

The Biblatex Package

Programmable Bibliographies and Citations

Philipp Lehman **Version 2.9a**
 (with Philip Kime, Audrey
 Boruvka and Joseph Wright) **24/06/2014**

Contents

List of Tables	1	4.5 Customization	422
1 Introduction	2	4.6 Auxiliary Commands .	505
1.1 About	2	4.7 Punctuation	559
1.2 License	3	4.8 Localization Strings .	575
1.3 Feedback	3	4.9 Localization Modules .	579
1.4 Acknowledgments . .	3	4.10 Formatting Commands	612
1.5 Prerequisites	4	4.11 Hints and Caveats . . .	649
2 Database Guide	13	Appendix	689
2.1 Entry Types	14	A Default Driver Source Map-	689
2.2 Entry Fields	54	pings	
2.3 Usage Notes	87	A.1 bibtex 689	
2.4 Hints and Caveats . . .	105	A.2 endnotexm	
3 User Guide	126	A.3 ris . . . 700	
3.1 Package Options	126	A.4 zoterordf	
3.2 Global Customization .	190	B Default Inheritance Setup	712
3.3 Standard Styles	191	C Default Sorting Schemes	722
3.4 Related Entries	210	C.1 Alphabetic 1	722
3.5 Sorting Options	215	C.2 Alphabetic 2	724
3.6 Bibliography Commands	218	C.3 Chronological	726
3.7 Citation Commands . .	254	D Option Scope	728
3.8 Localization Commands	282	E Revision History	739
3.9 Formatting Commands	286		
3.10 Language notes	309		
3.11 Usage Notes	312		
3.12 Hints and Caveats . . .	343		
4 Author Guide	356		
4.1 Overview	356		
4.2 Bibliography Styles . .	366		
4.3 Citation Styles	396		
4.4 Data Interface	403		

List of Tables

1 Biber/Biblatex compatibility matrix	13	4 Capacity of bibtex8	112
2 Supported Languages	78		
3 Date Specifications	100	5 Disambiguation counters . .	183

6	<code>mcite</code> -like commands	281	8	Date Interface	392
7	<code>mcite</code> -like syntax	283	9	Field types for <code>\mkcompran</code>	76
			10	setup	770

1 Introduction

This document is a systematic reference manual for the Biblalex package. Look at the sample documents which ship with Biblalex to get a first impression.¹ For a quick start guide, browse §§ 1.1, 2.1, 2.2, 2.3, 3.1, 3.3, 3.6, 3.7, 3.11.

1.1 About Biblalex

This package provides advanced bibliographic facilities for use with LaTeX in conjunction with BibTeX. The package is a complete reimplementation of the bibliographic facilities provided by LaTeX. A custom backend Biber by default is used which processes the BibTeX format data files and then performs all sorting, label generation (and a great deal more). Legacy BibTeX is also supported as a backend, albeit with a reduced feature set. Biblalex does not use the backend to format the bibliography information as with traditional BibTeX: instead of being implemented in BibTeX style files, the 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. There is no need to learn BibTeX's postfix stack language. 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. With Biber as the backend, features such as customisable sorting, multiple bibliographies with different sorting, customisable labels, dynamic data modification are available. Please refer to § 1.5.5 for information on Biber/Biblalex version compatibility. The package

is completely localized and can interface with the package. Please refer to table 2 for a list of languages currently supported by this package.

¹ `\biblalexctan/
doc/examples`

1.2 License

Copyright © 2006–2012 Philipp Lehman, 2012–2013 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 1.3.²

1.3 Feedback

Please use the Biblalex project page on GitHub to report bugs and submit feature requests.³ 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_{news-}
group or TeX-LaTeX Stack Exchange.⁴

1.4 Acknowledgments

The language modules of this package are made possible thanks to the following contributors: Augusto Ritter Stoffel, Mateus Araújo (Brazilian); Sebastià Vila-Marta (Catalan); Ivo Pletikosić (Croatian); Michal Hoftich (Czech); Johannes Wilm (Danish/Norwegian); Alexander van Loon, Pieter Belmans, Hendrik Maryns (Dutch); Hannu Väisänen, Janne Kujanpää (Finnish); Denis Bitouzé (French); Apostolos Syropoulos, Prokopis (Greek); Baldur Kristinsson (Icelandic); Enrico Gregorio, Andrea Marchitelli (Italian); Håkon Malmedal (Norwegian); Anastasia Kandulina, Yuriy

² <http://www.ctan.org/tex-archive/macros/latex/base/lppl.txt>

³ <http://github.com/plk/biblalex>

⁴ <http://tex.stackexchange.com/questions/tagged/biblalex>

Chernyshov (Polish); José Carlos Santos (Portuguese); Oleg Domanov (Russian); Tea Tušar and Bogdan Filipič (Slovene); Ignacio Fernández Galván (Spanish); Per Starbäck, Carl-Gustav Werner, Filip Åsblom (Swedish).

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 Biblalex to function. The package will not work if they are not available.

e-TeX The Biblalex 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. The Biblalex 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 **elatex** instead of **latex** or **pdfelatex** instead of **pdflatex**, respectively.

