind that `\DeclareLabeldate` can also be used to select non-date fields as a fallback and so `labeldatesource` might contain a field name. So, in summary, the rules are

\begin{verbatim}
\iffieldeundef{labeldatesource}
{} % labeldate package option is not set
{\iffieldeundef{\thefield{labeldatesource}year}
  % \DeclareLabeldate resolved to either a literal/
  \rightarrow localisation
  % string or a non-date field since
  % if a date is defined by a date field, there is
  % at least a year
  {\iffieldeundef{\thefield{labeldatesource}}
    {} % \DeclareLabeldate resolved to a literal/
    \rightarrow localisation string
    {} % \DeclareLabeldate resolved to a non-date
    \rightarrow field
  }
  {} % \DeclareLabeldate resolved a date field name
  \rightarrow prefix like "" or "orig"
}
\end{verbatim}

**entrytype** field (string)

The entry type (@book, @inbook, etc.), given in lowercase letters.

**childentrytype** field (string)

When citing a subentry of an entry set, `biblatex` provides the data of the parent @set entry to citation commands. This implies that the `entrytype` field holds the entry type of the parent. The entry type of the child entry being cited is provided in
the childentrytype field. This field is only available when citing a subentry of an entry set.

**entrysetcount** field (integer)

This field holds an integer indicating the position of a set member in the entry set (starting at 1). This field is only available in the subentries of an entry set.

**hash** field (string)

This field is special in that it is only available locally in name formatting directives. It holds a hash string which uniquely identifies individual names in a name list. This information is available for all names in all name lists. See also namehash and fullhash.

**namehash** field (string)

A hash string which uniquely identifies the labelname list. This is useful for recurrence checks. For example, a citation style which replaces recurrent authors or editors with a string like 'idem' could save the namehash field with \savefield and use it in a comparison with \iffieldequals later (see §§ 4.6.1 and 4.6.2). The namehash is derived from the truncated labelname list, i.e., it is responsive to maxnames and minnames. See also hash and fullhash.

**<namelist>namehash** field (string)

As namehash for the name list called ‘namelist’.

**fullhash** field (string)

A hash string which uniquely identifies the labelname list. This fields differs from namehash in two details: 1) The shortauthor and shorteditor lists are ignored when generating the hash. 2) The hash always refers to the full list, ignoring maxnames and minnames. See also hash and namehash.

**<namelist>fullhash** field (string)

As fullhash for the name list called ‘namelist’.

**pageref** list (literal)

If the backref package option is enabled, this list holds the page numbers of the pages on which the respective bibliography entry is cited. If there are refsection environments in the document, the back references are local to the reference sections.

**sortinit** field (literal)

This field holds the initial character of the information used during sorting.

**sortinitshash** field (string)

This field holds a hash of the (locale-specific) Unicode Collation Algorithm primary weight of the first extended grapheme cluster (essentially the first character) of the string used during sorting. This is useful when subdividing the bibliography alphabetically and is used internally by \bibinitsep (see § 3.10.4).

**clonesourcekey** field (string)

This field holds the entry key of the entry from which an entry was cloned. Clones are created for entries which are mentioned in related fields as part of related entry processing, for example.
4.2.4.2 Fields for Use in Citation Labels

**labelalpha** field (literal)

A label similar to the labels generated by the `alpha.bst` style of traditional BibTeX. This default label consists of initials drawn from the `labelname` list plus the last two digits of the publication year. The `label` field may be used to override its non-numeric portion. If the `label` field is defined, `biblatex` will use its value and append the last two digits of the publication year when generating `labelalpha`. The `shorthand` field may be used to override the entire label. If defined, `labelalpha` is the `shorthand` rather than an automatically generated label. Users can specify a template used to construct the alphabetic label (see §4.5.5) and the default template mirrors the format mentioned for `bibtex` above. A complete ‘alphabetic’ label consists of the fields `labelalpha` plus `extraalpha`. Note that the `labelalpha` and `extraalpha` fields need to be requested with the package option `labelalpha` (§ 3.1.2.3). See also `extraalpha` as well as \labelalphaothers in § 3.10.1.

**extraalpha** field (integer)

The ‘alphabetic’ citation scheme usually requires a letter to be appended to the label if the bibliography contains two or more works by the same author which were all published in the same year. In this case, the `extraalpha` field holds an integer which may be converted to a letter with \mknumalph or formatted in some other way. This field is similar to the role of `extrayear` in the author-year scheme. A complete ‘alphabetic’ label consists of the fields `labelalpha` plus `extraalpha`. Note that the `labelalpha` and `extraalpha` fields need to be requested with the package option `labelalpha`, see § 3.1.2.3 for details. See also `labelalpha` as well as \labelalphaothers in § 3.10.1. Table 7 summarises the various extra* disambiguation counters and what they track.

**labelname** list (name)

The name to be printed in citations. This list is a copy of either the `shortauthor`, the `author`, the `shorteditor`, the `editor`, or the `translator` list, which are normally checked for in this order. If no authors and editors are available, this list is undefined. Note that this list is also responsive to the `use<name>`, options, see § 3.1.3. Citation styles should use this list when printing the name in a citation. This list is provided for convenience only and does not carry any additional meaning. This field may be customized. See § 4.5.10 for details.

**labelnumber** field (literal)

The number of the bibliography entry, as required by numeric citation schemes. If the `shorthand` field is defined, `biblatex` does not assign a number to the respective entry. In this case `labelnumber` is the `shorthand` rather than a number. Numeric styles must use the value of this field instead of a counter. Note that this field needs to be requested with the package option `labelnumber`, see § 3.1.2.3 for details. Also see the package option `defernumbers` in § 3.1.2.1.

**labelprefix** field (literal)

If the `labelprefix` option of \newrefcontext has been set in order to prefix all entries in a subbibliography with a fixed string, this string is available in the `labelprefix` field of all affected entries. If no prefix has been set, the `labelprefix` field of the respective entry is undefined. See the `labelprefix`
option of \newrefcontext in § 3.7.10 for details. If the shorthand field is defined, biblatex does not assign the prefix to the labelprefix field of the respective entry. In this case, the labelprefix field is undefined.

**labeltitle** field (literal)

The printable title of a work. In some circumstances, a style might need to choose a title from a list of a possible title fields. For example, citation styles printing short titles may want to print the shorttitle field if it exists but otherwise print the title field. The list of fields to be considered when constructing labeltitle may be customized. See § 4.5.10 for details. Note that the extratitle field needs to be requested with the package option labeltitle, see § 3.1.2.3 for details. See also extratitle. Note also that the extratitleyear field needs to be requested with the package option labeltitleyear. See also extratitleyear.

**extratitle** field (integer)

It is sometimes useful, for example in author-title citation schemes, to be able to disambiguate works with the same title. For works by the same labelname with the same labeltitle, the extratitle field holds an integer which may be converted to a letter with \mnumalph or formatted in some other way (or it can be merely used as a flag to say that some other field such as a date should be used in conjunction with the labeltitle field). This field is undefined if there is only one work with the same labeltitle by the same labelname in the bibliography. Note that the extratitle field needs to be requested with the package option labeltitle, see § 3.1.2.3 for details. See also labeltitle. Table 7 summarises the various extra* disambiguation counters and what they track.

**extratitleyear** field (integer)

It is sometimes useful, for example in author-title citation schemes, to be able to disambiguate works with the same title in the same year but with no author. For works with the same labeltitle and with the same labelyear, the extratitleyear field holds an integer which may be converted to a letter with \mnumalph or formatted in some other way (or it can be merely used as a flag to say that some other field such as a publisher should be used in conjunction with the labelyear field). This field is undefined if there is only one work with the same labeltitle and labelyear in the bibliography. Note that the extratitleyear field needs to be requested with the package option labeltitleyear, see § 3.1.2.3 for details. See also labeltitleyear. Table 7 summarises the various extra* disambiguation counters and what they track.

**labelyear** field (literal)

The year of the date field selected by \DeclareLabeldate (§ 4.5.10) or the year field, for use in author-year labels. A complete author-year label consists of the fields labelyear plus extrayear. Note that the labelyear and extrayear fields need to be requested with the package option labeldateparts, see § 3.1.2.3 for details. See also extrayear.

**labelendyear** field (literal)

The end year of the date field selected by \DeclareLabeldate (§ 4.5.10) if the selected date is a range.
labelmonth field (datepart)

The month of the date field selected by `\DeclareLabeldate` (§ 4.5.10), or the month field for use in author-year labels. Note that the labelmonth field needs to be requested with the package option `labeldateparts`, see § 3.1.2.3 for details.

labelendmonth field (datepart)

The end month of the date field selected by `\DeclareLabeldate` (§ 4.5.10) if the selected date is a range.

labday field (datepart)

The month of the date field selected by `\DeclareLabeldate` (§ 4.5.10) for use in author-year labels. Note that the labday field needs to be requested with the package option `labeldateparts`, see § 3.1.2.3 for details.

celendday field (datepart)

The end day of the date field selected by `\DeclareLabeldate` (§ 4.5.10) if the selected date is a range.

extrayear field (integer)

The author-year citation scheme usually requires a letter to be appended to the year if the bibliography contains two or more works by the same author which were all published in the same year. In this case, the extrayear field holds an integer which may be converted to a letter with `\mknumalph` or formatted in some other way. This field is undefined if there is only one work by the author in the bibliography or if all works by the author have different publication years. A complete author-year label consists of the fields labelyear plus extrayear. Note that the labelyear and extrayear fields need to be requested with the package option `labeldateparts`, see § 3.1.2.3 for details. See also labelyear. Table 7 summarises the various extra* disambiguation counters and what they track.

4.2.4.3 Date Component Fields  Note that it is possible to define new date fields in the datamodel which behave exactly like the default data model date fields described in this section.

See table 10 for an overview of how the date fields in bib files are related to the date fields provided by the style interface. When testing for a field like origdate in a style, use code like:

\[
\text{\textbackslash iffieldundef{origyear}{...}{...}}
\]

This will tell you if the corresponding date is defined at all. This test:

\[
\text{\textbackslash iffieldundef{origendyear}{...}{...}}
\]

will tell you if the corresponding date is defined and a (fully specified) range. This test:

\[
\text{\textbackslash iffieldequalstr{origendyear}{}}{...}{...}{...}
\]
will tell you if the corresponding date is defined and an open-ended range. Open-ended ranges are indicated by an empty `endyear` component (as opposed to an undefined `endyear` component). See §2.3.8 and table 3 on page 37 for further examples.
<table>
<thead>
<tr>
<th>Field</th>
<th>Value (Example)</th>
<th>Field</th>
<th>Value (Example)</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>1988</td>
<td>day</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>month</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>year</td>
<td>1988</td>
</tr>
<tr>
<td></td>
<td></td>
<td>season</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endday</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endmonth</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endyear</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endseason</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hour</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>minute</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>second</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>timezone</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endhour</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endminute</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endseason</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endtimezone</td>
<td>undefined</td>
</tr>
<tr>
<td>date</td>
<td>1997/</td>
<td>day</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>month</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>year</td>
<td>1997</td>
</tr>
<tr>
<td></td>
<td></td>
<td>season</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endday</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endmonth</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>empty</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>endyear</td>
<td>empty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endseason</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hour</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>minute</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>second</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>timezone</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endhour</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endminute</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endseason</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>endtimezone</td>
<td>undefined</td>
</tr>
<tr>
<td>uridate</td>
<td>2009-01-31</td>
<td>urlday</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlmth</td>
<td>01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlyear</td>
<td>2009</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlseason</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlendday</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlendmonth</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlendyear</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlendseason</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlhour</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlminute</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlsecond</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urltimesone</td>
<td>undefined</td>
</tr>
<tr>
<td>uridate</td>
<td>2009-01-31T15:34:04Z</td>
<td>urlday</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlmth</td>
<td>01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlyear</td>
<td>2009</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlseason</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlendday</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlendmonth</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlendyear</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlendseason</td>
<td>undefined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlhour</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlminute</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td></td>
<td>urlsecond</td>
<td>04</td>
</tr>
</tbody>
</table>
Table 10: Date Interface

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hour</td>
<td>This field holds the hour component of the date field. If the date is a range, it holds the hour component of the start date.</td>
</tr>
<tr>
<td>minute</td>
<td>This field holds the minute component of the date field. If the date is a range, it holds the minute component of the start date.</td>
</tr>
<tr>
<td>second</td>
<td>This field holds the second component of the date field. If the date is a range, it holds the second component of the start date.</td>
</tr>
<tr>
<td>timezone</td>
<td>This field holds the timezone component of the date field. If the date is a range, it holds the timezone component of the start date.</td>
</tr>
<tr>
<td>day</td>
<td>This field holds the day component of the date field. If the date is a range, it holds the day component of the start date.</td>
</tr>
<tr>
<td>month</td>
<td>This field is the month as given in the database file or it holds the month component of the date field. If the date is a range, it holds the month component of the start date.</td>
</tr>
<tr>
<td>year</td>
<td>This field is the year as given in the database file or it holds the year component of the date field. If the date is a range, it holds the year component of the start date.</td>
</tr>
<tr>
<td>season</td>
<td>This field holds the season component of the date field as specified by EDF 5.2.5 (§2.3.8). It contains a season localisation string (§4.9.2.21). If the date is a range, it holds the season component of the start date.</td>
</tr>
<tr>
<td>endhour</td>
<td>If the date specification in the date field is a range, this field holds the hour component of the end date.</td>
</tr>
<tr>
<td>.endminute</td>
<td>If the date specification in the date field is a range, this field holds the minute component of the end date.</td>
</tr>
<tr>
<td>endsecond</td>
<td>If the date specification in the date field is a range, this field holds the second component of the end date.</td>
</tr>
<tr>
<td>endtimezone</td>
<td>If the date specification in the date field is a range, this field holds the timezone component of the end date.</td>
</tr>
<tr>
<td>endday</td>
<td>If the date specification in the date field is a range, this field holds the day component of the end date.</td>
</tr>
<tr>
<td>endmonth</td>
<td>If the date specification in the date field is a range, this field holds the month component of the end date.</td>
</tr>
</tbody>
</table>
endyear field (datepart)
If the date specification in the date field is a range, this field holds the year component of the end date. A blank (but defined) endyear component indicates an open ended date range.

dedseason field (datepart)
If the date specification in the date field is a range, this field holds the season component of the end date as specified by EDTF 5.2.5 (§ 2.3.8). It contains a season localisation string (§ 4.9.2.21). A blank (but defined) endseason component indicates an open ended date range.

orighour field (datepart)
This field holds the hour component of the origdate field. If the date is a range, it holds the hour component of the start date.

origminute field (datepart)
This field holds the minute component of the origdate field. If the date is a range, it holds the minute component of the start date.

origsecond field (datepart)
This field holds the second component of the origdate field. If the date is a range, it holds the second component of the start date.

origtimezone field (datepart)
This field holds the timezone component of the origdate field. If the date is a range, it holds the timezone component of the start date.

origday field (datepart)
This field holds the day component of the origdate field. If the date is a range, it holds the day component of the start date.

origmonth field (datepart)
This field holds the month component of the origdate field. If the date is a range, it holds the month component of the start date.

origyear field (datepart)
This field holds the year component of the origdate field. If the date is a range, it holds the year component of the start date.

origseason field (datepart)
This field holds the season component of the origdate field as specified by EDTF 5.2.5 (§ 2.3.8). It contains a season localisation string (§ 4.9.2.21). If the date is a range, it holds the season component of the start date.

origendhour field (datepart)
If the date specification in the origdate field is a range, this field holds the hour component of the end date.

origendminute field (datepart)
If the date specification in the origdate field is a range, this field holds the minute component of the end date.

origendsecond field (datepart)
If the date specification in the origdate field is a range, this field holds the second component of the end date.

origendtimezone field (datepart)
If the date specification in the origdate field is a range, this field holds the timezone component of the end date.

origendday field (datepart)
If the date specification in the origdate field is a range, this field holds the day component of the end date.

origendmonth field (datepart)
If the date specification in the origdate field is a range, this field holds the month component of the end date.

origendyear field (datepart)
If the date specification in the origdate field is a range, this field holds the year component of the end date. A blank (but defined) origendyear component indicates an open ended origendrange.

153
origendseason field (datepart)
If the date specification in the origdate field is a range, this field holds the season component of the end date as specified by edtf 5.2.5 (§2.3.8). It contains a season localisation string (§4.9.2.21). A blank (but defined) origendseason component indicates an open ended origdate range.

eventhour field (datepart)
This field holds the hour component of the eventdate field. If the date is a range, it holds the hour component of the start date.

eventminute field (datepart)
This field holds the minute component of the eventdate field. If the date is a range, it holds the minute component of the start date.

eventsecond field (datepart)
This field holds the second component of the eventdate field. If the date is a range, it holds the second component of the start date.

eventtimezone field (datepart)
This field holds the timezone component of the eventdate field. If the date is a range, it holds the timezone component of the start date.

eventday field (datepart)
This field holds the day component of the eventdate field. If the date is a range, it holds the day component of the start date.

eventmonth field (datepart)
This field holds the month component of the eventdate field. If the date is a range, it holds the month component of the start date.

eventyear field (datepart)
This field holds the year component of the eventdate field. If the date is a range, it holds the year component of the start date.

eventseason field (datepart)
This field holds the season component of the eventdate field as specified by edtf 5.2.5 (§2.3.8). It contains a season localisation string (§4.9.2.21). If the date is a range, it holds the season component of the start date.

eventendhour field (datepart)
If the date specification in the eventdate field is a range, this field holds the hour component of the end date.

eventendminute field (datepart)
If the date specification in the eventdate field is a range, this field holds the minute component of the end date.

eventendsecond field (datepart)
If the date specification in the eventdate field is a range, this field holds the second component of the end date.

eventendtimezone field (datepart)
If the date specification in the eventdate field is a range, this field holds the timezone component of the end date.

eventendday field (datepart)
If the date specification in the eventdate field is a range, this field holds the day component of the end date.

eventendmonth field (datepart)
If the date specification in the eventdate field is a range, this field holds the month component of the end date.

eventendyear field (datepart)
If the date specification in the eventdate field is a range, this field holds the year component of the end date. A blank (but defined) eventendyear component indicates an open ended eventdate range.
**eventendseason** field (datepart)

If the date specification in the *eventdate* field is a range, this field holds the season component of the end date as specified by *edtf* 5.2.5 (§ 2.3.8). It contains a season localisation string (§ 4.9.2.21). A blank (but defined) *eventendseason* component indicates an open ended *eventdate* range.

**urlhour** field (datepart)

This field holds the hour component of the *urldate* field. If the date is a range, it holds the hour component of the start date.

**urlminute** field (datepart)

This field holds the minute component of the *urldate* field. If the date is a range, it holds the minute component of the start date.

**urllsecond** field (datepart)

This field holds the second component of the *urldate* field. If the date is a range, it holds the second component of the start date.

**timezone** field (urldatepart)

This field holds the timezone component of the *urldate* field. If the date is a range, it holds the timezone component of the start date.

**urlday** field (datepart)

This field holds the day component of the *urldate* field.

**urllmonth** field (datepart)

This field holds the month component of the *urldate* field.

**urlyear** field (datepart)

This field holds the year component of the *urldate* field.

**urlseason** field (datepart)

This field holds the season component of the *urldate* field as specified by *edtf* 5.2.5 (§ 2.3.8). It contains a season localisation string (§ 4.9.2.21). If the date is a range, it holds the season component of the start date.

**urllendhour** field (datepart)

If the date specification in the *urldate* field is a range, this field holds the hour component of the end date.

**urllendminute** field (datepart)

If the date specification in the *urldate* field is a range, this field holds the minute component of the end date.

**urllendsecond** field (datepart)

If the date specification in the *urldate* field is a range, this field holds the second component of the end date.

**urllendtimezone** field (datepart)

If the date specification in the *urldate* field is a range, this field holds the timezone component of the end date.

**urllendday** field (datepart)

If the date specification in the *urldate* field is a range, this field holds the day component of the end date.

**urllendmonth** field (datepart)

If the date specification in the *urldate* field is a range, this field holds the month component of the end date.

**urllendyear** field (datepart)

If the date specification in the *urldate* field is a range, this field holds the year component of the end date. A blank (but defined) *urllendyear* component indicates an open ended *urldate* range.

**urllendseason** field (datepart)

If the date specification in the *urldate* field is a range, this field holds the season component of the end date as specified by *edtf* 5.2.5 (§ 2.3.8). It contains a season localisation string (§ 4.9.2.21). A blank (but defined) *urllendseason* component indicates an open ended *urldate* range.
4.3 Citation Styles

A citation style is a set of commands such as \cite which print different types of citations. Such styles are defined in files with the suffix cbx. The \biblatex package loads the selected citation style file at the end of the package. Note that a small repertory of frequently used macros shared by several of the standard citation styles is also included in \biblatex.def. This file is loaded at the end of the package as well, prior to the selected citation style. It also contains the definitions of the commands from § 3.8.5.

4.3.1 Citation Style Files

Before we go over the individual commands available in citation style files, consider this example of the overall structure of a typical cbx file:

\ProvidesFile{example.cbx}[2006/03/15 v1.0 biblatex citation style]
\DeclareCiteCommand{\cite}{...}{...}{...}{...}
\DeclareCiteCommand{\parencite}{\mkbibparens}{...}{...}{...}
\DeclareCiteCommand{\footcite}{\mkbibfootnote}{...}{...}{...}
\DeclareCiteCommand{\textcite}{...}{...}{...}{...}
\endinput

\RequireCitationStyle\{\textstyle\}

This command is optional and intended for specialized citation styles built on top of a more generic style. It loads the citation style style.cbx.

\InitializeCitationStyle\{\langle\text{code}\rangle\}

Specifies arbitrary \langle\text{code}\rangle required to initialize or reset the citation style. This hook will be executed once at package load-time and every time the \citereset command from § 3.8.8 is used. The \citereset command also resets the internal citation trackers of this package. The \citereset will affect the \ifciteseen, \ifentryseen, \ifciteibid, and \ifciteidem tests discussed in §4.6.2. When used in a refsection environment, the reset of the citation tracker is local to the current refsection environment.

\OnManualCitation\{\langle\text{code}\rangle\}

Specifies arbitrary \langle\text{code}\rangle required for a partial reset of the citation style. This hook will be executed every time the \mancite command from § 3.8.8 is used. It is particularly useful in citation styles which replace repeated citations by abbreviations like 'ibidem' or 'op. cit.' which may get ambiguous if automatically generated and manual citations are mixed. The \mancite command also resets the internal 'ibidem' and 'idem' trackers of this package. The \mancite will affect the \ifciteibid and \ifciteidem tests discussed in §4.6.2.

\DeclareCiteCommand\{\langle\text{command}\rangle\}\{\langle\text{wrapper}\rangle\}\{\langle\text{precode}\rangle\}\{\langle\text{loopcode}\rangle\}\{\langle\text{sepcode}\rangle\}\{\langle\text{postcode}\rangle\}
\DeclareCiteCommand*\{\langle\text{command}\rangle\}\{\langle\text{wrapper}\rangle\}\{\langle\text{precode}\rangle\}\{\langle\text{loopcode}\rangle\}\{\langle\text{sepcode}\rangle\}\{\langle\text{postcode}\rangle\}

This is the core command used to define all citation commands. It takes one optional and five mandatory arguments. The \langle\text{command}\rangle is the command to be defined, for example \cite. If the optional \langle\text{wrapper}\rangle argument is given, the entire citation will be passed to the \langle\text{wrapper}\rangle as an argument, i.e., the wrapper command must take one mandatory argument.\footnote{Typical examples of wrapper commands are \mkbibparens and \mkbibfootnote.} The \langle\text{precode}\rangle is arbitrary code to be executed at the beginning of the citation. It will typically handle the \langle\text{prenote}\rangle argument which is available in the \texttt{prenote} field. It may also be used to initialize macros required by the \langle\text{loopcode}\rangle. The \langle\text{loopcode}\rangle is arbitrary code to be executed for each entry key passed to the \langle\text{command}\rangle. This is the core code which prints the citation labels or any other data. The \langle\text{sepcode}\rangle is arbitrary code to be executed after each iteration of the \langle\text{loopcode}\rangle. It will only be executed if a list of entry keys is passed to the \langle\text{command}\rangle. The \langle\text{sepcode}\rangle will usually insert some kind of separator, such as a comma or a semicolon. The \langle\text{postcode}\rangle is arbitrary code to be executed at the end of the citation. The \langle\text{postcode}\rangle will typically handle the \langle\text{postnote}\rangle argument which is available in the \texttt{postnote} field.
The starred variant of \DeclareCiteCommand defines a starred ⟨command⟩. For example, \DeclareCiteCommand*{cite} would define cite*.\footnote{The bibliographic data available to the \loopcode is the data of the entry currently being processed.}

\DeclareAutoCiteCommand{⟨command⟩}⟨⟨command⟩⟩⟨⟨command⟩⟩⟨⟨command⟩⟩
\DeclareMultiCiteCommand{⟨command⟩}⟨⟨command⟩⟩⟨⟨command⟩⟩⟨⟨command⟩⟩

This command defines 'multicite' commands (§3.8.3). The ⟨command⟩ is the multicite command to be defined, for example \cite. It is automatically made robust. Multicite commands are built on top of backend commands defined with \DeclareCiteCommand and the ⟨cite⟩ argument specifies the name of the backend command to be used. Note that the wrapper of the backend command (i.e., the ⟨wrapper⟩) argument passed to \DeclareCiteCommand is ignored. Use the optional ⟨wrapper⟩ argument to specify an alternative wrapper. The ⟨delimiter⟩ is the string to be printed as a separator between the individual citations in the list. This will typically be \multicitedelim. The following examples are real definitions taken from biblatex.def:

\DeclareMultiCiteCommand{⟨command⟩}⟨⟨command⟩⟩⟨⟨command⟩⟩⟨⟨command⟩⟩
\DeclareAutoCiteCommand{⟨command⟩}⟨⟨command⟩⟩⟨⟨command⟩⟩⟨⟨command⟩⟩

A definition provided in the document preamble can be subsequently adopted with the following: (see §3.2.2).

\ExecuteBibliographyOptions{autocite=⟨name⟩}

4.3.2 Special Fields

The following fields are used by biblatex to pass data to citation commands. They are not used in bib files but defined automatically by the package. From the perspective of a citation style, they are not different from the fields in a bib file. See also §4.2.4.

\texttt{prenote} (literal)

The ⟨prenote⟩ argument passed to a citation command. This field is specific to citations and not available in the bibliography. If the ⟨prenote⟩ argument is missing or empty, this field is undefined.

\footnote{The bibliographic data available to the \loopcode is the data of the entry currently being processed. In addition to that, the data of the first entry is available to the \precode and the data of the last one is available to the \postcode. 'First' and 'last' refer to the order in which the citations are printed. If the \sortcites package option is active, this is the order of the list after sorting. Note that no bibliographic data is available to the \sepcodes.}

\footnote{Note that the regular variant of \DeclareCiteCommand defines a starred version of the ⟨command⟩ implicitly, unless the starred version has been defined before. This is intended as a fallback. The implicit definition is an alias for the regular variant.}
The \printfield command prints a \texttt{field} using the formatting directive \texttt{format}, as defined with \DeclareFieldFormat. If a type-specific \texttt{format} has been declared, the type-specific formatting directive takes precedence over the generic one. If the \texttt{field} is undefined, nothing is printed. If the \texttt{format} is omitted, \printfield tries using the name of the field as a format name. For example, if the \texttt{title} field is to be printed and the \texttt{format} is not specified, it will try to use the field format \texttt{title}.

This command loops over all items in a \texttt{literal list}, starting at item number \texttt{start} and stopping at item number \texttt{stop}, including \texttt{start} and \texttt{stop} (all lists are numbered starting at 1). Each item is printed using the formatting directive \texttt{format}, as defined with \DeclareListFormat. If a type-specific \texttt{format} has been declared, the type-specific formatting directive takes precedence over the generic one. If the \texttt{literal list} is undefined, nothing is printed. If the \texttt{format} is omitted, \printlist tries using the name of the list as a format name. In this case, any type-specific formatting directive will also take precedence over the generic one. If all of these formats are undefined, it falls back to default as a last resort. The \texttt{start} argument defaults to 1; \texttt{stop} defaults to the total number of items in the list. If the total number is greater than \texttt{maxitems}, \texttt{stop} defaults to \texttt{minitems} (see § 3.1.2.1). See \printfielddates for further details. Note that \printlist provides the name of the literal list currently being processed in \currentlist for use in list formatting directives.

\footnote{In other words, \printfield\{title\} is equivalent to \printfield[title]\{title\}.}
\printnames{(format)}{(start)-(stop)}{(name list)}

This command loops over all items in a \texttt{name list}, starting at item number \texttt{start} and stopping at item number \texttt{stop}, including \texttt{start} and \texttt{stop} (all lists are numbered starting at 1). Each item is printed using the formatting directive \texttt{format}, as defined with \texttt{\DeclareNameFormat}. If a type-specific \texttt{format} has been declared, the type-specific formatting directive takes precedence over the generic one. If the \texttt{name list} is undefined, nothing is printed. If the \texttt{format} is omitted, \texttt{\printnames} tries using the name of the list as a format name. In this case, any type-specific formatting directive will also take precedence over the generic one. If all of these formats are undefined, it falls back to default as a last resort. The \texttt{start} argument defaults to 1; \texttt{stop} defaults to the total number of items in the list. If the total number is greater than \texttt{(maxnames)}, \texttt{(stop)} defaults to \texttt{(minnames)} (see § 3.1.2). If you want to select a range but use the default list format, the first optional argument must still be given, but is left empty:

\printnames[]{1-3}{...}

One of \texttt{start} and \texttt{stop} may be omitted, hence the following arguments are all valid:

\printnames[]{...}[-1]{...}
\printnames[]{...}[2-]{...}
\printnames[]{...}[1-3]{...}

If you want to override \texttt{(maxnames)} and \texttt{(minnames)} and force printing of the entire list, you may refer to the \texttt{listtotal} counter in the second optional argument:

\printnames[]{...}[-\texttt{value(listtotal)}]{...}

Whenever \texttt{\printnames} and \texttt{\printlist} process a list, information concerning the current state is accessible by way of four counters: the \texttt{listtotal} counter holds the total number of items in the current list, \texttt{listcount} holds the number of the item currently being processed, \texttt{liststart} is the \texttt{start} argument passed to \texttt{\printnames} or \texttt{\printlist}, \texttt{liststop} is the \texttt{stop} argument. These counters are intended for use in list formatting directives. \texttt{listtotal} may also be used in the second optional argument to \texttt{\printnames} and \texttt{\printlist}. Note that these counters are local to list formatting directives and do not hold meaningful values when used anywhere else. For every list, there is also a counter by the same name which holds the total number of items in the corresponding list. For example, the \texttt{author} counter holds the total number of items in the author list. These counters are similar to \texttt{listtotal} except that they may also be used independently of list formatting directives. There are also \texttt{maxnames} and \texttt{minnames} as well as \texttt{maxitens} and \texttt{minitens} counters which hold the values of the corresponding package options. See § 4.10.5 for a complete list of such internal counters. Note that \texttt{\printnames} provides the name of the name list currently being processed in \texttt{\currentname} for use in name formatting directives.

\printtext{(format)}{(text)}

This command prints \texttt{text}, which may be printable text or arbitrary code generating printable text. It clears the punctuation buffer before inserting \texttt{text} and informs \texttt{biblatex} that printable text has been inserted. This ensures that all preceding and following \texttt{\newblock} and \texttt{\newunit} commands have the desired effect. \texttt{\printfield} and \texttt{\printnames} as well as \texttt{\bibstring} and its companion commands (see § 4.8) do that automatically. Using this command is required if a bibliography styles inserts literal text (including the commands from §§ 4.7.3 and 4.7.4) to ensure that block and unit punctuation works as advertised in § 4.7.1. The optional \texttt{format} argument specifies a field formatting directive to be used to format \texttt{text}. This may also be useful when several fields are to be printed as one chunk, for example, by enclosing the entire chunk in parentheses or quotation marks. If a type-specific \texttt{format} has been declared, the type-specific formatting directive takes precedence over the generic one. If the \texttt{format} is omitted, the \texttt{text} is printed as is. See also § 4.11.7 for some practical hints.

\printfile{(format)}{(file)}

This command is similar to \texttt{\printtext} except that the second argument is a file name rather than literal text. The \texttt{file} argument must be the name of a valid LaTeX file found in TeX’s search path. \texttt{\printfile} will use \texttt{\input} to load this \texttt{file}. If there is no such file, \texttt{\printfile} does nothing. The optional \texttt{format} argument specifies a field formatting directive to be applied to the \texttt{file}. If a
This command prints the date of the entry, as specified in the fields date or month/year. The date format is controlled by the package option date from §3.1.2.1. Additional formatting (fonts etc.) may be applied by adjusting the field format date (§4.10.4). Note that this command interfaces with the punctuation tracker. There is no need to wrap it in a \printtext command.

\printdateextra

Similar to \printdate but incorporates the extrayear field in the date specification. This is useful for bibliography styles designed for author-year citations.

\printlabelldate

Similar to \printdate but prints the date field determined by \DeclareLabeldate. The date format is controlled by the package option labeldate from §3.1.2.1. Additional formatting may be applied by adjusting the field format labeldate (§4.10.4).

\printlabelldateextra

Similar to \printlabelldate but incorporates the extrayear field in the date specification. This is useful for bibliography styles designed for author-year citations.

\print<datatype>date

As \printdate but prints the <datatype>date of the entry. The date format is controlled by the package option <datatype>date from §3.1.2.1. Additional formatting may be applied by adjusting the field format <datatype>date (§4.10.4). The <datatype>s in the default data model are ’’ (for the main date field), ’orig’, ’event’ and ’url’.

\printtime

This command prints the time range of the entry, as specified in the date field (see §2.3.8). The time format is controlled by the package option time from §3.1.2.1. Additional formatting (fonts etc.) may be applied by adjusting the field format time (§4.10.4). Relevant to time formatting are the timezeros option and the \bibtimesep and \bibtimelinesep macros (§3.10.3). Note that this command interfaces with the punctuation tracker. There is no need to wrap it in a \printtext command. Note that this command prints a stand-alone time range apart from the date elements. With the <datepart>datesettime option, you can have the printed along with a date when printing a date range instead of printing the time range completely separately, which is what this command allows for.

\print<datatype>time

As \printtime but prints the <datatype>time of the entry. The time format is controlled by the package option <datatype>time from §3.1.2.1. Additional formatting may be applied by adjusting the field format <datatype>time (§4.10.4). The <datatype>s in the default data model are ’’ (for the main date field), ’orig’, ’event’ and ’url’.

\indexfield{format}{field}

This command is similar to \indexfield except that the (field) is not printed but added to the index using the formatting directive (format), as defined with \DeclareIndexFieldFormat. If a type-specific (format) has been declared, it takes precedence over the generic one. If the (field) is undefined, this command does nothing. If the (format) is omitted, \indexfield tries using the name of the field as a format name. In this case, any type-specific formatting directive will also take precedence over the generic one. If all of these formats are undefined, it falls back to default as a last resort.

\indexlist{format}{start}--{stop}{literal list}

This command is similar to \indexlist except that the items in the list are not printed but added to the index using the formatting directive (format), as defined with \DeclareIndexListFormat. If a type-specific (format) has been declared, the type-specific formatting directive takes precedence over the generic one. If the (literal list) is undefined, this command does nothing. If the (format) is omitted, \indexlist tries using the name of the list as a format name. In this case, any type-specific formatting directive will also take precedence over the generic one. If all of these formats are undefined, it falls back to default as a last resort.

\indexnames{format}{start}--{stop}{name list}

This command is similar to \indexnames except that the items in the list are not printed but added to the index using the formatting directive (format), as defined with \DeclareIndexNameFormat. If a type-specific (format) has been declared, the type-specific formatting directive takes precedence over the generic one. If the (name list) is undefined, this command does nothing. If the (format) is omitted, \indexnames tries using the name of the list as a format name. In this case, any type-specific formatting directive will also take precedence over the generic one. If all of these formats are undefined, it falls back to default as a last resort.
Data commands like \printfield normally use the data of the entry currently being processed. You may use \entrydata to switch contexts locally. The \texttt{(key)} is the entry key of the entry to use locally. The \texttt{(code)} is arbitrary code to be executed in this context. This code will be executed in a group. See §4.11.6 for an example. Note that this command will automatically switch languages if the autolang package option is enabled. The starred version \entrydata* will clone all fields of the enclosing entry, using field, counter, and other resource names prefixed with the string ‘saved’. This is useful when comparing two data sets. For example, inside the \texttt{(code)} argument, the \texttt{author} field holds the author of entry \texttt{(key)} and the author of the enclosing entry is available as \texttt{savedauthor}. The author counter holds the number of names in the \texttt{author} field of \texttt{(key)}; the \texttt{savedauthor} counter refers to the author count of the enclosing entry.

\entryset{\texttt{(precode)}}{\texttt{(postcode)}}

This command is intended for use in bibliography drivers handling \texttt{@set} entries. It will loop over all members of the set, as indicated by the \texttt{entryset} field, and execute the appropriate driver for the respective set member. This is similar to executing the \texttt{\useldriver} command from §4.6.4 for each set member. The \texttt{(precode)} is arbitrary code to be executed prior to processing each item in the set. The \texttt{(postcode)} is arbitrary code to be executed immediately after processing each item. Both arguments are mandatory in terms of the syntax but may be left empty. See §4.11.1 for usage examples.

\DeclareFieldInputHandler{\texttt{(field)}}{\texttt{(code)}}

This command can be used to define a data input handler for \texttt{(field)} when it is read from the .bib file. Within the \texttt{(code)}, the macro \texttt{\NewValue} contains the value of the field. For example, to ignore the \texttt{volumes} field when it appears, you could do

\begin{verbatim}
\DeclareFieldInputHandler{volumes}{\def\NewValue{}}
\end{verbatim}

Generally, you would want to use \DeclareSourceMap (see §4.5.3) to remove and modify fields but this alternative method may be useful in some circumstances when the emphasis is on appearance rather than data since the \texttt{(code)} can be arbitrary TeX.

\DeclareListInputHandler{\texttt{(list)}}{\texttt{(code)}}

As \texttt{\DeclareFieldInputHandler} but for lists. Within the \texttt{(code)}, the macro \texttt{\NewValue} contains the value of the list and \texttt{\NewCount} contains the number of items in the list.

\DeclareNameInputHandler{\texttt{(name)}}{\texttt{(code)}}

As \texttt{\DeclareFieldInputHandler} but for names. Within the \texttt{(code)}, the macro \texttt{\NewValue} contains the value of the name, \texttt{\NewCount} contains the number of individual names in the name and \texttt{\NewOption} contains any per-name options passed in the .bib file.

### 4.4.2 Formatting Directives

This section introduces the commands used to define the formatting directives required by the data commands from §4.4.1. Note that all standard formats are defined in biblatex_.def.

\ DeclareFieldFormat{\texttt{\langle entrytype, ... \rangle}}{\texttt{(format)}}{\texttt{(code)}}

\ DeclareFieldFormat*{\texttt{\langle format \rangle}}{\texttt{(code)}}

Defines the field format \texttt{(format)}. This formatting directive is arbitrary \texttt{(code)} to be executed by \printfield. The value of the field will be passed to the \texttt{(code)} as its first and only argument. The name of the field currently being processed is available to the \texttt{(code)} as \texttt{\currentfield}. If an \texttt{\entrytype} is specified, the format is specific to that type. The \texttt{\entrytype} argument may be a comma-separated list of values. The starred variant of this command is similar to the regular version, except that all type-specific formats are cleared.

\ DeclareListFormat{\texttt{\langle entrytype, ... \rangle}}{\texttt{(format)}}{\texttt{(code)}}

\ DeclareListFormat*{\texttt{\langle format \rangle}}{\texttt{(code)}}

Defines the literal list format \texttt{(format)}. This formatting directive is arbitrary \texttt{(code)} to be executed for every item in a list processed by \printfield. The current item will be passed to the \texttt{(code)} as its first and only argument. The name of the literal list currently being processed is available to the \texttt{(code)} as \texttt{\currentlist}. If an \texttt{\entrytype} is specified, the format is specific to that type. The \texttt{\entrytype} argument may be a comma-separated list of values. Note that the formatting directive also handles the punctuation to be inserted between the individual items in the list. You need to check whether you are in the middle of or at the end of the list, i.e., whether \texttt{\listcount} is smaller than or equal to \texttt{\liststop}. The starred variant of this command is similar to the regular version, except that all type-specific formats are cleared.
\DeclareNameFormat*{⟨entrytype, …⟩}{⟨format⟩}{⟨code⟩}
\DeclareNameFormat*{⟨format⟩}{⟨code⟩}

Defines the name list format (⟨format⟩). This formatting directive is arbitrary (⟨code⟩) to be executed for every name in a list processed by \printnames. If an ⟨entrytype⟩ is specified, the format is specific to that type. The ⟨entrytype⟩ argument may be a comma-separated list of values. The individual parts of a name will be available in automatically created macros (see below). The default datamodel defines four name parts which correspond to the standard BibTeX name parts arguments:

family The family name(s), known as ‘last’ in BibTeX. If a name consists of a single part only (for example, ‘Aristotle’), this part will be treated as the family name.
given The given name(s). Note that given names are referred to as the ‘first’ names in the BibTeX file format documentation.
prefix Any name prefixes, for example von, van, of, da, de, del, della, etc. Note that name prefixes are referred to as the ‘von’ part of the name in the BibTeX file format documentation.
suffix Any name suffixes, for example Jr, Sr. Note that name suffixes are referred to as the ‘Jr’ part of the name in the BibTeX file format documentation.

The value of the datamodel ‘nameparts’ constant (see §4.2.3) creates two macros for each name part in the datamodel for the name. So, for example, in the default data model, name formats will have defined the following macros:

\namepartprefix \namepartprefixi
\namepartfamily \namepartfamilyi
\namepartsuffix \namepartsuffixi
\namepartgiven
\namepartgiveni

If a certain part of a name is not available, the corresponding macro will be empty, hence you may use, for example, the etoolbox tests like \ifdefvoid to check for the individual parts of a name. The name of the name list currently being processed is available to the (⟨code⟩) as \currentname. Note that the formatting directive also handles the punctuation to be inserted between separate names and between the individual parts of a name. You need to check whether you are in the middle of or at the end of the list, i.e., whether \listcount is smaller than or equal to \liststop. See also §3.13.4. The starred variant of this command is similar to the regular version, except that all type-specific formats are cleared.

\DeclareIndexFieldFormat*{⟨entrytype, …⟩}{⟨format⟩}{⟨code⟩}
\DeclareIndexFieldFormat*{⟨format⟩}{⟨code⟩}

Defines the field format (⟨format⟩). This formatting directive is arbitrary (⟨code⟩) to be executed by \indexfield. The value of the field will be passed to the (⟨code⟩) as its first and only argument. The name of the field currently being processed is available to the (⟨code⟩) as \currentfield. If an ⟨entrytype⟩ is specified, the format is specific to that type. The ⟨entrytype⟩ argument may be a comma-separated list of values. This command is similar to \DeclareFieldFormat except that the data handled by the (⟨code⟩) is not intended to be printed but written to the index. Note that \indexfield will execute the (⟨code⟩) as is, i.e., the (⟨code⟩) must include \index or a similar command. The starred variant of this command is similar to the regular version, except that all type-specific formats are cleared.

\DeclareIndexListFormat*{⟨entrytype, …⟩}{⟨format⟩}{⟨code⟩}
\DeclareIndexListFormat*{⟨format⟩}{⟨code⟩}

Defines the literal list format (⟨format⟩). This formatting directive is arbitrary (⟨code⟩) to be executed for every item in a list processed by \indexlist. The current item will be passed to the (⟨code⟩) as its only argument. The name of the literal list currently being processed is available to the (⟨code⟩) as \currentlist. If an ⟨entrytype⟩ is specified, the format is specific to that type. The ⟨entrytype⟩ argument may be a comma-separated list of values. This command is similar to \DeclareListFormat except that the data handled by the (⟨code⟩) is not intended to be printed but written to the index. Note that \indexlist will execute the (⟨code⟩) as is, i.e., the (⟨code⟩) must include \index or a similar command. The starred variant of this command is similar to the regular version, except that all type-specific formats are cleared.
The type of relationship. This serves three purposes. If the value of this field resolves to a localisation related
A separated list of keys of entries which are related to this entry in some way. Note the the order of
version, except that all type-specific formats are cleared. 

If no such macro exists, then the macro related:default is used. Lastly, if there is a format named related:*{relatedtype}, then it is used to format both the localised string and related entry data. If there is no related type specific format, the related format is used.

4.5 Customization

4.5.1 Related Entries

The related entries feature comprises the following components:

- Special fields in an entry to set up and describe relationships
- Optionally, localisation strings to prefix the related data
- Macros to extract and print the related data
- Formats to format the localisation string and related data

The special fields are related, relatedtype, relatedstring and relatedoptions:

related
A separated list of keys of entries which are related to this entry in some way. Note the the order of the keys is important. The data from multiple related entries is printed in the order of the keys listed in this field.

relatedtype
The type of relationship. This serves three purposes. If the value of this field resolves to a localisation string identifier, then the resulting localised string is printed before the data from the related entries. Secondly, if there is a macro called related:*(relatedtype), this is used to format the data from the related entries. If no such macro exists, then the macro related:default is used. Lastly, if there is a format named related:*{relatedtype}, then it is used to format both the localised string and related entry data. If there is no related type specific format, the related format is used.
relatedstring

If an entry contains this field, then if value of the field resolves to a localisation string identifier, the localisation key value specified is printed before data from the related entries. If the field does not specify a localisation key, its value is printed literally. If both relatedtype and relatedstring are present in an entry, relatedstring is used for the pre-data string (but relatedtype is still used to determine the macro and format to use when printing the data).

relatedoptions

A list of per-entry options to set on the related entry (actually on the clone of the related entry which is used as a data source—the actual related entry is not modified because it might be cited directly itself).

The related entry feature is enabled by default by the package option related from §3.1.2.1. The related information entry data from the related entries is included via a \usebibmacro{related} call. Standard styles call this macro towards the end of each driver. Style authors should ensure the existence of (or take note of existing) localisation strings which are useful as values for the relatedtype field, such as translationof or perhaps translatedas. A plural variant can be identified with the localisation key (relatedtype)s. This key’s corresponding string is printed whenever more than one entry is specified in related. Bibliography macros and formatting directives for printing entries related by (relatedtype) should be defined using the name related:(relatedtype).

The file biblatex.def contains macros and formats for some common relation types which can be used as templates. In particular, the \entrydata* command is essential in such macros in order to make the data of the related entries available. Examples of entries using this feature can be found in the biblatex distribution examples file biblatex-examples.bib. There are some specific formatting macros for this feature which control delimiters and separators in related entry information, see §4.10.1.

4.5.2 Datasource Sets

It is useful to be able to define named sets of datasource field names for use in loops etc. In addition, biber can use such sets in order to apply options and perform operations on particular sets of datasource fields. The following macros allow the user to define arbitrary sets of datasource fields, exposed to biblatex as etoolbox lists and to biber in the .bcf.

\DeclareDatafieldSet{(name)}{(specification)}

Declare a set of datasource fields with name {name}.

name={set name}
The name of the set.

The (specification) is one or more \member items:

\member

fieldtype={fieldtype}
datatype={datatype}
field={fieldname}

A \member specification appends fields to the set. Fields can be specified by datamodel {fieldtype} and/or {datatype} (see §4.5.4). Alternatively, fields can be explicitly added by name using the \field option. Once defined, the set is available as an etoolbox list called \datafieldset'setname' and is also passed via the .bcf to biber.

For example, here are the default sets defined by biblatex for name fields and title fields:

\DeclareDatafieldSet{setnames}{
  \member[datatype=name, fieldtype=list]
}
\DeclareDatafieldSet{settitles}{
  \member[field=title]
  \member[field=booktitle]
  \member[field=booktitle]
  \member[field=journaltitle]
  \member[field=maintitle]
  \member[field=origtitle]
}

This defines the macros \datafieldsetsetnames and \datafieldsetsettitles as etoolbox lists containing the names of the member datasource fields specified.
4.5.3 Dynamic Modification of Data

Bibliographic data sources which are automatically generated or which you have no control over can be a problem if you need to edit them in some way. For this reason, biber has the ability to modify data as it is read so that you can apply modifications to the source data stream without actually changing it. The modification can be defined in biber’s config file (see biber docs), or via biblatex macros in which case you can apply the modification only for specific documents, styles or globally.

Source mapping happens during data parsing and therefore before any other operation such as inheritance and sorting.

Source mappings can be defined at different “levels” which are applied in a defined order. See the biblatex manual regarding these macros:

\DeclareSourcemap{(specification)}

Defines source data modification (mapping) rules which can be used to perform any combination of the following tasks:

- Map data source entrytypes to different entrytypes
- Map datasource fields to different fields
- Add new fields to an entry
- Remove fields from an entry
- Modify the contents of a field using standard Perl regular expression match and replace
- Restrict any of the above operations to entries coming from particular datasources which you defined in \addresource macros
- Restrict any of the above operations to entries only of a certain entrytype
- Restrict any of the above operations to entries in a particular reference section

The \{specification\} is an undelimited list of \maps directives which specify containers for mappings rules applying to a particular data source type (§3.7.1). Spaces, tabs, and line endings may be used freely to visually arrange the \{specification\}.Blank lines are not permissible. This command may only be used in the preamble and may only be used once—subsequent uses will overwrite earlier definitions.

\maps\{(options)\}\{(elements)\}

Contains an ordered set of \map elements each of which is a logically related set of mapping steps to apply to the data source. The \{options\} are:

\datatype\{biblatex,bibtex,biblatexml\} default: bibtex

Data source type to which the contained \map directives apply (§3.7.1).
\overwrite\{true,false\} default: false

Specify whether a mapping rule is allowed to overwrite already existing data in an entry. If this option is not specified, the default is false. The short form overwrite is equivalent to overwrite=true.

\map\{(options)\}\{(restrictions,steps)\}

A container for an ordered set of map \steps, optionally restricted to particular entrytypes or data sources. This is a grouping element to allow a set of mapping steps to apply only to specific entrytypes or data sources. Mapping steps must always be contained within a \map element. The \{options\} are:

\overwrite\{true,false\}

As the same option on the parent \maps element. This option allows an override on a per-map group basis. If this option is not specified, the default is the parent \maps element option value. The short form overwrite is equivalent to overwrite=true.

foreach\{(loopval)\}

Loop over all \steps in this \map, setting the special variable MAPLOOP to each of the comma-separated values contained in \{loopval\}. \{loopval\} can either be the name of a datafield set defined with \DeclareDatafieldSet (see §4.5.2), a datasource field which is fetched and parsed as a comma-separated values list or an explicit comma-separated values list. \{loopval\} is determined in this order. This allows the user to repeat a group of \steps for each value \{loopval\}. Using regexp maps, it is possible to create a CSV field for use with this functionality. The special variable MAPUNIQ may also be used the \steps to generate a random unique string. This can be useful when creating keys for new entries. An example:
refsection=⟨integer⟩

Only apply the contained \step commands to entries in the reference section with number ⟨refsection⟩.

\perdatasource{⟨datasource⟩}

Restricts all \steps in this \map element to entries from the named ⟨datasource⟩. The ⟨datasource⟩ name should be exactly as given in a \addresource macro defining a data source for the document. Multiple \perdatasource restrictions are allowed within a \map element.

\pertype{⟨entrytype⟩}

Restricts all \steps in this \map element to entries of the named ⟨entrytype⟩. Multiple \pertype restrictions are allowed within a \map element.

\pernottype{⟨entrytype⟩}

Restricts all \steps in this \map element to entries not of the named ⟨entrytype⟩. Multiple \pernottype restrictions are allowed within a \map element.

\step{⟨options⟩}

A mapping step. Each step is applied sequentially to every relevant entry where ‘relevant’ means those entries which correspond to the data source type, entrytype and data source name restrictions mentioned above. Each step is applied to the entry as it appears after the application of all previous steps. The mapping performed by the step is determined by the following ⟨option⟩s:

- typesource=⟨entrytype⟩
- typetarget=⟨entrytype⟩
- fieldsource=⟨entryfield⟩
- notfield=⟨entryfield⟩
- fieldtarget=⟨entryfield⟩
- match=⟨regexp⟩
- notmatch=⟨regexp⟩
- replace=⟨regexp⟩
- fieldclone=⟨entryfield⟩
- fieldvalue=⟨string⟩
- entryclone=⟨clonekey⟩
- entrynew=⟨entrynewkey⟩
- entrynewtype=⟨string⟩
- entrytarget=⟨string⟩
- entrynull=true, false
- append=true, false
- final=true, false
- null=true, false
- origfield=true, false
- origfieldval=true, false
- origentrytype=true, false

For all boolean \step options, the short form option is equivalent to option=true. The following rules for a mapping step apply:

- If entrynew is set, a new entry is created with the entry key entrynewkey and the entry type given in the option entrynewtype. This entry is only in-scope during the processing of the current entry and can be referenced by entrytarget. In entrynewkey, you may use standard Perl regular expression backreferences to captures from a previous match step.
• When a fieldset step has entrytarget set to the entrykey of an entry created by entrynew, the target for the field set will be the entrytarget entry rather than the entry being currently processed. This allows users to create new entries and set fields in them.

• If entrynull is set, processing of the \map immediately terminates and the current entry is not created. It is as if it did not exist in the datasource. Obviously, you should select the entries which you want to apply this to using prior mapping steps.

• If entryclone is set, a clone of the entry is created with an entry key clonekey. Obviously this may cause labelling problems in author/year styles etc. and should be used with care. The cloned entry is in-scope during the processing of the current entry and can be modified by passing its key as the value to entrytarget. In clonekey, you may use standard Perl regular expression backreferences to captures from a previous match step.

• Change the typesource (entrytype) to the typetarget (entrytype), if defined. If final is true then if the (entrytype) of the entry is not typesource, processing of the parent \map immediately terminates.

• Change the fieldsource (entryfield) to fieldtarget, if defined. If final is true then if there is no fieldsource (entryfield) in the entry, processing of the parent \map immediately terminates.

• If notfield is used then only apply the step if the (entryfield) does not exist.

• If match is defined but replace is not, only apply the step if the fieldsource (entryfield) matches the match regular expression (logic is reversed if you use notmatch instead)\(^3\). You may use capture parenthesis as usual and refer to these ($1…$9) in later fieldvalue specifications. This allows you to pull out parts of some fields and put these parts in other fields.

• Perform a regular expression match and replace on the value of the fieldsource (entryfield) if match and replace are defined.

• If fieldset is defined, then its value is (entryfield) which will be set to a value specified by further options. If overwrite is false for this step and the field to set already exists then the map step is ignored. If final is also true for this step, then processing of the parent map stops at this point. If append is true, then the value to set is appended to the current value of (entryfield). The value to be set is specified by a mandatory one and only one of the following options:

  o fieldvalue — The fieldset (entryfield) is set to the fieldvalue (string)
  o null — The fieldset (entryfield) is ignored, as if it did not exist in the datasource
  o originentrytype — The fieldset (entryfield) is set to the most recently mentioned typesource (entrytype) name
  o origfield — The fieldset (entryfield) is set to the most recently mentioned fieldsource (entryfield) name
  o origfieldval — The fieldset (entryfield) is set to the most recently mentioned fieldsource value

With BibTeX datasources, you may specify the pseudo-field entrykey for fieldsource which is the citation key of the entry. With biblatexml the entrykey is a normal attribute and can be reference like any other attribute. Naturally, this ‘field’ cannot be changed (used as fieldset, fieldtarget or changed using replace).

\Declaresourcestyle{(specification)}

This command sets the source mappings used by a style. Such mappings are conceptually separate from user mappings defined with \Declaresourcemaps and are applied directly after user maps. The syntax is identical to \Declaresourcemap. This command is provided for style authors so that any maps defined for the style do not interfere with user maps or the default driver maps defined with \Declaresourcedrivemap. This command is for use in style files and can be used multiple times, the maps being run in order of definition.

\Declaresourcedrivemap[{datatype-driver}] {(specification)}

This command sets the driver default source mappings for the specified (driver). Such mappings are conceptually separate from user mappings defined with \Declaresourcemaps and style mapping defined with \Declaresourcestylemap. They consist of mappings which are part of the driver setup. Users should not normally need to change these. Driver default mapping are applied after user mappings (\Declaresourcemaps) and style mappings (\Declaresourcestylemap). These defaults are described in Appendix § A. The (specification) is identical to that for \Declaresourcemap but without the \maps elements: the (specification) is just a list of \map elements since each

\(^3\)Regular expressions are full Perl 5.16 regular expressions. This means you may need to deal with special characters, see examples.

167
\DeclareDriverSourcemap only applies to one datatype driver. See the default definitions in Appendix §A for examples.

Here are some data source mapping examples:

\begin{verbatim}
\DeclareSourcemap{
\maps[datatype=bibtex]{
  \map{
    \perdatasource{example1.bib}
    \perdatasource{example2.bib}
    \step[fieldset=keywords, fieldvalue={keyw1, keyw2}]
    \step[fieldset=entrykey]
    \step[fieldset=note, origfieldval]
  }
}
\end{verbatim}

This would add a \texttt{keywords} field with value 'keyw1, keyw2' and set the \texttt{note} field to the entry key to all entries which are found in either the \texttt{example1.bib} or \texttt{example2.bib} files.

\begin{verbatim}
\DeclareSourcemap{
\maps[datatype=bibtex]{
  \map{
    \step[fieldset=title]
    \step[fieldset=note, origfieldval]
  }
}
\end{verbatim}

Copy the \texttt{title} field to the \texttt{note} field unless the \texttt{note} field already exists.

\begin{verbatim}
\DeclareSourcemap{
\maps[datatype=bibtex]{
  \map{
    \step[typesource=chat, typetarget=customa, final]
    \step[fieldset=type, origentrytype]
  }
}
\end{verbatim}

Any chat entry types would become customa entry types and would automatically have a \texttt{type} field set to 'chat' unless the \texttt{type} field already exists in the entry (because \texttt{overwrite} is false by default). This mapping applies only to entries of type @chat since the first step has \texttt{final} set and so if the \texttt{typesource} does not match the entry entrytype, processing of this \map immediately terminates.

\begin{verbatim}
\DeclareSourcemap{
\maps[datatype=bibtex]{
  \map{
    \perdatasource{examples.bib}
    \pertype{article}
    \pertype{book}
    \step[fieldset=abstract, null]
    \step[fieldset=note, fieldvalue={Auto-created this field}]
  }
}
\end{verbatim}

Any entries of entrytype @article or @book from the \texttt{examples.bib} datasource would have their \texttt{abstract} fields removed and a \texttt{note} field added with value 'Auto-created this field'.
This removes abstract fields from any entry, changes conductor fields to namea fields and changes gps fields to usera fields.

Applies only to entries with pubmed fields and maps pubmedid fields to eprint fields, sets the eprinttype field to 'pubmedid' and also sets the userd field to the string 'Some string of things'.

Here, the contents of the series field have leading numbers stripped and the remainder of the contents lowercased. Since regular expressions usually contain all sort of special characters, it is best to enclose them in the provided \regexp macro as shown—this will pass the expression through to biber correctly.

Here, if for an entry, the maintitle field matches a particular regular expression, we set a special keyword so we can, for example, make a references section just for certain items.
If an entry has a `lista` field which matches regular expression `regexp`, then it is removed.

```latex
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map[overwrite=false]{
      \step[fieldsource=author]
      \step[fieldset=editor, origfieldval, final]
      \step[fieldsource=editor, match=\regexp{\A(.+?)\s+and.*}, replace=\{\$1\}]
    }
  }
}
```

For any entry with an `author` field, try to set `editor` to the same as `author`. If this fails because `editor` already exists, stop, otherwise truncate `editor` to just the first name in the name list.

```latex
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \step[fieldsource=author, match={Smith, Bill}, replace={Smith, William}]
      \step[fieldsource=author, match={Jones, Baz}, replace={Jones, Barry}]
    }
  }
}
```

Here, we use multiple match/replace for the same field to regularise some inconstant name variants. Bear in mind that \`step\` processing within a `map` element is sequential and so the changes from a previous `\step`s are already committed. Note that we don’t need the \`regexp\` macro to protect the regular expressions in this example as they contain no characters which need special escaping. Please note that due to the difficulty of protecting regular expressions in \LaTeX, there should be no literal spaces in the argument to \`regexp\`. Please use escape code equivalents if spaces are needed. For example, this example, if using \`regexp\`, should be:

```latex
\DeclareSourcemap{
  \maps[datatype=bibtex]{
    \map{
      \step[fieldsource=author, match=\regexp{Smith,\s+Bill}, replace=\regexp{Smith,\x20William}]
      \step[fieldsource=author, match=\regexp{Jones,\s+Baz}, replace=\regexp{Jones,\x20Barry}]
    }
  }
}
```

Here, we have used the hexadecimal escape sequence `\x20` in place of literal spaces in the replacement strings.

```latex
\DeclareSourcemap{
```
Only applies to entries with an author field matching 'Doe'. First the author field is copied to both the shortauthor and sortname fields, overwriting them if they already exist. Then, these two new fields are modified to canonicalise a particular name, which presumably has some variants in the data source.

This example demonstrates the sequential nature of the step processing and the append option. If an entry has a verba field then first, a forward slash is appended to the verbb field. Then, the contents of verba are appended to the verbb field. A slash is then appended to the verbc field and the contents of verbb are appended to the verbc field.

This example assumes you have created a field called autourl using the datamodel macros from § 4.5.4 in order to hold, for example, a Google Scholar query URL auto-created from elements of the entry. The example progressively extracts information from the entry, constructing the URL as it goes. It demonstrates that it is possible to refer to parenthetical matches from the most recent match in
any following fieldvalue which allows extracting the family name from the author, assuming a 'family, given' format. The resulting field could then be used as a hyperlink from, for example, the title of the work in the bibliography.

\DeclareSourcemap{
  \maps{datatype=bibtex}{
    \map{
      \step{fieldsource=title, match={A Title}, final}
      \step{entrynull}
    }
  }
}

Any entry with a title field matching 'A Title' will be completely ignored.

\DeclareSourcemap{
  \maps{datatype=bibtex}{
    \map{
      \pernottype{book}
      \pernottype{article}
      \step{entrynull}
    }
  }
}

Any entry which is not a @book or @article will be ignored.

\DeclareSourcemap{
  \maps{datatype=bibtex}{
    \map{
      \perdatasource{biblatex-examples.bib}
      \step{entryclone={rel-}}
    }
  }
}

Here, a clone of an entry from the specified data source will be created. The entry key of the clone will be the same as the original but prefixed by the value of the entryclone parameter. The cloned entry would still need to be cited in the document using its new entry key. This type of mapping step should be used with care as it may produce labelling problems in authoryear styles which use, for example, extrayear. One use case is for numeric styles which contain multiple bibliographies containing the same entry. In this case, you may need different bibliography number labels for the same entry and this is very tricky when there is only one entry which needs different labels. Creating clones with different entry keys solves this problem.

\DeclareSourcemap{
  \maps{datatype=biblatexml}{
    \map{
      \regexp{./bltx:names[@type='author']/bltx:name ↦ [2]/bltx:namepart[@type='family']},
      match=\regexp{\ASmith},
      replace={Jones}
    }
    \map{
      \step{fieldsource=editor, fieldtarget=translator}
    }
  }
}

biblatexml datasources are more structured than BibTeX since they are XML. Sourcemapping is possible with biblatexml too but the specifications of source and target fields etc. also support XPath 1.0 paths in order to be able to work with the structured data. Fields can be specified as per the BibTeX examples above and these are converted into XPath 1.0 queries internally as necessary. For example:
These maps, respectively,

- Replace the family name 'Smith' of the second author name with 'Jones'
- Move the editor to translator
- Move the editor to translator but with explicit XPaths
- Set the per-namelist useprefix option on the author name list to 'false'

### 4.5.4 Data Model Specification

The data model which `biblatex` uses consists of four main elements:

- Specification of constant strings and lists of strings
- Specification of valid Entrytypes
- Specification of valid Fields along with their type, datatype and any special flags
- Specification of which Fields are valid in which Entrytypes
- Specification of constraints which can be used to validate data against the data model

The default data model is defined in the core `biblatex` file `blx-dm.def` using the macros described in this section. The default data model is described in detail in §2. The data model is used internally by `biblatex` and also by the backend. In practice, changing the data model means that you can define the entrytypes and fields for your datasources and validate your data against the data model. Naturally, this is not much use unless your style supports any new entrytypes or fields and it raises issues of portability between styles (although this can be mitigated by using the dynamic data modification functionality described in §4.5.3).

Validation against the data model means that after mapping your data sources into the data model, `biber` (using its `--validate_datamodel` option) can check:

- Whether all entrytypes are valid entrytypes
- Whether all fields are valid fields for their entrytype
- Whether the fields obey various constraints on their format which you specify

Redefining the data model can be done in several places. Style authors can create a `.dbx` file which contains the data model macros required and this will be loaded automatically when using the `biblatex` package `style` option by looking for a file named after the style with a `.dbx` extension (just like the `.cbx` and `.bbx` files for a style). If the `style` option is not used but rather the `citestyle` and `bibstyle` options, then the package will try to load `.dbx` files called `<citestyle>.dbx` and `<bibstyle>.dbx`. Alternatively, the name of the data model file can be different from any of the style option names by specifying the name (without `.dbx` extension) to the package `datamodel` option. After loading the style data model file, `biblatex` then loads, if present, a users `biblatex-dm.cfg` which should be put somewhere `biblatex` can find it, just like the main configuration file `biblatex.cfg`. To summarise, the data model is determined by adding to the data model from each of these locations, in order:

```
blx-dm.def →
<datamodel option>.dbx →
<style option>.dbx →
<citestyle option>.dbx and <bibstyle option>.dbx →
        biblatex-dm.cfg
```
It is not possible to add to a loaded data model by using the macros below in your preamble as the preamble is read after \biblatex has defined critical internal macros based on the data model. If any data model macro is used in a document, it will be ignored and a warning will be generated. The data model is defined using the following macros:

\DeclareDatamodelConstant\[⟨options⟩\]{⟨name⟩}{⟨constantdef⟩}
Declares the ⟨name⟩ as a datamodel constant with definition ⟨constantdef⟩. Such constants are typically used internally by \biber.

\DeclareDatamodelEntrytypes\[⟨options⟩\]{⟨entrytypes⟩}
Declares the comma-separated list of ⟨entrytypes⟩ to be valid entrytypes in the data model. As usual in TeX csv lists, make sure each element is immediately followed by a comma or the closing brace—no extraneous whitespace.

\DeclareDatamodelFields\[⟨options⟩\]{⟨fields⟩}
Declares the comma-separated list of ⟨fields⟩ to be valid fields in the data model with associated comma-separated ⟨options⟩. The ⟨type⟩ and ⟨datatype⟩ options are mandatory. All valid ⟨options⟩ are:

- \texttt{type=⟨field type⟩}
  
  Set the type of the field as described in §2.2.1, typically 'field' or 'list'.

- \texttt{format=⟨field format⟩}
  
  Any special format of the field. Normally unspecified but can take the value 'xsv' which tells \biber that this field has a separated values format. The exact separator can be controlled with the \biber option \texttt{xsvsep} and defaults to the expected comma surrounded by optional whitespace.

- \texttt{datatype=⟨field datatype⟩}
  
  Set the datatype of the field as described in §2.2.1. For example, 'name' or 'literal'.

- \texttt{nullok=true, false}
  
  default: false
  
  The field is allowed to be defined but empty.

- \texttt{skipout=true, false}
  
  default: false
  
  The field is not output to the .bbl and is therefore not present during \biblatex style processing. As usual in TeX csv lists, make sure each element is immediately followed by a comma or the closing brace—no extraneous whitespace.

- \texttt{label=true, false}
  
  default: false
  
  The field can be used as a label in a bibliography or bibliography list. Specifying this causes \biblatex to create several helper macros for the field so that there are some internal lengths and headings etc. defined.

\DeclareDatamodelEntryfields\[⟨entrytypes⟩\]{⟨fields⟩}
Declares that the comma-separated list of ⟨fields⟩ is valid for the comma-separated list of ⟨entrytypes⟩. If ⟨entrytypes⟩ is not given, the fields are valid for all entrytypes. As usual in TeX csv lists, make sure each element is immediately followed by a comma or the closing brace—no extraneous whitespace.

\DeclareDatamodelConstraints\[⟨entrytypes⟩\]{⟨specification⟩}
If a comma-separated list of ⟨entrytypes⟩ is given, the constraints apply only to those entrytypes. The ⟨specification⟩ is an undelimited list of \constraint directives which specify a constraint. Spaces, tabs, and line endings may be used freely to visually arrange the ⟨specification⟩. Blank lines are not permissible.

\constraint\[⟨type=constrainttype⟩\]{⟨elements⟩}
Specifies a constraint of type ⟨constrainttype⟩. Valid constraint types are:

- \texttt{type=data, mandatory, conditional}

Constraints of type ‘data’ put restrictions on the value of a field. Constraints of type ‘mandatory’ specify which fields or combinations of fields an entrytype should have. Constraints of type ‘conditional’ allow more sophisticated conditional and quantified field constraints.
For constraints of type \textit{data}, constrain field values to be the given datatype.

\texttt{\textbackslash rangemin=}⟨\texttt{num}⟩

For constraints of \texttt{\textbackslash type} ‘data’ and \texttt{\textbackslash datatype} ‘integer’, constrain field values to be at least \texttt{⟨num⟩}.

\texttt{\textbackslash rangemax=}⟨\texttt{num}⟩

For constraints of \texttt{\textbackslash type} ‘data’ and \texttt{\textbackslash datatype} ‘integer’, constrain field values to be at most \texttt{⟨num⟩}.

\texttt{\textbackslash pattern=}⟨\texttt{patt}⟩

For constraints of \texttt{\textbackslash type} ‘data’ and \texttt{\textbackslash datatype} ‘pattern’, constrain field values to match regular expression pattern \texttt{⟨patt⟩}. It is best to wrap any regular expression in the macro \texttt{\regexp}, see §4.5.3.

A \texttt{\constraintmacro} may contain any of the following:

\texttt{\constraintfieldsor\{\texttt{\fields}\}}

For constraints of \texttt{\textbackslash type} ‘mandatory’, specifies that an entry must contain a boolean OR of the \texttt{\constraintfields}.

\texttt{\constraintfieldsxor\{\texttt{\fields}\}}

For constraints of \texttt{\textbackslash type} ‘mandatory’, specifies that an entry must contain a boolean XOR of the \texttt{\constraintfields}.

\texttt{\antecedent\{\texttt{\quantifier=}⟨\texttt{quantspec}⟩\}\{\texttt{\fields}\}}

For constraints of \texttt{\textbackslash type} ‘conditional’, specifies a quantified set of \texttt{\constraintfields} which must be satisfied before the \texttt{\consequent} of the constraint is checked. \texttt{⟨quantspec⟩} should have one of the following values:

\texttt{\quantifier=all,one,none}

Specifies how many of the \texttt{\constraintfields} inside the \texttt{\antecedent} have to be present to satisfy the antecedent of the conditional constraint.

\texttt{\consequent\{\texttt{\quantifier=}⟨\texttt{quantspec}⟩\}\{\texttt{\fields}\}}

For constraints of \texttt{\textbackslash type} ‘conditional’, specifies a quantified set of \texttt{\constraintfields} which must be satisfied if the preceding \texttt{\antecedent} of the constraint was satisfied. \texttt{⟨quantspec⟩} should have one of the following values:

\texttt{\quantifier=all,one,none}

Specifies how many of the \texttt{\constraintfields} inside the \texttt{\consequent} have to be present to satisfy the consequent of the conditional constraint.

\texttt{\constraintfield\{\texttt{\field}\}}

For constraints of \texttt{\textbackslash type} ‘data’, the constraint applies to this \texttt{\field}. For constraints of \texttt{\textbackslash type} ‘mandatory’, the entry must contain this \texttt{\field}.

The data model declaration macros may be used multiple times as they append to the previous definitions. In order to replace, change or remove existing definitions (such as the default model which is loaded with \texttt{biblatex}), you should reset (clear) the current definition and then set what you want using the following macros. Typically, these macros will be the first things in any \texttt{biblatex-dm.cfg} file:

\texttt{\ResetDatamodelEntrytypes}

Clear all data model entrytype information.

\texttt{\ResetDatamodelFields}

Clear all data model field information.

\texttt{\ResetDatamodelEntryfields}

Clear all data model fields for entrytypes information.

\texttt{\ResetDatamodelConstraints}

Clear all data model fields Constraints information.

Here is an example of a simple data model. Refer to the core \texttt{biblatex} file \texttt{blx-dm.def} for the default data model specification.

\ResetDatamodelEntrytypes
\ResetDatamodelFields
\ResetDatamodelEntryfields
\ResetDatamodelConstraints

\DeclareDatamodelEntrytypes{entrytype1, entrytype2}

\DeclareDatamodelFields[type=field, datatype=literal]{field1,field2,field3,field4}

\DeclareDatamodelEntryfields{field1}
\DeclareDatamodelEntryfields[entrytype1]{field2,field3}
\DeclareDatamodelEntryfields[entrytype2]{field2,field3,field4}

\DeclareDatamodelConstraints[entrytype1]{
\constraint[type=data, datatype=integer, rangemin=3, rangemax=10]{
  \constraintfield{field1}
}
\constraint[type=mandatory]{
  \constraintfield{field1}
  \constraintfieldsxor{
    \constraintfield{field2}
    \constraintfield{field3}
  }
}
}
\DeclareDatamodelConstraints{
\constraint[type=conditional]{
  \antecedent[quantifier=none]{
    \constraintfield{field2}
  }
  \consequent[quantifier=all]{
    \constraintfield{field3}
    \constraintfield{field4}
  }
}
}

This model specifies:

- Clear the default data model completely
- Two valid entry types @entrytype1 and @entrytype2
- Four valid literal field fields
- field1 is valid for all entrytypes
- field2 and field3 are valid for entrytype1
- field2, field3 and field4 are valid for @entrytype2
- For @entrytype1:
  - field1 must be an integer between 3 and 10
  - field1 must be present
  - One and only one of field2 or field3 must be present
- For any entrytype, if field2 is not present, field3 and field4 must be present

### 4.5.5 Labels

Alphabetic styles use a label to identify bibliography entries. This label is constructed from components of the entry using a template which describes how to build the label. The template can be customised on a global or per-type basis. A separate template is used to specify how to extract parts of name fields for labels, since names can be quite complex fields.

\DeclareLabelalphaTemplate[⟨entrytype, …⟩]{⟨specification⟩}

Defines the alphabetic label template for the given entrytypes. If no entrytypes are specified in the first argument, then the global label template is defined. The ⟨specification⟩ is an undelimited list of \labelelement directives which specify the elements used to build the label. Spaces, tabs, and line endings may be used freely to visually arrange the ⟨specification⟩. Blank lines are not permissible. This command may only be used in the preamble.
\labelelement\{(elements)\}

Specifies the elements used to build the label. The \{elements\} are an undelimited list of \field or \literals which are evaluated in the order in which they are given. The first \field or \literal which expands to a non-empty string is used as the \labelelement expansion and the next \labelelement, if any, is then processed.

\field\{\{options\}\}\{\{field\}\}

If \{field\} is non-empty, use it as the current label \labelelement, subject to the options below. Useful values for \{field\} are typically the name list type fields, date fields, and title fields. You may also use the 'citekey' pseudo-field to specify the citation key as part of the label. Name list fields are treated specially and when a name list field is specified, the template defined with \DeclareLabelalphaNameTemplate is used to extract parts from the name which then returns the string that the \field option uses.

\final=true, false  \text{default: false}

This option marks a \field directive as the final one in the \{specification\}. If the \{field\} is non-empty, then this field is used for the label and the remainder of the \{specification\} will be ignored. The short form \final is equivalent to \final=true.

\lowercase=true, false  \text{default: false}

Forces the label part derived from the field to lowercase. By default, the case is taken from the field source and not modified.

\strwidth=\{integer\}  \text{default: 1}

The number of characters of the \{field\} to use. This setting may be overridden by an individual name part when extracting characters from a name. See \DeclareLabelalphaNameTemplate below.

\strside=left, right  \text{default: left}

The side of the string from which to take the \strwidth number of characters. This setting may be overridden by an individual name part when extracting characters from a name. See \DeclareLabelalphaNameTemplate below.

\padside=left, right  \text{default: right}

Side to pad the label part when using the \padchar option. Only for use with fixed-width label strings (\strwidth).

\padchar=\{character\}

If present, pads the label part on the \padside side with the specified character to the length of \strwidth. Only for use with fixed-width label strings (\strwidth).

\uppercase=true, false  \text{default: false}

Forces the label part derived from the field to uppercase. By default, the case is taken from the field source and not modified.

\varwidth=true, false  \text{default: false}

Use a variable width, left-side substring of characters from the string returned for \{field\}. The length of the string is determined by the minimum length needed to disambiguate the substring from all other \{field\} elements in the same position in the label. For name list fields, this means that each name substring is disambiguated from all other name substrings which occur in the same position in the name list (see examples below). This option overrides \strwidth if both are used. The short form \varwidth is equivalent to \varwidth=true. For name list fields, the \nameparts with the pre option set are prepended to the string returned from this disambiguation.