Biber Biber is the default backend of Biblalex. You only need one backend, either BibTeX or Biber. Biber comes with TeXLive and is also available from SourceForge.⁵ There are some slight differences in name parsing of ‘von’ parts with Biber as compared with BibTeX which you probably won’t normally notice. Biber uses the

btparse C library for BibTeX format file parsing which aimed to be compatible with BibTeX’s parsing rules but also aimed at

⁵ <http://biblalex-biber.sourceforge.net/>

correcting some of the common problems. For details, see the manual page for the Perl

Text::BibTeX_{module⁶}

BibTeX The Biblatex package can use legacy BibTeX as a data backend. While a legacy BibTeX

binary is sufficient to run Biblatex, using **bibtex8**
bibtex8,
is recommended. If your TeX distribution does not ship with
you can get it from CTAN.⁷

etoolbox This LaTeX package, which is loaded automatically, provides generic programming
facilities required by Biblatex. It is available from CTAN.⁸

kvoptions This LaTeX package, which is also loaded automatically, is used for internal option

⁶ <http://search.cpan.org/~ambs/Text-BibTeX>
⁷ <http://www.ctan.org/tex-archive/biblio/bibtex/8-bit/>
⁸ <http://www.ctan.org/tex-archive/macros/latex/contrib/etoolbox/>

oberdiek

handling. It is available with the package bundle from CTAN.⁹

`logreq` 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.¹⁰

Apart from the above resources, Biblatex also requires the standard LaTeX packages

`keyval` and `ifthen` as well as the `url` package. These package are included in all common TeX distributions and will be loaded automatically.

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 provides 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. If loaded, Biblatex package will detect `babel`

⁹ <http://www.ctan.org/pkg/kvoptions>

¹⁰ <http://www.ctan.org/tex-archive/macros/latex/contrib/logreq/>

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 any other language, loading

csquotes is strongly recommended.¹¹

xpatch The **xpatch** package extends the patching commands

of **etoolbox** to Biblatex bibliography macros, drivers and formatting directives.¹²

1.5.3 Compatible Classes and Packages

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

¹¹ <http://www.ctan.org/tex-archive/macros/latex/contrib/csquotes/>

¹² <http://www.ctan.org/tex-archive/macros/latex/contrib/xpatch/>

`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 § 3.12.2 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 § 3.12.1 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
§ 3.8.

backref
[backref](#) The package creates back references in the bibliography. See the package
options **hyperref** and
backref in § 3.1.2.1 for comparable func-
tionality.

bibtopic The package provides
support for bibliographies subdivided by topic, type, or other crite-
ria. For bibliographies subdivided by topic, see the category feature
in § 3.6.7 and the corresponding filters in § 3.6.2. Alternatively, you
may use the **keywords** field in
conjunction with the **keyword** and
notkeyword filters for
comparable functionality, see §§ 2.2.3 and 3.6.2 for details. For bib-
liographies subdivided by type, use the **type** and
nottype filters. Also see § 3.11.4 for ex-
amples.

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 **section** environment and the **section** filter for comparable functionality. Alternatively, you might also want to use the

refsegment **segment** environment and the **segment** filter. See §§ 3.6.5, 3.6.6, 3.6.2 for details. Also see § 3.11.3 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** **numeric-comp** package option in § 3.1.2.1 and the **numeric-comp** citation style in § 3.3.1. For configurable punctuation, see § 3.9.

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 §§ 3.11.5, 3.6.11, 3.7.10 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 § 3.7, and the facilities discussed in §§ 3.6.8, 3.6.9, 3.9 for comparable functionality. Also see § 3.7.9.

splitbib

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

bibtopic.

titlesec

The titlesec package redefines user-level document division commands

\chapter

such as or

\section

. This approach is not compatible

with internal command changes applied by the Biblatex

section
segment

and

ref-
ref-

option settings described in § 3.1.2.1.

ucs

The ucs package provides support for UTF-8 encoded in-

put. 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 warn you during processing if it encounters

<i>Biber version</i>	<i>Biblatex version</i>
1.9	2.9
1.8	2.8
1.7	2.7
1.6	2.6
1.5	2.5
1.4	2.4
1.3	2.3
1.2	2.1, 2.2
1.1	2.1
1.0	2.0
0.9.9	1.7x
0.9.8	1.7x
0.9.7	1.7x
0.9.6	1.7x
0.9.5	1.6x
0.9.4	1.5x
0.9.3	1.5x
0.9.2	1.4x
0.9.1	1.4x
0.9	1.4x

Table 1: Biber/Biblatex compatibility matrix

information which comes from a Biblatex version which is incompatible. table 1 shows a compatibility matrix for the recent versions.

2 Database Guide

It is important to distinguish between BibTeX the program and BibTeX the file format. Biblatex can be used with or without BibTeX the program since its default

backend Biber uses fully supports the BibTeX file format (**bib**) and users should be able to move to Biblatex will little or no changes to their BibTeX data files when using Biber as a backend. If using BibTeX as the data backend, note

that you cannot use arbitrary **bst** files because the package depends on a special BibTeX interface. When using BibTeX as a backend, Biblatex uses its

own special **bst** file only. The entry guide below is backend agnostic unless otherwise stated.

This section describes the default data model defined in the **blx-dm.d** file which is part of **biblatex**. The data model is defined using the macros documented in § 4.5.3. 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 file and to read § 4.5.3.

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,
seriesvolume,
numbereid,
issuemonth,
pagesversion,
noteissn,
addendum.

[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,
edition¹⁸,
volumes,

`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,
seriesnumber,
note,
publisher,
location, isbn,

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,
mainsubtitle,
maintitleaddon,
booksubtitle,
booktitleaddon,
language,
origlanguage,
volume part

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 **howpublished** field to supply publishing information in free format, if applicable. The **type** field 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,
origlanguage,
volume part,
edition,
volumes,
series number,

`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,
origlanguage,
edition,
volumes,
seriesnumber,
note,
publisher,
location,isbn,
pagetotal,

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,
booksubtitle,
booktitleaddon,
language,
origlanguage,
volume part,

[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,
editiontype,
seriesnumber,
versionnote,
organization,
publisher,
location,isbn,
chapterpages,
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 applica-

ble. 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,
author,

[online](#) An online resource.