\varwidthnorm=true, false  \text{default: false}

As \varwidth but will force the disambiguated substrings for the \{field\} to be the same length as the longest disambiguated substring. This can be used to regularise the format of the labels if desired. This option overrides \strwidth if both are used. The short form \varwidthnorm is equivalent to \varwidthnorm=true.

\varwidthlist=true, false  \text{default: false}

Alternative method of automatic label disambiguation where the field as a whole is disambiguated from all other fields in the same label position. For non-name list fields, this is equivalent to \varwidth. For name list fields, names in a name list are not disambiguated from other names in the same position in their name lists but instead the entire name list is disambiguated as a whole from other name lists (see examples below). This option overrides \strwidth if both are used. The short form \varwidthlist is equivalent to \varwidthlist=true. For name list fields, the \nameparts with the pre option set are prepended to the string returned from this disambiguation.

\strwidthmax=\{integer\}

When using \varwidth, this option sets a limit (in number of characters) on the length of variable width substrings. This option can be used to regularise the label.
\texttt{strfixedcount=(integer)} \hspace{1cm} \text{default: 1}

When using \texttt{varwidthnorm}, there must be at least \texttt{strfixedcount} disambiguated substrings with the same, maximal length to trigger the forcing of all disambiguated substrings to this same maximal length.

\textbf{ifnames=(range)}

Only use this \texttt{\field} specification if it is a name list field with a number of names matching the \texttt{ifnames} range value. This allows a \texttt{\labelelement} to be conditionalised on name length (see below). The range can specified as in the following examples:

<table>
<thead>
<tr>
<th>\texttt{ifnames}</th>
<th>\hspace{1cm} Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Only apply to name lists containing exactly 3 names</td>
</tr>
<tr>
<td>{2-4}</td>
<td>Only apply to name lists containing minimum 2 and maximum 4 names</td>
</tr>
<tr>
<td>{3}</td>
<td>Only apply to name lists containing at most 3 names</td>
</tr>
<tr>
<td>{2-}</td>
<td>Only apply to name lists containing at least 2 names</td>
</tr>
</tbody>
</table>

\textbf{names=(range)}

By default, for name list fields, the names used range from the first name to the \texttt{maxalphanames/minalphanames} truncation. This option can be used to override this with an explicit range of names to consider. The plus ‘+’ sign is a special end of range marker denoting the truncation point of \texttt{max/minalphanames}. The range separator can be any number of characters with the Unicode Dash property. For example:

<table>
<thead>
<tr>
<th>\texttt{names}</th>
<th>\hspace{1cm} Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Use first 3 names in the name list</td>
</tr>
<tr>
<td>{2-3}</td>
<td>Use second and thirds names only</td>
</tr>
<tr>
<td>{1-3}</td>
<td>Same as 1-3</td>
</tr>
<tr>
<td>{2-}</td>
<td>Use all names starting with the second name (ignoring max/minalphanames truncation)</td>
</tr>
<tr>
<td>{2-+}</td>
<td>Use all names starting with the second name (respecting max/minalphanames truncation)</td>
</tr>
</tbody>
</table>

\textbf{namessep=(string)} \hspace{1cm} \text{default: empty}

An arbitrary string separator to put between names in a namelist.

\textbf{noalphaothers=true, false} \hspace{1cm} \text{default: false}

By default, \texttt{\labelalphaothers} is appended to label parts derived from name lists if there are more names in the list than are shown in the label part. This option can be used to disable the default behaviour.

\textit{\texttt{\literal{}}}

Insert the literal \texttt{\literal{}} into the label at this point.

When a name list \texttt{\field} is specified, the method of extracting the string is specified by a separate template specified by the following command:

\texttt{\DeclareLabelalphaNameTemplate\{\texttt{entrytype,...}\}\{\texttt{specification}\}}

Specifies the template to use to extract a label string from a name list when a \texttt{\field} specification in \texttt{\DeclareLabelalphaTemplate} contains a name list. The template can be specified per-entrytype.

\texttt{\namepart\{\texttt{options}\}\{\texttt{namepart}\}}

\texttt{namepart} is one of the datamodel nameparts defined with the \texttt{\DeclareDatamodelConstant} command (see §4.2.3). The options are:

\hspace{1cm} \textbf{use=true, false} \hspace{1cm} \text{default: false}

Only use the \texttt{namepart} in constructing the label information if there is a corresponding option \texttt{use\texttt{namepart}} and that option is true.

\hspace{1cm} \textbf{pre=true, false} \hspace{1cm} \text{default: false}

When constructing label strings from names, the \texttt{namepart} \emph{without} a \texttt{pre} option will be used to construct label string, passing through disambiguation, substring etc. operations as specified by the \texttt{\field} options in \texttt{\DeclareLabelalphaTemplate}. Then the \texttt{namepart} options with the \texttt{pre} option set will be prepended to the result, (in the order given, if there are more than one such \texttt{nameparts}). This allows to unconditionally prepend certain namepart information to name label strings, like name prefixes. Note that the uppercase and lowercase options of \texttt{\field} in \texttt{\DeclareLabelalphaTemplate} are applied to the entire label returned from \texttt{\DeclareLabelalphaTemplate}, both \texttt{pre} parts and non-\texttt{pre}.
For static (non-varwidth) disambiguation in `\DeclareLabelalphaTemplate`, nameparts separated by whitespace or hyphens (compound names) as separate names for label generation. This means that when forming a label out of, for example the surname 'Ballam Forsyth' with a 1 character, left-side substring, this name would give 'BF' with compound=true and 'B' with compound=false. The short form compound is equivalent to compound=true.

\begin{verbatim}
\DeclareLabelalphaTemplate{
  \labelelement{
    \field[final]{shorthand}
    \field{label}
    \field[strwidth=3,strside=left,ifnames=1]{labelname}
  }
  \labelelement{
    \field[strwidth=2,strside=right]{year}
  }
}
\DeclareLabelalphaNameTemplate{
  \namepart[use=true, pre=true, strwidth=1, compound=true]{prefix}
  \namepart{family}
}
\end{verbatim}

To get an idea of how the label automatic disambiguation works, consider the following author lists:

```
Agassi, Chang, Laver  (2000)
Agassi, Connors, Lendl  (2001)
Agassi, Courier, Laver  (2002)
Borg, Connors, Edberg  (2003)
Borg, Connors, Emerson  (2004)
```

Assuming a template declaration such as:

```
\DeclareLabelalphaTemplate{
  \labelelement{
    \field[varwidth]{labelname}
  }
}
```
Then the labels would be:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Agassi, Chang, Laver</td>
<td>[AChLa]</td>
</tr>
<tr>
<td>Agassi, Connors, Lendl</td>
<td>[AConLe]</td>
</tr>
<tr>
<td>Agassi, Courier, Laver</td>
<td>[ACouLa]</td>
</tr>
<tr>
<td>Borg, Connors, Edberg</td>
<td>[BConEd]</td>
</tr>
<tr>
<td>Borg, Connors, Emerson</td>
<td>[BConEm]</td>
</tr>
</tbody>
</table>

With normalised variable width labels defined:

\[
\text{\textbackslash DeclareLabelalphaTemplate{}\textbackslash labelelement{\textbackslash field[varwidthnorm]{\textit{labelname}}}}
\]

You would get the following as the substrings of names in each position are extended to the length of the longest substring in that same position:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Agassi, Chang, Laver</td>
<td>[AChLa]</td>
</tr>
<tr>
<td>Agassi, Connors, Lendl</td>
<td>[AConLe]</td>
</tr>
<tr>
<td>Agassi, Courier, Laver</td>
<td>[ACouLa]</td>
</tr>
<tr>
<td>Borg, Connors, Edberg</td>
<td>[BConEd]</td>
</tr>
<tr>
<td>Borg, Connors, Emerson</td>
<td>[BConEm]</td>
</tr>
</tbody>
</table>

With a restriction to two characters for the name components of the label element defined like this:

\[
\text{\textbackslash DeclareLabelalphaTemplate{}\textbackslash labelelement{\textbackslash field[varwidthnorm,strwidthmax=2]{\textit{labelname}}}}
\]

This would be the result (note that the individual family name label parts are no longer unambiguous):

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Agassi, Chang, Laver</td>
<td>[AChLa]</td>
</tr>
<tr>
<td>Agassi, Connors, Lendl</td>
<td>[ACoLe]</td>
</tr>
<tr>
<td>Agassi, Courier, Laver</td>
<td>[ACoLa]</td>
</tr>
<tr>
<td>Borg, Connors, Edberg</td>
<td>[BCoEd]</td>
</tr>
<tr>
<td>Borg, Connors, Emerson</td>
<td>[BCEm]</td>
</tr>
</tbody>
</table>

Alternatively, you could choose to disambiguate the name lists as a whole with:

\[
\text{\textbackslash DeclareLabelalphaTemplate{}\textbackslash labelelement{\textbackslash field[varwidthlist]{\textit{labelname}}}}
\]

Which would result in:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Agassi, Chang, Laver</td>
<td>[AChL]</td>
</tr>
<tr>
<td>Agassi, Connors, Lendl</td>
<td>[ACoL]</td>
</tr>
<tr>
<td>Agassi, Courier, Laver</td>
<td>[ACL]</td>
</tr>
<tr>
<td>Borg, Connors, Edberg</td>
<td>[BCEd]</td>
</tr>
<tr>
<td>Borg, Connors, Emerson</td>
<td>[BCE]</td>
</tr>
</tbody>
</table>

Perhaps you only want to consider at most two names for label generation but disambiguate at the whole name list level:
Which would result in:

<table>
<thead>
<tr>
<th>Team</th>
<th>Opening Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agassi, Chang, Laver</td>
<td>[ACh+]</td>
</tr>
<tr>
<td>Agassi, Connors, Lendl</td>
<td>[ACo+]</td>
</tr>
<tr>
<td>Agassi, Courier, Laver</td>
<td>[AC+]</td>
</tr>
<tr>
<td>Borg, Connors, Edberg</td>
<td>[BC+a]</td>
</tr>
<tr>
<td>Borg, Connors, Emerson</td>
<td>[BC+b]</td>
</tr>
</tbody>
</table>

In this last example, you can see \labelalphaothers has been appended to show that there are more names. The last two labels now require disambiguating with \extraalpha as there is no way of disambiguating this label name list with only two names.

Finally, here is an example using multiple label elements:

Which would result in:

<table>
<thead>
<tr>
<th>Team</th>
<th>Opening Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agassi, Chang, Laver</td>
<td>[AChL-000]</td>
</tr>
<tr>
<td>Agassi, Connors, Lendl</td>
<td>[ACoL-001]</td>
</tr>
<tr>
<td>Agassi, Courier, Laver</td>
<td>[ACouL-002]</td>
</tr>
<tr>
<td>Borg, Connors, Edberg</td>
<td>[BCEd-003]</td>
</tr>
<tr>
<td>Borg, Connors, Emerson</td>
<td>[BCEm-004]</td>
</tr>
</tbody>
</table>

Here is another rather contrived example showing that you don’t need to specially quote LaTeX special characters (apart from ‘%’, obviously) when specifying padding characters and literals:
which given:

```latex
@Book{test,  
  author = {XXX YY and WWW ZZ},  
  title = {T},  
  year = {2007},  
}
```

would resulting a label looking like this:

```
[>|YY/ZZ__&&T07]
```

Generating labels from fields may involve some difficulties when you have fields containing diacritics, hyphens, spaces etc. Often, you want to ignore things like separator characters or spaces when generating labels. An option is provided to customise the regular expression(s) to strip from a field before it is passed to the label generation system.

\begin{verbatim}
\DeclareNolabel{⟨specification⟩}
\end{verbatim}

Defines regular expressions to strip from any field before generating a label part for the field. The (specification) is an undelimited list of \nolabel directives which specify the regular expressions to remove from fields. Spaces, tabs and line endings may be used freely to visually arrange the (specification). Blank lines are not permissible. This command may only be used in the preamble.

\begin{verbatim}
\nolabel{⟨regexp⟩}
\end{verbatim}

Any number of \nolabel commands can be given each of which specifies to remove the ⟨regexp⟩ from the copy of the field which the label generation system sees. Since regular expressions usually contain special characters, it is best to enclose them in the provided \regexp macro as shown—this will pass the expression through to biber correctly.

If there is no \DeclareNolabel specification, biber will default to:

```
\DeclareNolabel{  
  % strip punctuation, symbols, separator and control characters  
  \nolabel{\regexp{[\p{P}\p{S}\p{C}]}}
}
```

This biber default strips punctuation, symbol, separator and control characters from fields before passing the field string to the label generation system.

\begin{verbatim}
\DeclareNolabelwidthcount{⟨specification⟩}
\end{verbatim}

Defines regular expressions to ignore from any field when counting characters in fixed-width substrings. The (specification) is an undelimited list of \nolabelwidthcount directives which specify the regular expressions to ignore when counting characters for fixed-width substrings. Spaces, tabs and line endings may be used freely to visually arrange the (specification). Blank lines are not permissible. This command may only be used in the preamble.

\begin{verbatim}
\nolabelwidthcount{⟨regexp⟩}
\end{verbatim}

Any number of \nolabelwidthcount commands can be given each of which specifies to ignore the ⟨regexp⟩ when generating fixed-width substrings during label generation. Since regular expressions usually contain special characters, it is best to enclose them in the provided \regexp macro as shown—this will pass the expression through to biber correctly.

There is no default \DeclareNolabelwidthcount specification. Note that this setting is only taken into account when using fixed-width substrings (non-varwidth) during label part generation. See § 4.5.5.

### 4.5.6 Sorting

In addition to the predefined sorting schemes discussed in § 3.5, it is possible to define new ones or modify the default definitions. The sorting process may be customized further by excluding certain fields from sorting on a per-type basis and by automatically populating the presort field on a per-type basis.
\DeclareSortingScheme[{options}]{{(name)}}{{(specification)}}

Defines the sorting scheme \(\text{(name)}\). The \(\text{(name)}\) is the identifier passed to the sorting option (§ 3.1.2.1) when selecting the sorting scheme. The \\text{\DeclareSortingScheme} command supports the following optional arguments:

\text{locale}={locale}

The locale for the sorting scheme which then overrides the global sorting locale in the sortlocale option discussed in § 3.1.2.1.

The \(\text{(specification)}\) is an undelimited list of \text{\sort} directives which specify the elements to be considered in the sorting process. Spaces, tabs, and line endings may be used freely to visually arrange the \(\text{(specification)}\). Blank lines are not permissible. This command may only be used in the preamble.

\text{\sort}\{(elements)\}

Specifies the elements considered in the sorting process. The \(\text{(elements)}\) are an undelimited list of \text{\field}, \text{\literal}, and \text{\citeorder} commands which are evaluated in the order in which they are given. If an element is defined, it is added to the sort key and the sorting routine skips to the next \text{\sort} directive. If it is undefined, the next element is evaluated. Since literal strings are always defined, any \text{\literal} commands should be the sole or the last element in a \text{\sort} directive. All \(\text{(elements)}\) should be the same datatype as described in § 2.2.2 since they will be potentially compared to any of the other \(\text{(elements)}\) in other entries.. The \\text{\sort} command supports the following optional arguments:

\text{locale}={locale}

Override the locale used for sorting at the level of a particular set of sorting elements. If specified, the locale overrides the locale set at the level of \text{\DeclareSortingScheme} and also the global setting. See also the discussion of the global sorting locale option sortlocale in § 3.1.2.1.

\text{direction}={ascending, descending}

default: ascending

The sort direction, which may be either ascending or descending. The default is ascending order.

\text{final}={true, false}

default: false

This option marks a \text{\sort} directive as the final one in the \(\text{(specification)}\). If one of the \(\text{(elements)}\) is available, the remainder of the \(\text{(specification)}\) will be ignored. The short form \text{final} is equivalent to \text{final}={true}.

\text{sortcase}={true, false}

Whether or not to sort case-sensitively. The default setting depends on the global sortcase option.

\text{sortupper}={true, false}

Whether or not to sort in 'uppercase before lowercase' (true) or 'lowercase before uppercase' order (false). The default setting depends on the global sortupper option.

\text{\field}\{(key=value, ...)}\{(field)\}

The \text{\field} element adds a \(\text{(field)}\) to the sorting specification. If the \(\text{(field)}\) is undefined, the element is skipped. The \text{\field} command supports the following optional arguments:

\text{padside}={left, right}

default: left

Pads a field on the left or right side using \text{padchar} so that its width is \text{padwidth}. If no padding option is set, no padding is done at all. If any padding option is specified, then padding is performed and the missing options are assigned built-in default values. If padding and substring matching are both specified, the substring match is performed first.

\text{padwidth}={integer}

default: 4

The target width in characters.

\text{padchar}={character}

default: 0

The character to be used when padding the field.

\text{strside}={left, right}

default: left

Performs a substring match on the left or right side of the field. The number of characters to match is specified by the corresponding \text{strwidth} option. If no substring option is set, no substring matching is performed at all. If any substring option is specified, then substring matching is performed and the missing options are assigned built-in default values. If padding and substring matching are both specified, the substring match is performed first.

\text{strwidth}={integer}

default: 4

The number of characters to match.

\text{\literal}\{(string)\}

The \text{\literal} element adds a literal \(\text{(string)}\) to the sorting specification. This is useful as a fallback if some fields are not available.
The `\citeorder` element has a special meaning. It requests a sort based on the lexical order of the actual citations. For entries cited within the same citation command like:

```
cite{one,two}
```

there is a distinction between the lexical order and the semantic order. Here "one" and "two" have the same semantic order but a unique lexical order. The semantic order only matters if you specify further sorting to disambiguate entries with the same semantic order. For example, this is the definition of the `none` sorting scheme:

```
\DeclareSortingScheme{none}{
  \sort{\citeorder}
}
```

This sorts the bibliography purely lexically by the order of the keys in the citation commands. In the example above, it sorts "one" before "two". However, suppose that you consider "one" and "two" to have the same order (semantic order) since they are cited at the same time and want to further sort these by year. Suppose "two" has an earlier `year` than "one":

```
\DeclareSortingScheme{noneyear}{
  \sort{\citeorder}
  \sort{year}
}
```

This sorts "two" before "one", even though lexically, "one" would sort before "two". This is possible because the semantic order can be disambiguated by the further sorting on year. With the standard `none` sorting scheme, the lexical order and semantic order are identical because there is nothing further to disambiguate them. This means that you can use `\citeorder` just like any other sorting specification element, choosing how to further sort entries cited at the same time (in the same citation command).

```
\DeclareSortingNamekeyScheme[⟨schemename⟩]{{⟨specification⟩}}
```

Defines how the sorting keys for names are constructed. This can change the sorting order of names arbitrarily because you can choose how to put together the name parts when constructing the string to compare when sorting. The sorting key construction scheme so defined is called `⟨schemename⟩` which defaults to “global” if this optional parameter is absent. When constructing the sorting key for a name, a sorting key for each name part is constructed and the key for each name is formed into an ordered key list with a special internal separator. The point of this option is to accommodate languages or situations where sorting of names needs to be customised (for example, Icelandic names are sometimes sorted by given names rather than by family names). This macro may be used multiple times to define schemes with different names which can then be referred to later. Sorting name key schemes can have the following scopes, in order of increasing precedence:

- The default scheme defined without the optional name argument
- Given as the `sortnamekeyscheme` option to a reference context (see §3.7.10)
- Given as a per-entry option `sortnamekeyscheme` in a bibliography data source entry
- Given as a per-namelist option `sortnamekeyscheme`
- Given as a per-name option `sortnamekeyscheme`

By default there is only a global scheme which has the following `⟨specification⟩`:

```
\DeclareSortingNamekeyScheme{
  \keypart{
    \namepart[use=true]{prefix}
  }
  \keypart{
    \namepart{family}
  }
  \keypart{
    \namepart{given}
  }
```

This means that the key is constructed by concatenating, in order, the name prefix (only if the `useprefix` option is true), the family name(s), the given name(s), the name suffix and then the name prefix (only if the `useprefix` option is false).

\keypart{(part)}
(part) is an ordered list of \namepart{} and \literal{} specifications which are concatenated together when constructing a part of the name sorting key.

\literal{(string)}
A literal string to insert into the name sorting key.

\namepart{(name)}
Specifies the (name) of a namepart to use in constructing the name sorting key.

\use{true,false} default: true
Indicates that the namepart (name) is only to be used in this concatenation position if the corresponding use ‘name’ option is set to the specified boolean value.

\initst{true,false} default: true
Indicates that only the initials of namepart (name) are to be used in constructing the sorting specification.

As an example, suppose you wanted to be able to sort names by given name rather than family name, you could define a sorting name key scheme like this:

\Declarationsortingnamekeyscheme[givenfirst]{
  \namepart{given}
  \namepart[use=true]{prefix}
  \namepart{family}
  \namepart[use=false]{prefix}
}

You can then use the name `givenfirst` at the appropriate scope in order to make `biber` use this scheme when constructing sorting name keys. For example, you could enable this for one bibliography list like this:

\begin{refcontext}{sortnamekeyscheme=givenfirst}
\printbibliography
\end{refcontext}

or perhaps you only want to do this for a particular entry:

@BOOK{key,
  OPTIONS = {sortnamekeyscheme=givenfirst},
  AUTHOR = {Arnar Vigfusson}
}
or just a name list by using the option as a pseudo-name which will be ignored:

```latex
@BOOK{key,
  AUTHOR = {sortnamekeyscheme=givenfirst and Arnar Vigfusson}
}
```

or just a single name by passing the option as part of the extended name information format which biber supports (see biber doc):

```latex
@BOOK{key,
  AUTHOR = {given=Arnar, family=Vigfusson, sortnamekeyscheme=

  ↪
  →
  givenfirst}
}
```

Now we give some examples of sorting schemes. In the first example, we define a simple name/title/year scheme. The name element may be either the author, the editor, or the translator. Given this specification, the sorting routine will use the first element which is available and continue with the title. Note that the options use<name> options are considered automatically in the sorting process:

```latex
\DeclareSortingScheme{sample}{
  \sort{
    \field{author}
    \field{editor}
    \field{translator}
  }
  \sort{
    \field{title}
  }
  \sort{
    \field{year}
  }
}
```

In the next example, we define the same scheme in a more elaborate way, considering special fields such as presort, sortkey, sortname, etc. Since the sortkey field specifies the master sort key, it needs to override all other elements except for presort. This is indicated by the final option. If the sortkey field is available, processing will stop at this point. If not, the sorting routine continues with the next \sort directive. This setup corresponds to the default definition of the nty scheme:

```latex
\DeclareSortingScheme{nty}{
  \sort{
    \field{presort}
  }
  \sort[final]{
    \field{sortkey}
  }
  \sort{
    \field{sortname}
    \field{author}
    \field{editor}
    \field{translator}
    \field{sorttitle}
    \field{title}
  }
  \sort{
    \field{sorttitle}
    \field{title}
  }
  \sort{
    \field{sortyear}
  }
}
```
Finally, here is an example of a sorting scheme which overrides the global sorting locale and additionally overrides again when sorting by the \texttt{origtitle} field. Note the use in the scheme-level override of a \texttt{babel/polyglossia} language name instead of a real locale identifier. \texttt{biber} will map this to a suitable, real locale identifier (in this case, \texttt{sv_SE}):}

\begin{verbatim}
\DeclareSortingScheme[locale=swedish]{custom}{
  \sort{
    \field{sortname}
    \field{author}
    \field{editor}
    \field{translator}
    \field{sorttitle}
    \field{title}
  }
  \sort[locale=de_DE_phonebook]{
    \field{origtitle}
  }
}
\end{verbatim}

\begin{verbatim}
\DeclareSortExclusion{(entrytype, ...)}{(field, ...)}
\end{verbatim}

Specifies fields to be excluded from sorting on a per-type basis. The \texttt{(entrytype)} argument and the \texttt{(field)} argument may be a comma-separated list of values. A blank \texttt{(field)} argument will clear all exclusions for this \texttt{(entrytype)}. A value of "+" for \texttt{(entrytype)} will exclude \texttt{(field,...)} for every \texttt{entrytype}. This is equivalent to simply deleting the field from the sorting specification and is only normally used in combination with \texttt{\DeclareSortInclusion} when one wishes to exclude a field for all but explicitly included \texttt{entrytypes}. See example in \texttt{\DeclareSortInclusion} below. This command may only be used in the preamble.

\begin{verbatim}
\DeclareSortInclusion{(entrytype, ...)}{(field, ...)}
\end{verbatim}

Only used along with \texttt{\DeclareSortExclusion}. Specifies fields to be included in sorting on a per-type basis. This allows the user to exclude a field from sorting for all \texttt{entrytypes} and then to override this for certain \texttt{entrytypes}. This is easier sometimes than using \texttt{\DeclareSortExclusion} to list exclusions for many \texttt{entrytypes}. The \texttt{(entrytype)} argument and the \texttt{(field)} argument may be a comma-separated list of values. This command may only be used in the preamble. For example, this would use \texttt{title} during sorting only for \texttt{article}s:

\begin{verbatim}
\DeclareSortExclusion{*}{title}
\DeclareSortInclusion{article}{title}
\end{verbatim}

\begin{verbatim}
\DeclarePresort{(entrytype, ...)}{(string)}
\end{verbatim}

Specifies a string to be used to automatically populate the \texttt{presort} field of entries without a \texttt{presort} field. The \texttt{presort} may be defined globally or on a per-type basis. If the optional \texttt{(entrytype)} argument is given, the \texttt{(string)} applies to the respective entry type. If not, it serves as the global default value. Specifying an \texttt{(entrytype)} in conjunction with a blank \texttt{(string)} will clear the type-specific setting. The \texttt{(entrytype)} argument may be a comma-separated list of values. This command may only be used in the preamble.

\begin{verbatim}
\DeclareSortTranslit{(entrytype)}{(specification)}
\end{verbatim}

Languages which can be written in different scripts or alphabets often only have CLDR sorting tailoring for one script and it is expected that you transliterate into the supported script for sorting purposes. A common example is Sanskrit which is often written in academic contexts in IAST romanised script but which needs to be sorted in the ‘sa’ locale which expects the Devanāgarī script. This means that it is necessary to transliterate into the sorting script internally. \texttt{\DeclareSortTranslit} declares which parts of an entry you would like to transliterate for sorting purposes. Without the \texttt{(entrytype)} parameter, the \texttt{(specification)} applies to all \texttt{entrytypes}. The \texttt{(specification)} is one or more \texttt{\translit} commands:
<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iast</td>
<td>devanagari</td>
<td>Sanskrit IAST\textsuperscript{14} to Devanāgarī</td>
</tr>
</tbody>
</table>

**Table 11: Valid transliteration pairs**

\transit{(\textit{field} or \textit{fieldset})}{\langle\textit{from}\rangle}{\langle\textit{to}\rangle}

Specifies that the data field \textit{field} or all fields in a fieldset \textit{fieldset} declared with \texttt{\DeclareDatafieldSet} (see §4.5.2) should be transliterated from script \langle\textit{from}\rangle to script \langle\textit{to}\rangle for sorting purposes. The field/set argument can also be "" to apply transliteration to all fields. The valid \langle\textit{from}\rangle and \langle\textit{to}\rangle values are given in table 11. Note that \biblatex does not aim to support general transliteration, only those which are useful for sorting purposes. Please open a GitHub ticket for \biblatex if you think you need additional transliterations.

An example of transliterating titles so that they sort correctly in Sanskrit:

\begin{verbatim}
\DeclareDatafieldSet{settitles}{
  \member[\textit{field}=title]
  \member[\textit{field}=booktitle]
  \member[\textit{field}=eventtitle]
  \member[\textit{field}=issuetitle]
  \member[\textit{field}=journaltitle]
  \member[\textit{field}=maintitle]
  \member[\textit{field}=origtitle]
}
\DeclareSortTranslit{\transit{\textit{settitles}}{\textit{iast}}{\textit{devanagari}}}
\end{verbatim}

4.5.7 Bibliography List Filters

When using customisable bibliography lists (See §3.7.3), usually one wants to return in the \texttt{.bbl} only those entries which have the particular fields which the bibliography list is summarising. For example, when printing a normal list of shorthands, you want the list returned by \texttt{biber} in the \texttt{.bbl} to contain only those entries which have a shorthand field. This is accomplished by defining a bibliography list filter using the \texttt{\DeclareBiblistFilter} command. This differs from the filters defined using \texttt{\defbibfilter} (see §3.7.9) since the filters defined by \texttt{\defbibfilter} run inside \biblatex after the \texttt{.bbl} has been generated. In addition, bibliography lists in the \texttt{.bbl} do not contain entry data, only the citation keys for the entries and so no filtering by \biblatex using \texttt{\defbibfilter} is possible for bibliography lists.

\begin{verbatim}
\DeclareBiblistFilter{(\textit{name})}{(\textit{specification})}
\end{verbatim}

Defines a bibliography list filter with \textit{name}. The \textit{specification} consists of one or more \texttt{\filter} or \texttt{\filteror} macros, all of which must be satisfied for the entry to pass the filter:

\begin{verbatim}
\filter{(\textit{filterspec})}{\langle\textit{filter}\rangle}
\end{verbatim}

Filter entries according to the \textit{filterspec} and \textit{filter}. \textit{filterspec} can be one of:

- \texttt{type/nottype} Entry is/is not of \textit{entrytype} \texttt{\filter}
- \texttt{subtype/notsubtype} Entry is/is not of \textit{subtype} \texttt{\filter}
- \texttt{keyword/notkeyword} Entry has/does not have \textit{keyword} \texttt{\filter}
- \texttt{field/notfield} Entry has/does not have a field called \texttt{\filter}

\begin{verbatim}
\filteror{(\textit{filters})}{\langle\textit{filters}\rangle}
\end{verbatim}

A wrapper around one or more \texttt{\filter} commands specifying that they form a disjunctive set, i.e. any one of the \texttt{\filters} must be satisfied.

Fields in the datamodel which are marked as 'Label fields' (see §4.5.4) automatically have a filter defined for them with the same name and which filters out any entries which do not contain the field. For example, \biblatex automatically generates a filter for the \texttt{shorthand} field:
4.5.8 Controlling Name Initials Generation

Generating initials for name parts from a given name involves some difficulties when you have names with prefixes, diacritics, hyphens etc. Often, you want to ignore things like prefixes when generating initials so that the initials for “al-Hasan” is just “H” instead of “a-H”. This is tricky when you also have names like “Ho-Pun” where you want the initials to be “H-P”, for example.

\DeclareNoinit{⟨specification⟩}

Defines regular expressions to strip from names before generating initials. The ⟨specification⟩ is an undelimited list of \noinit directives which specify the regular expressions to remove from the name. Spaces, tabs and line endings may be used freely to visually arrange the ⟨specification⟩. Blank lines are not permissible. This command may only be used in the preamble.

\noinit{⟨regexp⟩}

Any number of \noinit commands can be given each of which specifies to remove the ⟨regexp⟩ from the copy of the name which the initials generation system sees. Since regular expressions usually contain special characters, it is best to enclose them in the provided \regexp macro as shown—this will pass the expression through to biber correctly.

If there is no \DeclareNoinit specification, biber will default to:

\DeclareNoinit{
% strip lowercase prefixes like 'al-' when generating initials from names
\noinit{\regexp{\b\p{Ll}{2}\p{Pd}}} \\
% strip some common diacritics when generating initials from names
\noinit{\regexp{[{\x{2bf}\x{2018}]}}}
}

This biber default strips a couple of diacritics and also strips lowercase prefixes from names before generating initials.

4.5.9 Fine Tuning Sorting

It can be useful to fine tune sorting so that it ignores certain parts of particular fields.

\DeclareNosort{⟨specification⟩}

Defines regular expressions to strip from particular fields or types of fields when sorting. The ⟨specification⟩ is an undelimited list of \nosort directives which specify the regular expressions to remove from particular fields or type of field. Spaces, tabs and line endings may be used freely to visually arrange the ⟨specification⟩. Blank lines are not permissible. This command may only be used in the preamble.

\nosort{⟨field or field type⟩}{⟨regexp⟩}

Any number of \nosort commands can be given each of which specifies to remove the ⟨regexp⟩ from the ⟨field⟩ or ⟨field type⟩. A ⟨field type⟩ is simple a convenience grouping of semantically similar fields from which you might want to remove a regexp. Table 12 shows the available field types. Since regular expressions usually contain special characters, it is best to enclose them in the provided \regexp macro as shown—this will pass the expression through to biber correctly.

The default is:

\DeclareNosort{
% strip prefixes like 'al-' when sorting names
\nosort{type_names}{\regexp{\A\p{L}{2}\p{Pd}}} \\
% strip some diacritics when sorting names
\nosort{type_names}{\regexp{[{\x{2bf}\x{2018}]}}}
}
This `biber` default strips a couple of diacritics and also strips prefixes from names when sorting. Suppose you wanted to ignore "The" at the beginning of a `title` field when sorting:

\begin{verbatim}
\DeclareNosort{
  \nosort{title}{\regexp{\AThe\s+}}
}
\end{verbatim}

Or if you wanted to ignore "The" at the beginning of any title field:

\begin{verbatim}
\DeclareNosort{
  \nosort{type_title}{\regexp{\AThe\s+}}
}
\end{verbatim}

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>type_name</td>
<td>author</td>
</tr>
<tr>
<td></td>
<td>afterword</td>
</tr>
<tr>
<td></td>
<td>annotator</td>
</tr>
<tr>
<td></td>
<td>bookauthor</td>
</tr>
<tr>
<td></td>
<td>commentator</td>
</tr>
<tr>
<td></td>
<td>editor</td>
</tr>
<tr>
<td></td>
<td>editora</td>
</tr>
<tr>
<td></td>
<td>editorb</td>
</tr>
<tr>
<td></td>
<td>editorc</td>
</tr>
<tr>
<td></td>
<td>foreword</td>
</tr>
<tr>
<td></td>
<td>holder</td>
</tr>
<tr>
<td></td>
<td>introduction</td>
</tr>
<tr>
<td></td>
<td>namea</td>
</tr>
<tr>
<td></td>
<td>nameb</td>
</tr>
<tr>
<td></td>
<td>namec</td>
</tr>
<tr>
<td></td>
<td>shortauthor</td>
</tr>
<tr>
<td></td>
<td>shorteditor</td>
</tr>
<tr>
<td></td>
<td>translator</td>
</tr>
<tr>
<td>type_title</td>
<td>booktitle</td>
</tr>
<tr>
<td></td>
<td>eventtitle</td>
</tr>
<tr>
<td></td>
<td>issuettile</td>
</tr>
<tr>
<td></td>
<td>journaltitle</td>
</tr>
<tr>
<td></td>
<td>maintitle</td>
</tr>
<tr>
<td></td>
<td>origtitle</td>
</tr>
<tr>
<td></td>
<td>title</td>
</tr>
</tbody>
</table>

\textbf{Table 12: Field types for \texttt{\nosort}}

\subsection*{4.5.10 Special Fields}

Some of the automatically generated fields from §4.2.4.2 may be customized.

\begin{verbatim}
\DeclareLabelname{⟨entrytype, …⟩}{⟨specification⟩}
\end{verbatim}

Defines the fields to consider when generating the `labelname` field (see §4.2.4.2). The `⟨specification⟩` is an ordered list of \texttt{\field} commands. The fields are checked in the order listed and the first field which is available will be used as `labelname`. This is the default definition:

\begin{verbatim}
\DeclareLabelname{%
  \field{shortauthor}
  \field{author}
  \field{shorteditor}
  \field{editor}
  \field{translator}
  \field{translator}
}
\end{verbatim}
The labelname field may be customized globally or on a per-type basis. If the optional \langle entrytype \rangle argument is given, the specification applies to the respective entry type. If not, it is applied globally. The \langle entrytype \rangle argument may be a comma-separated list of values. This command may only be used in the preamble.

\DeclareLabeldate{\langle entrytype, ... \rangle}{\langle specification \rangle}

Defines the date components to consider when generating labelyear, labelmonth, labelday, labelendyear, labelendmonth and labelendday fields (see §4.2.4.2). The \langle specification \rangle is an ordered list of \field or \literal commands. The items are checked in the order listed and the first item which is available will be used to populate the mentioned fields. Note that the \field items do not have to be datetype 'date' in the data model so that you can create pseudo-year labels by, for example, using a pubstate field contents, if available, as the year label by defining \DeclareLabeldate suitably. Note also that a \literal command will always be used when found and so this should always be the last thing in the list. If the value of a \literal command is a valid localisation string, then this will be resolved in the current language, otherwise the value is used as a literal string as-is. This is the default definition:

\DeclareLabeldate{
  \field{date}
  \field{year}
  \field{eventdate}
  \field{urldate}
  \literal{nodate}
}

Note that the date field is split by the backend into year, month which are also valid fields in the default data model. In order to support legacy data which directly sets year and/or month, the specification 'date' in \DeclareLabeldate will also match year and month fields, if present. The label* fields may be customized globally or on a per-type basis. If the optional \langle entrytype \rangle argument is given, the specification applies to the respective entry type. If not, it is applied globally. The \langle entrytype \rangle argument may be a comma-separated list of values. This command may only be used in the preamble. See also §4.2.4.3.

\DeclareLabeltitle{\langle entrytype, ... \rangle}{\langle specification \rangle}

Defines the fields to consider when generating the labeltitle field (see §4.2.4.2). The \langle specification \rangle is an ordered list of \field commands. The fields are checked in the order listed and the first field which is available will be used as labeltitle. This is the default definition:

\DeclareLabeltitle{
  \field{shorttitle}
  \field{title}
}

The labeltitle field may be customized globally or on a per-type basis. If the optional \langle entrytype \rangle argument is given, the specification applies to the respective entry type. If not, it is applied globally. The \langle entrytype \rangle argument may be a comma-separated list of values. This command may only be used in the preamble.

4.5.11 Data Inheritance (crossref)

biber features a highly customizable cross-referencing mechanism with flexible data inheritance rules. This sections deals with the configuration interface. See appendix B for the default configuration. A note on terminology: the child or target is the entry with the crossref field, the parent or source is the entry the crossref field points to. The child inherits data from the parent.

\DefaultInheritance{\langle exceptions \rangle}{\langle options \rangle}

Configures the default inheritance behavior. This command may only be used in the preamble. The default behavior may be customized be setting the following \langle options \rangle:

- all=true, false  
  default: true

Whether or not to inherit all fields from the parent by default.
all\text{=true} means that the child entry inherits all fields from the parent, unless a more specific
inheritance rule has been set up with \texttt{\DeclareDataInheritance}. If an inheritance rule is
defined for a field, data inheritance is controlled by that rule. all\text{=false} means that no data is
inherited from the parent by default and each field to be inherited requires an explicit inheritance rule
set up with \texttt{\DeclareDataInheritance}. The package default is all\text{=true}.

\texttt{\override=true, false}

Whether or not to overwrite target fields with source fields if both are defined. This applies both to
automatic inheritance and to explicit inheritance rules. The package default is override\text{=false},
i.e., existing fields of the child entry are not overwritten.

\texttt{\ignore=(csv list of uniqueness options)}

This option takes a comma-separated list of one of more of 'singletitle', 'uniquetitle', 'uniquebaretitle'
and/or 'uniquework'. The purpose of this option is to ignore tracking information for these three options
when the field which would trigger the tracking (table 6) is inherited. An example—Suppose that you
have several @book entries which all crossref a @mvbook from which they get their author field. You
might reasonably want the \texttt{\ifsingletitle} test to return 'true' for this author as their only 'work'
is the @mvbook. Similar comments would apply to situations involving the \texttt{\ifuniquetitle},
\texttt{\ifuniquebaretitle} and \texttt{\ifuniquework} tests. The ignore option lists which of these
should have their tracking information ignored when the fields which would trigger them are inherited.
The idea is that the presence of an inherited field does not contribute towards the determination of
whether some combination of name/title is unique in the bibliographic data. For example, this modified
default setting would ignore \texttt{singletitle} and \texttt{uniquetitle} tracking:

\begin{verbatim}
\DefaultInheritance{ignore={singletitle, uniquetitle}, all=true, override=false}
\end{verbatim}

Of course, the ignoring of tracking does nothing if the fields inherited do not play a role in tracking.
Only the fields listed in table 6 are relevant to this option.

The optional \texttt{(exceptions)} are an undelimited list of \texttt{\except} directives. Spaces, tabs, and line endings
may be used freely to visually arrange the \texttt{(exceptions)}. Blank lines are not permissible.

\begin{verbatim}
\except{{\textlangle source\textrangle}{{\textrangle target}}{{\textlangle options\textrangle}}}
\end{verbatim}

Defines an exception to the default inheritance rules.

\texttt{\DeclareDataInheritance sets the inheritance \texttt{(options)} for a specific \texttt{(source)} and \texttt{(target)}
combination. The \texttt{(source)} and \texttt{(target)} arguments specify the parent and the child entry type. The
asterisk matches all types and is permissible in either argument.

\begin{verbatim}
\DeclareDataInheritance{{\textlangle options\textrangle}}{{\textlangle source, ...\textrangle}}{{\textlangle target, ...\textrangle}}{{\textlangle rules\textrangle}}
\end{verbatim}

Declares inheritance rules. The \texttt{(source)} and \texttt{(target)} arguments specify the parent and the child
entry type. Either argument may be a single entry type, a comma-separated list of types, or an asterisk.
The asterisk matches all entry types. The \texttt{(rules)} are an undelimited list of \texttt{\inherit} and/or
\texttt{\noinherit} directives. Spaces, tabs, and line endings may be used freely to visually arrange the
\texttt{(rules)}. Blank lines are not permissible. This command may only be used in the preamble. The options
are:

\texttt{\ignore=(csv list of uniqueness options)}

As the ignore option on \texttt{\DefaultInheritance} explained above. When set here, it takes
precedence over any global options set with \texttt{\DefaultInheritance}. For example, this would
ignore \texttt{singletitle} and \texttt{uniquetitle} tracking for a @book inheriting from a @mvbook.

\begin{verbatim}
\DeclareDataInheritance[ignore={singletitle, uniquetitle}]{mvbook}{...}
\end{verbatim}

\begin{verbatim}
\inherit{{\textlangle option\textrangle}}{{\textlangle source\textrangle}}{{\textlangle target\textrangle}}
\end{verbatim}

Defines an inheritance rule by mapping a \texttt{(source)} field to a \texttt{(target)} field. \texttt{(option)} can be one of

\texttt{\override=true, false}

As the override option for \texttt{\DefaultInheritance} explained above. When set here, it takes
precedence over any global options set with \texttt{\DefaultInheritance}.

\texttt{\noinherit{(source)}}

Unconditionally prevents inheritance of the \texttt{(source)} field.
\ResetDataInheritance \Clears all inheritance rules defined with \DeclareDataInheritance. This command may only be used in the preamble.

Here are some practical examples:

\DefaultInheritance{all=true,override=false}

This example shows how to configure the default inheritance behavior. The above settings are the package defaults.

\DefaultInheritance{
  \except*{online}{all=false}
}{all=true,override=false}

This example is similar to the one above but adds one exception: entries of type @online will, by default, not inherit any data from any parent.

\DeclareDataInheritance{collection}{incollection}{
  \inherit{title}{booktitle}
  \inherit{subtitle}{bookssubtitle}
  \inherit{titleaddon}{booktitleaddon}
}

So far we have looked at setting up standard inheritance. For example, all=true means that the publisher field of a source entry is copied to the publisher field of the target entry. In some cases, however, asymmetric mappings are required. They are defined with \DeclareDataInheritance. The above example sets up three typical rules for @incollection entries referencing a @collection. We map the title and related fields of the source to the corresponding booktitle fields of the target.

\DeclareDataInheritance{mvbook,book}{inbook,bookinbook}{
  \inherit{author}{author}
  \inherit{author}{bookauthor}
}

This rule is an example of one-to-many mapping: it maps the author field of the source to both the author and the bookauthor fields of the target in order to allow for compact inbook/bookinbook entries. The source may be either a @mvbook or a @book entry, the target either an @inbook or a @bookinbook entry.

\DeclareDataInheritance{*}{inbook,incollection}{
  \noinherit{introduction}
}

This rule prevents inheritance of the introduction field. It applies to all targets of type @inbook or @incollection, regardless of the source entry type.

\DeclareDataInheritance{*}{*}{
  \noinherit{abstract}
}

This rule, which applies to all entries, regardless of the source and target entry types, prevents inheritance of the abstract field.

\DefaultInheritance{all=true,override=false}
\ResetDataInheritance
This example demonstrates how to emulate traditional BibTeX's cross-referencing mechanism. It enables inheritance by default, disables overwriting, and clears all other inheritance rules and mappings.

In a bibliography entry, you can give an option ‘noinherit’ where the value is a datafield set defined with \DeclareDatafieldSet (§ 4.5.2). This will block inheritance of the fields in the set on a per-entry basis. For example:

```latex
\DeclareDatafieldSet{nobtitle}{
  \member[field=booktitle]
}

@INBOOK{s1,
  OPTIONS = {noinherit=nobtitle},
  TITLE = {Subtitle},
  CROSSREF = {s2}
}

@BOOK{s2,
  TITLE = {Title}
}
```

Here, s1 will not inherit the TITLE of s2 as BOOKTITLE as this is blocked by the datafield set given as the value to the noinherit option. One important thing to note is that children will never inherit any dateparts of a given type if they already contain a datepart of that type. So, for example:

```latex
@INBOOK{b1,
  DATE = {2004-03-03},
  ORIGDATE = {2004-03},
  CROSSREF = {b2}
}

@BOOK{b2,
  DATE = {2004-03-03/2005-08-09},
  ORIGDATE = {2004-03/2005-08},
  EVENTDATE = {2004-03/2005-08},
}
```

Here, b1 will not inherit any of endyear, endmonth, endday, origendyear or origendmonth as this would make a mess of its own dates. It will, given the inheritance defaults, inherit all of the event* date parts.

### 4.6 Auxiliary Commands

The facilities in this section are intended for analyzing and saving bibliographic data rather than formatting and printing it.

#### 4.6.1 Data Commands

The commands in this section grant low-level access to the unformatted bibliographic data. They are not intended for typesetting but rather for things like saving data to a temporary macro so that it may be used in a comparison later.

\texttt{\thefield{(field)}}

Expands to the unformatted (field). If the (field) is undefined, this command expands to an empty string.

\texttt{\strfield{(field)}}

Similar to \thefield, except that the field is automatically sanitized such that its value may safely be used in the formation of a control sequence name.

\texttt{\csfield{(field)}}

Similar to \thefield, but prevents expansion.
\usefield{⟨command⟩}{⟨field⟩}

Executes ⟨command⟩ using the unformatted ⟨field⟩ as its argument.

\thelist{⟨literal list⟩}

Expands to the unformatted ⟨literal list⟩. If the list is undefined, this command expands to an empty string. Note that this command will dump the ⟨literal list⟩ in the internal format used by this package. This format is not suitable for printing.

\strlist{⟨literal list⟩}

Similar to \thelist, except that the list internal representation is automatically sanitized such that its value may safely be used in the formation of a control sequence name.

\thename{⟨name list⟩}

Expands to the unformatted ⟨name list⟩. If the list is undefined, this command expands to an empty string. Note that this command will dump the ⟨name list⟩ in the internal format used by this package. This format is not suitable for printing.

\strname{⟨name list⟩}

Similar to \thename, except that the name internal representation is automatically sanitized such that its value may safely be used in the formation of a control sequence name.

\savefield{⟨field⟩}{⟨macro⟩}
\savefield*{⟨field⟩}{⟨macro⟩}

Copies an unformatted ⟨field⟩ to a ⟨macro⟩. The regular variant of this command defines the ⟨macro⟩ globally, the starred one works locally.

\savelist{⟨literal list⟩}{⟨macro⟩}
\savelist*{⟨literal list⟩}{⟨macro⟩}

Copies an unformatted ⟨literal list⟩ to a ⟨macro⟩. The regular variant of this command defines the ⟨macro⟩ globally, the starred one works locally.

\savename{⟨name list⟩}{⟨macro⟩}
\savename*{⟨name list⟩}{⟨macro⟩}

Copies an unformatted ⟨name list⟩ to a ⟨macro⟩. The regular variant of this command defines the ⟨macro⟩ globally, the starred one works locally.

\savefieldcs{⟨field⟩}{⟨csname⟩}
\savefieldcs*{⟨field⟩}{⟨csname⟩}

Similar to \savefield, but takes the control sequence name ⟨csname⟩ (without a leading backslash) as an argument, rather than a macro name.

\savelistcs{⟨literal list⟩}{⟨csname⟩}
\savelistcs*{⟨literal list⟩}{⟨csname⟩}

Similar to \savelist, but takes the control sequence name ⟨csname⟩ (without a leading backslash) as an argument, rather than a macro name.

\savenamecs{⟨name list⟩}{⟨csname⟩}
\savenamecs*{⟨name list⟩}{⟨csname⟩}

Similar to \savename, but takes the control sequence name ⟨csname⟩ (without a leading backslash) as an argument, rather than a macro name.

\restorefield{⟨field⟩}{⟨macro⟩}

Restores a ⟨field⟩ from a ⟨macro⟩ defined with \savefield before. The field is restored within a local scope.

\restorelist{⟨literal list⟩}{⟨macro⟩}

Restores a ⟨literal list⟩ from a ⟨macro⟩ defined with \savelist before. The list is restored within a local scope.

\renovname{⟨name list⟩}{⟨macro⟩}

Restores a ⟨name list⟩ from a ⟨macro⟩ defined with \savename before. The list is restored within a local scope.

\clearfield{⟨field⟩}

Clears the ⟨field⟩ within a local scope. A field cleared this way is treated as undefined by subsequent data commands.
\clearlist{(literal list)}

Clears the (literal list) within a local scope. A list cleared this way is treated as undefined by subsequent data commands.

\clearname{(name list)}

Clears the (name list) within a local scope. A list cleared this way is treated as undefined by subsequent data commands.

### 4.6.2 Stand-alone Tests

The commands in this section are various kinds of stand-alone tests for use in bibliography and citation styles.

\if<datatype>julian{(true)}{(false)}

Expands to (true) if the date `datatype`date (date, urldate, eventdate etc.) Was converted to the Julian Calendar due to the settings of the julian and gregorianstart options.

\ifdatejulian{(true)}{(false)}

As \if<datatype>julian but for use in \mkbibdate* formatting commands (§ 4.10.2) inside which the appropriate \if<datatype>julian command is aliased to this command.

\if<datatype>dateera{(era)}{(true)}{(false)}

Expands to (true) if the date `datatype`date (date, urldate, eventdate etc.) has an era specification equal to (era) and (false) otherwise. The supported (era) strings which biber determines and passes in the .bbl are:

- BCE/BCE era
- CE/AD era

This command is useful for determining whether to print the location strings in § 4.9.2.21.

\ifdateera{(era)}{(true)}{(false)}

As \if<datatype>dateera but for use in \mkbibdate* formatting commands (§ 4.10.2) inside which the appropriate \if<datatype>dateera command is aliased to this command.

\if<datatype>datecirca{(true)}{(false)}

Expands to (true) if the date `datatype`date (date, urldate, eventdate etc.) had a ‘circa’ marker in the source and (false) otherwise. See § 2.3.8. This command is useful for determining whether to print the location strings in § 4.9.2.21.

\ifdatecirca{(true)}{(false)}

As \if<datatype>datecirca but for use in \mkbibdate* formatting commands (§ 4.10.2) inside which the appropriate \if<datatype>datecirca command is aliased to this command.

\if<datatype>dateuncertain{(true)}{(false)}

Expands to (true) if the date `datatype`date (date, urldate, eventdate etc.) had an uncertainty marker in the source and (false) otherwise. See § 2.3.8. This command is useful for determining whether to print, for example, a question mark after a year.

\ifdateuncertain{(true)}{(false)}

As \if<datatype>dateuncertain but for use in \mkbibdate* formatting commands (§ 4.10.2) inside which the appropriate \if<datatype>dateuncertain command is aliased to this command.

\ifndefdateuncertain{(true)}{(false)}

As \ifndef<datatype>dateuncertain but for use in \mkbibdate* formatting commands (§ 4.10.2) inside which the appropriate \ifndef<datatype>dateuncertain command is aliased to this command.

\ifcaselang{(language)}{(true)}{(false)}

Expands to (true) if the the optional (language) is one of those declared by \DeclareCaseLangs (see § 4.6.4) and to (false) otherwise. Without the optional argument, checks the current value of \currentlang.

\ifsortnamekeyscheme{(string)}{(true)}{(false)}

Expands to (true) if the (string) is equal to the current in scope sorting name key scheme name (see 4.5.6), and to (false) otherwise.
\iffieldequal{\textit{field}}{\texttt{true}}{\texttt{false}}

Expands to \texttt{true} if the \textit{field} is undefined, and to \texttt{false} otherwise.

\iffieldequals{\textit{field}}{\texttt{true}}{\texttt{false}}

Expands to \texttt{true} if the \textit{field} is undefined, and to \texttt{false} otherwise.

\iffieldequalsalcs{\textit{field}}{\texttt{true}}{\texttt{false}}

Similar to \iffieldequals but takes the control sequence name \texttt{csname} (without a leading backslash) as an argument, rather than a macro name.

\iffieldequalsalcs{\textit{field}}{\texttt{true}}{\texttt{false}}

Similar to \iffieldequalsalcs but takes the control sequence name \texttt{csname} (without a leading backslash) as an argument, rather than a macro name.

\iffieldequalstr{\textit{field}}{\texttt{true}}{\texttt{false}}

Executes \texttt{true} if the value of the \textit{field} is equal to \texttt{string}, and \texttt{false} otherwise. This command is robust.

\iffielderef{\textit{field}}{\texttt{true}}{\texttt{false}}

If the crossref/xref field of an entry is defined, this command checks if the \textit{field} is related to the cross-referenced parent entry. It executes \texttt{true} if the \textit{field} of the child entry is equal to the corresponding \textit{field} of the parent entry, and \texttt{false} otherwise. If the crossref/xref field is undefined, it always executes \texttt{false}. This command is robust. See the description of the crossref and xref fields in §2.2.3 as well as §2.4.1 for further information concerning cross-referencing.

\iflistref{\textit{literal list}}{\texttt{true}}{\texttt{false}}

Similar to \iffielderef but checks if a \textit{literal list} is related to the cross-referenced parent entry. See the description of the crossref and xref fields in §2.2.3 as well as §2.4.1 for further information concerning cross-referencing.

\ifnameref{\textit{name list}}{\texttt{true}}{\texttt{false}}

Similar to \iffielderef but checks if a \textit{name list} is related to the cross-referenced parent entry. See the description of the crossref and xref fields in §2.2.3 as well as §2.4.1 for further information concerning cross-referencing.

\ifcurrentfield{\textit{field}}{\texttt{true}}{\texttt{false}}

Executes \texttt{true} if the current field is \textit{field}, and \texttt{false} otherwise. This command is robust. It is intended for use in field formatting directives and always executes \texttt{false} when used in any other context.

\textbf{context.}

\textit{intended for use in field formatting directives and always executes (false) otherwise.}

\textit{Similar to crossref, it always executes (false) otherwise.}

\textit{Similar to xref, undefined, it always executes (false) otherwise.}

\textit{Similar to backslash) as an argument, rather than a macro name.}

\textit{Expands to ⟨true⟩ otherwise. This command is robust.}

\textit{Expands to ⟨false⟩ otherwise. This command is robust. See the description of the crossref and xref fields in §2.2.3 as well as §2.4.1 for further information concerning cross-referencing.}

\textit{Expands to ⟨false⟩ otherwise. This command is robust. See the description of the crossref and xref fields in §2.2.3 as well as §2.4.1 for further information concerning cross-referencing.}

\textit{Expands to ⟨false⟩ otherwise. This command is robust. It is intended for use in field formatting directives and always executes (false) when used in any other context.}
\currentlist{(literal list)}{(true)}{(false)}

Executes \texttt{(true)} if the current list is \texttt{(literal list)}, and \texttt{(false)} otherwise. This command is robust. It is intended for use in list formatting directives and always executes \texttt{(false)} when used in any other context.

\currentname{(name list)}{(true)}{(false)}

Executes \texttt{(true)} if the current list is \texttt{(name list)}, and \texttt{(false)} otherwise. This command is robust. It is intended for use in list formatting directives and always executes \texttt{(false)} when used in any other context.

\useprefix{(true)}{(false)}

Expands to \texttt{(true)} if the \texttt{useprefix} option is enabled (either globally or for the current entry), and \texttt{(false)} otherwise. See §3.1.3 for details on this option.

\useauthor{(true)}{(false)}

This is just a particular case of the \texttt{use<name>} macro below but is mentioned here as \texttt{author} is part of the default data model. Expands to \texttt{(true)} if the \texttt{useauthor} option is enabled (either globally or for the current entry), and \texttt{(false)} otherwise. See §3.1.3 for details on this option.

\useeditor{(true)}{(false)}

This is just a particular case of the \texttt{use<name>} macro below but is mentioned here as \texttt{editor} is part of the default data model. Expands to \texttt{(true)} if the \texttt{useeditor} option is enabled (either globally or for the current entry), and \texttt{(false)} otherwise. See §3.1.3 for details on this option.

\usetranslator{(true)}{(false)}

This is just a particular case of the \texttt{use<name>} macro below but is mentioned here as \texttt{translator} is part of the default data model. Expands to \texttt{(true)} if the \texttt{usetranslator} option is enabled (either globally or for the current entry), and \texttt{(false)} otherwise. See §3.1.3 for details on this option.

\use{name} {(true)}{(false)}

Expands to \texttt{(true)} if the \texttt{use<name>} option is enabled (either globally or for the current entry), and \texttt{(false)} otherwise. See §3.1.3 for details on this option.

\crossrefs{(true)}{(false)}

Expands to \texttt{(true)} if the entry was included in the .\texttt{bbl} due to being referenced more than \texttt{mincrossrefs} times and \texttt{false} otherwise. See §3.1.2.1. Also expands to \texttt{false} if the entry was directly cited.

\xrefs{(true)}{(false)}

Expands to \texttt{(true)} if the entry was included in the .\texttt{bbl} due to being referenced more than \texttt{minxrefs} times and \texttt{false} otherwise. See §3.1.2.1. Also expands to \texttt{false} if the entry was directly cited.

\singletitle{(true)}{(false)}

Expands to \texttt{(true)} if there is only one work by the \texttt{labelname} name in the bibliography, and to \texttt{(false)} otherwise. If \texttt{labelname} is not set for an entry, this will always expand to \texttt{(false)}. Note that this feature needs to be enabled explicitly with the package option \texttt{singletitle}.

\uniquetitle{(true)}{(false)}

Expands to \texttt{(true)} if there is only one work with the title \texttt{labeltitle} and to \texttt{(false)} otherwise. If \texttt{labeltitle} is not set for an entry, this will always expand to \texttt{(false)}. Note that this feature needs to be enabled explicitly with the package option \texttt{uniquetitle}.

\uniquetitlebare{(true)}{(false)}

Expands to \texttt{(true)} if \texttt{labelname} is empty and there is only one work with the title \texttt{labeltitle} and to \texttt{(false)} otherwise. If \texttt{labeltitle} is not set for an entry, this will always expand to \texttt{(false)}. Note that this feature needs to be enabled explicitly with the package option \texttt{uniquetitlebare}.

\uniquework{(true)}{(false)}

Expands to \texttt{(true)} if there is only one work by the \texttt{labelname} name with the \texttt{labeltitle} title in the bibliography, and to \texttt{(false)} otherwise. If neither \texttt{labelname} nor \texttt{labeltitle} are set for an entry, this will always expand to \texttt{(false)}. Note that this feature needs to be enabled explicitly with the package option \texttt{uniquework}. If both \texttt{singletitle} and \texttt{uniquetitle} are \texttt{false} for the same entry, this could be because another entry has the same \texttt{labdname} and yet another, different, entry has the same \texttt{labeltitle}. \texttt{uniquework} would let you know that there is another entry that has \texttt{both} the same \texttt{labelname} and the same \texttt{labeltitle}. This could be helpful in cases where multiple people maintain bibliography datasources and there is a risk of adding the same work with different keys without other parties realising this. This test could help to find such duplicates.
\ifuniqueprimaryauthor{(true)}{(false)}\x Expands to (true) if there is only one work by the primary (first) author with the family name of \labelname\ and to (false) otherwise. If \labelname\ is not set for an entry, this will always expand to (false). Note that this feature needs to be enabled explicitly with the package option \texttt{uniqueprimaryauthor}.

\ifandothers{\texttt{(list)}}{(true)}{(false)}
\x Expands to (true) if the \texttt{(list)} is defined and has been truncated in the \texttt{bib} file with the keyword 'and others'\ and to (false) otherwise. The \texttt{(list)} may be a literal list or a name list.

\ifmorenames{(true)}{(false)}
\x Expands to (true) if the current name list has been or will be truncated, and to (false) otherwise. This command is intended for use in formatting directives for name lists. It will always expand to (false) when used elsewhere. This command performs the equivalent of an \ifandothers\ test for the current list. If this test is negative, it also checks if the \texttt{listtotal} counter is larger than \texttt{liststop}. This command may be used in a formatting directive to decide if a note such as ‘and others’ or ‘et al.’ is to be printed at the end of the list. Note that you still need to check whether you are in the middle or at the end of the list, i.e., whether \texttt{listcount} is smaller than or equal to \texttt{liststop}, see §4.4.1 for details.

\ifmoreitems{(true)}{(false)}
\x This command is similar to \ifmorenames but checks the current literal list. It is intended for use in formatting directives for literal lists. It will always expand to (false) when used elsewhere.

\if<namepart>\texttt{init}{(true)}{(false)}
\x Expands to (true) or (false), depending on the state of the \texttt{namepart}\init\ package option (see §3.1.2.3). This command is intended for use in formatting directives for name lists.

\if\terseinits\{(true}\{(false)}
\x Expands to (true) or (false), depending on the state of the \terseinits\ package option (see §3.1.23). This command is intended for use in formatting directives for name lists.

\ifentrytype\{(type\}\{(true\)}{(false)}
\x Executes (true) if the entry type of the entry currently being processed is \texttt{(type)}, and (false) otherwise.

\ifkeyword\{(keyword\}\{(true\)}{(false)}
\x Executes (true) if the \texttt{(keyword)} is found in the \texttt{keywords} field of the entry currently being processed, and (false) otherwise.

\ifentrykeyword\{(entrykey\}\{(keyword\}\{(true\)}{(false)}
\x A variant of \ifkeyword which takes an entry key as its first argument. This is useful for testing an entry other than the one currently processed.

\ifcategory\{(category\}\{(true\)}{(false)}
\x Executes (true) if the entry currently being processed has been assigned to a \texttt{(category)} with \texttt{addtocategory}, and (false) otherwise.

\ifentrycategory\{(entrykey\}\{(category\}\{(true\)}{(false)}
\x A variant of \ifcategory which takes an entry key as its first argument. This is useful for testing an entry other than the one currently processed.

\ifciteseen\{(true\)}{(false)}
\x Executes (true) if the entry currently being processed has been cited before, and (false) otherwise. This command is robust and intended for use in citation styles. If there are any \texttt{refsection} environments in the document, the citation tracking is local to these environments. Note that the citation tracker needs to be enabled explicitly with the package option \texttt{citetracker}. The behavior of this test depends on the mode the citation tracker is operating in, see §3.1.23 for details. If the citation tracker is disabled, the test always yields (false). Also see the \citetracker\true\ and \citetracker\false\ switches in §4.6.4.

\ifentryseen\{(entrykey\}\{(true\)}{(false)}
\x A variant of \ifciteseen which takes an entry key as its first argument. Since the \texttt{(entrykey)} is expanded prior to performing the test, it is possible to test for entry keys in a field such as \texttt{xref}:

\ifentryseen\\texttt{\texttt{\texttt{xref}}}\{(true\}\{(false)}

Apart from the additional argument, \ifentryseen behaves like \ifciteseen.
\ifentryinbib{(entrykey)}{(true)}{(false)}

Executes \texttt{(true)} if the entry \texttt{(entrykey)} appears in the current bibliography, and \texttt{(false)} otherwise. This command is intended for use in bibliography styles.

\iffirstcitekey{(true)}{(false)}

Executes \texttt{(true)} if the entry currently being processed is the first one in the citation list, and \texttt{(false)} otherwise. This command relies on the \texttt{citecount}, \texttt{citetotal}, \texttt{multicitecount} and \texttt{multicitetotal} counters (§4.10.5) and thus is intended for use only in the \texttt{(loccode)} of a citation command defined with \texttt{\DeclareCiteCommand}.

\iflastcitekey{(true)}{(false)}

Similar \texttt{\iffirstcitekey}, but executes \texttt{(true)} if the entry currently being processed is the last one in the citation list, and \texttt{(false)} otherwise.

\ifciteibid{(true)}{(false)}

Expands to \texttt{(true)} if the entry currently being processed is the same as the last one, and to \texttt{(false)} otherwise. This command is intended for use in citation styles. If there are any \texttt{refsection} environments in the document, the tracking is local to these environments. Note that the \texttt{'ibidem'} tracker needs to be enabled explicitly with the package option \texttt{ibidtracker}. The behavior of this test depends on the mode the tracker is operating in, see §3.1.2.3 for details. If the tracker is disabled, the test always yields \texttt{(false)}. Also see the \texttt{\citetrackertrue} and \texttt{\citetrackerfalse} switches in §4.6.4.

\ifciteidem{(true)}{(false)}

Expands to \texttt{(true)} if the primary name (i.e., the author or editor) in the entry currently being processed is the same as the last one, and to \texttt{(false)} otherwise. This command is intended for use in citation styles. If there are any \texttt{refsection} environments in the document, the tracking is local to these environments. Note that the \texttt{'idem'} tracker needs to be enabled explicitly with the package option \texttt{idemtracker}. The behavior of this test depends on the mode the tracker is operating in, see §3.1.2.3 for details. If the tracker is disabled, the test always yields \texttt{(false)}. Also see \texttt{\citetrackertrue} and \texttt{\citetrackerfalse} switches in §4.6.4.

\ifopcit{(true)}{(false)}

This command is similar to \texttt{\ifciteidem} except that it expands to \texttt{(true)} if the entry currently being processed is the same as the last one \emph{by this author or editor}. Note that the \texttt{'opcit'} tracker needs to be enabled explicitly with the package option \texttt{opcittracker}. The behavior of this test depends on the mode the tracker is operating in, see §3.1.2.3 for details. If the tracker is disabled, the test always yields \texttt{(false)}. Also see \texttt{\citetrackertrue} and \texttt{\citetrackerfalse} switches in §4.6.4.

\ifoocit{(true)}{(false)}

This command is similar to \texttt{\ifopcit} except that it also compares the \texttt{(postnote)} arguments and expands to \texttt{(true)} only if they match and are numerical (in the sense of \texttt{\ifnumerals} from §4.6.2), i.e., \texttt{\ifoocit} will yield \texttt{true} if the citation refers to the same page cited before. Note that the \texttt{'loccit'} tracker needs to be enabled explicitly with the package option \texttt{loccittracker}. The behavior of this test depends on the mode the tracker is operating in, see §3.1.2.3 for details. If the tracker is disabled, the test always yields \texttt{(false)}. Also see \texttt{\citetrackertrue} and \texttt{\citetrackerfalse} switches in §4.6.4.

\iffirstonpage{(true)}{(false)}

The behavior of this command is responsive to the package option \texttt{pagetracker}. If the option is set to \texttt{page}, it expands to \texttt{(true)} if the current item is the first one on the page, and to \texttt{(false)} otherwise. If the option is set to \texttt{spread}, it expands to \texttt{(true)} if the current item is the first one on the double-page spread, and to \texttt{(false)} otherwise. If the page tracker is disabled, this test always yields \texttt{(false)}. Depending on the context, the \texttt{'item'} may be a citation or an entry in the bibliography or a bibliography list. Note that this test distinguishes between body text and footnotes. For example, if used in the first footnote on a page, it will expand to \texttt{(true)} even if there is a citation in the body text prior to the footnote. Also see the \texttt{\pagetrackertrue} and \texttt{\pagetrackerfalse} switches in §4.6.4.

\ifsamepage{(instance 1)}{(instance 2)}{(true)}{(false)}

This command expands to \texttt{(true)} if two instances of a reference are located on the same page or double-page spread, and to \texttt{(false)} otherwise. An instance of a reference may be a citation or an entry in the bibliography or a bibliography list. These instances are identified by the value of the \texttt{instcount} counter, see §4.10.5. The behavior of this command is responsive to the package option \texttt{pagetracker}. If this option is set to \texttt{spread}, \texttt{\ifsamepage} is in fact an ‘if same spread’ test. If
the page tracker is disabled, this test always yields (false). The arguments (instance 1) and (instance 2) are treated as integer expressions in the sense of e-TeX’s \numexpr. This implies that it is possible to make calculations within these arguments, for example:

\ifsamepage{\value{instcount}}{\value{instcount} = 1}{\true}{\false}

Note that \value is not prefixed by \the and that the subtraction is included in the second argument in the above example. If (instance 1) or (instance 2) is an invalid number (for example, a negative one), the test yields (false). Also note that this test does not distinguish between body text and footnotes. Also see the \pagetrackertrue and \pagetrackerfalse switches in § 4.6.4.

\ifinteger{\string}{{\true}}{\false}

Executes (true) if the (\string) is a positive integer, and (false) otherwise. This command is robust.

\ifnumeral{\string}{{\true}}{\false}

Executes (true) if the (\string) is an Arabic or Roman numeral, and (false) otherwise. This command is robust. See also \DeclareNumChars and \NumCheckSetup in § 4.6.4.

\ifnumerals{\string}{{\true}}{\false}

Executes (true) if the (\string) is a range or a list of Arabic or Roman numerals, and (false) otherwise. This command is robust. In contrast to \ifnumeral, it will also execute (true) with arguments like ”52–58”, ”14/15”, ”1, 3, 5”, and so on. See also \DeclareNumChars, \DeclareRangeChars, \DeclareRangeCommands, and \NumCheckSetup in § 4.6.4.

\ifpages{\string}{{\true}}{\false}

Similar to \ifnumerals, but also considers \DeclarePageCommands from § 4.6.4.

\iffielddint{\field}{{\true}}{\false}

Similar to \ifinteger, but uses the value of a (\field) rather than a literal string in the test. If the (\field) is undefined, it executes (false).

\iffielddnum{\field}{{\true}}{\false}

Similar to \ifnumeral, but uses the value of a (\field) rather than a literal string in the test. If the (\field) is undefined, it executes (false).

\iffielddnums{\field}{{\true}}{\false}

Similar to \ifnumerals, but uses the value of a (\field) rather than a literal string in the test. If the (\field) is undefined, it executes (false).

\iffielddpages{\field}{{\true}}{\false}

Similar to \ifpages, but uses the value of a (\field) rather than a literal string in the test. If the (\field) is undefined, it executes (false).

\ifbibstring{\string}{{\true}}{\false}

Expands to (true) if the (\string) is a known localisation key, and to (false) otherwise. The localisation keys defined by default are listed in § 4.9.2. New ones may be defined with \NewBibliographyString.

\ifbibxstring{\string}{{\true}}{\false}

Similar to \ifbibstring, but the (\string) is expanded.

\iffieldbibstring{\field}{{\true}}{\false}

Similar to \ifbibstring, but uses the value of a (\field) rather than a literal string in the test. If the (\field) is undefined, it expands to (false).

\ifdriver{\entrytype}{{\true}}{\false}

Expands to (true) if a driver for the (\entrytype) is available, and to (false) otherwise.

\ifcapital{\true}{\false}

Executes (true) if \biblatex’s punctuation tracker would capitalize a localisation string at the current location, and (false) otherwise. This command is robust. It may be useful for conditional capitalization of certain parts of a name in a formatting directive.

\ifcitation{\true}{\false}

Expands to (true) when located in a citation, and to (false) otherwise. Note that this command is responsive to the outermost context in which it is used. For example, if a citation command defined with \DeclareCiteCommand executes a driver defined with \DeclareBibliographyDriver, any \ifcitation tests in the driver code will yield (true). See § 4.11.6 for a practical example.

201
\ifbibliography{(true)}{(false)}

Expands to (true) when located in a bibliography, and to (false) otherwise. Note that this command is responsive to the outermost context in which it is used. For example, if a driver defined with \DeclareBibliographyDriver executes a citation command defined with \DeclareCiteCommand, any \bibliography tests in the citation code will yield (true). See §4.11.6 for a practical example.

\ifnatbibmode{(true)}{(false)}

Expands to (true) or (false) depending on the natbib option from §3.1.1.

\ifciteindex{(true)}{(false)}

Expands to (true) or (false) depending on the indexing option from §3.1.2.1.

\ifbibindex{(true)}{(false)}

Expands to (true) or (false) depending on the indexing option from §3.1.2.1.

\iffootnote{(true)}{(false)}

Expands to (true) when located in a footnote, and to (false) otherwise. Note that footnotes in minipage environments are considered to be part of the body text. This command will only expand to (true) in footnotes a the bottom of the page and in endnotes as provided by the endnotes package.

citecounter

This counter indicates how many times the entry currently being processed is cited in the current reference section. Note that this feature needs to be enabled explicitly with the package option citecounter. If the option is set to context, citations in the body text and in footnotes are counted separately. In this case, citecounter will hold the value of the context it is used in.

uniquename

This counter refers to the labelname list. It is set on a per-name basis. Its value is 0 if the base name (by default the 'family' part of the name) is unique, 1 if adding the other parts of the name (as specified in the uniquename template defined by \DeclareUniquenameTemplate) as initials will make it unique, and 2 if the full name is required to disambiguate the name. This information is required by author-year and author-title citation schemes which add additional parts of the name when citing different authors with the same last name. For example, (given the default \DeclareUniquenameTemplate definition) if there is one 'John Doe' and one 'Edward Doe' in the list of references, this counter will be set to 1. If there is one 'John Doe' and one 'Jane Doe', the value of the counter will be 2. If the option is set to allinit/mininit, the counter will be limited to 1. This is useful for citations styles which use initials to disambiguate names but never print the full name in citations. If adding the initials is not sufficient to disambiguate the name, uniquename will also be set to 0 for that name. This feature needs to be enabled explicitly with the package option uniquename. Note that the uniquename counter is local to \printnames and that it is only set for the labelname list and to the name list labelname has been derived from (typically author or editor). Its value is zero in any other context, i.e., it must be evaluated in the name formatting directives handling name lists. See §4.11.4 for further details and practical examples.

uniquelist

This counter refers to the labelname list. It is set on a per-field basis. Its value indicates the number of names required to disambiguate the name list if automatic maxnames/minnames truncation would lead to ambiguous citations. For example, if there is one work by 'Doe/Smith/Johnson' and another one by 'Doe/Edwards/Williams', setting maxnames=1 would lead to 'Doe et al.' in both cases. In this case, uniquelist would be set to 2 on the labelname lists of both entries because at least the first two names are required to disambiguate them. Note that the uniquelist counter is local to \printnames and that it is only set for the labelname list and to the name list labelname has been derived from (typically author or editor). Its value is zero in any other context. If available, the uniquelist value will be used automatically by \printnames when processing the name list, i.e., it will automatically override maxnames/minnames. This feature needs to be enabled explicitly with the package option uniquelist. See §4.11.4 for further details and practical examples.

parenlevel

The current nesting level of parentheses and/or brackets. This information is only available if the parentracker from §3.1.2.3 is enabled.

4.6.3 Tests with \ifboolexpr and \ifthenelse

The tests introduced in §4.6.2 may also be used with the \ifboolexpr command provided by the etoolbox package and the \ifthenelse command provided by the ifthen package. The syntax of the tests is slightly different in this case: the (true) and (false) arguments are omitted from the test itself and passed to the \ifboolexpr or \ifthenelse command instead. Note that the use of these commands implies some processing overhead. If you do not need any boolean operators, it is more efficient to use the stand-alone tests from §4.6.2.
\ifboolexpr{⟨expression⟩}{⟨true⟩}{⟨false⟩}
	etoolbox command which allows for complex tests with boolean operators and grouping:

\begin{verbatim}
\ifboolexpr{
    test \ifnameundef{editor}
    and
    not test \iflistundef{location}
}
\or test \iffieldundef{year}
\}{...}
\{...
\end{verbatim}

\ifthenelse{⟨tests⟩}{⟨true⟩}{⟨false⟩}

ifthen command which allows for complex tests with boolean operators and grouping:

\begin{verbatim}
\ifthenelse{
    \ifnameundef{editor}
    \and
    \not \iflistundef{location}
\} \or \iffieldundef{year}
\}{...}
\{...
\end{verbatim}

The additional tests provided by biblatex are only available when \ifboolexpr or \ifthenelse are used in citation commands and in the bibliography.

### 4.6.4 Miscellaneous Commands

The section introduced miscellaneous commands and little helpers for use in bibliography and citation styles.

\newbibmacro{⟨name⟩}{⟨arguments⟩}{⟨optional⟩}{⟨definition⟩}
\newbibmacro*{⟨name⟩}{⟨arguments⟩}{⟨optional⟩}{⟨definition⟩}

Defines a macro to be executed via \usebibmacro later. The syntax of this command is very similar to \newcommand except that \texttt{⟨name⟩} may contain characters such as numbers and punctuation marks and does not start with a backslash. The optional argument \texttt{⟨arguments⟩} is an integer specifying the number of arguments taken by the macro. If \texttt{⟨optional⟩} is given, it specifies a default value for the first argument of the macro, which automatically becomes an optional argument. In contrast to \texttt{\newcommand}, \texttt{\newbibmacro} issues a warning message if the macro is already defined, and automatically falls back to \texttt{\renewbibmacro}. As with \texttt{\newcommand}, the regular variant of this command uses the \texttt{\long} prefix in the definition while the starred one does not. If a macro has been declared to be long, it may take arguments containing \texttt{\par} tokens. \texttt{\newbibmacro} and \texttt{\renewbibmacro} are provided for convenience. Style authors are free to use \texttt{\newcommand} or \texttt{\def} instead. However, note that most shared definitions found in biblatex.def are defined with \texttt{\newbibmacro}, hence they must be used and modified accordingly.