`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,
url, date

patent A patent or patent request. The number or record token is given in the

number field. Use the type field

location to specify the type and the 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,
title add-on,
type, version,
location, note,
date, month,
year, addendum,
pubstate, doi,
eprint,
eprint class,
eprint type,
url, url date

[periodical](#) An complete issue of a periodical, such as a special issue of a journal. The ti-

title
title of the periodical is given in the field. If the
issue has its own title in addition to the main title of the periodical, it goes

issuetitle
in the field. The

editor
is omissible in terms of § 2.3.2.

editor,
Required fields:

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

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,

`mvproceedings`

A multi-volume

entry.

proceedings

@proceedings

For

backwards

compatibility,

multi-volume

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,
eventtitle,
eventtitleaddon,
eventdate,
venue,
language,
volumes,
seriesnumber,
note,
organization,
publisher,
location,
month,isbn,
pagetotal,
addendum,
pubstate,doi,
eprint,
eprintclass

`inproceedings` An article in a conference proceedings. This type is similar to

`@incollection`
`organization`

It supports an optional
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,

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 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 uni-

versity 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,
versionnote,
location,
month, isrn,
chapter, pages,
pagetotal,
addendum,
pubstate, doi,
eprint,
eprintclass,
eprinttype,
url, urldate

[set](#) An entry set. This entry type is special, see § 3.11.5 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.

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

Optional fields: subtitle,
titleaddon,
language, note,
location,
month, isbn,
chapter, pages,
page total,
addendum,
pubstate, doi,
eprint,
eprintclass,
eprinttype,
url, urldate

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
note

and 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 `Biber only`

`xdata`
which may be inherited by other entries using the
field. Entries of this type only serve as data containers; they may not be cited or
added to the bibliography. See § 3.11.6 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**.

@inproceedings
conference A BibTeX legacy alias for

@online
electronic An alias for

@thesis
mastersthesis Similar to **type** except that the field is optional and defaults to the localized term

type
'Master's thesis'. You may still use the **type** field to override that.

@thesis
phdthesis Similar to **type** except that the field is optional and defaults to the localized term

type
'PhD thesis'. You may still use the **type** field to override that.

@report
techreport Similar to **type** except that the field is optional and defaults to the localized term

type
'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**.

[bibnote](#) This special entry type is not meant to be used in the **bib** file like other types. It is provided for third-party pack-

ages 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.

[commentary](#) Commentaries which have a status different from regular books, such as legal commentaries.

image Images, pictures, photographs, and similar media.

jurisdiction Court decisions, court recordings, and similar things.

legislation Laws, bills, legislative proposals, and similar things.

legal Legal documents such as treaties.

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

movie Motion pictures. See also **@video**.

music Musical recordings. This is a more specific variant of

@audio.

performance 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.

review Reviews of some other work. This is a more specific vari-

ant of the **@article** type.

The standard styles will treat this entry type as an alias for

@article.

software Computer software.

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

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

@movie.

2.2 Entry Fields

This section gives an overview of the fields supported by the Biblatex default data model. See § 2.2.1 for an introduction to the data types used by the data model specification and §§ 2.2.2 and 2.2.3 for the actual field listings.

2.2.1 Data Types

In datasources such as a **bib** file, all bibliographic data is specified in **author** fields. Some of those fields, for example and

editor , may contain a list of items. This list structure

and 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 § 4.5.3). This section gives an overview of the data types supported by this package. See §§ 2.2.2 and 2.2.3 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 four name components: the first name, the name prefix (von, van, of, da, de, della, ...), the last name, and the name suffix (junior, senior,

...). Name lists may be truncated in the bib file with the

keyword ‘and others’.

Typical examples of name lists are author and

editor.

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 localization keys. For each item in the list, a test is performed to determine whether it is a known localization key (the localization keys defined by default are listed in § 4.9.2). If so, the localized 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 option-

ally 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 Biber only

command which can be used to customise the separator between multiple ranges. With Biber, 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

\DeclareSourcemacro

(see § 4.5.2).

Integer fields hold unformatted integers which may be converted to ordinals or strings as they are printed. A typical example is the

extrayear field.

Datepart fields hold unformatted integers which may be converted to ordinals or strings as they are printed. A typical example is the

month field.

Date fields hold a date specification in **yyyy-mm-dd** format or a date range in **yyyy-mm-dd/yyyy-mm-dd** format. Date fields are special in that the date is parsed and split up into its components. See § 2.3.8 for details. 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 characters. 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 is always a comma when using BibTeX as the backend but can be configured to be any Perl regular expression when using Biber 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 localization keys. A test is performed to determine whether the value of the field is a known localization key (the localization keys defined by default are listed in § 4.9.2). If so, the

localized 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 Biblatex and its backend. The 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 § 3.6.4. Biblatex automatically creates supporting macros for such fields. See § 3.6.4.

Biber only

abstract field (literal)

bib

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

addendum field (literal)

Miscellaneous bibliographic data to be printed at the end of the entry. This is

note

similar to the **note** field except that it is printed at the end of the bibliography entry.

afterword list (name)

The author(s) of an afterword to the work. If the author of the after-

word is identical to the **editor** and/or

translator, the standard styles will automatically concatenate these fields in the bibliography. See also

introduction and **foreword**.

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 com-

pletely unrelated to **annotator**. The

annotator is the author of annotations which are part of the work cited.

annotator list (name)

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

commentator.

author list (name)

The author(s) of the **title**.

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.

bookauthor list (name)

The author(s) of the **booktitle**.

bookpagination field (key)

If the work is published as part of another one, this is the pagination scheme of the enclosing work, i. e.,

bookpagination

relates to **pagination**

like **booktitle** to

title.