\renewbibmacro{⟨name⟩}{⟨arguments⟩}{⟨optional⟩}{⟨definition⟩}
\renewbibmacro*{⟨name⟩}{⟨arguments⟩}{⟨optional⟩}{⟨definition⟩}

Similar to \texttt{\newbibmacro} but redefines \texttt{⟨name⟩}. In contrast to \texttt{\renewcommand}, \texttt{\renewbibmacro} issues a warning message if the macro is undefined, and automatically falls back to \texttt{\newbibmacro}.

\providebibmacro{⟨name⟩}{⟨arguments⟩}{⟨optional⟩}{⟨definition⟩}
\providebibmacro*{⟨name⟩}{⟨arguments⟩}{⟨optional⟩}{⟨definition⟩}

Similar to \texttt{\newbibmacro} but only defines \texttt{⟨name⟩} if it is undefined. This command is similar in concept to \texttt{\providecommand}.

\usebibmacro{⟨name⟩}
\usebibmacro*{⟨name⟩}

This command executes the macro \texttt{⟨name⟩}, as defined with \texttt{\newbibmacro}. If the macro takes any arguments, they are simply appended after \texttt{⟨name⟩}. The regular variant of this command sanitizes \texttt{⟨name⟩} while the starred variant does not.
\savecommand{⟨command⟩}
\restorecommand{⟨command⟩}

These commands save and restore any ⟨command⟩, which must be a command name starting with a backslash. Both commands work within a local scope. They are mainly provided for use in localisation files.

\savebibmacro{⟨name⟩}
\restorebibmacro{⟨name⟩}

These commands save and restore the macro ⟨name⟩, where ⟨name⟩ is the identifier of a macro defined with \newbibmacro. Both commands work within a local scope. They are mainly provided for use in localisation files.

\savefieldformat{⟨entry type⟩}{⟨format⟩}
\restorefieldformat{⟨entry type⟩}{⟨format⟩}

These commands save and restore the formatting directive ⟨format⟩, as defined with \DeclareFieldFormat. Both commands work within a local scope. They are mainly provided for use in localisation files.

\savelistformat{⟨entry type⟩}{⟨format⟩}
\restorelistformat{⟨entry type⟩}{⟨format⟩}

These commands save and restore the formatting directive ⟨format⟩, as defined with \DeclareListFormat. Both commands work within a local scope. They are mainly provided for use in localisation files.

\savenameformat{⟨entry type⟩}{⟨format⟩}
\restorenameformat{⟨entry type⟩}{⟨format⟩}

These commands save and restore the formatting directive ⟨format⟩, as defined with \DeclareNameFormat. Both commands work within a local scope. They are mainly provided for use in localisation files.

\ifbibmacroundef{⟨name⟩}{⟨true⟩}{⟨false⟩}

Expands to ⟨true⟩ if the bibliography macro ⟨name⟩ is undefined, and to ⟨false⟩ otherwise.

\iffieldformatundef{⟨entry type⟩}{⟨name⟩}{⟨true⟩}{⟨false⟩}
\iflistformatundef{⟨entry type⟩}{⟨name⟩}{⟨true⟩}{⟨false⟩}
\ifnameformatundef{⟨entry type⟩}{⟨name⟩}{⟨true⟩}{⟨false⟩}

Expands to ⟨true⟩ if the formatting directive ⟨format⟩ is undefined, and to ⟨false⟩ otherwise.

\usedriver{⟨code⟩}{⟨entrytype⟩}

Executes the bibliography driver for an ⟨entrytype⟩. Calling this command in the ⟨loopcode⟩ of a citation command defined with \DeclareCiteCommand is a simple way to print full citations similar to a bibliography entry. Commands such as \newblock, which are not applicable in a citation, are disabled automatically. Additional initialization commands may be passed as the ⟨code⟩ argument. This argument is executed inside the group in which \usedriver runs the respective driver. Note that it is mandatory in terms of the syntax but may be left empty. Also note that this command will automatically switch languages if the autolang package option is enabled.

\bibhypertarget{⟨name⟩}{⟨text⟩}

A wrapper for hyperref’s \hypertarget command. The ⟨name⟩ is the name of the anchor, the ⟨text⟩ is arbitrary printable text or code which serves as an anchor. If there are any refsection environments in the document, the ⟨name⟩ is local to the current environment. If the hyperref package option is disabled or the hyperref package has not been loaded, this command will simply pass on its ⟨text⟩ argument. See also the formatting directive \bibhypertarget in § 4.10.4.

\bibhyperlink{⟨name⟩}{⟨text⟩}

A wrapper for hyperref’s \hyperlink command. The ⟨name⟩ is the name of an anchor defined with \bibhyperlink, the ⟨text⟩ is arbitrary printable text or code to be transformed into a link. If there are any refsection environments in the document, the ⟨name⟩ is local to the current environment. If the hyperref package option is disabled or the hyperref package has not been loaded, this command will simply pass on its ⟨text⟩ argument. See also the formatting directive \bibhyperlink in § 4.10.4.
\bibhyperref{⟨entrykey⟩}{⟨text⟩}
Transforms ⟨text⟩ into an internal link pointing to ⟨entrykey⟩ in the bibliography. If ⟨entrykey⟩ is omitted, this command uses the key of the entry currently being processed. This command is employed to transform citations into clickable links pointing to the corresponding entry in the bibliography. The link target is marked automatically by biblatex. If there are multiple bibliographies in a document, the target will be the first occurrence of ⟨entrykey⟩ in one of the bibliographies. If there are refsection environments, the links are local to the environment. See also the formatting directive bibhyperref in § 4.10.4.

\ifhyperref{⟨true⟩}{⟨false⟩}
Expands to ⟨true⟩ if the hyperref package option is enabled (which implies that the hyperref package has been loaded), and to ⟨false⟩ otherwise.

\docsvfield{⟨field⟩}
Similar to the \docsvlist command from the etoolbox package, except that it takes a field name as its argument. The value of this field is parsed as a comma-separated list. If the ⟨field⟩ is undefined, this command expands to an empty string.

\forcsvfield{⟨handler⟩}{⟨field⟩}
Similar to the \forcsvlist command from the etoolbox package, except that it takes a field name as its argument. The value of this field is parsed as a comma-separated list. If the ⟨field⟩ is undefined, this command expands to an empty string.

\MakeCapital{⟨text⟩}
Similar to \MakeUppercase but only converts the first printable character in ⟨text⟩ to uppercase. Note that the restrictions that apply to \MakeUppercase also apply to this command. Namely, all commands in ⟨text⟩ must either be robust or prefixed with \protect since the ⟨text⟩ is expanded during capitalization. Apart from Ascii characters and the standard accent commands, this command also handles the active characters of the inputenc package as well as the shorthands of the babel package. If the ⟨text⟩ starts with a control sequence, nothing is capitalized. This command is robust.

\MakeSentenceCase{⟨text⟩}
\MakeSentenceCase*[⟨text⟩]
Converts its ⟨text⟩ argument to sentence case, i.e., the first word is capitalized and the remainder of the string is converted to lowercase. This command is robust. The starred variant differs from the regular version in that it considers the language of the entry, as specified in the langid field. If the langid field is defined and holds a language declared with \DeclareCaseLangs (see below)\textsuperscript{35}, then the sentence case conversion is performed. If the langid field is undefined, then the language list declared with \DeclareCaseLangs is checked for the presence of the main document language derived from the language option. If found, sentence case conversion is performed, if not, the ⟨text⟩ is not altered in any way. It is recommended to use \MakeSentenceCase* rather than the regular variant in formatting directives. Both variants support the traditional BibTeX convention for bib files that anything wrapped in a pair of curly braces is not modified when changing the case. For example:

\MakeSentenceCase{an Introduction to LaTeX}
\MakeSentenceCase*[{an Introduction to LaTeX}]

would yield:

\begin{verbatim}
An introduction to latex
An introduction to LaTeX
\end{verbatim}

In bib files designed with traditional BibTeX in mind, it has been fairly common to only wrap single letters in braces to prevent case-changing:

\begin{verbatim}
title = {An Introduction to {L}a{T}e{X}}
\end{verbatim}

The problem with this convention is that the braces will suppress the kerning on both sides of the enclosed letter. It is preferable to wrap the entire word in braces as shown in the first example.

\textsuperscript{35}By default, converting to sentence case is enabled for the following language identifiers: american, british, canadian, english, australian, newzealand as well as the aliases USenglish and UKenglish. Use \DeclareCaseLangs to extend or change this list.
\textbf{Table 13: \texttt{\textbackslash mkcomprange} setup}

\texttt{\textbackslash mkpageprefix\{\texttt{\textbackslash \langle pagination\rangle}\}\{\texttt{\textbackslash \langle postpro\rangle}\}\{\texttt{\textbackslash \langle text\rangle}\}}

This command is intended for use in field formatting directives which format the page numbers in the \langle text\rangle argument of citation commands and the \texttt{\textbackslash pages} field of bibliography entries. It will parse its \langle text\rangle argument and prefix it with 'p.' or 'pp.' by default. The optional \langle pagination\rangle argument holds the name of a field indicating the pagination type. This may be either \texttt{\textbackslash pagination} or \texttt{\textbackslash bookpagination}, with \texttt{\textbackslash pagination} being the default. The spacing between the prefix and the \langle text\rangle may be modified by redefining \texttt{\textbackslash ppstep}. The default is an unbreakable interword space. See §§ 2.3.10 and 3.13.3 for further details. See also \texttt{\textbackslash DeclareNumChars}, \texttt{\textbackslash DeclareRangeChars}, \texttt{\textbackslash DeclareRangeCommands}, and \texttt{\textbackslash NumCheckSetup}. The optional \langle postpro\rangle argument specifies a macro to be used for post-processing the \langle text\rangle. If only one optional argument is given, it is taken as \langle pagination\rangle. Here are two typical examples:

\texttt{\textbackslash DeclareFieldFormat\{postnote\}\{\texttt{\textbackslash mkpageprefix\{\texttt{\textbackslash \langle pagination\rangle}\}\{\texttt{\textbackslash \langle postpro\rangle}\}\{\texttt{\textbackslash \langle text\rangle}\}}\}}

\texttt{\textbackslash DeclareFieldFormat\{pages\}\{\texttt{\textbackslash mkpageprefix\{\texttt{\textbackslash \langle bookpagination\rangle}\}\{\texttt{\textbackslash \langle text\rangle}\}}\}}

The optional argument \texttt{\textbackslash \langle pagination\rangle} in the first example is omissible.

\texttt{\textbackslash mkpagetotal\{\texttt{\textbackslash \langle pagination\rangle}\}\{\texttt{\textbackslash \langle postpro\rangle}\}\{\texttt{\textbackslash \langle text\rangle}\}}

This command is similar to \texttt{\textbackslash mkpageprefix} except that it is intended for the pagetotal field of bibliography entries, i.e., it will print "123 pages" rather than "page 123". The optional \langle pagination\rangle argument defaults to \texttt{\textbackslash bookpagination}. The spacing inserted between the pagination suffix and the \langle text\rangle may be modified by redefining the macro \texttt{\textbackslash ppstep}. The optional \langle postpro\rangle argument specifies a macro to be used for post-processing the \langle text\rangle. If only one optional argument is given, it is taken as \langle pagination\rangle. Here is a typical example:

\texttt{\textbackslash DeclareFieldFormat\{pagetotal\}\{\texttt{\textbackslash mkpagetotal\{\texttt{\textbackslash \langle bookpagination\rangle}\}\{\texttt{\textbackslash \langle text\rangle}\}}\}}

The optional argument \texttt{\textbackslash \langle pagination\rangle} in this case is omissible.

\texttt{\textbackslash mkcomprange\{\texttt{\langle postpro\rangle}\}\{\texttt{\langle text\rangle}\}}
\texttt{\textbackslash mkcomprange*\{\texttt{\langle postpro\rangle}\}\{\texttt{\langle text\rangle}\}}

This command, which is intended for use in field formatting directives, will parse its \langle text\rangle argument for page ranges and compress them. For example, "125–129" may be formatted as "125–9". You may configure the behavior of \texttt{\textbackslash mkcomprange} by adjusting the \LaTeX\ counters \texttt{\textbackslash mincomprange}, \texttt{\textbackslash maxcomprange}, and \texttt{\textbackslash mincompwidth}, as illustrated in table 13. The default settings are \texttt{10}, \texttt{100000}, and \texttt{1}, respectively. This means that the command tries to compress as much as possible by default. Use \texttt{\textbackslash setcounter} to adjust the parameters. The scanner recognizes \texttt{\textbackslash bibrangedash} and hyphens as range dashes. It will normalize the dash by replacing any number of consecutive hyphens with \texttt{\textbackslash bibrangedash}. Lists of ranges delimited with \texttt{\textbackslash bibrangesep} are also supported. The backend will normalise any comma or semi-colons surrounded by optional space by replacing them with \texttt{\textbackslash bibrangesep}. If you want to hide a character from the list/range scanner for some reason, wrap...
the character or the entire string in curly braces. The optional \langle postpro \rangle argument specifies a macro to be used for post-processing the \langle text \rangle. This is important if you want to combine \texttt{\textbackslash mkcomprange} with other formatting macros which also need to parse their \langle text \rangle argument, such as \texttt{\textbackslash mkpageprefix}. Simply nesting these commands will not work as expected. Use the \langle postpro \rangle argument to set up the processing chain as follows:

\begin{verbatim}
\DeclareFieldFormat{postnote}{\texttt{\textbackslash mkcomprange}[\texttt{\textbackslash mkpageprefix[\texttt{\leftrightarrow pagination}]}](#1)}
\end{verbatim}

Note that \texttt{\textbackslash mkcomprange} is executed first, using \texttt{\textbackslash mkpageprefix} as post-processor. Also note that the \langle postpro \rangle argument is wrapped in an additional pair of braces. This is only required in this particular case to prevent LaTeX's optional argument scanner from getting confused by the nested brackets. The starred version of this command differs from the regular one in the way the \langle postpro \rangle argument is applied to a list of values. For example:

\begin{verbatim}
\texttt{\textbackslash mkcomprange}[\texttt{\textbackslash mkpageprefix}]{5, 123-129, 423-439}
\texttt{\textbackslash mkcomprange*}[\texttt{\textbackslash mkpageprefix}]{5, 123-129, 423-439}
\end{verbatim}

will output:

\begin{verbatim}
pp. 5, 123-9, 423-39
p. 5, pp. 123-9, pp. 423-39
\end{verbatim}

\begin{verbatim}
\texttt{\textbackslash mkfirstpage}[\langle postpro \rangle]{\langle text \rangle}
\texttt{\textbackslash mkfirstpage*}[\langle postpro \rangle]{\langle text \rangle}
\end{verbatim}

This command, which is intended for use in field formatting directives, will parse its \langle text \rangle argument for page ranges and print the start page of the range only. The scanner recognizes \texttt{\textbackslashibirangedash} and hyphens as range dashes. Lists of ranges delimited with \texttt{\textbackslashibrangesep} are also supported. If you want to hide a character from the list/range scanner for some reason, wrap the character or the entire string in curly braces. The optional \langle postpro \rangle argument specifies a macro to be used for post-processing the \langle text \rangle. See \texttt{\textbackslash mkcomprange} on how to use this argument. The starred version of this command differs from the regular one in the way the \langle postpro \rangle argument is applied to a list of values. For example:

\begin{verbatim}
\texttt{\textbackslash mkfirstpage}[\texttt{\textbackslash mkpageprefix}]{5, 123-129, 423-439}
\texttt{\textbackslash mkfirstpage*}[\texttt{\textbackslash mkpageprefix}]{5, 123-129, 423-439}
\end{verbatim}

will output:

\begin{verbatim}
pp. 5, 123, 423
p. 5, p. 123, p. 423
\end{verbatim}

\begin{verbatim}
\texttt{\textbackslash rangelen}{\langle rangefield \rangle}
\end{verbatim}

Takes the name of a bibfield declared as a range field in the data model and returns the length of the range. This is calculated by \texttt{biber} and can handle many special cases. It will return −1 for open ended ranges. Specifically \texttt{\textbackslash rangelen} can:

• Calculate the total of multiple ranges in the same field such as ’1-10, 20-30’
• Handle implicit ranges such as ’22-4’ and ’130-33’
• Handle roman numeral ranges in upper and lower case and consisting of both ASCII and Unicode roman numeral representations.

Here are some examples:
The \rangenlen command can be used in tests:

\ifnumcomp{\rangenlen{pages}}{=}{}{1}{add 'f'}{do nothing}

\DeclareNumChars{⟨characters⟩}
\DeclareNumChars*{⟨characters⟩}

This command configures the \ifnumeral, \ifnumerals, and \ifpages tests from §4.6.2. The setup will also affect \iffielldnum, \iffielldnums, \iffielldpages as well as \mklpageprefix and \mkpagetotal. The \rangenlen argument is an undelimited list of characters which are to be considered as being part of a number. The regular version of this command replaces the current setting, the starred version appends its argument to the current list. The default setting is:

\DeclareNumChars{.}

This means that a (section or other) number like ‘3.4.5’ will be considered as a number. Note that Arabic and Roman numerals are detected by default, there is no need to declare them explicitly.

\DeclareRangeChars{⟨characters⟩}
\DeclareRangeChars*{⟨characters⟩}

This command configures the \ifnumerals and \ifpages tests from §4.6.2. The setup will also affect \iffielldnums and \iffielldpages as well as \mklpageprefix and \mkpagetotal. The \rangenlen argument is an undelimited list of characters which are to be considered as range indicators. The regular version of this command replaces the current setting, the starred version appends its argument to the current list. The default setting is:

\DeclareRangeChars{~,;+-/+}

This means that strings like ‘3–5’, ‘35+’, ‘8/9’ and so on will be considered as a range by \ifnumerals and \ifpages. Non-range characters in such strings are recognized as numbers. So strings like ‘3a–5a’ and ‘35b+’ are not deemed to be ranges by default. See also §§2.3.10 and 3.13.3 for further details.

\DeclareRangeCommands{⟨commands⟩}
\DeclareRangeCommands*{⟨commands⟩}

This command is similar to \DeclareRangeChars, except that the \rangenlen argument is an undelimited list of commands which are to be considered as range indicators. The regular version of this command replaces the current setting, the starred version appends its argument to the current list. The default list is rather long and should cover all common cases; here is a shorter example:

\DeclareRangeCommands{\& ↦ \bibrangedash\textendash\textemdash\psq\psqq}

See also §§2.3.10 and 3.13.3 for further details.
This command is similar to \DeclareRangeCommands, except that it only affects the \ifpages and \iffieldpages tests but not \ifnumerals and \iffieldnum. The default setting is:

\DeclarePageCommands{\pno\ppno}

\NumCheckSetup{⟨code⟩}

Use this command to temporarily redefine any commands which interfere with the tests performed by \ifnumeral, \ifnumerals, and \ifpages from §4.6.2. The setup will also affect \iffieldnum, \iffieldnumerals, \iffieldpages as well as \mkpageprefix and \mkpagetotal. The ⟨code⟩ will be executed in a group by these commands. Since the above mentioned commands will expand the string to be analyzed, it is possible to remove commands to be ignored by the tests by making them expand to an empty string. See also §§2.3.10 and 3.13.3 for further details.

\DeclareCaseLangs{⟨languages⟩}

\DeclareCaseLangs*{⟨languages⟩}

Defines the list of languages which are considered by the \MakeSentenceCase* command as it converts a string to sentence case. The ⟨languages⟩ argument is a comma-separated list of babel/polyglossia language identifiers. The regular version of this command replaces the current setting, the starred version appends its argument to the current list. The default setting is:

\DeclareCaseLangs{%
  american,british,canadian,english,australian,newzealand,USenglish
  ↪,UKenglish}

See the babel/polyglossia manuals and table2 for a list of languages identifiers.

\BibliographyWarning{⟨message⟩}

This command is similar to \PackageWarning but prints the entry key of the entry currently being processed in addition to the input line number. It may be used in the bibliography as well as in citation commands. If the ⟨message⟩ is fairly long, use \MessageBreak to include line breaks. Note that the standard \PackageWarning command does not provide a meaningful clue when used in the bibliography since the input line number is the line on which the \printbibliography command was given.

\pagetrackertrue
\pagetrackerfalse

These commands activate or deactivate the citation tracker locally (this will affect the \iffirstonpage and \ifsamepage test from §4.6.2). They are intended for use in the definition of citation commands or anywhere in the document body. If a citation command is to be excluded from page tracking, use \pagetrackerfalse in the ⟨precode⟩ argument of \DeclareCiteCommand. See §4.3.1 for details. Note that these commands have no effect if page tracking has been disabled globally.

\citetrackertrue
\citetrackerfalse

These commands activate or deactivate all citation trackers locally (this will affect the \ifciteseen, \ifentryseen, \ifcitebib, and \ifciteidem tests from §4.6.2). They are intended for use in the definition of citation commands or anywhere in the document body. If a citation command is to be excluded from tracking, use \citetrackerfalse in the ⟨precode⟩ argument of \DeclareCiteCommand. See §4.3.1 for details. Note that these commands have no effect if tracking has been disabled globally.

\backtrackertrue
\backtrackerfalse

These commands activate or deactivate the backref tracker locally. They are intended for use in the definition of citation commands or anywhere in the document body. If a citation command is to be excluded from backtracking, use \backtrackerfalse in the ⟨precode⟩ argument of \DeclareCiteCommand. Note that these commands have no effect if the backref option has been not been set globally.

## 4.7 Punctuation and Spacing

The biblatex package provides elaborate facilities designed to manage and track punctuation and spacing in the bibliography and in citations. These facilities work on two levels. The high-level commands discussed in §4.7.1 deal with punctuation and whitespace inserted by the bibliography style
between the individual segments of a bibliography entry. The commands in §§4.7.2, 4.7.3, 4.7.4 work at a lower level. They use TeX’s space factor and modified space factor codes to track punctuation in a robust and efficient way. This way it is possible to detect trailing punctuation marks within fields, not only those explicitly inserted between fields. The same technique is also used for automatic capitalization of localisation strings, see \texttt{\textbackslash DeclareCapitalPunctuation} in §4.7.5 as well as §4.8 for details. Note that these facilities are only made available locally in citations and bibliographies. They will not affect any other part of a document.

### 4.7.1 Block and Unit Punctuation

The major segments of a bibliography entry are ‘blocks’ and ‘units’. A block is the larger segment of the two, a unit is shorter or at most equal in length. For example, the values of fields such as title or note usually form a unit which is separated from subsequent data by a period or a comma. A block may comprise several fields which are treated as separate units, for example publisher, location, and year. The segmentation of an entry into blocks and units is at the discretion of the bibliography style. An entry is segmented by inserting \texttt{\textbackslash newblock} and \texttt{\textbackslash newunit} commands at suitable places and \texttt{\textbackslash finentry} at the very end (see §4.23 for an example). See also §4.11.7 for some practical hints.

\texttt{\textbackslash newblock}  
**Records the end of a block.** This command does not print anything, it merely marks the end of the block. The block delimiter \texttt{\textbackslash newblockpunct} will be inserted by a subsequent \texttt{\textbackslash printtext}, \texttt{\textbackslash printfield}, \texttt{\textbackslash printlist}, \texttt{\textbackslash printnames}, or \texttt{\textbackslash bibstring} command. You may use \texttt{\textbackslash newblock} at suitable places without having to worry about spurious blocks. A new block will only be started by the next \texttt{\textbackslash printfield} (or similar) command if this command prints anything. See §4.11.7 for further details.

\texttt{\textbackslash newunit}  
**Records the end of a unit and puts the default delimiter \texttt{\textbackslash newunitpunct} in the punctuation buffer.** This command does not print anything, it merely marks the end of the unit. The punctuation buffer will be inserted by the next \texttt{\textbackslash printtext}, \texttt{\textbackslash printfield}, \texttt{\textbackslash printlist}, \texttt{\textbackslash printnames}, or \texttt{\textbackslash bibstring} command. You may use \texttt{\textbackslash newunit} after commands like \texttt{\textbackslash printfield} without having to worry about spurious punctuation and whitespace. The buffer will only be inserted by the next \texttt{\textbackslash printfield} or similar command if both fields are non-empty. This also applies to \texttt{\textbackslash printtext}, \texttt{\textbackslash printlist}, \texttt{\textbackslash printnames}, and \texttt{\textbackslash bibstring}. See §4.11.7 for further details.

\texttt{\textbackslash finentry}  
**Inserts \texttt{\textbackslash finentrypunct}.** This command should be used at the very end of every bibliography entry.

\texttt{\textbackslash setunit}\{\texttt{punctuation}\}  
\texttt{\textbackslash setunit*}\{\texttt{punctuation}\}  
The \texttt{\textbackslash setunit} command is similar to \texttt{\textbackslash newunit} except that it uses \texttt{\textbackslash punctuation} instead of \texttt{\textbackslash newunitpunct}. The starred variant differs from the regular version in that it checks if the last \texttt{\textbackslash printtext}, \texttt{\textbackslash printfield}, \texttt{\textbackslash printlist}, \texttt{\textbackslash printnames}, or \texttt{\textbackslash bibstring} command did actually print anything. If not, it does nothing.

\texttt{\textbackslash printunit}\{\texttt{punctuation}\}  
\texttt{\textbackslash printunit*}\{\texttt{punctuation}\}  
The \texttt{\textbackslash printunit} command is similar to \texttt{\textbackslash setunit} except that \texttt{\textbackslash punctuation} persists in the buffer. This ensures that \texttt{\textbackslash punctuation} is inserted before the next non-empty field printed by the \texttt{\textbackslash printtext}, \texttt{\textbackslash printfield}, \texttt{\textbackslash printlist}, \texttt{\textbackslash printnames}, or \texttt{\textbackslash bibstring} commands—regardless of any intermediate calls to \texttt{\textbackslash newunit} or \texttt{\textbackslash setunit}.

\texttt{\textbackslash setpunctfont}\{\texttt{\textbackslash command}\}  
This command, which is intended for use in field formatting directives, provides an alternative way of dealing with unit punctuation after a field printed in a different font (for example, a title printed in italics). The standard LaTeX way of dealing with this is adding a small amount of space, the so-called italic correction. This command allows adapting the punctuation to the font of the preceding field. The \texttt{\textbackslash command} should be a text font command which takes one argument, such as \texttt{\textbackslash emph} or \texttt{\textbackslash textbf}. This command will only affect punctuation marks inserted by one of the commands from §4.7.3. The font adaption is applied to the next punctuation mark only and will be reset automatically thereafter. If you want to reset it manually before it takes effect, issue \texttt{\textbackslash resetpunctfont}. If the \texttt{\textbackslash punctfont} package option is disabled, this command does nothing. Note that the \texttt{\textbackslash mkbibemph}, \texttt{\textbackslash mkbibitalic}, and \texttt{\textbackslash mkbibbold} wrappers from §4.10.4 incorporate this feature by default.

\texttt{\textbackslash resetpunctfont}  
This command resets the unit punctuation font defined with \texttt{\textbackslash setpunctfont} before it takes effect. If the \texttt{\textbackslash punctfont} package option is disabled, this command does nothing.
4.7.2 Punctuation Tests

The following commands may be used to test for preceding punctuation marks at any point in citations and the bibliography.

\ifpunct{⟨true⟩}{⟨false⟩}
Executes ⟨true⟩ if preceded by any punctuation mark except for an abbreviation dot, and ⟨false⟩ otherwise.

\ifterm{⟨true⟩}{⟨false⟩}
Executes ⟨true⟩ if preceded by a terminal punctuation mark, and ⟨false⟩ otherwise. A terminal punctuation mark is any punctuation mark which has been registered for automatic capitalization, either with \DeclareCapitalPunctuation or by default, see §4.7.5 for details. By default, this applies to periods, exclamation marks, and question marks.

\ifpunctmark{⟨character⟩}{⟨true⟩}{⟨false⟩}
Executes ⟨true⟩ if preceded by the punctuation mark ⟨character⟩, and ⟨false⟩ otherwise. The ⟨character⟩ may be a comma, a semicolon, a colon, a period, an exclamation mark, a question mark, or an asterisk. Note that a period denotes an end-of-sentence period. Use the asterisk to test for the dot after an abbreviation. If this command is used in a formatting directive for name lists, i.e., in the argument to \DeclareNameFormat, the ⟨character⟩ may also be an apostrophe.

\ifprefchar{⟨true⟩}{⟨false⟩}
Executes ⟨true⟩ if preceded by any prefix character declared by \DeclarePrefChars.

4.7.3 Adding Punctuation

The following commands are designed to prevent double punctuation marks. Bibliography and citation styles should always use these commands instead of literal punctuation marks. All \add... commands in this section automatically remove preceding whitespace with \unspace (see §4.7.4). Note that the behavior of all \add... commands discussed below is the package default, which is restored whenever \biblatex switches languages. This behavior may be adjusted with \DeclarePunctuationPairs from §4.7.5.

\adddot
Adds a period unless it is preceded by any punctuation mark. The purpose of this command is inserting the dot after an abbreviation. Any dot inserted this way is recognized as such by the other punctuation commands. This command may also be used to turn a previously inserted literal period into an abbreviation dot.

\addcomma
Adds a comma unless it is preceded by another comma, a semicolon, a colon, or a period.

\addsemicolon
Adds a semicolon unless it is preceded by a comma, another semicolon, a colon, or a period.

\addcolon
Adds a colon unless it is preceded by a comma, a semicolon, another colon, or a period.

\addperiod
Adds a period unless it is preceded by an abbreviation dot or any other punctuation mark. This command may also be used to turn a previously inserted abbreviation dot into a period, for example at the end of a sentence.

\addexclam
Adds an exclamation mark unless it is preceded by any punctuation mark except for an abbreviation dot.

\addquestion
Adds a question mark unless it is preceded by any punctuation mark except for an abbreviation dot.

\isdot
Turns a previously inserted literal period into an abbreviation dot. In contrast to \adddot, nothing is inserted if this command is not preceded by a period.

\nopunct
Adds an internal marker which will cause the next punctuation command to print nothing.

4.7.4 Adding Whitespace

The following commands are designed to prevent spurious whitespace. Bibliography and citation styles should always use these commands instead of literal whitespace. In contrast to the commands in §§4.7.2 and 4.7.3, they are not restricted to citations and the bibliography but available globally.

\unspace
Removes preceding whitespace, i.e., removes all skips and penalties from the end of the current horizontal list. This command is implicitly executed by all of the following commands.

\addspace
Adds a breakable interword space.

\addnbspace
Adds a non-breakable interword space.

\addthinspace
Adds a breakable thin space.
\addnbspace Adds a non-breakable thin space. This is similar to \, and \thinspace.

\addlowpenspace Adds a space penalized by the value of the lownamepenalty counter, see §§ 3.10.4 and 4.10.3 for details.

\addhighpenspace Adds a space penalized by the value of the highnamepenalty counter, see §§ 3.10.4 and 4.10.3 for details.

\addlpthinspace Similar to \addlowpenspace but adds a breakable thin space.

\addhpthinspace Similar to \addhighpenspace but adds a breakable thin space.

\addabbrvspace Adds a space penalized by the value of the abbrvpenalty counter, see §§ 3.10.4 and 4.10.3 for details.

\addabthinspace Similar to \addabbrvspace but using a thin space.

\adddotspace Executes \adddot and adds a space penalized by the value of the abbrvpenalty counter, see §§ 3.10.4 and 4.10.3 for details.

\addslash Adds a breakable slash. This command differs from the \slash command in the LaTeX kernel in that a linebreak after the slash is not penalized at all.

Note that the commands in this section implicitly execute \unspace to remove spurious whitespace, hence they may be used to override each other. For example, you may use \addnbspace to transform a previously inserted interword space into a non-breakable one and \addspace to turn a non-breakable space into a breakable one.

4.7.5 Configuring Punctuation and Capitalization

The following commands configure various features related to punctuation and automatic capitalization.

\DeclarePrefChars{⟨characters⟩}

This command declares characters that are to be treated specially when testing to see if \bibnamedelimc is to be inserted between a name prefix and a family name. If a character is in the list of ⟨characters⟩, \bibnamedelimc is not inserted. It is used to allow abbreviated name prefixes like ‘d’Argent’ where no space should be inserted after the apostrophe. The default setting is:

\DeclarePrefChars{'}

\DeclareAutoPunctuation{⟨characters⟩}

This command defines the punctuation marks to be considered by the citation commands as they scan ahead for punctuation. Note that ⟨characters⟩ is an undelimited list of characters. Valid ⟨characters⟩ are period, comma, semicolon, colon, exclamation and question mark. The default setting is:

\DeclareAutoPunctuation{.,;:!?}

This definition is restored automatically whenever the autopunct package option is set to true. Executing \DeclareAutoPunctuation{} is equivalent to setting autopunct=false, i.e., it disables this feature.

\DeclareCapitalPunctuation{⟨characters⟩}

When \bibtex inserts localisation strings, i.e., key terms such as ‘edition’ or ‘volume’, it automatically capitalizes them after terminal punctuation marks. This command defines the punctuation marks which will cause localisation strings to be capitalized if one of them precedes a string. Note that ⟨characters⟩ is an undelimited list of characters. Valid ⟨characters⟩ are period, comma, semicolon, colon, exclamation and question mark. The package default is:

\DeclareCapitalPunctuation{.!?}

Using \DeclareCapitalPunctuation with an empty argument is equivalent to disabling automatic capitalization. Since this feature is language specific, this command must be used in the argument to \DefineBibliographyExtras (when used in the preamble) or \DeclareBibliographyExtras (when used in a localisation module). See §§ 3.9 and 4.9 for details. By default, strings are capitalized after periods, exclamation marks, and question marks. All strings are generally capitalized at the beginning of a paragraph (in fact whenever TeX is in vertical mode).
\DeclarePunctuationPairs{⟨identifier⟩}{⟨characters⟩}

Use this command to declare valid pairs of punctuation marks. This will affect the punctuation commands discussed in §4.7.3. For example, the description of \addcomma states that this command adds a comma unless it is preceded by another comma, a semicolon, a colon, or a period. In other words, commas after abbreviation dots, exclamation marks, and question marks are permitted. These valid pairs are declared as follows:

\DeclarePunctuationPairs{comma}{*!?}

The (⟨identifier⟩) selects the command to be configured. The identifiers correspond to the names of the punctuation commands from §4.7.3 without the \add prefix, i.e., valid (⟨identifier⟩) strings are dot, comma, semicolon, colon, period, exclam, question. The (⟨characters⟩) argument is an undelimited list of punctuation marks. Valid (⟨characters⟩) are comma, semicolon, colon, period, exclamation mark, question mark, and asterisk. A period in the (⟨characters⟩) argument denotes an end-of-sentence period, an asterisk the dot after an abbreviation. This is the default setup, which is automatically restored whenever \biblatex switches languages and corresponds to the behavior described in §4.7.3:

\DeclarePunctuationPairs{dot}{}
\DeclarePunctuationPairs{comma}{*!?}
\DeclarePunctuationPairs{semicolon}{*!?}
\DeclarePunctuationPairs{colon}{*!?}
\DeclarePunctuationPairs{period}{}
\DeclarePunctuationPairs{exclam}{*}
\DeclarePunctuationPairs{question}{*}

Since this feature is language specific, \DeclarePunctuationPairs must be used in the argument to \DefineBibliographyExtras (when used in the preamble) or \DeclareBibliographyExtras (when used in a localisation module). See §§3.9 and 4.9 for details. Note that some localisation modules may use a setup which is different from the package default.\footnote{As of this writing, the \texttt{american} module uses different settings for ‘American-style’ punctuation.}

\DeclareQuotePunctuation{⟨characters⟩}

This command controls ‘American-style’ punctuation. The \mkbibquote wrapper from §4.10.4 can interact with the punctuation facilities discussed in §§ 4.7.1, 4.7.3, 4.7.4. Punctuation marks after \mkbibquote will be moved inside the quotes if they have been registered with \DeclareQuotePunctuation. Note that (⟨characters⟩) is an undelimited list of characters. Valid (⟨characters⟩) are period, comma, semicolon, colon, exclamation and question mark. Here is an example:

\DeclareQuotePunctuation{.,}

Executing \DeclareQuotePunctuation{} is equivalent to disabling this feature. This is the package default. Since this feature is language specific, this command must be used in the argument to \DefineBibliographyExtras (when used in the preamble) or \DeclareBibliographyExtras (when used in a localisation module). See §§3.9 and 4.9 for details. See also §3.11.1.

\uspunctuation

A shorthand using the lower-level commands \DeclareQuotePunctuation and \DeclarePunctuationPairs to activate ‘American-style’ punctuation. See §3.11.1 for details. This shorthand is provided for convenience only. The effective settings are applied by the lower-level commands.

\stdpunctuation

Undoes the settings applied by \uspunctuation, restoring standard punctuation. As standard punctuation is the default setting, you only need this command to override a previously executed \uspunctuation command. See §3.11.1 for details.

4.7.6 Correcting Punctuation Tracking

The facilities for punctuation tracking and automatic capitalization are very reliable under normal circumstances, but there are always marginal cases which may require manual intervention. Typical
cases are localisation strings printed as the first word in a footnote (which is usually treated as the
beginning of a paragraph as far as capitalization is concerned, but TeX is not in vertical mode at this
point) or punctuation after periods which are not really end-of-sentence periods (for example, after an
ellipsis like " [...]" a command such as \addperiod would do nothing since parentheses and brackets
are transparent to the punctuation tracker). In such cases, use the following commands in bibliography
and citation styles to mark the beginning or middle of a sentence if and where required:

\bibsentence
This command marks the beginning of a sentence. A localisation string immediately after this command
will be capitalized and the punctuation tracker is reset, i.e., this command hides all preceding
punctuation marks from the punctuation tracker and enforces capitalization.

\midsentence
This command marks the middle of a sentence. A localisation string immediately after this command
will not be capitalized and the punctuation tracker is reset, i.e., this command hides all preceding
punctuation marks from the punctuation tracker and suppresses capitalization.

\midsentence*
The starred variant of \midsentence differs from the regular one in that a preceding abbreviation
dot is not hidden from the punctuation tracker, i.e., any code after \midsentence* will see a
preceding abbreviation dot. All other punctuation marks are hidden from the punctuation tracker and
capitalization is suppressed.

4.8 Localization Strings
Localization strings are key terms such as ‘edition’ or ‘volume’ which are automatically translated
by \biblatex’s localisation modules. See §4.9 for an overview and §4.9.2 for a list of all strings
supported by default. The commands in this section are used to print the localised term.

\bibstring{\{wrapper\}}\{(key)\}
Prints the localisation string \{key\}, where \{key\} is an identifier in lowercase letters (see §4.9.2). The
string will be capitalized as required, see §4.7.5 for details. Depending on the abbreviate package
option from §3.1.2.1, \bibstring prints the short or the long version of the string. If localisation
strings are nested, i.e., if \bibstring is used in another string, it will behave like \bibxstring. If the \{wrapper\} argument is given, the string is passed to the \{wrapper\} for formatting. This is intended
for font commands such as \textit.

\biblstring{\{wrapper\}}\{(key)\}
Similar to \bibstring but always prints the long string, ignoring the abbreviate option.

\bibsstring{\{wrapper\}}\{(key)\}
Similar to \bibstring but always prints the short string, ignoring the abbreviate option.

\bibcpstring{\{wrapper\}}\{(key)\}
Similar to \bibstring but the term is always capitalized.

\bibcplstring{\{wrapper\}}\{(key)\}
Similar to \biblstring but the term is always capitalized.

\bibcpsstring{\{wrapper\}}\{(key)\}
Similar to \bibstring but the term is always capitalized.

\bibucstring{\{wrapper\}}\{(key)\}
Similar to \bibstring but the whole term is uppercased.

\bibuclstring{\{wrapper\}}\{(key)\}
Similar to \biblstring but the whole term is uppercased.

\bibucsstring{\{wrapper\}}\{(key)\}
Similar to \bibstring but the whole term is uppercased.

\biblcstring{\{wrapper\}}\{(key)\}
Similar to \bibstring but the whole term is lowercased.

\biblclstring{\{wrapper\}}\{(key)\}
Similar to \biblstring but the whole term is lowercased.

\biblcsstring{\{wrapper\}}\{(key)\}
Similar to \bibbsstring but the whole term is lowercased.
A simplified but expandable version of \bibstring. Note that this variant does not capitalize automatically, nor does it hook into the punctuation tracker. It is intended for special cases in which strings are nested or an expanded localisation string is required in a test.

\bibxstring{\langle key \rangle}

Similar to \bibxstring but always uses the long string, ignoring the abbreviate option.

\bibxsstring{\langle wrapper \rangle}{\langle key \rangle}

Similar to \bibxstring but always uses the short string, ignoring the abbreviate option.

\mainlang

Switches from the current language to the main document language. This can be used the \langle wrapper \rangle argument in the localisation string commands above.

### 4.9 Localization Modules

A localisation module provides translations for key terms such as ‘edition’ or ‘volume’ as well as definitions for language specific features such as the date format and ordinals. These definitions are provided in files with the suffix \texttt{lbx}. The base name of the file must be a language name known to the babel/polyglossia packages. The \texttt{lbx} files may also be used to map babel/polyglossia language names to the backend modules of the \texttt{biblatex} package. All localisation modules are loaded on demand in the document body. Note that the contents of the file are processed in a group and that the category code of the character $@$ is temporarily set to ‘letter’.

#### 4.9.1 Localization Commands

The user-level versions of the localisation commands were already introduced in §3.9. When used in \texttt{lbx} files, however, the syntax of localisation commands is different from the user syntax in the preamble and the configuration file. When used in localisation files, there is no need to specify the \langle language \rangle because the mapping of strings to a language is already provided by the name of the \texttt{lbx} file.

\DeclareBibliographyStrings{\langle definitions \rangle}

This command is only available in \texttt{lbx} files. It is used to define localisation strings. The \langle definitions \rangle consist of \langle key \rangle=\langle value \rangle pairs which assign an expression to an identifier. A complete list of all keys supported by default is given in §4.9.2. Note that the syntax of the value is different in \texttt{lbx} files. The value assigned to a key consists of two expressions, each of which is wrapped in an additional pair of brackets. This is best shown by example:

\begin{verbatim}
\DeclareBibliographyStrings{
  bibliography = {{Bibliography}{Bibliography}},
  shorthands = {{List of Abbreviations}{Abbreviations}},
  editor = {{editor}{ed.}},
  editors = {{editors}{eds.}},
}
\end{verbatim}

The first value is the long, written out expression, the second one is an abbreviated or short form. Both strings must always be given even though they may be identical if an expression is always (or never) abbreviated. Depending on the setting of the abbreviate package option (see §3.1.2.1), \texttt{biblatex} selects one expression when loading the \texttt{lbx} file. There is also a special key named \texttt{inherit} which copies the strings from a different language. This is intended for languages which only differ in a few expressions, such as German and Austrian or American and British English. For example, here are the complete definitions for Austrian:

\begin{verbatim}
\DeclareBibliographyStrings{
  inherit = {german},
  january = {{J"anner}{J"an.}},
}
\end{verbatim}

The above examples are slightly simplified. Real localisation files should use the punctuation and formatting commands discussed in §§4.7.3 and 3.10 instead of literal punctuation. Here is an excerpt from a real localisation file:
Note the handling of abbreviation dots, the spacing in abbreviated expressions, and the capitalization in the example above. All expressions should be capitalized as they usually are when used in the middle of a sentence. The \texttt{biblatex} package will automatically capitalize the first word when required at the beginning of a sentence, see \texttt{\DeclareCapitalPunctuation} in §4.7.5 for details. Expressions intended for use in headings are special. They should be capitalized in a way that is suitable for titling and should not be abbreviated (but they may have a short form).

\InheritBibliographyStrings{(language)}

This command is only available in \texttt{lbx} files. It copies the localisation strings for \texttt{language} to the current language, as specified by the name of the \texttt{lbx} file.

\DeclareBibliographyExtras{(code)}

This command is only available in \texttt{lbx} files. It is used to adapt language specific features such as the date format and ordinals. The \texttt{code}, which may be arbitrary \LaTeX\ code, will usually consist of redefinitions of the formatting commands from §4.10.2.

\UndeclareBibliographyExtras{(code)}

This command is only available in \texttt{lbx} files. It is used to restore any formatting commands modified with \texttt{\DeclareBibliographyExtras}. If a redefined command is included in §4.10.2, there is no need to restore its previous definition since these commands are localised by all language modules anyway.

\InheritBibliographyExtras{(language)}

This command is only available in \texttt{lbx} files. It copies the bibliography extras for \texttt{language} to the current language, as specified by the name of the \texttt{lbx} file.

\DeclareHyphenationExceptions{(text)}

This command corresponds to \texttt{\DefineHyphenationExceptions} from §3.9. The difference is that it is only available in \texttt{lbx} files and that the \texttt{language} argument is omitted. The hyphenation exceptions will affect the language of the \texttt{lbx} file currently being processed.

\DeclareRedundantLanguages{(language, language, ...) }{(langid, langid, ...)}

This command provides the language mappings required by the \texttt{clearlang} option from §3.1.2.1. The \texttt{language} is the string given in the language field (without the optional lang prefix); \texttt{langid} is \texttt{babel/polyglossia}'s language identifier, as given in the optional argument of \texttt{\usepackage} when loading \texttt{babel} or the argument of \texttt{\setdefaultlanguage} or \texttt{\setotherlanguages} when using \texttt{polyglossia}. This command may be used in \texttt{lbx} files or in the document preamble. Here are some examples:

\begin{verbatim}
\DeclareRedundantLanguages{french}{french}
\DeclareRedundantLanguages{german}{german,ngerman,austrian,
\hspace{1em}nswissgerman,swissgerman}
\DeclareRedundantLanguages{english,american}{english,american,
\hspace{1em}british,
\hspace{1em}canadian,austalian,newzealand,USEnglish,UKenglish}
\end{verbatim}

Note that this feature needs to be enabled globally with the \texttt{clearlang} option from §3.1.2.1. If it is disabled, all mappings will be ignored. If the \texttt{langid} parameter is blank, \texttt{biblatex} will clear the mappings for the corresponding \texttt{language}, i.e., the feature will be disabled for this \texttt{language} only.

\DeclareLanguageMapping{(language)}{(file)}

This command maps a \texttt{babel/polyglossia} language identifier to an \texttt{lbx} file. The \texttt{language} must be a language name known to the \texttt{babel/polyglossia} package, i.e., one of the identifiers listed in table 2. The \texttt{file} argument is the name of an alternative \texttt{lbx} file without the .\texttt{lbx} suffix. Declaring the same mapping more than once is possible. Subsequent declarations will simply overwrite any previous ones. This command may only be used in the preamble. See §4.11.8 for further details.

216
This command, which may be used in the preamble (including cbx and bbx files) as well as in lbx files, declares new localisation strings, i.e., it initializes a new ⟨key⟩ to be used in the ⟨definitions⟩ of \DefineBibliographyStrings or \DeclareBibliographyStrings. The ⟨key⟩ argument may also be a comma-separated list of key names. When used in an lbx, the ⟨key⟩ is initialized only for the language specified by the name of the lbx file. The keys listed in § 4.9.2 are defined by default.

### 4.9.2 Localization Keys

The localisation keys in this section are defined by default and covered by the localisation files which come with \biblatex. Note that these strings are only available in citations, the bibliography and bibliography lists. All expressions should be capitalized as they usually are when used in the middle of a sentence. \biblatex will capitalize them automatically at the beginning of a sentence. The only exceptions to these rules are the three strings intended for use in headings.

#### 4.9.2.1 Headings

The following strings are special because they are intended for use in headings and made available globally via macros. For this reason, they should be capitalized for use in headings and they must not include any local commands which are part of \biblatex's author interface.

- **bibliography**
  - The term 'bibliography', also available as \bibname.
- **references**
  - The term 'references', also available as \refname.
- **shorthands**
  - The term 'list of shorthands' or 'list of abbreviations', also available as \blistname.

#### 4.9.2.2 Roles, Expressed as Functions

The following keys refer to roles which are expressed as a function ('editor', 'translator') rather than as an action ('edited by', 'translated by').

- **editor**
  - The term 'editor', referring to the main editor. This is the most generic editorial role.
- **editors**
  - The plural form of editor.
- **compiler**
  - The term 'compiler', referring to an editor whose task is to compile a work.
- **compilers**
  - The plural form of compiler.
- **founder**
  - The term 'founder', referring to a founding editor.
- **founders**
  - The plural form of founder.
- **continuator**
  - An expression like 'continuator', 'continuation', or 'continued', referring to a past editor who continued the work of the founding editor but was subsequently replaced by the current editor.
- **continuators**
  - The plural form of continuator.
- **redactor**
  - The term 'redactor', referring to a secondary editor.
- **redactors**
  - The plural form of redactor.
- **reviser**
  - The term 'reviser', referring to a secondary editor.
- **revisers**
  - The plural form of reviser.
- **collaborator**
  - A term like 'collaborator', 'collaboration', 'cooperator', or 'cooperation', referring to a secondary editor.
- **collaborators**
  - The plural form of collaborator.
- **translator**
  - The term 'translator'.
- **translators**
  - The plural form of translator.
- **commentator**
  - The term 'commentator', referring to the author of a commentary to a work.
- **commentators**
  - The plural form of commentators.
- **annotator**
  - The term 'annotator', referring to the author of annotations to a work.
- **annotators**
  - The plural form of annotators.

#### 4.9.2.3 Concatenated Editor Roles, Expressed as Functions

The following keys are similar in function to editor, translator, etc. They are used to indicate additional roles of the editor, e.g., 'editor and translator', 'editor and foreword'.

- **editortr**
  - Used if editor/translator are identical.
- **editorstr**
  - The plural form of editortr.
- **editorco**
  - Used if editor/commentator are identical.
- **editorsco**
  - The plural form of editorco.
- **editoran**
  - Used if editor/annotator are identical.
- **editorsan**
  - The plural form of editoran.
editorin Used if editor/introduction are identical.
editorsin The plural form of editorin.
editorfo Used if editor/foreword are identical.
editorsfo The plural form of editorfo.
editoraf Used if editor/aftword are identical.
editorsaf The plural form of editoraf.

Keys for editor/translator/(role) combinations:
editortrco Used if editor/translator/commentator are identical.
editorstrco The plural form of editortrco.
editortran Used if editor/translator/annotator are identical.
editorstran The plural form of editortran.
editortrin Used if editor/translator/introduction are identical.
editorstrin The plural form of editortrin.
editortrfo Used if editor/translator/foreword are identical.
editorstrfo The plural form of editortrfo.
editortraf Used if editor/translator/aftword are identical.
editorstraf The plural form of editortraf.

Keys for editor/commentator/(role) combinations:
editorcoin Used if editor/commentator/introduction are identical.
editorscoin The plural form of editorcoin.
editorcofo Used if editor/commentator/foreword are identical.
editorscofo The plural form of editorcofo.
editorcoaf Used if editor/commentator/aftword are identical.
editorscoaf The plural form of editorcoaf.

Keys for editor/annotator/(role) combinations:
editoranin Used if editor/annotator/introduction are identical.
editorsanin The plural form of editoranin.
editoranfo Used if editor/annotator/foreword are identical.
editorsanfo The plural form of editoranfo.
editoranaf Used if editor/annotator/aftword are identical.
editorsanaf The plural form of editoranaf.

Keys for editor/translator/commentator/(role) combinations:
editortrcoin Used if editor/translator/commentator/introduction are identical.
editorstrcoin The plural form of editortrcoin.
editortrcofo Used if editor/translator/commentator/foreword are identical.
editorstrcofo The plural form of editortrcofo.
editortrcoaf Used if editor/translator/commentator/aftword are identical.
editorstrcoaf The plural form of editortrcoaf.

Keys for editor/annotator/commentator/(role) combinations:
editortrnain Used if editor/annotator/commentator/introduction are identical.
editorstrnain The plural form of editortrnain.
editortranfo Used if editor/annotator/commentator/foreword are identical.
editorstranfo The plural form of editortranfo.
editortrnaf Used if editor/annotator/commentator/aftword are identical.
editorstrnaf The plural form of editortrnaf.
4.9.2.4 Concatenated Translator Roles, Expressed as Functions

The following keys are similar in function to translator. They are used to indicate additional roles of the translator, e.g., 'translator and commentator', 'translator and introduction'.

- **translatorco**: Used if translator/commentator are identical.
- **translatorsco**: The plural form of translatorco.
- **translatorsan**: Used if translator/annotator are identical.
- **translatorsan**: The plural form of translatorsan.
- **translatorin**: Used if translator/introduction are identical.
- **translatorsin**: The plural form of translatorin.
- **translatorfo**: Used if translator/foreword are identical.
- **translatorsfo**: The plural form of translatorfo.
- **translatoraf**: Used if translator/aftword are identical.
- **translatorsaf**: The plural form of translatoraf.

**Keys for translator/commentator/(role) combinations:**

- **translatorcoin**: Used if translator/commentator/introduction are identical.
- **translatorscoin**: The plural form of translatorcoin.
- **translatorcofo**: Used if translator/commentator/foreword are identical.
- **translatorscofo**: The plural form of translatorcofo.
- **translatorcoaf**: Used if translator/commentator/aftword are identical.
- **translatorscoaf**: The plural form of translatorscoaf.

**Keys for translator/annotator/(role) combinations:**

- **translatoranin**: Used if translator/annotator/introduction are identical.
- **translatorsanin**: The plural form of translatoranin.
- **translatoranfo**: Used if translator/annotator/foreword are identical.
- **translatorsanfo**: The plural form of translatoranfo.
- **translatoranaf**: Used if translator/annotator/aftword are identical.
- **translatorsanaf**: The plural form of translatorsanaf.

4.9.2.5 Roles, Expressed as Actions

The following keys refer to roles which are expressed as an action ('edited by', 'translated by') rather than as a function ('editor', 'translator').

- **byauthor**: The expression '[created] by ⟨name⟩'.
- **byeditor**: The expression 'edited by ⟨name⟩'.
- **bycompiler**: The expression 'compiled by ⟨name⟩'.
- **byfounder**: The expression 'founded by ⟨name⟩'.
- **bycontinuator**: The expression 'continued by ⟨name⟩'.
- **byreconstructor**: The expression 'reconstructed by ⟨name⟩'.
- **byreviser**: The expression 'revised by ⟨name⟩'.
- **byreviewer**: The expression 'reviewed by ⟨name⟩'.
- **bycollaborator**: An expression like 'in collaboration with ⟨name⟩' or 'in cooperation with ⟨name⟩'.
- **bytranslator**: The expression 'translated by ⟨name⟩' or 'translated from ⟨language⟩ by ⟨name⟩'.
- **bycommentator**: The expression 'commented by ⟨name⟩'.
- **byannotator**: The expression 'annotated by ⟨name⟩'.

4.9.2.6 Concatenated Editor Roles, Expressed as Actions

The following keys are similar in function to byeditor, bytranslator, etc. They are used to indicate additional roles of the editor, e.g., 'edited and translated by', 'edited and furnished with an introduction by', 'edited, with a foreword, by'.

- **byeditortr**: Used if editor/translator are identical.
- **byeditorco**: Used if editor/commentator are identical.
- **byeditoran**: Used if editor/annotator are identical.
- **byeditorin**: Used if editor/introduction are identical.
- **byeditorfo**: Used if editor/foreword are identical.
- **byeditoraf**: Used if editor/aftword are identical.
4.9.2.8 Roles, Expressed as Objects

Roles which are related to supplementary material may also be expressed as objects (‘with a commentary by’) rather than as functions (‘commentator’) or as actions (‘commented by’).

The expression ‘with a commentary by <name>’.

The expression ‘with annotations by <name>’.

The expression ‘with an introduction by <name>’.

The expression ‘with a foreword by <name>’.

The expression ‘with an afterword by <name>’.

4.9.2.7 Concatenated Translator Roles, Expressed as Actions

The following keys are similar in function to bytranslator. They are used to indicate additional roles of the translator, e.g., ‘translated and commented by’, ‘translated and furnished with an introduction by’, ‘translated, with a foreword, by’.

The expression ‘with a foreword by’.

The expression ‘with an introduction by’.

The expression ‘with a commentary by’.

The expression ‘with annotations by’.

The expression ‘with a foreword, by’.

Keys for editor/translator/(role) combinations:

- byeditortrco Used if editor/translator/commentator are identical.
- byeditortran Used if editor/translator/annotator are identical.
- byeditortrin Used if editor/translator/introduction are identical.
- byeditortrfo Used if editor/translator/foreword are identical.
- byeditortraf Used if editor/translator/afterword are identical.

Keys for editor/commentator/(role) combinations:

- byeditortrco Used if editor/commentator/introduction are identical.
- byeditortrco Used if editor/commentator/foreword are identical.
- byeditortrco Used if editor/commentator/afterword are identical.

Keys for editor/annotator/(role) combinations:

- byeditoranin Used if editor/annotator/introduction are identical.
- byeditoranfo Used if editor/annotator/foreword are identical.
- byeditoranaf Used if editor/annotator/afterword are identical.

Keys for translator/commentator/(role) combinations:

- bytranslatortrco Used if translator/commentator/introduction are identical.
- bytranslatortrco Used if translator/commentator/foreword are identical.
- bytranslatortrco Used if translator/commentator/afterword are identical.

Keys for translator/annotator/(role) combinations:

- bytranslatoranin Used if translator/annotator/introduction are identical.
- bytranslatoranfo Used if translator/annotator/foreword are identical.
- bytranslatoranaf Used if translator/annotator/afterword are identical.

Keys for translator/(role) combinations:

- bytranslatorco Used if translator/commentator are identical.
- bytranslatoran Used if translator/annotator are identical.
- bytranslatorin Used if translator/introduction are identical.
- bytranslatorfo Used if translator/foreword are identical.
- bytranslatoraf Used if translator/afterword are identical.

4.9.2.8 Roles, Expressed as Objects

Roles which are related to supplementary material may also be expressed as objects (‘with a commentary by’) rather than as functions (‘commentator’) or as actions (‘commented by’).
4.9.2.9 Supplementary Material

The term 'commentary'.

The term 'annotations'.

The term 'introduction'.

The term 'foreword'.

The term 'afterword'.

4.9.2.10 Publication Details

The term 'volume', referring to a book.

The plural form of volume.

The term 'in', as used in expressions like 'in ⟨number of volumes⟩ volumes'.

The term 'book', referring to a journal.

The term 'part', referring to a part of a book or a periodical.

The term 'issue', referring to a periodical.

The expression 'new series', referring to a journal.

The expression 'old series', referring to a journal.

The term 'edition'.

The term 'in', referring to the title of a work published as part of another one, e.g., '⟨title of article⟩ in ⟨title of journal⟩'.

The term 'of', as used in expressions like 'volume ⟨number⟩ of ⟨name of series⟩'.

The term 'number', referring to an issue of a journal.

The term 'chapter', referring to a chapter in a book.

The term 'version', referring to a revision number.

The term 'reprint'.

The expression 'reprint of ⟨title⟩'.

The expression 'reprinted as ⟨title⟩'.

The expression 'reprinted from ⟨title⟩'.

The expression 'translation of ⟨title⟩'.

The expression 'translated as ⟨title⟩'.

The expression 'translated from [the] ⟨language⟩'.

The expression 'review of ⟨title⟩'.

The expression 'originally published as ⟨title⟩'.

The expression 'originally published in ⟨year⟩'.

The term 'as', as used in expressions like 'published by ⟨publisher⟩ as ⟨title⟩'.

The term 'by', as used in expressions like 'published by ⟨publisher⟩'.

4.9.2.11 Publication State

The expression 'in preparation' (the manuscript is being prepared for publication).

The expression 'submitted' (the manuscript has been submitted to a journal or conference).

The expression 'forthcoming' (the manuscript has been accepted by a press or journal).

The expression 'in press' (the manuscript is fully copyedited and out of the author's hands; it is in the final stages of the production process).

The expression 'pre-published' (the manuscript is published in a preliminary form or location, such as online version in advance of print publication).

4.9.2.12 Pagination

The term 'page'.

The plural form of page.

The term 'column', referring to a column on a page.

The plural form of column.

The term 'section', referring to a document division (usually abbreviated as §).

The plural form of section (usually abbreviated as §§).
The term 'paragraph' (i.e., a block of text, not to be confused with section).

The plural form of paragraph.

The term 'verse' as used when referring to a work which is cited by verse numbers.

The plural form of verse.

The term 'line' as used when referring to a work which is cited by line numbers.

The plural form of line.

### 4.9.2.13 Types

The following keys are typically used in the type field of @thesis, @report, @misc, and other entries:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mathesis</td>
<td>An expression equivalent to the term 'Master’s thesis'.</td>
</tr>
<tr>
<td>phdthesis</td>
<td>The term 'PhD thesis', 'PhD dissertation', 'doctoral thesis', etc.</td>
</tr>
<tr>
<td>candthesis</td>
<td>An expression equivalent to the term 'Candidate thesis'. Used for 'Candidate' degrees that have no clear equivalent to the Master’s or doctoral level.</td>
</tr>
<tr>
<td>techreport</td>
<td>The term 'technical report'.</td>
</tr>
<tr>
<td>resreport</td>
<td>The term ‘research report’.</td>
</tr>
<tr>
<td>software</td>
<td>The term 'computer software'.</td>
</tr>
<tr>
<td>datacd</td>
<td>The term 'data cd' or 'cd-rom'.</td>
</tr>
<tr>
<td>audiocd</td>
<td>The term 'audio cd'.</td>
</tr>
</tbody>
</table>

### 4.9.2.14 Miscellaneous

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodate</td>
<td>The term to use in place of a date when there is no date for an entry e.g., ‘n.d.’</td>
</tr>
<tr>
<td>and</td>
<td>The term ‘and’, as used in a list of authors or editors, for example.</td>
</tr>
<tr>
<td>andothers</td>
<td>The expression ‘and others’ or ‘et alii’, used to mark the truncation of a name list.</td>
</tr>
<tr>
<td>andmore</td>
<td>Like andothers but used to mark the truncation of a literal list.</td>
</tr>
</tbody>
</table>

### 4.9.2.15 Labels

The following strings are intended for use as labels, e.g., 'Address: ⟨url⟩' or 'Abstract: ⟨abstract⟩'.

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>The term 'address' in the sense of an internet address.</td>
</tr>
<tr>
<td>urlfrom</td>
<td>An expression like 'available from ⟨url⟩' or 'available at ⟨url⟩'.</td>
</tr>
<tr>
<td>urlseen</td>
<td>An expression like 'accessed on ⟨date⟩', 'retrieved on ⟨date⟩', 'visited on ⟨date⟩', referring to the access date of an online resource.</td>
</tr>
<tr>
<td>file</td>
<td>The term 'file'.</td>
</tr>
<tr>
<td>library</td>
<td>The term 'library'.</td>
</tr>
<tr>
<td>abstract</td>
<td>The term 'abstract'.</td>
</tr>
<tr>
<td>annotation</td>
<td>The term ‘annotations’.</td>
</tr>
</tbody>
</table>

### 4.9.2.16 Citations

Traditional scholarly expressions used in citations:

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idem</td>
<td>The term equivalent to the Latin ‘idem’ ('the same [person]').</td>
</tr>
<tr>
<td>idemsf</td>
<td>The feminine singular form of idem.</td>
</tr>
<tr>
<td>idemsm</td>
<td>The masculine singular form of idem.</td>
</tr>
<tr>
<td>idemsn</td>
<td>The neuter singular form of idem.</td>
</tr>
<tr>
<td>idempf</td>
<td>The feminine plural form of idem.</td>
</tr>
<tr>
<td>idempm</td>
<td>The masculine plural form of idem.</td>
</tr>
<tr>
<td>idempn</td>
<td>The neuter plural form of idem.</td>
</tr>
<tr>
<td>idempfp</td>
<td>The plural form of idem suitable for a mixed gender list of names.</td>
</tr>
<tr>
<td>ibidem</td>
<td>The term equivalent to the Latin ‘ibidem’ ('in the same place').</td>
</tr>
<tr>
<td>opcit</td>
<td>The term equivalent to the Latin term 'opere citato' ('[in] the work [already] cited').</td>
</tr>
<tr>
<td>loccit</td>
<td>The term equivalent to the Latin term 'loco citato' ('[at] the place [already] cited').</td>
</tr>
<tr>
<td>confer</td>
<td>The term equivalent to the Latin ‘confer’ ('compare').</td>
</tr>
<tr>
<td>sequens</td>
<td>The term equivalent to the Latin ‘sequens’ ('[and] the following [page]'), as used to indicate a range of two pages when only the starting page is provided (e.g., ‘25 sq.’ or ‘25 f.’ instead of ‘25–26’).</td>
</tr>
<tr>
<td>sequentes</td>
<td>The term equivalent to the Latin ‘sequentes’ ('[and] the following [pages]'), as used to indicate an open-ended range of pages when only the starting page is provided (e.g., ‘25 sq.’ or ‘25 ff.’).</td>
</tr>
<tr>
<td>passim</td>
<td>The term equivalent to the Latin ‘passim’ ('throughout', 'here and there', 'scatteredly').</td>
</tr>
</tbody>
</table>
Other expressions frequently used in citations:

**see** The term 'see'.
**seealso** The expression 'see also'.
**seenote** An expression like 'see note (footnote)' or 'as in (footnote)', used to refer to a previous footnote in a citation.

**backrefpage** An expression like 'see page (page)' or 'cited on page (page)', used to introduce back references in the bibliography.

**backrefpages** The plural form of backrefpage, e.g., 'see pages (pages)' or 'cited on pages (pages)'.

**quotedin** An expression like 'quoted in (citation)', used when quoting a passage which was already a quotation in the cited work.

**citedas** An expression like 'henceforth cited as (shorthand)', used to introduce a shorthand in a citation.

**thiscite** The expression used in some verbose citation styles to differentiate between the page range of the cited item (typically an article in a journal, collection, or conference proceedings) and the page number the citation refers to. For example: "Author, Title, in: Book, pp. 45–61, thiscite p. 52."

### 4.9.2.17 Month Names

- **january** The name 'January'.
- **february** The name 'February'.
- **march** The name 'March'.
- **april** The name 'April'.
- **may** The name 'May'.
- **june** The name 'June'.
- **july** The name 'July'.
- **august** The name 'August'.
- **september** The name 'September'.
- **october** The name 'October'.
- **november** The name 'November'.
- **december** The name 'December'.

### 4.9.2.18 Language Names

- **langamerican** The language 'American' or 'American English'.
- **langbrazilian** The language 'Brazilian' or 'Brazilian Portuguese'.
- **langcatalan** The language 'Catalan'.
- **langcroatian** The language 'Croatian'.
- **langczech** The language 'Czech'.
- **langdanish** The language 'Danish'.
- **langdutch** The language 'Dutch'.
- **langenglish** The language 'English'.
- **langestonian** The language 'Estonian'.
- **langfinnish** The language 'Finnish'.
- **langfrench** The language 'French'.
- **langgerman** The language 'German'.
- **langgreek** The language 'Greek'.
- **langitalian** The language 'Italian'.
- **langjapanese** The language 'Japanese'.
- **langlatin** The language 'Latin'.
- **langnorwegian** The language 'Norwegian'.
- **langpolish** The language 'Polish'.
- **langportuguese** The language 'Portuguese'.
- **langrussian** The language 'Russian'.
- **langslovene** The language 'Slovene'.
- **langslovak** The language 'Slovak'.
- **langslovene** The language 'Slovene'.
- **langspanish** The language 'Spanish'.
- **langswedish** The language 'Swedish'.
The following strings are intended for use in phrases like 'translated from [the] English by ⟨translator⟩':

| fromamerican | The expression 'from [the] American' or 'from [the] American English'. |
| frombrazilian | The expression 'from [the] Brazilian' or 'from [the] Brazilian Portuguese'. |
| fromcatalan | The expression 'from [the] Catalan'. |
| fromcroatian | The expression 'from [the] Croatian'. |
| fromczech | The expression 'from [the] Czech'. |
| fromdanish | The expression 'from [the] Danish'. |
| fromdutch | The expression 'from [the] Dutch'. |
| fromenglish | The expression 'from [the] English'. |
| fromestonian | The expression 'from [the] Estonian'. |
| fromfinnish | The expression 'from [the] Finnish'. |
| fromfrench | The expression 'from [the] French'. |
| fromgerman | The expression 'from [the] German'. |
| fromgreek | The expression 'from [the] Greek'. |
| fromitalian | The expression 'from [the] Italian'. |
| fromjapanese | The expression 'from [the] Japanese'. |
| fromlatin | The expression 'from [the] Latin'. |
| fromnorwegian | The expression 'from [the] Norwegian'. |
| frompolish | The expression 'from [the] Polish'. |
| fromportuguese | The expression 'from [the] Portuguese'. |
| fromrussian | The expression 'from [the] Russian'. |
| fromslovak | The expression 'from [the] Slovak'. |
| fromslovene | The expression 'from [the] Slovene'. |
| fromspanish | The expression 'from [the] Spanish'. |
| fromswedish | The expression 'from [the] Swedish'. |

**4.9.2.19 Country Names**  Country names are localised by using the string country plus the iso-3166 country code as the key. The short version of the translation should be the iso-3166 country code. Note that only a small number of country names is defined by default, mainly to illustrate this scheme. These keys are used in the location list of @patent entries but they may be useful for other purposes as well.

| countryde | The name 'Germany', abbreviated as DE. |
| countryeu | The name 'European Union', abbreviated as EU. |
| countryep | Similar to countryeu but abbreviated as EP. This is intended for patent entries. |
| countryfr | The name 'France', abbreviated as FR. |
| countryuk | The name 'United Kingdom', abbreviated (according to iso-3166) as GB. |
| countryus | The name 'United States of America', abbreviated as US. |

**4.9.2.20 Patents and Patent Requests**  Strings related to patents are localised by using the term patent plus the iso-3166 country code as the key. Note that only a small number of patent keys is defined by default, mainly to illustrate this scheme. These keys are used in the type field of @patent entries.

| patent | The generic term 'patent'. |
| patentde | The expression 'German patent'. |
| patenteu | The expression 'European patent'. |
| patentfr | The expression 'French patent'. |
| patentuk | The expression 'British patent'. |
| patentus | The expression 'U.S. patent'. |

Patent requests are handled in a similar way, using the string patreq as the base name of the key:

| patreq | The generic term 'patent request'. |
| patreqde | The expression 'German patent request'. |
| patreqeu | The expression 'European patent request'. |
| patreqfr | The expression 'French patent request'. |
| patrequk | The expression 'British patent request'. |
| patrequs | The expression 'U.S. patent request'. |
4.9.2.21 Dates and Times

Abbreviation strings for standard eras. Both secular and Christian variants are supported.

- **commonera** The era ‘CE’
- **beforecommonera** The era ‘BCE’
- **annodomini** The era ‘AD’
- **beforechrist** The era ‘BC’

Abbreviation strings for ‘circa’ dates:

- **circa** The string ‘circa’

Abbreviation strings for seasons parsed from EDTF dates:

- **spring** The string ‘spring’
- **summer** The string ‘summer’
- **autumn** The string ‘autumn’
- **winter** The string ‘winter’

Abbreviation strings for AM/PM:

- **am** The string ‘AM’
- **pm** The string ‘PM’

4.10 Formatting Commands

This section corresponds to §3.10 in the user part of this manual. Bibliography and citation styles should incorporate the commands and facilities discussed in this section in order to provide a certain degree of high-level configurability. Users should not be forced to write new styles if all they want to do is modify the spacing in the bibliography or the punctuation used in citations.

4.10.1 User-definable Commands and Hooks

This section corresponds to §3.10.1 in the user part of the manual. The commands and hooks discussed here are meant to be redefined by users, but bibliography and citation styles may provide a default definition which is different from the package default. These commands are defined in `biblatex.def`. Note that all commands starting with `\mk...` take one mandatory argument.

- **\bibnamedelima** This delimiter controls the spacing between the elements which make up a name part. It is inserted automatically by the backend after the first name element if the element is less than three characters long and before the last element. The default definition is `\addhighpenspace`, i.e., a space penalized by the value of the `highnamepenalty` counter (§3.10.4). Please refer to §3.13.4 for further details.

- **\bibnamedelimb** This delimiter controls the spacing between the elements which make up a name part. It is inserted automatically by the backend between all name elements where `\bibnamedelima` does not apply. The default definition is `\addlowpenspace`, i.e., a space penalized by the value of the `lownamepenalty` counter (§3.10.4). Please refer to §3.13.4 for further details.

- **\bibnamedelimc** This delimiter controls the spacing between name parts. The default name formats use it between the name prefix and the last name if `useprefix=true`. The default definition is `\addhighpenspace`, i.e., a space penalized by the value of the `highnamepenalty` counter (§3.10.4). Please refer to §3.13.4 for further details.

- **\bibnamedelimd** This delimiter controls the spacing between name parts. The default name formats use it between all name parts where `\bibnamedelimc` does not apply. The default definition is `\addlowpenspace`, i.e., a space penalized by the value of the `lownamepenalty` counter (§3.10.4). Please refer to §3.13.4 for further details.

- **\bibnamedelimi** This delimiter replaces `\bibnamedelima/b` after initials. Note that this only applies to initials given as such in the `bib` file, not to the initials automatically generated by `biblatex` which use their own set of delimiters.

- **\bibinitperiod** The punctuation inserted automatically by the backend after all initials unless `\bibinithyphenendelim` applies. The default definition is a period (`\adddot`). Please refer to §3.13.4 for further details.

- **\bibinitdelim** The spacing inserted automatically by the backend between multiple initials unless `\bibinithyphenendelim` applies. The default definition is an unbreakable interword space. Please refer to §3.13.4 for further details.
\bibinithyphendelim The punctuation inserted automatically by the backend between the initials of hyphenated name parts, replacing \bibinitperiod and \bibinitdelim. The default definition is a period followed by an unbreakable hyphen. Please refer to §3.13.4 for further details.

\bibindexnamedelim Replaces \bibnamedelim in the index.
\bibindexnamedelimb Replaces \bibnamedelimib in the index.
\bibindexnamedelimc Replaces \bibnamedelimic in the index.
\bibindexnamedelimd Replaces \bibnamedelimid in the index.
\bibindexnamedelimi Replaces \bibnamedelimi in the index.
\bibinitperiod Replaces \bibinitperiod in the index.
\bibinitdelim Replaces \bibinitdelim in the index.
\bibinitperioddelim Replaces \bibinitperioddelim in the index.
\bibinitdelimdelim Replaces \bibinitdelimdelim in the index.
\bibinitdelimperiod Replaces \bibinitdelimperiod in the index.
\bibinitdeliminit Replaces \bibinitdeliminit in the index.
\revsdnamepunct The punctuation to be printed between the first and last name parts when a name is reversed. The default is a comma. This command should be incorporated in formatting directives for name lists. Please refer to §3.13.4 for further details.
\bibnamedash The dash to be used as a replacement for recurrent authors or editors in the bibliography. The default is an 'em' or an 'en' dash, depending on the indentation of the list of references.
\labelnamepunct The separator to be printed after the name used for alphabetizing in the bibliography (author or editor, if the author field is undefined). Use this separator instead of \newunitpunct at this location. The default is \newunitpunct, i.e., it is not handled differently from regular unit punctuation but permits convenient reconfiguration.
\subtitlepunct The separator to be printed between the fields title and subtitle, booktitle and booksubtitle, as well as maintitle and mainsubtitle. Use this separator instead of \newunitpunct at this location. The default is \newunitpunct, i.e., it is not handled differently from regular unit punctuation but permits convenient reconfiguration.
\intitlepunct The separator to be printed between the word "in" and the following title in entry types such as @article, @inbook, @incollection, etc. Use this separator instead of \newunitpunct at this location. The default is a colon plus an interword space.
\bibpagespunct The separator to be printed before the pages field. Use this separator instead of \newunitpunct at this location. The default is a comma plus an interword space.
\bibpagemorepunct The separator to be printed before the pageref field. Use this separator instead of \newunitpunct at this location. The default is an interword space.
\multinamedelim The delimiter to be printed between multiple items in a name list like author or editor if there are more than two names in the list. If there are only two names in the list, use the \finalnamedelim instead. This command should be incorporated in all formatting directives for name lists.
\finalnamedelim Use this command instead of \multinamedelim before the final name in a name list.
\revsdnamelim The extra delimiter to be printed after the first name in a name list consisting of two names (in addition to \finalnamedelim) if the first name is reversed. This command should be incorporated in all formatting directives for name lists.
\andsdelim The delimiter to be printed before the localisation string 'and others' if a name list like author or editor is truncated. This command should be incorporated in all formatting directives for name lists.
\multilistdelim The delimiter to be printed between multiple items in a literal list like publisher or location if there are more than two names in the list. If there are only two items in the list, use the \finallistdelim instead. This command should be incorporated in all formatting directives for literal lists.
\finallistdelim Use this command instead of \multilistdelim before the final item in a literal list.
\andmoredelim The delimiter to be printed before the localisation string 'and more' if a literal list like publisher or location is truncated. This command should be incorporated in all formatting directives for literal lists.
\multicitedelim The delimiter printed between citations if multiple entry keys are passed to a single citation command. This command should be incorporated in the definition of all citation commands, for example in the (separate) argument passed to \DeclareCiteCommand. See §4.3.1 for details.
\supercitedelim Similar to \multicitedelim, but intended for the \supercite command only.
\compcitedelim Similar to \multicitedelim, but intended for citation styles that 'compress' multiple citations, i.e., print the author only once if subsequent citations share the same author etc.
\textcitedelim  Similar to \multicitedelim, but intended for \textcite and related commands (§ 3.8.2).
\nametitledelim  The delimiter to be printed between the author/editor and the title. This command should be incorporated in the definition of all citation commands of author-title and some verbose citation styles.
\nameyeardelim  The delimiter to be printed between the author/editor and the year. This command should be incorporated in the definition of all citation commands of author-year citation styles.
\namelabeldelim  The delimiter printed between the name/title and the label. This command should be incorporated in the definition of all citation commands of alphabetic and numeric citation styles.
\nonameyeardelim  The delimiter printed between the substitute for the labelname when it does not exist (usually the label or title in standard styles) and the year in author-year citation styles. This is only used when there is no labelname since when the labelname exists, \nameyeardelim is used.
\volcitedelim  The delimiter to be printed between the volume portion and the page/text portion of \volcite and related commands (§ 3.8.6).
\prenotedelim  The delimiter to be printed after the (prenote) argument of a citation command.
\postnotedelim  The delimiter to be printed after the (postnote) argument of a citation command.
\extpostnotedelim  The delimiter printed between the citation and the parenthetical (postnote) argument of a citation command when the postnote occurs outside of the citation parentheses. In the standard styles, this occurs when the citation uses the shorthand field of the entry.
\mkbibnamefamily\{\text\}\} Formatting hook for the family name, to be used in all formatting directives for name lists.
\mkbibnamegiven\{\text\}\} Similar to \mkbibnamefamily, but intended for the given name.
\mkbibnameprefix\{\text\}\} Similar to \mkbibnamefamily, but intended for the name prefix.
\mkbibnamesuffix\{\text\}\} Similar to \mkbibnamefamily, but intended for the name suffix.
\relatedpunct  The separator between the relatedtype bibliography localisation string and the data from the first related entry.
\relateddelim  The separator between the data of multiple related entries. The default definition is a linebreak.
\relateddelim<\relatedtype>  The separator between the data of multiple related entries inside related entries of type ‘relatedtype’. There is no default, if such a type-specific delimiter does not exist, \relateddelim is used.