The value of this field will affect the formatting of the **pages** and

pagetotal fields. The key should be given in the singular form. Possible keys are **page**, **column**, **line**, **verse**, **section**, and **paragraph**. See also **pagination** as well as § 2.3.10.

booksubtitle field (literal)

The subtitle related to the **booktitle**.
subtitle
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 **subtitle**.
this field. See also **subtitle**.

booktitle field (literal)

title
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)

booktitle
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.

edition 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`
`editor` 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.

`editortype` field (key)

The type of editorial role performed by the
`editor`. Roles supported by de-
fault are `editor`, `com-
piler`, `founder`,
`continuator`,
`redactor`, `re-`

viser, collaborator, editor.

The role 'editor' is the default. In this case, the field is omissible. See § 2.3.6 for further hints.

editoratype field (key)

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

editorbtype field (key)

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

editorctype field (key)

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

eid field (literal)

The electronic identifier of an @article.

entrysubtype 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.

eprint 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 § 3.11.7 for de-

tails. Also see **eprinttype** and **eprintclass**.

eprintclass field (literal)

Additional information related to the resource indicated by the **eprinttype** field. This could be a section of an archive, a path indicating a service, a classification of some sort,

etc. See § 3.11.7 for details. Also see **eprint** and **eprinttype**.

eprinttype field (literal)

The type of **eprint** identifier, e. g., the name of the

archive, repository, service, or system the **eprint**

field refers to. See § 3.11.7 for details. Also see **eprint**

and **eprintclass**.

eventdate 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**

eventdate field, respectively. See also **venue** and

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 fore-

word 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 in-
troduction is identical to the **editor** and/
or **translator**, the stan-
dard styles will automatically concatenate these fields in the biblio-

graphy. 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)

The International Standard Technical Report Number of a technical report.

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 sim-

ilar 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 localization keys. If localization 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, de-

pending on the entry type. Traditional BibTeX uses the field name

address, which is supported as an alias.
@patent

See also §§ 2.2.5 and 2.3.4. With 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**.
subtitle

See also

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}
month={1}

but . The bibliography style converts this to a language dependent string or ordinal where required. See

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 (literal)

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.

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 edi-
tion. 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 origi-
nal 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 origi-
nal 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**.

paragraph. bookpagination

and . See also

as well as §§ 2.3.10 and 3.12.3.

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**
part field and the number of the part of that volume in the 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.

BibTeX only

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)

Label field

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) Label field

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) Label field

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) Label field

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

shortseries field (literal) Label field

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

shorttitle field (literal) Label field

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 **editor**, entry type. If the translator is identical to the the standard styles will automatically concatenate these fields in the bibliography.

`type` field (key)

The type of a **manual**, **patent**, **report**, **thesis**, or . 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), with Biber, 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

location § 2.1.3. Note that the list holds the place of publication. It therefore corresponds

to the publisher and
institution
venue lists.
The location of the event is given in the
eventdate and
eventtitle field. See also

version field (literal)

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

volume field (literal)

The volume of a multi-volume book or a periodical. See also

part.

volumes field (literal)

The total number of volumes of a multi-volume work. Depending on

the entry type, this field refers to title or
maintitle.

year field (literal)

The year of publication. See also date and § 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 § 3.11.5 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**

bib from § 4.10.6, except that it is definable on a per-entry basis in the 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 localize strings in the bibliography. Note that the language names are case sensitive. The languages currently supported by this pack-

age are given in table 2. Note that **babel**

treats the identifier **english**

as an alias for **british** or

Language	Region/Dialect	Identifiers
Catalan	Spain, France, Andorra, Italy	catalan
Croatian	Croatia, Bosnia and Herzegovina, Serbia	croatian
Czech	Czech Republic	czech
Danish	Denmark	danish
Dutch	Netherlands	dutch
English	USA	american,
		USenglish,
		english
	United Kingdom	british,
		UKenglish
		canadian
	Canada	australian
	Australia	newzealand
	New Zealand	finnish
	Finland	french
French	France, Canada	german
German	Germany	austrian
	Austria	ngerman
German (new)	Germany	naustrian
	Austria	greek
Greek	Greece	italian
Italian	Italy	norwegian,
Norwegian	Norway	norsk,

american, depending on the
babel version. The Biblatex package always treats it as

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 in
§ 2.2.2.

langidopts field (literal)

For polyglossia users,
allows per-entry language specific options. The literal value of this field is

polyglossia's
passed to language switching facility when using the package option

autolang=langname

For example, the fields:

langid	=
langidopts	=

would wrap the bibliography entry in:

`\english[variant=`
`\endenglish`

`ids` field (separated list of entrykeys)

Biber only

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 `.bb1`.

`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:

<code>title</code>	<code>=</code>
<code>indextitle</code>	<code>=</code>
<code>indexsorttitle</code>	<code>=</code>

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 §§ 3.6.2 and 3.11.4), they are usually not printed. Note that with the default separator (comma), spaces around the separator are ignored.

options field (separated $\langle key \rangle = \langle value \rangle$ options)

A separated list of entry options in $\langle key \rangle = \langle value \rangle$ 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 § 4.5.5. This field is consumed by the backend processing

and does not appear in the .bb1.

related field (separated values)

Biber only

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)

Biber only

Per-type options to set for a related entry. Note that this does not set the options on

the related entry itself, only the clone which is used as a datasource for the parent entry.

dataonly

relatedtype field (identifier)

Biber only

An identifier which specified the type of relationship for the keys listed in the

related

field. The identifier is a localized 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)

Biber only

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 bibliogra-

phy. 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)

Biber only

Similar to **sortkey** but used in the list of shorthands. If present, Biblatex uses this field in-

stead of **shorthand** when sorting the list of shorthands. This is useful if the

shorthand field holds short-
hands with formatting commands such as **\emph** or

`\textbf`. 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 bibliogra-

phy. The `sorttitle` field may come in handy if you have an entry with a title like “An Introduction to...” and want that alphabetized under ‘T’ 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 (literal)