4.10.2 Language-specific Commands

This section corresponds to § 3.10.3 in the user part of the manual. The commands discussed here are usually handled by the localisation modules, but may also be redefined by users on a per-language basis. Note that all commands starting with \mkbib... take one or more mandatory arguments.

\bibrangedash  The language specific range dash. Defaults to \textendash.
\ibrangessep  The language specific separator to be used between multiple ranges. Defaults to a comma followed by a space.
\bibdatesep  The language specific separator used between date components in terse date formats. Defaults to \hyphen.
\bibdaterangesep  The language specific separator to be used for date ranges. Defaults to \textendash for all date formats apart from ymd which defaults to a \textbackslash slash. The date format option edtf is hard-coded to \slash since this is a standards compliant format.
\mkbibdatelong  Takes the names of three field as arguments which correspond to three date components (in the order year/month/day) and uses their values to print the date in the language specific long date format.
\mkbibdateshort  Similar to \mkbibdatelong but using the language specific short date format.
\mkbibtimezone  Modifies a timezone string passed in as the only argument. By default this changes ‘Z’ to the value of \bibtimezone.
\bibdateuncertain  The language specific marker to be used after uncertain dates when the global option dateuncertain is enabled. Defaults to a space followed by a question mark.
\bibdateeraprefix  The language specific marker which is printed as a prefix to beginning BCE/BC dates in a date range when the option dateera is set to ‘astronomical’. Defaults to \textminus, if defined and \textendash otherwise.
\bibdateeraendprefix  The language specific marker which is printed as a prefix to end BCE/BC dates in a date range when the option dateera is set to ‘astronomical’. Defaults to a thin space followed by \bibdateeraprefix when \bibdaterangesep is set to a dash and to \bibdateeraprefix otherwise. This is a separate macro so that you may add extra space before a negative date marker which, for example follows a dash date range marker as this can look a little odd.
\bibtimesep  The language specific marker which separates time components. Default to a colon.
\butctimezone  The language specific string printed for the UTC timezone. Defaults to ‘Z’.
\btimezonesep  The language specific marker which separates an optional time zone component from a time. Empty by default.
\datetimesep  The language specific separator printed between date and time components when printing time components along with date components (see the \dateusetime option in §3.1.2.1). Defaults to a space for non-EDTF output formats, and 'T' for EDTF output format.
\finalandcomma  Prints the comma to be inserted before the final ‘and’ in an enumeration, if applicable in the respective language.
\finalandsemicolon  Prints the semicolon to be inserted before the final ‘and’ in an enumeration, if applicable in the respective language.
\mkbibordinal{\langle integer\rangle}  Takes an integer argument and prints it as an ordinal number.
\mkbibmascord{\langle integer\rangle}  Similar to \mkbibordinal, but prints a masculine ordinal, if applicable in the respective language.
\mkbibfemord{\langle integer\rangle}  Similar to \mkbibordinal, but prints a feminine ordinal, if applicable in the respective language.
\mkbibneutord{\langle integer\rangle}  Similar to \mkbibordinal, but prints a neuter ordinal, if applicable in the respective language.
\mkbibordedition{\langle integer\rangle}  Similar to \mkbibordinal, but intended for use with the term ‘edition’.
\mkbibordseries{\langle integer\rangle}  Similar to \mkbibordinal, but intended for use with the term ‘series’.

4.10.3 User-definable Lengths and Counters

This section corresponds to §3.10.4 in the user part of the manual. The length registers and counters discussed here are meant to be altered by users. Bibliography and citation styles should incorporate them where applicable and may also provide a default setting which is different from the package default.

\bibhang  The hanging indentation of the bibliography, if applicable. This length is initialized to \parindent at load-time. If \parindent is zero length for some reason, \bibhang will default to 1em.
\labelsep  The horizontal space between entries and their corresponding labels. Bibliography styles which use list environments and print a label should set \labelsep to \biblabelsep in the definition of the respective environment.
\itemsep  The vertical space between the individual entries in the bibliography. Bibliography styles using list environments should set \itemsep to \bibitemsep in the definition of the respective environment.
\parsep  The vertical space between paragraphs within an entry in the bibliography. Bibliography styles using list environments should set \parsep to \bibparsep in the definition of the respective environment.
\urlnumpenalty  If this counter is set to a value greater than zero, \biblatex will permit linebreaks after numbers in all strings formatted with the \url command from the url package. This will affect urls and dois in the bibliography. The breakpoints will be penalized by the value of this counter. If urls and/or dois in the bibliography run into the margin, try setting this counter to a value greater than zero but less than 10000 (you normally want to use a high value like 9000). Setting the counter to zero disables this feature. This is the default setting.
\urlucpenalty  Similar to \urlnumpenalty, except that it will add a breakpoint after all uppercase letters.
4.10.4 Auxiliary Commands and Hooks

The auxiliary commands and facilities in this section serve a special purpose. Some of them are used by \biblatex to communicate with bibliography and citation styles in some way or other.

\mkbibemph{(text)}

A generic command which prints its argument as emphasized text. This is a simple wrapper around the standard \textbf{emph} command. Apart from that, it uses \setpunctfont from §4.7.1 to adapt the font of the next punctuation mark following the text set in italics. If the punctfont package option is disabled, this command behaves like \textbf{emph}.

\mkbibitalic{(text)}

Similar in concept to \mkbibemph but prints italicized text. This is a simple wrapper around the standard \textit command which incorporates \setpunctfont. If the punctfont package option is disabled, this command behaves like \textit).

\mkbibbold{(text)}

Similar in concept to \mkbibemph but prints bold text. This is a simple wrapper around the standard \textbf command which incorporates \setpunctfont. If the punctfont package option is disabled, this command behaves like \textbf.

\mkbibquote{(text)}

A generic command which wraps its argument in quotation marks. If the csquotes package is loaded, this command uses the language sensitive quotation marks provided by that package. \mkbibquote also supports ‘American-style’ punctuation, see \DeclareQuotePunctuation in §4.7.5 for details.

\mkbibparens{(text)}

A generic command which wraps its argument in parentheses. This command is nestable. When nested, it will alternate between parentheses and brackets, depending on the nesting level.

\mkbibbrackets{(text)}

A generic command which wraps its argument in square brackets. This command is nestable. When nested, it will alternate between brackets and parentheses, depending on the nesting level.

\bibopenparen{\text} \bibcloseparen

Alternative syntax for \mkbibparens. This will also work across groups. Note that \bibopenparen and \bibcloseparen must always be balanced.

\bibopenbracket{\text} \bibclosebracket

Alternative syntax for \mkbibbrackets. This will also work across groups. Note that \bibopenbracket and \bibclosebracket must always be balanced.

\mkbibfootnote{(text)}

A generic command which prints its argument as a footnote. This is a wrapper around the standard \LaTeX \texttt{footnote} command which removes spurious whitespace preceding the footnote mark and prevents nested footnotes. By default, \mkbibfootnote requests capitalization at the beginning of the note and automatically adds a period at the end. You may change this behavior by redefining the \bibfootnotewrapper macro introduced below.

\mkbibfootnotetext{(text)}

Similar to \mkbibfootnote but uses the \texttt{footnotetext} command.

\mkbibendnote{(text)}

Similar in concept to \mkbibfootnote except that it prints its argument as an endnote. \mkbibendnote removes spurious whitespace preceding the endnote mark and prevents nested notes. It supports the \texttt{endnote} command provided by the endnotes package as well as the \texttt{pagenote} command provided by the pagenote package and the \texttt{memoir} class. If both commands are available, \texttt{endnote} takes precedence. If no endnote support is available, \mkbibendnote issues an error and falls back to \texttt{footnote}. By default, \mkbibendnote requests capitalization at the beginning of the note and automatically adds a period at the end. You may change this behavior by redefining the \bibendnotewrapper macro introduced below.

\mkbibendnotetext{(text)}

Similar to \mkbibendnote but uses the \texttt{endnotetext} command. Please note that as of this writing, neither the pagenote package nor the memoir class provide a corresponding \texttt{pagenotetext} command. In this case, \mkbibendnote will issue an error and fall back to \texttt{footnotetext}.
\bibfootnotewrapper{\text}

An inner wrapper which encloses the \text argument of \mkbibfootnote and \mkbibfootnotetext. For example, \mkbibfootnote eventually boils down to this:

\footnote{\bibfootnotewrapper{\text}}

The wrapper ensures capitalization at the beginning of the note and adds a period at the end. The default definition is:

\newcommand{\bibfootnotewrapper}[1]{\bibsentence #1\addperiod}

If you don’t want capitalization at the beginning or a period at the end of the note, do not modify \mkbibfootnote but redefine \bibfootnotewrapper instead.

\bibendnotewrapper{\text}

Similar in concept to \bibfootnotewrapper but related to the \mkbibendnote and \mkbibendnotetext commands.

\mkbibsuperscript{\text}

A generic command which prints its argument as superscripted text. This is a simple wrapper around the standard LaTeX \textsuperscript command which removes spurious whitespace and allows hyphenation of the preceding word.

\mkbibmonth{\text}

This command takes an integer argument and prints it as a month name. Even though the output of this command is language specific, its definition is not, hence it is normally not redefined in localisation modules.

\mkbibseason{\text}

This command takes a season localisation string and prints the version of the string corresponding to the setting of the dateabbrev package option. Even though the output of this command is language specific, its definition is not, hence it is normally not redefined in localisation modules.

\mkyearzeros{\text}

This command strips leading zeros from a year or enforces them, depending on the datezeros package option (§ 3.1.2.1). It is intended for use in the definition of date formatting macros.

\mkmonthzeros{\text}

This command strips leading zeros from a month or enforces them, depending on the datezeros package option (§ 3.1.2.1). It is intended for use in the definition of date formatting macros.

\mkdayzeros{\text}

This command strips leading zeros from a day or enforces them, depending on the datezeros package option (§ 3.1.2.1). It is intended for use in the definition of date formatting macros.

\mtimezeros{\text}

This command strips leading zeros from a number or preserves them, depending on the timezeros package option (§ 3.1.2.1). It is intended for use in the definition of time formatting macros.

\forcezerosy{\text}

This command adds zeros to a year (or any number supposed to be 4-digits). It is intended for date formatting and ordinals.

\forcezerosmdt{\text}

This command adds zeros to a month, day or time part (or any number supposed to be 2-digits). It is intended for date/time formatting and ordinals.

\stripzeros{\text}

This command strips leading zeros from a number. It is intended for date formatting and ordinals.

For every field marked as a 'Label field' in the data model, a formatting directive is created as per shorthandwidth above. Since shorthand is so marked in the default data model, this functionality is a superset of that described for shorthandwidth.
4.10.5 Auxiliary Lengths, Counters, and Other Features

The length registers and counters discussed here are used by biblatex to pass information to bibliography and citation styles. Think of them as read-only registers. Note that all counters are LaTeX counters. Use \value{counter} to read out the current value.

\labelnumberwidth For every field marked as a 'label' field in the data model, a length register is created as per shorthandwidth above. Since shorthand is so marked in the default data model, this functionality is a superset of that described for shorthandwidth.

\labelnumberwidth This length register indicates the width of the widest \labelnumber. Numeric bibliography styles should incorporate this length in the definition of the bibliography environment.

\labelalpha This length register indicates the width of the widest \labelalpha. Alphabetic bibliography styles should incorporate this length in the definition of the bibliography environment.

\maxextraalpha This counter holds the highest number found in any extralabel field.

\maxextrayear This counter holds the highest number found in any extrayear field.

\reftsegment This counter indicates the current \reftsegment environment. When queried in a bibliography heading, the counter returns the value of the \reftsegment option passed to \printbibliography.

\refsection This counter indicates the current \refsection environment. When queried in a bibliography heading, the counter returns the value of the \refsection option passed to \printbibliography.

\maxnames This counter holds the setting of the \maxnames package option.

\minnames This counter holds the setting of the \minnames package option.

\maxitems This counter holds the setting of the \maxitems package option.

\minitems This counter holds the setting of the \minitems package option.

\instcount This counter is incremented by biblatex for every citation as well as for every entry in the bibliography and bibliography lists. The value of this counter uniquely identifies a single instance of a reference in the document.
\AtBeginBibliography{(code)}

Appends the (code) to an internal hook executed at the beginning of the bibliography. The (code) is executed at the beginning of the list of references, immediately after the (begin code) of \defbibenvironment. This command may only be used in the preamble.

\AtBeginShorthands{(code)}

Appends the (code) to an internal hook executed at the beginning of the list of shorthands. The (code) is executed at the beginning of the list of shorthands, immediately after the (begin code) of \defbibenvironment. This command may only be used in the preamble.

This is just an alias for:

\AtBeginBiblist{shorthand}(code)

\AtBeginBibliography{(biblistname)}{(code)}

Appends the (code) to an internal hook executed at the beginning of the bibliography list (biblistname). The (code) is executed at the beginning of the bibliography list, immediately after the (begin code) of \defbibenvironment. This command may only be used in the preamble.
\AtEveryBibitem{(code)}
Appends the (code) to an internal hook executed at the beginning of every item in the bibliography. The (code) is executed immediately after the (item code) of \defbibenvironment. The bibliographic data of the respective entry is available at this point. This command may only be used in the preamble.

\AtEveryLositem{(code)}
Appends the (code) to an internal hook executed at the beginning of every item in the list of short-hands. The (code) is executed immediately after the (item code) of \defbibenvironment. The bibliographic data of the respective entry is available at this point. This command may only be used in the preamble.

This is just an alias for:
\AtEveryBiblistitem{shorthand}{code}
\AtEveryBiblistitem{biblistname}{code}

\AtNextBibliography{(code)}
Similar to \AtBeginBibliography but only affecting the next \printbibliography. The internal hook is cleared after being executed once. This command may be used in the document body.

\AtEveryCite{(code)}
Appends the (code) to an internal hook executed at the beginning of every citation command. The (code) is executed immediately before the (precode) of the command (see §4.3.1). No bibliographic data is available at this point. This command may only be used in the preamble.

\AtEveryCitekey{(code)}
Appends the (code) to an internal hook executed once for every entry key passed to a citation command. The (code) is executed immediately before the (loopcode) of the command (see §4.3.1). The bibliographic data of the respective entry is available at this point. This command may only be used in the preamble.

\AtEveryMultiCite{(code)}
Appends the (code) to an internal hook executed at the beginning of every multicite command. The (code) is executed immediately before the multiprenote field (§4.3.2) is printed. No bibliographic data is available at this point. This command may only be used in the preamble.

\AtNextCite{(code)}
Similar to \AtEveryCite but only affecting the next citation command. The internal hook is cleared after being executed once. This command may be used in the document body.

\AtEachCitekey{(code)}
Similar to \AtEveryCitekey but only affecting the current citation command. This command may be used in the document body. The (code) is appended to the internal hook locally when located in a citation, as determined by \ifcitation.

\AtNextCitekey{(code)}
Similar to \AtEveryCitekey but only affecting the next entry key. The internal hook is cleared after being executed once. This command may be used in the document body.

\AtNextMultiCite{(code)}
Similar to \AtEveryMultiCite but only affecting the next multicite command. The internal hook is cleared after being executed once. This command may be used in the document body.

\AtDataInput{entrytype}{(code)}
Appends the (code) to an internal hook executed once for every entry as the bibliographic data is imported from the bbl file. The (entrytype) is the entry type the (code) applies to. If it applies to all entry types, omit the optional argument. The (code) is executed immediately after the entry has been imported. This command may only be used in the preamble. Note that (code) may be executed multiple times for an entry. This occurs when the same entry is cited in different refsection environments or the sorting option settings incorporate more than one sorting scheme. The refsection counter holds the number of the respective reference section while the data is imported.
\UseBibitemHook
Executes the internal hook corresponding to \AtEveryBibitem.

\UseEveryCiteHook
Executes the internal hook corresponding to \AtEveryCite.

\UseEveryCitekeyHook
Executes the internal hook corresponding to \AtEveryCitekey.

\UseEveryMultiCiteHook
Executes the internal hook corresponding to \AtMultiEveryCite.

\UseNextCiteHook
Executes and clears the internal hook corresponding to \AtNextCite.

\UseNextCitekeyHook
Executes and clears the internal hook corresponding to \AtNextCitekey.

\UseNextMultiCiteHook
Executes and clears the internal hook corresponding to \AtNextMultiCite.

\DeferNextCitekeyHook
Locally un-defines the internal hook specified by \AtNextCitekey. This essentially defers the
hook to the next entry key in the citation list, when executed in the \langle \texttt{precode} \rangle
argument of \DeclareCiteCommand (§ 4.3.1).

4.11 Hints and Caveats
This section provides some additional hints concerning the author interface of this package. It also
addresses common problems and potential misconceptions.

4.11.1 Entry Sets
Entry sets have already been introduced in §3.12.5. This section discusses how to process entry sets in a
bibliography style. From the perspective of the driver, there is no difference between static and dynamic
entry sets. Both types are handled in the same way. You will normally use the \entryset command
from §4.4.1 to loop over all set members (in the order in which they are listed in the \entryset field
of the @set entry, or in the order in which they were passed to \defbibentryset, respectively)
and append \finentry at the end. That’s it. The formatting is handled by the drivers for the entry
types of the individual set members:

\DeclareBibliographyDriver{set}{%}
  \entryset{}{}
  \finentry}

You may have noticed that the numeric styles which ship with this package support subdivided entry
sets, i.e., the members of the set are marked with a letter or some other marker such that citations may
either refer to the entire set or to a specific set member. The markers are generated as follows by the
bibliography style:

\DeclareBibliographyDriver{set}{%}
  \entryset
    {\printfield{entrysetcount}%
     \setunit*{\addnbspace}}%
  \finentry

The \entrysetcount field holds an integer indicating the position of a set member in the entry set.
The conversion of this number to a letter or some other marker is handled by the formatting directive
of the \entrysetcount field. All the driver needs to do is print the field and add some white space
(or start a new line). Printing the markers in citations works in a similar way. Where a numeric style
normally says \printfield{labelnumber}, you simply append the \entrysetcount field:
Since this field is only defined when processing citations referring to a set member, there is no need to add any additional tests.

4.11.2 Electronic Publishing Information

The standard styles feature dedicated support for arXiv references. Support for other resources is easily added. The standard styles handle the eprint field as follows:

\iffieldundef{eprinttype}
  {\printfield{eprint}}
  {\printfield[eprint:\strfield{eprinttype}]{eprint}}

If an eprinttype field is available, the above code tries to use the field format eprint:(<eprinttype>). If this format is undefined, \printfield automatically falls back to the field format eprint. There are two predefined field formats, the type-specific format eprint:arxiv and the fallback format eprint:

\DeclareFieldFormat{eprint}{...}
\DeclareFieldFormat{eprint:arxiv}{...}

In other words, adding support for additional resources is as easy as defining a field format named eprint:⟨resource⟩ where ⟨resource⟩ is an identifier to be used in the eprinttype field.

4.11.3 External Abstracts and Annotations

External abstracts and annotations have been discussed in §3.12.8. This section provides some more background for style authors. The standard styles use the following macros (from biblatex.def) to handle abstracts and annotations:

\newbibmacro*{annotation}{%  \iffieldundef{annotation}
  {\printfile{annotation}\bibannotationprefix\thefield{entrykey}.↪→tex}}%
  {\printfield{annotation}}}
\newcommand*{\bibannotationprefix}{bibannotation-}
\newbibmacro*{abstract}{%  \iffieldundef{abstract}
  {\printfile{abstract}\bibabstractprefix\thefield{entrykey}.\tex↪→}}%
  {\printfield{abstract}}}
\newcommand*{\bibabstractprefix}{bibabstract-}

If the abstract/annotation field is undefined, the above code tries to load the abstracts/annotations from an external file. The \printfile commands also incorporate file name prefixes which may be redefined by users. Note that you must enable \printfile explicitly by setting the loadfiles package option from §3.1.2.1. This feature is disabled by default for performance reasons.

4.11.4 Name Disambigation

The uniquename and uniquelist options introduced in §3.1.2.3 support various modes of operation. This section explains the differences between these modes by way of example. The uniquename option disambiguates individual names in the labelname list. The uniquelist option disambiguates the labelname list if it has become ambiguous after maxnames/minnames truncation. You can use either option stand-alone or combine both.

Name disambiguation works by taking a ‘base’ which is composed of one or more nameparts and then determining what needs to be added, if anything, to this ‘base’ to make the name unique in the
current refsection. Name disambiguation is controlled by the uniquename template declared with the following command:

\DeclareUniquenameTemplate{⟨specification⟩}

The ⟨specification⟩ is a list of \namepart commands which define the nameparts to use in determining the uniquename information.

\namepart[⟨options⟩]{⟨namepart⟩}

⟨namepart⟩ is one of the datamodel nameparts defined with the \DeclareDatamodelConstant command (see §4.2.3). The options are:

use=true, false default: false

Only use the ⟨namepart⟩ in constructing the uniquename information if there is a corresponding option use='namepart' and that option is true.

base=true, false default: false

The ⟨namepart⟩ is part of the 'base' which is the main piece of namepart(s) information which is being disambiguated by uniqueness information.

The default uniquename template is:

\DeclareUniquenameTemplate{
 \namepart[use=true, base=true]{prefix}
 \namepart[base=true]{family}
 \namepart{given}
}

This means that the 'base' to be disambiguated consists of the 'family' namepart, along with any prefix, if the useprefix option is true. The disambiguation is performed by adding aspects of any non 'base' nameparts in the specification, here just the 'given' namepart.

### 4.11.4.1 Individual Names (uniquename)

Let's start off with some uniquename examples. Consider the following data:

<table>
<thead>
<tr>
<th>Name</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe</td>
<td>2008</td>
</tr>
<tr>
<td>Edward Doe</td>
<td>2008</td>
</tr>
<tr>
<td>John Smith</td>
<td>2008</td>
</tr>
<tr>
<td>Jane Smith</td>
<td>2008</td>
</tr>
</tbody>
</table>

Let’s assume we’re using an author-year style and set uniquename=false. In this case, we would get the following citations:

- Doe 2008a
- Doe 2008b
- Smith 2008a
- Smith 2008b

Since the family names are ambiguous and all works have been published in the same year, an extra letter is appended to the year to disambiguate the citations. Many style guides, however, mandate that the extra letter be used to disambiguate works by the same authors only, not works by different authors with the same family name. In order to disambiguate the author’s family name, you are expected to add additional parts of the name, either as initials or in full. This requirement is addressed by the uniquename option. Here are the same citations with uniquename=init:

- J. Doe 2008
- E. Doe 2008
- Smith 2008a
- Smith 2008b

uniquename=init restricts name disambiguation to initials. Since 'J. Smith' would still be ambiguous, no additional name parts are added for the 'Smiths'. With uniquename=full, names are printed in full where required:

J. Doe 2008
E. Doe 2008
Smith 2008a
Smith 2008b
In order to illustrate the difference between `unique_name=init/full` and `allinit/allfull`, we need to introduce the notion of a 'visible' name. In the following, 'visible' names are all names at a position before the `max_names/min_names/uniquelist` truncation point. For example, given this data:

William Jones/Edward Doe/Jane Smith
John Doe
John Smith

and `max_names=1,min_names=1,unique_name=init/full`, we would get the following names in citations:

Jones et al.
Doe
Smith

When disambiguating names, `unique_name=init/full` only consider the visible names. Since all visible last names are distinct in this example, no further name parts are added. Let's compare that to the output of `unique_name=allinit`:

Jones et al.
J. Doe
Smith

`allinit` considers all names in all `labelname` lists, including those which are hidden and replaced by 'et al.' as the list is truncated. In this example, 'John Doe' is disambiguated from 'Edward Doe'. Since the ambiguity of the two 'Smiths' can't be resolved by adding initials, no initials are added in this case. Now let's compare that to the output of `unique_name=allfull` which also disambiguates 'John Smith' from 'Jane Smith':

Jones et al.
J. Doe
John Smith

The options `unique_name=mininit/minfull` are similar to `init/full` in that they only consider visible names, but they perform minimal disambiguation. That is, they will disambiguate individual names only if they occur in identical lists of last names. Consider the following data:

John Doe/William Jones
Edward Doe/William Jones
John Smith/William Edwards
Edward Smith/Allan Johnson

With `unique_name=init/full`, we would get:

J. Doe and Jones
E. Doe and Jones
J. Smith and Edwards
E. Smith and Johnson

With `unique_name=mininit/minfull`:

J. Doe and Jones
E. Doe and Jones
Smith and Edwards
Smith and Johnson
The 'Smiths' are not disambiguated because the visible name lists are not ambiguous and the mininit/minfull options serve to disambiguate names occurring in identical last name lists only. Another way of looking at this is that they globally disambiguate family name lists. When it comes to ambiguous lists, note that a truncated list is considered to be distinct from an untruncated one even if the visible names are identical. For example, consider the following data:

<table>
<thead>
<tr>
<th>John Doe/William Jones</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edward Doe</td>
</tr>
</tbody>
</table>

With maxnames=1, uniquename=init/full, we would get:

<table>
<thead>
<tr>
<th>J. Doe et al.</th>
</tr>
</thead>
<tbody>
<tr>
<td>E. Doe</td>
</tr>
</tbody>
</table>

With uniquename=mininit/minfull:

<table>
<thead>
<tr>
<th>Doe et al.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doe</td>
</tr>
</tbody>
</table>

Because the lists differ in the `et al.'; the names are not disambiguated.

### 4.11.4.2 Lists of Names (uniquelist)

Ambiguity is also an issue with name lists. If the labelname list is truncated by the maxnames/minnames options, it may become ambiguous. This type of ambiguity is addressed by the uniquelist option. Consider the following data:

<table>
<thead>
<tr>
<th>Doe/Jones/Smith 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith/Johnson/Doe 2005</td>
</tr>
<tr>
<td>Smith/Doe/Edwards 2005</td>
</tr>
<tr>
<td>Smith/Doe/Jones 2005</td>
</tr>
</tbody>
</table>

Many author-year styles truncate long author/editor lists in citations. For example, with maxnames=1 we would get:

<table>
<thead>
<tr>
<th>Doe et al. 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith et al. 2005a</td>
</tr>
<tr>
<td>Smith et al. 2005b</td>
</tr>
<tr>
<td>Smith et al. 2005c</td>
</tr>
</tbody>
</table>

Since the authors are ambiguous after truncation, the extra letter is added to the year to ensure unique citations. Here again, many style guides mandate that the extra letter be used to disambiguate works by the same authors only. In order to disambiguate author lists, you are usually required to add more names, exceeding the maxnames/minnames truncation point. The uniquelist feature addresses this requirement. With uniquelist=true, we would get:

<table>
<thead>
<tr>
<th>Doe et al. 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith, Johnson et al. 2005</td>
</tr>
<tr>
<td>Smith, Doe and Edwards 2005</td>
</tr>
<tr>
<td>Smith, Doe and Jones 2005</td>
</tr>
</tbody>
</table>

The uniquelist option overrides maxnames/minnames on a per-entry basis. Essentially, what happens is that the 'et al.' part of the citation is expanded to the point of no ambiguity – but no further than that. uniquelist may also be combined with uniquename. Consider the following data:

<table>
<thead>
<tr>
<th>John Doe/Allan Johnson/William Jones 2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Doe/Edward Johnson/William Jones 2009</td>
</tr>
<tr>
<td>John Doe/Jane Smith/William Jones 2009</td>
</tr>
<tr>
<td>John Doe/John Smith/William Jones 2009</td>
</tr>
<tr>
<td>John Doe/John Edwards/William Jones 2009</td>
</tr>
<tr>
<td>John Doe/John Edwards/Jack Johnson 2009</td>
</tr>
</tbody>
</table>
**With** `maxnames=1`:

Doe et al. 2009a  
Doe et al. 2009b  
Doe et al. 2009c  
Doe et al. 2009d  
Doe et al. 2009e  
Doe et al. 2009f

**With** `maxnames=1, uniquename=full, uniquelist=true`:

Doe, A. Johnson et al. 2009  
Doe, E. Johnson et al. 2009  
Doe, Jane Smith et al. 2009  
Doe, John Smith et al. 2009  
Doe, Edwards and Jones 2009  
Doe, Edwards and Johnson 2009

**With** `uniquelist=minyear`, list disambiguation only happens if the visible list is identical to another visible list with the same `labelyear`. This is useful for author-year styles which only require that the citation as a whole be unique, but do not guarantee unambiguous authorship information in citations. This mode is conceptually related to `uniquename=mininit/minfull`. Consider this example:

Smith/Jones 2000  
Smith/Johnson 2001

**With** `maxnames=1` and `uniquelist=true`, we would get:

Smith and Jones 2000  
Smith and Johnson 2001

**With** `uniquelist=minyear`:

Smith et al. 2000  
Smith et al. 2001

**With** `uniquelist=minyear`, it is not clear that the authors are different for the two works but the citations as a whole are still unambiguous since the year is different. In contrast to that, `uniquelist=true` disambiguates the authorship even if this information is not required to uniquely locate the works in the bibliography. Let’s consider another example:

Vogel/Beast/Garble/Rook 2000  
Vogel/Beast/Tremble/Bite 2000  
Vogel/Beast/Acid/Squeeze 2001

**With** `maxnames=3, minnames=1, uniquelist=true`, we would get:

Vogel, Beast, Garble et al. 2000  
Vogel, Beast, Tremble et al. 2000  
Vogel, Beast, Acid et al. 2001

**With** `uniquelist=minyear`:

Vogel, Beast, Garble et al. 2000  
Vogel, Beast, Tremble et al. 2000  
Vogel et al. 2001

In the last citation, `uniquelist=minyear` does not override `maxnames/minnames` as the citation does not need disambiguating from the other two because the year is different.
4.11.5 Trackers in Floats and TOC/LOT/LOF

If a citation is given in a float (typically in the caption of a figure or table), scholarly back references like ‘ibidem’ or back references based on the page tracker get ambiguous because floats are objects which are (physically and logically) placed outside the flow of text, hence the logic of such references applies poorly to them. To avoid any such ambiguities, the citation and page trackers are temporarily disabled in all floats. In addition to that, these trackers plus the back reference tracker (\texttt{backref}) are temporarily disabled in the table of contents, the list of figures, and the list of tables.

4.11.6 Mixing Programming Interfaces

The \texttt{biblatex} package provides two main programming interfaces for style authors. The \texttt{\DeclareBibliographyDriver} command, which defines a handler for an entry type, is typically used in \texttt{bbx} files. \texttt{\DeclareCiteCommand}, which defines a new citation command, is typically used in \texttt{cbx} files. However, in some cases it is convenient to mix these two interfaces. For example, the \texttt{\fullcite} command prints a verbose citation similar to the full bibliography entry. It is essentially defined as follows:

\begin{verbatim}
\DeclareCiteCommand{\fullcite}
  {...}
  {\usedriver{...}{\thefield{entrytype}}}
  {...}
  {...}
\end{verbatim}

As you can see, the core code which prints the citations simply executes the bibliography driver defined with \texttt{\DeclareBibliographyDriver} for the type of the current entry. When writing a citation style for a verbose citation scheme, it is often convenient to use the following structure:

\begin{verbatim}
\ProvidesFile{example.cbx}[2007/06/09 v1.0 biblatex citation style]
\DeclareCiteCommand{\cite}
  {...}
  {\usedriver{...}{cite:\thefield{entrytype}}}
  {...}
  {...}
\DeclareBibliographyDriver{cite:article}{...}
\DeclareBibliographyDriver{cite:book}{...}
\DeclareBibliographyDriver{cite:inbook}{...}
...
\end{verbatim}

Another case in which mixing interfaces is helpful are styles using cross-references within the bibliography. For example, when printing an @incollection entry, the data inherited from the @collection parent entry would be replaced by a short pointer to the respective parent entry:


One way to implement such cross-references within the bibliography is to think of them as citations which use the value of the \texttt{xref} or \texttt{crossref} field as the entry key. Here is an example:

\begin{verbatim}
\ProvidesFile{example.bbx}[2007/06/09 v1.0 biblatex bibliography → style]
\DeclareCiteCommand{\bbx@xref}
  {}
  {...}% code for cross-references
  {}
  {}
\DeclareBibliographyDriver{incollection}{...
\end{verbatim}
When defining \bbx@xref, the \verb+⟨precode⟩+, \verb+⟨postcode⟩+, and \verb+⟨sepcode⟩+ arguments of \DeclareCiteCommand are left empty in the above example because they will not be used anyway. The cross-reference is printed by the \verb+⟨loopcode⟩+ of \bbx@xref. For further details on the xref field, refer to §2.2.3 and to the hints in §2.4.1. Also see the \verb+\iffieldxref+, \verb+\iflistxref+, and \verb+\ifnamexref+ tests in §4.6.2. The above could also be implemented using the \verb+\entrydata+ command from §4.4.1:

\ProvidesFile{example.bbx}[2007/06/09 v1.0 biblatex bibliography ↪ → style]
\DeclareBibliographyDriver{incollection}{% ...
\iffieldundef{xref}{...}% code if no cross-reference
\entrydata{\thefield{xref}}{%
% code for cross-references
... }
}}%
...
}

4.11.7 Using the Punctuation Tracker

4.11.7.1 The Basics

There is one fundamental principle style authors should keep in mind when designing a bibliography driver: block and unit punctuation is handled asynchronously. This is best explained by way of example. Consider the following code snippet:

\printfield{title}%
\newunit
\printfield{edition}%
\newunit
\printfield{note}%

If there is no \verb+edition+ field, this piece of code will not print:

Title. . Note

but rather:

Title. Note

because the unit punctuation tracker works asynchronously. \verb+\newunit+ will not print the unit punctuation immediately. It merely records a unit boundary and puts \verb+\newunitpunct+ on the punctuation buffer. This buffer will be handled by subsequent \printfield, \printlist, or similar commands but only if the respective field or list is defined. Commands like \printfield will consider three factors prior to inserting any block or unit punctuation:

- Has a new unit/block been requested at all?
  - Is there any preceding \newunit or \newblock command?
- Did the preceding commands print anything?
  - Is there any preceding \printfield or similar command?
  - Did this command actually print anything?
• Are we about to print anything now?
  = Is the field/list to be processed now defined?

Block and unit punctuation will only be inserted if all of these conditions apply. Let’s reconsider the above example:

\printfield{title}\
\newunit
\printfield{edition}\
\newunit
\printfield{note}

Here’s what happens if the edition field is undefined. The first \printfield command prints the title and sets an internal ‘new text’ flag. The first \newunit sets an internal ‘new unit’ flag. No punctuation has been printed at this point. The second \printfield does nothing because the edition field is undefined. The next \newunit command sets the internal flag ‘new unit’ again. Still no punctuation has been printed. The third \printfield checks if the note field is defined. If so, it looks at the ‘new text’ and ‘new unit’ flags. If both are set, it inserts the punctuation buffer before printing the note. It then clears the ‘new unit’ flag and sets the ‘new text’ flag again.

This may all sound more complicated than it is. In practice, it means that it is possible to write large parts of a bibliography driver in a sequential way. The advantage of this approach becomes obvious when trying to write the above code without using the punctuation tracker. Such an attempt will lead to a rather convoluted set of iffieldundef tests required to check for all possible field combinations (note that the code below handles three fields; a typical driver may need to cater for some two dozen fields):

\iffieldundef{title}\
  \iffieldundef{edition}\
    {\printfield{note}}\
    {\printfield{edition}\
      \iffieldundef{note}\
        {. \printfield{note}}}\
      {\printfield{title}\
        \iffieldundef{edition}\
        {. \printfield{edition}}\
        \iffieldundef{note}\
        {. \printfield{note}}}}

4.11.7.2 Common Mistakes It is a fairly common misconception to think of the unit punctuation as something that is handled synchronously. This typically causes problems if the driver includes any literal text. Consider this erroneous code snippet which will generate misplaced unit punctuation:

\printfield{title}\
\newunit\
\printfield{series} \printfield{number}\

This code will yield the following result:

Title (. Series Number)

Here’s what happens. The first \printfield prints the title. Then \newunit marks a unit boundary but does not print anything. The unit punctuation is printed by the next \printfield command. That’s the asynchronous part mentioned before. However, the opening parenthesis is printed immediately before the next \printfield inserts the unit punctuation, leading to a misplaced period. When inserting any literal text such as parentheses (including those printed by commands such as \bibopenparen and \mkbibparens), always wrap the text in a \printtext command. For
the punctuation tracker to work as expected, it needs to know about all literal text inserted by a driver. This is what \texttt{printtext} is all about. \texttt{printtext} interfaces with the punctuation tracker and ensures that the punctuation buffer is inserted before the literal text gets printed. It also sets the internal ‘new text’ flag. Note there is in fact a third piece of literal text in this example: the space after \texttt{printfield{series}}. In the corrected example, we will use the punctuation tracker to handle that space.

\begin{verbatim}
\printfield{title}\
\newunit
\printtext{()}
\printfield{series}\
\setunit*{\addspace}
\printfield{number}\
\printtext{)}
\end{verbatim}

While the above code will work as expected, the recommended way to handle parentheses, quotes, and other things which enclose more than one field, is to define a field format:

\begin{verbatim}
\DeclareFieldFormat{parens}{\mkbibparens{#1}}
\end{verbatim}

Field formats may be used with both \texttt{printfield} and \texttt{printtext}, hence we can use them to enclose several fields in a single pair of parentheses:

\begin{verbatim}
\printtext[parens]{\
\printfield{series}\
\setunit*{\addspace}\
\printfield{number}}
\end{verbatim}

We still need to handle cases in which there is no series information at all, so let’s improve the code some more:

\begin{verbatim}
\iffieldundef{series}
{}\else\
\printtext[parens]{\
\printfield{series}\
\setunit*{\addspace}\
\printfield{number}}\fi
\end{verbatim}

One final hint: localisation strings are not literal text as far as the punctuation tracker is concerned. Since \texttt{\textbf{bibstring}} and similar commands interface with the punctuation tracker, there is no need to wrap them in a \texttt{printtext} command.

\textbf{4.11.7.3 Advanced Usage} The punctuation tracker may also be used to handle more complex scenarios. For example, suppose that we want the fields location, publisher, and year to be rendered in one of the following formats, depending on the available data:

\begin{verbatim}
...text. Location: Publisher, Year. Text...
...text. Location: Publisher. Text...
...text. Location: Year. Text...
...text. Publisher, Year. Text...
...text. Location. Text...
...text. Publisher. Text...
...text. Year. Text...
\end{verbatim}

This problem can be solved with a rather convoluted set of \texttt{\iflistundef} and \texttt{\iffieldundef} tests which check for all possible field combinations:
The above could be written in a somewhat more readable way by employing \ifthenelse and the boolean operators discussed in §4.6.3. The approach would still be essentially the same. However, it may also be written sequentially:

\newunit
\printlist{location}\
\setunit*{\addcolon\space}\
\printlist{publisher}\
\setunit*{\addcomma\space}\
\printfield{year}\
\newunit

In practice, you will often use a combination of explicit tests and the implicit tests performed by the punctuation tracker. For example, consider the following format (note the punctuation after the location if there is no publisher):

...text. Location: Publisher, Year. Text...
...text. Location: Publisher. Text...
...text. Location, Year. Text...
...text. Publisher, Year. Text...
...text. Location. Text...
...text. Publisher. Text...
...text. Year. Text...

This can be handled by the following code:

\newunit
\printlist{location}\
\iflistundef{publisher}\
{\setunit*{\addcomma\space}}\
{\setunit*{\addcolon\space}}\
\printlist{publisher}\
{\setunit*{\addcomma\space}}\
\printfield{year}\
\newunit

Since the punctuation after the location is special if there is no publisher, we need one \iflistundef test to catch this case. Everything else is handled by the punctuation tracker.

### 4.11.8 Custom Localization Modules

Style guides may include provisions as to how strings like ‘edition’ should be abbreviated or they may mandate certain fixed expressions. For example, the MLA style guide requires authors to use the term
Localization commands such as `\DefineBibliographyStrings` from §3.9 may indeed be used in cbx and bbx files to handle such cases. However, overloading style files with translations is rather inconvenient. This is where `\DeclareLanguageMapping` from §4.9.1 comes into play. This command maps an lbx file with alternative translations to a babel/polyglossia language. For example, you could create a file named `french-humanities.lbx` which provides French translations adapted for use in the humanities and map it to the babel/polyglossia language `french` in the preamble or in the configuration file:

```
\DeclareLanguageMapping{french}{french-humanities}
```

If the document language is set to `french`, `french-humanities.lbx` will replace `french.lbx`. Coming back to the MLA example mentioned above, an MLA style may come with an `american-mla.lbx` file to provide strings which comply with the MLA style guide. It would declare the following mapping in the cbx and/or bbx file:

```
\DeclareLanguageMapping{american}{american-mla}
```

Since the alternative lbx file can inherit strings from the standard `american.lbx` module, `american-mla.lbx` may be as short as this:

```
\ProvidesFile{american-mla.lbx}[2008/10/01 v1.0 biblatex ↪ localization]
\InheritBibliographyExtras{american}
\DeclareBibliographyStrings{%
    inherit = {american},
    bibliography = {{Works Cited}{Works Cited}},
    references = {{Works Cited}{Works Cited}},
}
\endinput
```

Alternative lbx files must ensure that the localisation module is complete. They should do so by inheriting data from the corresponding standard module. If the language `american` is mapped to `american-mla.lbx`, `biblatex` will not load `american.lbx` unless this module is requested explicitly. In the above example, inheriting `strings` and `extras` will cause `biblatex` to load `american.lbx` before applying the modifications in `american-mla.lbx`.

Note that `\DeclareLanguageMapping` is not intended to handle language variants (e.g., American English vs. British English) or babel/polyglossia language aliases (e.g., USenglish vs. american). For example, babel/polyglossia offers the USenglish option which is similar to american. Therefore, `biblatex` ships with an USenglish.lbx file which simply inherits all data from `american.lbx` (which in turn gets the `strings` from `english.lbx`). In other words, the mapping of language variants and babel/polyglossia language aliases happens on the file level, the point being that `biblatex`'s language support can be extended simply by adding additional lbx files. There is no need for centralized mapping. If you need support for, say, Portuguese (babel/polyglossia: portuges), you create a file named `portuges.lbx`. If `babel/polyglossia` offered an alias named `brasil`, you would create `brasil.lbx` and inherit the data from `portuges.lbx`. In contrast to that, the point of `\DeclareLanguageMapping` is handling stylistic variants like ‘humanities vs. natural sciences’ or ‘MLA vs. APA’ etc. which will typically be built on top of existing lbx files.

### 4.11.9 Grouping

In a citation or bibliography style, you may need to set flags or store certain values for later use. In this case, it is crucial to understand the basic grouping structure imposed by this package. As a rule of thumb, you are working in a large group whenever author commands such as those discussed in §4.6 are available because the author interface of this package is only enabled locally. If any bibliographic data is available, there is at least one additional group. Here are some general rules:

- The entire list of references printed by `\printbibliography` and similar commands is processed in a group. Each entry in the list is processed in an additional group which encloses the (item code) of `\defbibenvironment` as well as all driver code.
• The entire bibliography list printed by \printbiblist is processed in a group. Each entry in the list is processed in an additional group which encloses the \textit{item code} of \texttt{defbibenvironment} as well as all driver code.

• All citation commands defined with \DeclareCiteCommand are processed in a group holding the complete citation code consisting of the \textit{precode}, \textit{sepcode}, \textit{loopcode}, and \textit{postcode} arguments. The \textit{loopcode} is enclosed in an additional group every time it is executed. If any \textit{wrapper} code has been specified, the entire unit consisting of the wrapper code and the citation code is wrapped in an additional group.

• In addition to the grouping imposed by all backend commands defined with \DeclareCiteCommand, all ‘autocite’ and ‘multicite’ definitions imply an additional group.

• \texttt{\printfile}, \texttt{\printtext}, \texttt{\printfield}, \texttt{\printlist}, and \texttt{\printnames} form groups. This implies that all formatting directives will be processed within a group of their own.

• All lbx files are loaded and processed in a group. If an lbx file contains any code which is not part of \texttt{\DeclareBibliographyExtras}, the definitions must be global.

Note that using \texttt{\aftergroup} in citation and bibliography styles is unreliable because the precise number of groups employed in a certain context may change in future versions of this package. If the above list states that something is processed in a group, this means that there is \textit{at least one} group. There may also be several nested ones.

4.11.10 Namespaces

In order to minimize the risk of name clashes, LaTeX packages typically prefix the names of internal macros with a short string specific to the package. For example, if the foobar package requires a macro for internal use, it would typically be called \texttt{\FB@macro} or \texttt{\foo@macro} rather than \texttt{\macro} or \texttt{\@macro}. Here is a list of the prefixes used or recommended by biblatex:

\texttt{blx} All macros with names like \texttt{\blx@name} are strictly reserved for internal use. This also applies to counter names, length registers, boolean switches, and so on. These macros may be altered in backwards-incompatible ways, they may be renamed or even removed at any time without further notice. Such changes will not even be mentioned in the revision history or the release notes. In short: never use any macros with the string \texttt{blx} in their name in any styles.

\texttt{abx} Macros prefixed with abx are also internal macros but they are fairly stable. It is always preferable to use the facilities provided by the official author interface, but there may be cases in which using an abx macro is convenient.

\texttt{bbx} This is the recommended prefix for internal macros defined in bibliography styles.

\texttt{cbx} This is the recommended prefix for internal macros defined in citation styles.

\texttt{lbx} This is the recommended base prefix for internal macros defined in localisation modules. The localisation module should add a second prefix to specify the language. For example, an internal macro defined by the Spanish localisation module would be named \texttt{\lbx@es@macro}.

Appendix

A Default Driver Source Mappings

These are the driver default source mappings.

A.1 bibtex

The bibtex driver is of course the most comprehensive and mature of the biblatex/biber supported data formats. These source mapping defaults are how the aliases from sections § 2.1.2 and § 2.2.5 are implemented.

```latex
\DeclareDriverSourcemap[datatype=bibtex]{
  \map{
    \step[typesource=conference, typetarget=inproceedings]
    \step[typesource=electronic, typetarget=online]
    \step[typesource=www, typetarget=online]
  }
}
```
### B Default Inheritance Setup

The following table shows the biber cross-referencing rules defined by default. Please refer to §§ 24.1 and 4.5.11 for explanation.

<table>
<thead>
<tr>
<th>Types</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Target</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>mvbook, book</td>
<td>inbook, bookinbook, suppbook</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>mvbook</td>
<td>book, inbook, bookinbook, suppbook</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Types

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>mvcollection,</td>
<td>collection, reference,</td>
</tr>
<tr>
<td>mvreference</td>
<td>incollection, inreference,</td>
</tr>
<tr>
<td></td>
<td>suppcollection</td>
</tr>
<tr>
<td></td>
<td>mvproceedings proceedings, inproceedings</td>
</tr>
<tr>
<td>book</td>
<td>inbook, bookinbook, suppbook</td>
</tr>
<tr>
<td></td>
<td>collection, incollection, inreference,</td>
</tr>
<tr>
<td></td>
<td>suppcollection</td>
</tr>
<tr>
<td></td>
<td>proceedings inproceedings</td>
</tr>
<tr>
<td></td>
<td>periodical article, suppperiodical</td>
</tr>
</tbody>
</table>

### Fields

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>maintitle</td>
</tr>
<tr>
<td>subtitle</td>
<td>mainsubtitle</td>
</tr>
<tr>
<td>titleaddon</td>
<td>maintitleaddon</td>
</tr>
<tr>
<td>shorttitle</td>
<td>-</td>
</tr>
<tr>
<td>sorttitle</td>
<td>-</td>
</tr>
<tr>
<td>indextitle</td>
<td>-</td>
</tr>
<tr>
<td>indexsorttitle</td>
<td>-</td>
</tr>
<tr>
<td>title</td>
<td>booktitle</td>
</tr>
<tr>
<td>subtitle</td>
<td>booksubtitle</td>
</tr>
<tr>
<td>titleaddon</td>
<td>booktitleaddon</td>
</tr>
<tr>
<td>shorttitle</td>
<td>-</td>
</tr>
<tr>
<td>sorttitle</td>
<td>-</td>
</tr>
<tr>
<td>indextitle</td>
<td>-</td>
</tr>
<tr>
<td>indexsorttitle</td>
<td>-</td>
</tr>
<tr>
<td>title</td>
<td>booktitle</td>
</tr>
<tr>
<td>subtitle</td>
<td>booksubtitle</td>
</tr>
<tr>
<td>titleaddon</td>
<td>booktitleaddon</td>
</tr>
<tr>
<td>shorttitle</td>
<td>-</td>
</tr>
<tr>
<td>sorttitle</td>
<td>-</td>
</tr>
<tr>
<td>indextitle</td>
<td>-</td>
</tr>
<tr>
<td>indexsorttitle</td>
<td>-</td>
</tr>
</tbody>
</table>

### C Default Sorting Schemes

#### C.1 Alphabetic Schemes 1

The following table shows the standard alphabetic sorting schemes defined by default. Please refer to § 3.5 for explanation.

<table>
<thead>
<tr>
<th>Option</th>
<th>Sorting scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>nty</td>
<td>presort→sortname→sorttitle→sortyear→volume</td>
</tr>
<tr>
<td></td>
<td>←mm ←author ←title ←year</td>
</tr>
<tr>
<td></td>
<td>←editor</td>
</tr>
<tr>
<td></td>
<td>←translator</td>
</tr>
<tr>
<td></td>
<td>←sorttitle</td>
</tr>
<tr>
<td></td>
<td>←title</td>
</tr>
</tbody>
</table>
### C.2 Alphabetic Schemes 2

The following table shows the alphabetic sorting schemes for alphabetic styles defined by default. Please refer to §3.5 for explanation.

<table>
<thead>
<tr>
<th>Option</th>
<th>Sorting scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>nyt</td>
<td>presort→sortname→sortyear→sorttitle→volume</td>
</tr>
<tr>
<td></td>
<td>←mm ←author ←year ←title</td>
</tr>
<tr>
<td></td>
<td>←editor ←translator ←sorttitle ←title</td>
</tr>
<tr>
<td>nyvt</td>
<td>presort→sortname→sortyear→volume→sorttitle</td>
</tr>
<tr>
<td></td>
<td>←mm ←author ←year ←title</td>
</tr>
<tr>
<td></td>
<td>←editor ←translator ←sorttitle ←title</td>
</tr>
<tr>
<td>all</td>
<td>presort→sortkey</td>
</tr>
<tr>
<td></td>
<td>←mm</td>
</tr>
</tbody>
</table>

### C.3 Chronological Schemes

The following table shows the chronological sorting schemes defined by default. Please refer to §3.5 for explanation.

<table>
<thead>
<tr>
<th>Option</th>
<th>Sorting scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>ynt</td>
<td>presort→sortyear→sortname→sorttitle→volume</td>
</tr>
<tr>
<td></td>
<td>←mm ←year 9999</td>
</tr>
<tr>
<td></td>
<td>←sortname ←sorttitle</td>
</tr>
<tr>
<td></td>
<td>←author ←title</td>
</tr>
<tr>
<td></td>
<td>←editor ←translator ←sorttitle ←title</td>
</tr>
<tr>
<td>ydnt</td>
<td>presort→sortyear(desc.)→sortname→sorttitle</td>
</tr>
<tr>
<td></td>
<td>←mm ←year(desc.) 9999</td>
</tr>
<tr>
<td></td>
<td>←sortname ←sorttitle</td>
</tr>
<tr>
<td></td>
<td>←author ←title</td>
</tr>
<tr>
<td></td>
<td>←editor ←translator ←sorttitle ←title</td>
</tr>
<tr>
<td>all</td>
<td>presort→sortkey</td>
</tr>
<tr>
<td></td>
<td>←mm</td>
</tr>
</tbody>
</table>
The biblatexml XML datasource format is designed to be an extensible and modern data source format for biblatex users. There are limitations with BibTeX format .bib files, in particular one might mention UTF-8 support and name formats. biber goes some way to addressing the UTF-8 limitations by using a modified version of the btparse C library but the rather archaic name parsing rules for BibTeX are hard-coded and specific to simple Western names.

biblatexml is an XML format for bibliographic data. When biber either reads or writes biblatexml format datasources, it automatically writes a RelaXNG XML schema for the datasources which is dynamically generated from the active biblatex datamodel. There is no static schema for biblatexml datasources because the allowable fields etc. depend on the data model. The format of biblatexml datasources is relatively self-explanatory—it is usually only necessary to generate a biblatexml datasource from existing BibTeX format datasources (using biber’s ‘tool’ mode) in order to understand the format. biber also allows users to validate biblatexml datasources against the data model generated schema.

Since the biblatexml format is XML and depends on the data model and the data model is extensible by the user (see § 4.5.4), the biblatexml format can deal with extensions that BibTeX format data sources cannot, e.g. new nameparts, options at sub-entry scope. Since it is an XML format, it is relatively easy to transform it into other XML formats or HTML using standard XML processing libraries and tools.

Here is an explanation of the format with examples. By convention, biblatexml files have a .bltxml extension and kpsewhich understands this file extension.

D.1 Header

biblatexml files begin with the standard XML header:

```xml
<?xml version="1.0" encoding="UTF-8"?>
```

The schema model, type and schema type namespace are given in the following line:

```xml
<?xml-model href="biblatexml.rng"
    type="application/xml"
    schematypens="http://relaxng.org/ns/structure/1.0"?>
```

When biber generates biblatexml data sources, it automatically adds this line and points the schema model (href) attribute at the automatically generated RelaXNG XML schema for ease of validation.

D.2 Body

The body of a biblatexml data source looks like:

```xml
<bltx:entries
    xmlns:bltx="http://biblatex-biber.sourceforge.net/biblatexml">
    <bltx:entry id="" entrytype=""
        <bltx:entry id="" entrytype="">
            <bltx:entry id="" entrytype="">
                <bltx:entry id="" entrytype="">
                    <bltx:entry id="" entrytype="">
                        <bltx:entry id="" entrytype="">
                            <bltx:entry/>
                        </bltx:entries>
                    </bltx:entry>
                </bltx:entries>
            </bltx:entries>
        </bltx:entries>
    </bltx:entries>
</bltx:entries>
```

The body is one or more entry elements inside the top-level entries element and everything is in the bltx namespace. An entry has an id attribute corresponding to a BibTeX entry key and a entrytype attribute corresponding to a BibTeX entrytype. For example, the biblatexml
In general, the XML elements in a \texttt{biblatexml} format datasource file have names corresponding to the fields in the datamodel, just like \texttt{BibTeX} format datasources. So for example, the \texttt{BibTeX} format source

\begin{verbatim}
@book{key1,
    TITLE = {...},
    ISSUE = {...},
    NOTE = {...}
}
\end{verbatim}

would be, in \texttt{biblatexml}

\begin{verbatim}
<bltx:entry id="key1" entrytype="book">
    <bltx:title>...</bltx:title>
    <bltx:issue>...</bltx:issue>
    <bltx:note>...</bltx:note>
</bltx:entry>
\end{verbatim}

The following exceptions to this simple mapping are to be noted

\section*{D.2.1 Key aliases}

Citation key aliases are specified like this:

\begin{verbatim}
<bltx:ids>
    <bltx:key>alias1</bltx:key>
    <bltx:key>alias2</bltx:key>
</bltx:ids>
\end{verbatim}

this corresponds to the \texttt{BibTeX} format

\begin{verbatim}
@book{key1,
    IDS = {alias1,alias2}
}
\end{verbatim}

\section*{D.2.2 Names}

Name specifications in \texttt{biblatexxml} are somewhat more complex in order to generalise the name handling abilities of \texttt{biblatex}. The user has to be more explicit about the name parts and this allows a much great scope for the handling of different types of names and name parts. A name in \texttt{biblatexml} format looks like this
A name list field is contained in the names element with the mandatory type attribute giving the name of the name list. Things to note:

- The optional morenames attribute performs the same task as the BibTeX datasource format ‘and others’ string at the end of a name.
- Note that optional useprefix option can be specified can be specified at the level of a name list or an individual name in the name list. This is impossible with BibTeX datasources.
- Individual names may have an optional gender attribute which must be one of those defined in the datamodel ‘gender’ constant list. This is currently not used by standard styles but is available in biblatex name formats if necessary.
- A name list is composed of one or more name elements.
- Each name is composed of name parts of a type defined by the data model 'nameparts' constant.
- Each name part may have an option initial attribute which makes explicit the initial of the name part. If this is not present, biber attempts to automatically determine the initial from the name part.
- Name parts may have name parts so that compound names can be handled.

Ignoring the biblatexml-only features, a corresponding BibTeX format datasource would look like this:

```
AUTHOR = {von Smith, John Arthur and Brown, Raymond and others}
```

### D.2.3 Lists

Datasource list fields (see §2.2.1) can be represented in two ways, depending on whether there is more than one element in the list:

```
<bltx:publisher>London</bltx:publisher>
<bltx:location>
  <bltx:item>London</bltx:item>
  <bltx:item>Moscow</bltx:item>
</bltx:location>
```

### D.2.4 Ranges

Datasource range fields (see §2.2.1) are represented like this:

```
<bltx:pages>
  <bltx:item>1</bltx:start>
</bltx:pages>
```
A range field is a list of ranges, each with its own item. A range item has a start element and an optional end element, since ranges can be open-ended.

**D.2.5 Dates**

Datasource date fields (see §2.2.1) can be represented in two ways, depending on whether they constitute a date range:

```xml
<bltx:date>1985-04-02</bltx:date>
<bltx:date type="event">
  <bltx:start>1990-05-16</bltx:start>
  <bltx:end>1990-05-17</bltx:end>
</bltx:date>
```

The type attribute on a date element corresponds to a particular type of date defined in the data model.

**D.2.6 Related Entries**

Related entries are specified as follows:

```xml
<bltx:related>
  <bltx:item type="reprint"
    ids="rel1,rel2"
    string="Somestring"
    options="skipbiblist"/>
</bltx:related>
```

This corresponds to the BibTeX format:

```latex
@book{key1,
  RELATED = {rel2,rel2},
  RELATEDTYPE = {reprint},
  RELATEDSTRING = {Somestring},
  RELATEDOPTIONS = {skipbiblist}
}
```

As per §4.5.1, the string and options attributes are optional.

**E Option Scope**

The following table provides an overview of the scope (global/per-type/per-entry) of various package options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviate</td>
<td><img src="#" alt="Abbreviate" /></td>
</tr>
<tr>
<td>Alldates</td>
<td><img src="#" alt="Alldates" /></td>
</tr>
<tr>
<td>Alldatesusetime</td>
<td><img src="#" alt="Alldatesusetime" /></td>
</tr>
<tr>
<td>Alltimes</td>
<td><img src="#" alt="Alltimes" /></td>
</tr>
<tr>
<td>Arxiv</td>
<td><img src="#" alt="Arxiv" /></td>
</tr>
<tr>
<td>Autocite</td>
<td><img src="#" alt="Autocite" /></td>
</tr>
<tr>
<td>Option</td>
<td>Scope</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>Load-time</td>
</tr>
<tr>
<td>autopunct</td>
<td>•</td>
</tr>
<tr>
<td>autolang</td>
<td>•</td>
</tr>
<tr>
<td>backend</td>
<td>•</td>
</tr>
<tr>
<td>backref</td>
<td>•</td>
</tr>
<tr>
<td>backrefsetstyle</td>
<td>•</td>
</tr>
<tr>
<td>backrefstyle</td>
<td>•</td>
</tr>
<tr>
<td>bibencoding</td>
<td>•</td>
</tr>
<tr>
<td>bibstyle</td>
<td>•</td>
</tr>
<tr>
<td>bibwarn</td>
<td>•</td>
</tr>
<tr>
<td>block</td>
<td>•</td>
</tr>
<tr>
<td>citecounter</td>
<td>•</td>
</tr>
<tr>
<td>citereset</td>
<td>•</td>
</tr>
<tr>
<td>citestyle</td>
<td>•</td>
</tr>
<tr>
<td>citetracker</td>
<td>•</td>
</tr>
<tr>
<td>clearlang</td>
<td>•</td>
</tr>
<tr>
<td>datamodel</td>
<td>–</td>
</tr>
<tr>
<td>dataonly</td>
<td>–</td>
</tr>
<tr>
<td>date</td>
<td>•</td>
</tr>
<tr>
<td>labeldate</td>
<td>•</td>
</tr>
<tr>
<td>dateabbrev</td>
<td>•</td>
</tr>
<tr>
<td>datecirca</td>
<td>•</td>
</tr>
<tr>
<td>dateera</td>
<td>•</td>
</tr>
<tr>
<td>dateerauto</td>
<td>•</td>
</tr>
<tr>
<td>dateuncertain</td>
<td>•</td>
</tr>
<tr>
<td>datezeros</td>
<td>•</td>
</tr>
<tr>
<td>defernumbers</td>
<td>•</td>
</tr>
<tr>
<td>doi</td>
<td>•</td>
</tr>
<tr>
<td>eprint</td>
<td>•</td>
</tr>
<tr>
<td>inits</td>
<td>•</td>
</tr>
<tr>
<td>gregorianstart</td>
<td>•</td>
</tr>
<tr>
<td>hyperref</td>
<td>•</td>
</tr>
<tr>
<td>ibidtracker</td>
<td>•</td>
</tr>
<tr>
<td>idemtracker</td>
<td>•</td>
</tr>
<tr>
<td>indexing</td>
<td>•</td>
</tr>
<tr>
<td>isbn</td>
<td>•</td>
</tr>
<tr>
<td>julian</td>
<td>•</td>
</tr>
<tr>
<td>labelalpha</td>
<td>•</td>
</tr>
<tr>
<td>labelnamefield</td>
<td>–</td>
</tr>
<tr>
<td>labelnumber</td>
<td>•</td>
</tr>
<tr>
<td>labeltitle</td>
<td>•</td>
</tr>
<tr>
<td>labeltitlefield</td>
<td>–</td>
</tr>
<tr>
<td>labeltitleyear</td>
<td>•</td>
</tr>
<tr>
<td>labeldateparts</td>
<td>•</td>
</tr>
<tr>
<td>labeltime</td>
<td>•</td>
</tr>
<tr>
<td>labeldateusetime</td>
<td>•</td>
</tr>
<tr>
<td>time</td>
<td>•</td>
</tr>
<tr>
<td>dateusetime</td>
<td>•</td>
</tr>
<tr>
<td>language</td>
<td>•</td>
</tr>
<tr>
<td>loadfiles</td>
<td>•</td>
</tr>
<tr>
<td>loccittracker</td>
<td>•</td>
</tr>
<tr>
<td>maxalphenames</td>
<td>•</td>
</tr>
<tr>
<td>maxbibnames</td>
<td>•</td>
</tr>
<tr>
<td>maxcitenames</td>
<td>•</td>
</tr>
<tr>
<td>maxitems</td>
<td>•</td>
</tr>
<tr>
<td>maxnames</td>
<td>•</td>
</tr>
<tr>
<td>maxparens</td>
<td>•</td>
</tr>
<tr>
<td>mbcite</td>
<td>•</td>
</tr>
<tr>
<td>minalphenames</td>
<td>•</td>
</tr>
<tr>
<td>minibibnames</td>
<td>•</td>
</tr>
<tr>
<td>Option</td>
<td>Scope</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>Load-time</td>
</tr>
<tr>
<td>mincitenames</td>
<td>●</td>
</tr>
<tr>
<td>mincrossrefs</td>
<td>●</td>
</tr>
<tr>
<td>minxrefs</td>
<td>●</td>
</tr>
<tr>
<td>minitems</td>
<td>●</td>
</tr>
<tr>
<td>minnames</td>
<td>●</td>
</tr>
<tr>
<td>natbib</td>
<td>●</td>
</tr>
<tr>
<td>noinherit</td>
<td>–</td>
</tr>
<tr>
<td>notype</td>
<td>●</td>
</tr>
<tr>
<td>opcitittracker</td>
<td>●</td>
</tr>
<tr>
<td>openbib</td>
<td>●</td>
</tr>
<tr>
<td>pagetracker</td>
<td>●</td>
</tr>
<tr>
<td>parenttracker</td>
<td>●</td>
</tr>
<tr>
<td>punctfont</td>
<td>●</td>
</tr>
<tr>
<td>refsection</td>
<td>●</td>
</tr>
<tr>
<td>refsegment</td>
<td>●</td>
</tr>
<tr>
<td>safeinputenc</td>
<td>●</td>
</tr>
<tr>
<td>seconds</td>
<td>●</td>
</tr>
<tr>
<td>singletitle</td>
<td>●</td>
</tr>
<tr>
<td>skipbib</td>
<td>–</td>
</tr>
<tr>
<td>skipbiblist</td>
<td>–</td>
</tr>
<tr>
<td>skiplab</td>
<td>–</td>
</tr>
<tr>
<td>sortcase</td>
<td>●</td>
</tr>
<tr>
<td>sortcites</td>
<td>●</td>
</tr>
<tr>
<td>sorting</td>
<td>●</td>
</tr>
<tr>
<td>sortnamekeyscheme</td>
<td>–</td>
</tr>
<tr>
<td>sortlocale</td>
<td>●</td>
</tr>
<tr>
<td>sortlos</td>
<td>●</td>
</tr>
<tr>
<td>sortupper</td>
<td>●</td>
</tr>
<tr>
<td>style</td>
<td>●</td>
</tr>
<tr>
<td>terseinits</td>
<td>●</td>
</tr>
<tr>
<td>texencoding</td>
<td>●</td>
</tr>
<tr>
<td>timezeros</td>
<td>●</td>
</tr>
<tr>
<td>timezones</td>
<td>●</td>
</tr>
<tr>
<td>uniquelist</td>
<td>●</td>
</tr>
<tr>
<td>uniquename</td>
<td>●</td>
</tr>
<tr>
<td>uniquetitle</td>
<td>●</td>
</tr>
<tr>
<td>uniquetarget</td>
<td>●</td>
</tr>
<tr>
<td>uniquework</td>
<td>●</td>
</tr>
<tr>
<td>uniquework</td>
<td>●</td>
</tr>
<tr>
<td>uniqueprimaryauthor</td>
<td>●</td>
</tr>
<tr>
<td>url</td>
<td>●</td>
</tr>
<tr>
<td>useprefix</td>
<td>●</td>
</tr>
<tr>
<td>use&lt;name&gt;</td>
<td>●</td>
</tr>
</tbody>
</table>

**F Revision History**

This revision history is a list of changes relevant to users of this package. Changes of a more technical nature which do not affect the user interface or the behavior of the package are not included in the list. More technical details are to be found in the `CHANGES.org` file. The numbers on the right indicate the relevant section of this manual.

### 3.7 2016

Corrected default for \bibdateeraprefix................................. 4.10.2

**Added** \DeclareSortInclusion ................................. 4.5.6

**Added** \relateddelim<relatedtype> ................................. 3.10.1

### 3.6 2016-09-15

Corrected some documentation and fixed a bug with labeldate localisation strings.
3.5 2016-09-10

Added `\ifuniquebaretitle` ......................................... 4.6.2
Documented `\labelnamesource` and `\labeltitlesource` ........... 4.2.4.1
Added `\bibdatetangesep` ........................................ 3.10.3
Added resection option to `\Declaresourcemap` .................. 4.5.3
Added suppress option to inheritance specifications .......... 4.5.11
Added `\ifuniquework` ........................................... 4.6.2
Changed `\DeclareStyleSourcemap` so that it can be used multiple times ... 4.5.3
Added `\forcezerosy` and `\forcezerosmdt` ...................... 4.10.4
Changed `\mkdatezeros` to `\mkyearzeros`, `\mkmonthzeros` and `\mkdayzeros` 4.10.4
Added `\DeclareLabelalphaTemplate` .......................... 4.5.5
Added `\DeclareLabelalphaNameTemplate` ....................... 4.5.6
Added `\DeclareLabelalphadatamapTemplate` .................... 4.5.6

3.4 2016-05-10

Added `\ifcrossrefsresource` and `\ifxrefsresource` ........... 4.6.2
Added data annotation feature .................................. 3.6
Added package option `minxrefs` ................................ 3.1.2.1
Added `\ifuniqueprimaryauthor` and associated global option ... 4.6.2
Added `\DeprecateField`, `\DeprecateList` and `\DeprecateName` 4.4.1
Added `\ifcaselang` ............................................. 4.6.2
Added `\DeclareSortTranslit` .................................. 4.5.6
Added uniquetitle test ......................................... 4.6.2
Added `\namelabeldelim` ...................................... 3.10.1
New starred variants of the `\assignrefercontext`* macros .... 3.7.10
New context-sensitive delimiter interface ...................... 3.10.2
Moved `prefixnumbers` option to `\newreffontext` and renamed to `labelprefix` 3.7.10
Added `\DeclareDatafieldSet` .................................. 4.5.3

3.3 2016-03-01

New macros for auto-assignment of refcontexts ................. 3.7.10
Schema documentation for `biblatexml` ........................ D
Sourcemap documentation and examples for `biblatexml` .... 4.5.3
Changes for name formats to generalise available name parts ... 4.4.2
`userefprefix` can now be specified per-namelist and per-name in `biblatexml` datasources 4.4.2
New sourcemap options for creating new entries dynamically and looping over map steps... 4.5.3

3.2 2015-11-06

256
Removed messy experimental endnote and zoterdfl support for biber

Added \nonameyeardelim ........................................... 3.10.1
Added \extpostnotedelim ........................................... 3.10.1

3.2 2015-12-28
Added pstrwidth and pcompound to \DeclareLabelalphaTemplate ...... 4.5.5
Added \AtEachCitekey ............................................ 4.10.6

3.1 2015-09
Added \DeclareNolabel ........................................... 4.5.5
Added \DeclareNolabelwidthcount ................................. 4.5.5

3.0 2015-04-20
Improved Danish (Jonas Nyrup) and Spanish (ludenticus) translations
labelname and labeltitle are now resolved by biblatex instead of biber for more flexibility and future extensibility
New \entryclone sourcemap verb for cloning entries during sourcemapping ........ 4.5.3
New \pernottype negated per-type sourcemap verb .......................... 4.5.3
New range calculation command \frangelen ................................ 4.6.4
New bibliography context functionality .................................. 3.7.10
Name lists in the data model now automatically create internals for \ifuse{name} tests and booleans .................... 3.1.3.1 and 4.6.2

2.9a 2014-06-25
resetnumbers now allows passing a number to reset to ..................... 3.7.2

2.9 2014-02-25
Generalised shorthands facility ......................................... 3.7.3
Sorting locales can now be defined as part of a sorting scheme ........ 4.5.6
Added sortinithash ................................................. 4.2.4.1
Added Slovene localisation (Tea Tušar and Bogdan Filipič)
Added \mkbibitalic .................................................. 4.10.4
Recommend begentry and finentry bibliography macros .................. 4.2.3

2.8a 2013-11-25
Split option language=auto into language=autocite and language=autobib 3.1.2.1

2.8 2013-10-21
New langidopts ....................................................... 2.2.3
hyphenation field renamed to langid .................................. 2.2.3
polyglossia support
Renamed babel option to autolang ...................................... 3.1.2.1
Corrected Dutch localisation
Added datelabel=year option ........................................... 3.1.2.1
Added datelabelsource field ........................................... 4.2.4.1

2.7a 2013-07-14
Bugfix - respect maxnames and uniquelist in \finalandsemicolon
Corrected French localisation

2.7 2013-07-07
Added field eventtitleaddon to default datamodel and styles ............. 2.2.2
Added \ifentryinbib, \iffirstcitekey and \iflastcitekey ............. 4.6.2
Added postpunct special field, documented multiprenote and multipostnote special fields 4.3.2
Added \UseBibitemHook, \AtEveryMultiCite, \AtNextMultiCite,
\UseEveryCiteHook, \UseEveryCitekeyHook, \UseEveryMultiCiteHook,
\UseNextCiteHook, \UseNextCitekeyHook, \UseNextMultiCiteHook,
\DeferNextCitekeyHook ............................................ 4.10.6
Fixed \textcite and related commands in the numeric and verbose styles ........ 3.8.2
Added multicite variants of \volcite and related commands

Added \finalandsemicolon

Added citation delimiter \textcitedelim for \textcite and related commands to styles

Updated Russian localisation (Oleg Domanov)

Fixed Brazilian and Finnish localisation

2.6 2013-04-30

Added \printunit

Added field clonesourcekey

New options for \DeclareLabelalphaTemplate

Added \DeclareLabeldate and retired \DeclareLabelyear

Added nodate localisation string

Added \rangelen

Added starred variants of \citeauthor and \Citeauthor

Restored original url format. Added urlfrom localisation key

Added \AtNextBibliography

Fixed related entry processing to allow nested and cyclic related entries

Added Croatian localisation (Ivo Pletikosić)

Added Polish localisation (Anastasia Kandulina, Yuriy Chernyshov)

Fixed Catalan localisation

Added smart "of" for titles to Catalan and French localisation

Misc bug fixes

2.5 2013-01-10

Made url work as a localisation string, defaulting to previously hard-coded value 'URL'.

Changed some biber option names to cohere with biber 1.5.

New sourcemap step for conditionally removing entire entries

Updated Catalan localisation (Sebastià Vila-Marta)

2.4 2012-11-28

Added relatedoptions field

Added \DeclareStyleSourcemap

Renamed \DeclareDefaultSourcemap to \DeclareDriverSourcemap

Documented \DeclareFieldInputHandler, \DeclareListInputHandler and \DeclareNameInputHandler.

Added Czech localisation (Michal Hoftich)

Updated Catalan localisation (Sebastià Vila-Marta)

2.3 2012-11-01

Better detection of situations which require a biber or \LaTeX re-run

New append mode for \DeclareSourcemap so that fields can be combined

Extended auxiliary indexing macros

Added support for plural localisation strings with relatedtype

Added \csfield and \usefield

Added starred variant of \usebibmacro

Added \ifbibmacrounder, \iffieldformatundef, \iflistformatundef and \ifnameformatundef

Added Catalan localisation (Sebastià Vila-Marta)

Misc bug fixes

2.2 2012-08-17

Misc bug fixes

Added \revsdnamepunct

Added \ifterseinits

2.1 2012-08-01

Misc bug fixes

Updated Norwegian localisation (Håkon Malmedal)

Increased data model auto-loading possibilities
2.0 2012-07-01

Misc bug fixes
Generalised singletitle test a little ........................................ 4.6.2
Added new special field extratitleyear ..................................... 4.2.4
Customisable data model .......................................................... 4.5.4
Added \DeclareDefaultSourcemap .............................................. 4.5.3
Added labeltitle option ........................................................... 3.1.2.3
Added new special field extratitle ............................................ 4.2.4
Made special field labeltitle customisable .................................. 4.2.4
Removed field reprinttitle ....................................................... 3.4
Added related entry feature ..................................................... 3.4
Added \DeclareNoInit .............................................................. 4.5.8
Added \DeclareNoSort ............................................................. 4.5.9
Added sorting option for \printbibliography and \printshorthands .... 3.7.2
Added ids field for citekey aliasing ......................................... 2.2
Added sortfirstinits option ..................................................... 3.1.2.3
Added data stream modification feature ..................................... 4.5.3
Added customisable labels feature .......................................... 4.5.5
Added \citeyear* and \citedate* ................................................. 3.8.5