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)

Biber only

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 paren-

t/child relation without inheriting data; **xdata** inherits

xdata
data without establishing a relation. The value of the
may be a single entry key or a separated list of keys. See § 3.11.6 for further de-
tails. This field is consumed by the backend processing and does not appear in the

.bb1.

xref field (entry key)

This field is an alternative cross-referencing mechanism. It differs from

crossref in that the child entry will not

xref
inherit any data from the parent entry specified in the
field. If the number of child entries referencing a specific parent entry hits
a certain threshold, the parent entry is automatically added to the bibliogra-
phy even if it has not been cited explicitly. The threshold is settable with the

mincrossrefs pack-
age 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 Bibla-
tex level depends on the availability of the parent entry. If the parent entry

is available, the **xref** fields of the child entries will be de-

xref
fined. If not, their **xref** fields will be undefined. Whether
the parent entry is added to the bibliography implicitly because of the thresh-
old 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**type and **editor**type but referring to the **name[a-c]** fields. 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,

bib 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 3.11.7.

`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 3.11.7.

`school` list (literal)

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 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

bib files which were generated with a traditional BibTeX style in mind. Unfortunately, it is not possible to handle all legacy files automatically because Biblatex's data model is slightly different from traditional BibTeX. Therefore, such **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 **@inbook**. See §§ 2.1.1 and 2.3.1 for details.
- The fields **institution**, **organization**, **publisher** and **address** as well as the aliases **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 **jurabib** package should note that the **shortauthor** field is treated as a name list by Biblatex, see § 2.3.3 for details.

2.3.1 The Entry Type **@inbook**

Use the **@inbook** entry type for a self-contained part of a book with its own title

only. It relates to **@book** just like

@incollection

relates to **@collection**. See § 2.3.5 for examples. If you want to refer to a chapter

or section of a book, simply use the **book**

type and add a **chapter** and/or

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.

label You may generally use the **label** field to provide a substitute for any missing data required for citations. How the

label field is employed depends on the citation style. The author-year citation styles which come with this pack-

age use the **label** field as a fallback if either the

author / **editor** or **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

title 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

label 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 em-

ployes 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` = { {
`shortauthor` = { N

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` = { {

shorteditor = {N

Users switching from the jurabib package to
Biblatex should note that the 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
address and to the field aliases

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	=	{W
institution	=	{O
organization	=	{A
		a
		A

Note the difference between a literal ‘**{and}**’ and the list separator ‘**and**’ in the above examples. You may also wrap the entire name in braces:

publisher	=	{ {
institution	=	{ {
organization	=	{ {
		a
		{

Legacy files which have not been updated for use with 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.

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:

@MvBook{works,		
author	=	{S
title	=	{C
volumes	=	{5
. . .		

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

@Book{works:4,		
author	=	{S
maintitle	=	{C
title	=	{S
volume	=	{4
. . .		

If the individual volumes do not have a title, we put the main title in the

title	field and include a volume number:
-------	------------------------------------

@Book{works:4,		
author	=	{S
title	=	{C

volume = {4
.
.
.

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:

@InBook{lear,
author = {S
bookauthor = {S
maintitle = {C
booktitle = {T
title = {K
volume = {1
pages = {5
.
.
.

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:

@InBook{stage,
author = {E
title = {S

bookauthor	=	{S
editor	=	{B
maintitle	=	{C
booktitle	=	{T
volume	=	{1
pages	=	{7

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

fields (i. e., `editora`, `editorb`, `editorc`) may be specified in the corresponding

`editor...type` field. The following roles are supported by default. The role

`editor` is the default. In this case, the `editor` field is omissible.

`editor` The main editor. This is the most generic editorial role and the default value.

editor

compiler Similar to 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 corre-

sponding **editortype** field:

```
@Collection{ . . . ,
    editor      = {
    editortype  = {
    . . . }
```

There may also be secondary editors in addition to the main editor:

```
@Book{ . . . ,
    author      = {
    editor      = {
    editora     = {
```


editoratype	=	{
editorb	=	{
editorbtype	=	{
. . .		

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:

@Book{ . . . ,		
author	=	{
editor	=	{
editora	=	{
editoratype	=	{
. . .		

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:

@Collection{ . . . ,		
editor	=	{

editortype = {
 editora = {
 . . .

You may add more roles by initializing and defining a new localization key whose name corresponds to the identifier in the

editor...type

field. See §§ 3.8 and 4.9.1 for details.

2.3.7 Publication and Journal Series

series

The 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
 series

field for multi-volume works and employs for publication series only. The volume or number of a book in the series goes in the

number field in this case:

@Book{ . . . ,
 author =
 title =
 series =
 number =
 . . .

The **@article** entry type makes use of the **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 localization key. If so, the localized string is printed. If not, the value is printed as is. Consider the following example of a journal published in numbered series:

```
@Article{... ,
  journal
  series
  volume
  number
  year
  . . .}
```

This entry will be printed as “*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 **series** field as well, either as a literal string or as a localization key. Consider the following example which makes use of the localization key

```
newseries:
@Article{... ,
  journal
```

1850
1997/
1967-02

2009-01-31

1988/1992
2002-01/2002-02

1995-03-30/1995-04-05

Table 3: Date Specifications

series
volume
year
■ ■ ■

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

2.3.8 Date Specifications

The date fields `date`, `origdate`, `eventdate`, and `urldate` require a date specification in `yyyy-mm-dd` format. Date ranges are given as `yyyy-mm-dd/` or `yyyy-mm-dd`. Partial dates are valid provided that date components are omitted at the end only. You may specify an open ended date range by giving the range separator and omitting the end date (e. g., `yyyy/`). See table 3 for some examples of valid date specifications and the formatted date 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 `year` and `month` fields for backwards compatibility with traditional BibTeX. Style author 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 components are made available to styles by way of the special fields discussed in § 4.2.4.3. See this section and table 8 on page 392 for further information.

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

date the 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 field of an entry or in the $\langle postnote \rangle$ 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:

```
@InBook{key,  
  title  
  pagination  
  booktitle  
  bookpagination  
  pages  
}
```

The `bookpagination` and `pages`

field affects the formatting of the `pagetotal` fields in the list of references. Since `page` is the default, this field is omis-

sible 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 re-

flected by the

field, which affects the formatting of the `<postnote>` argument to any citation command. With a citation like

```
\cite[17]{key},
```

the postnote will be formatted as ‘v. 17’. Setting the

field to

would yield ‘§ 17’. See § [3.12.3](#)

for further usage instructions.

The `pagination` and `bookpagination`

fields are key fields. This package will try to use their value as a localization 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 localization keys for the pagination scheme required by a certain entry, you can simply add them. See the commands

`\NewBibliographyS`
and `\DefineBibliograph`

in § 3.8. You need to define two localization strings for each additional pagination scheme: the singular form (whose localization key corresponds to the value of

the `pagination` field) and the

plural form (whose localization 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

2.4.1.1 The `crossref` field (BibTeX)

The `crossref` field is a convenient way to establish a parent/child relation between two associated entries. Unfortunately, the BibTeX program uses symmetric field mapping which reduces the usefulness of the

`crossref` field significantly. There are two issues with symmetric field mapping, as seen in the following example:

`@Book{book,`
`author`
`bookauthor`

=
=

title	=
booktitle	=
subtitle	=
booksubtitle	=
publisher	=
location	=
date	=
}	
@InBook{inbook,	
crossref	=
title	=
subtitle	=
pages	=
}	

As BibTeX is not capable of mapping the **title** field of the parent to the **booktitle** field of the child, the title of the book needs to be given twice. The style then needs to ignore the **booktitle** of the parent since it is only required to work around this fundamental limitation of BibTeX. The problem with the **subtitle** field is the inverse of that. Since the **subtitle** of the parent would become the **booksubtitle**, rather than in the **subtitle** of the child, we need to add an empty **booksubtitle** field to the child entry to prevent inheritance of this field. Of course we also need to duplicate the subtitle in the parent entry to ensure that it is available as **booksubtitle** in the child entry. In short, using BibTeX's **crossref** field tends to bloat database files and corrupt the data model.

2.4.1.2 The **crossref** field (Biber)

With Biber, the limitations of BibTeX's **crossref** field belong to the past. 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:

@Book{book,

```

author
title
subtitle
publisher
location
date
}
@InBook{inbook,
  crossref
  title
  pages
}
title
booktitle
subtitle
booksubtitle

```

The title field of the parent will be copied to
the booktitle field of the
child, the subtitle becomes the
booksubtitle.

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	=
bookauthor	=
title	=
booktitle	=
booksubtitle	=
publisher	=
location	=
date	=
pages	=

See appendix [B](#) for a list of mapping rules set up by default. Note that all of this is customizable. See § [4.5.10](#) on how to configure Biber's cross-referencing mechanism. See also § [2.2.3](#).

2.4.1.3 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

mincrossrefs package option from § 3.1.2.1. The **xref** field is supported with all backends. See also § 2.2.3.

2.4.2 Capacity Issues

2.4.2.1 BibTeX

A venerable tool originally developed in the 1980s, BibTeX uses static memory allocation, much to the dismay of users working with large bibliographical

databases. With a large **bib** file which contains several hundred entries, BibTeX is very likely to run out of memory. The number of entries it can cope with depends on the number of fields defined by the BibTeX style

bst (). Style files which define a considerable number of fields, such

as **biblatex.bst**, are more likely to trigger such problems. Unfortunately, traditional BibTeX does not output a clear error message when it runs out of memory but exposes a rather cryptical kind of faulty behavior. The warning messages printed in this case look like this:

Warning - - I'm igno

--line 422 of fil
Warning--I'm igno
--line 423 of fil

These warning messages could indeed indicate that the en-

try Jones1995 is
faulty because it includes two year and two

volume fields. If that is not the case and the
bib file is fairly large, this is most likely a capacity issue. What
makes these warnings so confusing is that they are not tied to a specific entry. If
you remove the allegedly faulty entry, a different one will trigger similar warnings.

bibtex8
This is one reason why switching to
or Biber is advisable.

2.4.2.2 **bibtex8**

bibtex8 is a venerable tool as well and will also
run out of memory with its default capacity. Switching from traditional BibTeX

bibtex8 is still an improvement because the
capacity of the latter may be increased at run-time via command-line switches and it
also prints unambiguous error messages, for example:

17289 fields:
Sorry - - - you've ex

	Default	—big—huge—wolfg
max_cites	750	5000
mcites	2000	7500
max_ent_ints	3000	7500
ment_ints	4000	
max_ent_strs	3000	10000
ment_strs	6000	
max_fields	17250	125000
mfields	20000	
max_strings	4000	30000
mstrings	10000	
pool_size	65530	750000
mpool	20000	
wiz_fn_space	3000	10000
mwizfuns	6000	
hash_prime	4253	30011
hash_size	5000	35000
	10000	

Table 4: Capacity and Switches of

bibtex8

Table 4 gives an overview of the various capacity parameters of

bibtex8 and the command-line switches used to increase their default values. There are two ways to increase the capacity on the command-line. You may use a high-level switch like

--huge to select a different set of defaults or low-level

switches such as **--mfields** to modify a single parameter. The first thing you should always

do is run **bibtex8** with the

--wolfgang switch. Don't

even bother trying anything else. With a very large database, however, even that capacity may be too small. In this case, you need to resort to the low-level switches.

Here is an example of a set of switches which should cope with a **bib** file containing about 1000 entries:

```
bibtex8 --wolfgang
--mfields 25000
auxfile
```

When taking a closer look at table 4, you will notice that there are two parameters which can not be modified directly,

hash_prime and

hash_size. Increasing these

values is only possible with the high-level switches. That is why the above com-

mand includes the **--wolfgang**

switch in addition to the low-level switches. This situation is very unfortunate

because the hash size effectively sets a cap on some other parameters. For ex-

ample, `max_strings` can

not be greater than `hash_size`.

If you hit this cap, all you can do is recompile

`bibtex8` with a larger capacity. Also note that

the `wiz_fn_space`

parameter is not related to the `bib` file but

to the memory requirements of the `bst` file.

`biblatex.bst`

needs a value of about 6000. The value 10000 implicitly used by the

`--wolfgang` switch is fine.

2.4.2.3 Biber

Biber eliminates all of the above limitations.

2.4.3 Sorting and Encoding Issues

2.4.3.1 BibTeX

Traditional BibTeX can only alphabetize Ascii characters correctly. If the bibliographic data includes non-Ascii characters, they have to be given in Ascii notation. For example, instead of typing a letter like ‘ä’ directly, you need to input it as

`\”a`, using an accent command and the Ascii letter. This Ascii notation needs to be wrapped in a pair of curly braces. Traditional BibTeX will then ignore the accent and use the Ascii letter for sorting. Here are a few examples:

<code>author</code>	<code>= {S{\</code>
<code>editor</code>	<code>= {Ma{</code>

translator = {Fer
title = {\0

Apart from it being inconvenient, there are two major issues with this convention. One subtle problem is that the extra set of braces suppresses the kerning on both sides of all non-Ascii letters. But first and foremost, simply ignoring all accents may not be the correct way to handle them. For example, in Danish, the letter ‘å’ is the very last letter of the alphabet, so it should be alphabetized after ‘z’. BibTeX will sort it like an ‘a’. The ‘æ’ ligature and the letter ‘ø’ are also sorted after ‘z’ in this language. There are similar cases in Norwegian. In Swedish, the letter ‘ö’ is the very last letter of the alphabet and the letters ‘å’ and ‘ä’ are also alphabetized after ‘z’, rather than like an ‘a’. What’s more, even the sorting of Ascii characters is done in a rather peculiar way by traditional BibTeX because the sorting algorithm uses Ascii codepage

order (0 - 9, A - Z, a - z). This implies that the lowercase letter ‘a’ would end up after the uppercase ‘Z’, which is not even acceptable in the language BibTeX was originally designed for. The traditional

bst files work around this problem by converting all strings used for sorting to lowercase, i. e., sorting is effectively case-insensitive. See also § 2.4.3.4.

2.4.3.2 bibtex8

Switching to bibtex8 will help in such cases. bibtex8 can sort case-sensitively and it can handle 8-bit characters properly, provided that

you supply it with a suitable csf file and give the --csfile switch on the command line. This also implies that it is possible to apply language specific sorting rules to the biblio-

graphy. The Biblatex package comes with csf files for some common Western European encodings. bibtex8

also ships with a few **csf** files. Note that

biblatex.bst
can not detect if it is running under traditional BibTeX

or **bibtex8**, hence the

bibtex8 package option. By default, sorting is case-insensitive since this is required for traditional BibTeX. If the

bibtex8 package option is enabled, sorting is case-sensitive.

Since **bibtex8** is backwards compatible with traditional BibTeX, it is possible to mix 8-bit input and Ascii notation. This is

useful if the encoding used in the **bib** file does not cover all required characters. There are also a few marginal cases in which the Ascii notation scheme would yield better sorting results. A typical example is the ligature ‘œ’.

bibtex8 will handle this ligature like a single

character. Depending on the sorting scheme defined in the **csf** file, it could be treated like an ‘o’ or alphabetized after the letter ‘o’ but it can not be sorted

as ‘oe’. The Ascii notation (**\oe**) is equivalent to ‘oe’ during sorting:

title	=	{Æuv
title	=	{\0

Sometimes even that is not sufficient and further tricks are required. For example, the letter ‘ß’ in German is particularly tricky. This letter is essentially alphabetized as ‘ss’ but after ‘ss’. The name ‘Baßmann’ would be alphabetized as follows: Basmann/Bassmann/Baßmann/Bastmann. In this case, the Ascii notation

(`\ss`) would yield slightly better sorting results than ‘ß’ in conjunction with a `csf` file which treats ‘ß’ like ‘s’:

`author` = `{Ba{`

To get it absolutely right, however, you need to resort to the `sortname` field:

`author` = `{Baß`
`sortname` = `{Bas`

Not only BibTeX, LaTeX needs to know about the encoding as well. See § 2.4.3.4 on how to specify encodings.

2.4.3.3 *Biber*

Biber handles Ascii, 8-bit encodings such as Latin 1, and UTF-8. It features true

`bib` Unicode support and is capable of reencoding the 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 #10.¹³ Collation tailoring based on the Unicode Common Locale Data Repository (CLDR) is also supported.¹⁴ The bottom line is that Biber will deliver sorting results far superior

`bibtex8` to both BibTeX and in many cases. If

¹³ <http://unicode.org/reports/tr10/>

¹⁴ <http://cldr.unicode.org/>

you are interested in the technical details, section 1.8 of Unicode Technical Standard #10 will provide you with a very concise summary of why the inadequateness of

traditional BibTeX and even **bibtex8** is of a very general nature and not limited to the lack of UTF-8 support.¹⁵

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 example, by using the Unicode Collation Algorithm, Biber can handle the character ‘ß’ mentioned as an example in § 2.4.3.2 without any manual intervention. All you need to do to get localized sorting is specify the locale:

\usepackage[backe

or if you are using german as the main document language via Babel or Polyglossia:

\usepackage[backe

This will make Biblalex pass the Babel/Polyglossia main document language as the locale which Biber will map into a suitable default locale. 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 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 UTF-8 input. See § 2.4.3.4 on how to specify input and data encodings properly.

2.4.3.4 Specifying Encodings

When using a non-Ascii encoding in the **bib** file, it is important to understand what Biblalex 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

¹⁵ http://unicode.org/reports/tr10/#Common_Misperceptions

imported from the `bbl` file is interpreted correctly, provided that
 the `bibencoding` package option is set properly. Depending on the backend, the BibTeX side may de-
 mand attention, too. When using `bibtex8`,
 you need to supply `bibtex8` with a match-
 ing `csf` file as it needs to know about the encoding of
 the `bib` file to be able to alphabetize the entries correctly.
 Unfortunately, there is no way for BibLaTeX to pass this information to
`bibtex8` automatically. The only way is setting
 its `--csfile` option on the command line
 when running `bibtex8`. When using Biber,
 all of this is handled automatically and no further steps, apart from setting
 the `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 `tex` and the `bib`
 file with pdfTeX or traditional TeX (this will work with BibTeX,
`bibtex8`, and Biber):

```
\usepackage{bibl
```

- Latin 1 encoding (ISO-8859-1) in the `tex` file, Ascii nota-

tion in the `bib` file with pdfTeX or traditional TeX (BibTeX,
`bibtex8`, Biber):

```
\usepackage[lati
\usepackage[bibe
tex
```

- Latin 9 encoding (iso-8859-15) in both the `bib`
 and the `bib` file with pdfTeX or traditional TeX
 (`bibtex8`, Biber):

```
\usepackage[lati
\usepackage[bibe
```

Since `bibencod-`
`ing=auto` is the default setting, the
 option is omissible. The following setup will have the same effect:

```
\usepackage[lati
\usepackage{bibl
```


- UTF-8 encoding in the `tex` file, Latin 1 (ISO-8859-1) in the `bib` file with pdfTeX or traditional TeX (`bibtex8`, Biber):

```
\usepackage[utf8]
\usepackage[bibe
```

The same scenario with XeTeX or LuaTeX in native UTF-8 mode:

```
\usepackage[bibe
```

- Using UTF-8 encoding in both the `tex` and the `bib` file is not possible with traditional BibTeX or `bibtex8` since neither of them is capable of handling UTF-8. Unless you switch to Biber, you need to use an 8-bit encoding such as Latin 1 (see above) or resort to Ascii notation in this case:

```
\usepackage[utf8]
\usepackage[bibe
```

The same scenario with XeTeX or LuaTeX in native UTF-8 mode:

```
\usepackage[bibe
```

Biber can handle Ascii notation, 8-bit encodings such as Latin 1, and UTF-8. It

is also capable of reencoding the `bib` data on the fly (replacing the limited macro-level reencoding feature of Bibl_{at}ex). This will happen automatically

if required, provided that you specify the encoding of the `bib` files properly. In addition to the scenarios discussed above, Biber can also handle the following cases:

- Transparent UTF-8 workflow, i.e., UTF-8 encoding in both the

`tex` and the `bib` file with pdfTeX or traditional TeX:

```
\usepackage[utf8]
\usepackage[bibe
```

Since `bibencod-`
`ing=auto` is the default setting, the option is omissible:

```
\usepackage[utf8]
\usepackage{bibl
```

The same scenario with XeTeX or LuaTeX in native UTF-8 mode:

```
\usepackage{bibl
```

- It is even possible to combine an 8-bit encoded `tex` file with

UTF-8 encoding in the `bib` file, provided that all characters in

the `bib` file are also covered by the selected 8-bit encoding:

```
\usepackage[lati
\usepackage[bibe
```

Some workarounds may be required when using traditional TeX or pdfTeX

with UTF-8 encoding because `inputenc`'s

`utf8` module does not cover all of Unicode. Roughly speaking, it only covers the Western European Unicode range. When

loading `inputenc` with the

`utf8` option, Biblatex will normally instruct Biber

to reencode the `bib` data to UTF-8. This may lead to

`inputenc` errors if some of the char-

acters in the `bib` file are outside the limited Unicode range supported

by `inputenc`.

- If you are affected by this problem, try setting the

```
safeinputenc
```

option:

`\usepackage[utf8`
`\usepackage[safe`

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 con-

vert `S` to `\k{S}`. This option is similar to setting

`texencoding=`

`ascii` but will only take effect in this spe-

cific 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 `Š`). 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`

but also less obvious things like:

! Argument of `\UT`

2.4.4 Editors and Compiler Scripts

This section is in need of an update to match the new script interface used by Biblatex. For the time being, see the documentation of the

`logreq`

package¹⁶ and the Biblatex Developer's Wiki
for a draft spec.¹⁷

¹⁶ <http://www.ctan.org/tex-archive/macros/latex/contrib/logreq/>

¹⁷ http://sourceforge.net/apps/mediawiki/biblatex/index.php?title=Workflow_Automation

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 $\langle key \rangle = \langle value \rangle$ 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`,
`bibtexu`,
`biber`
default: `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 reen-coding, 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`, re-
spectively.

`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. Depending on the file, case-sensitive sorting may be supported.

`bibtexu` `bibtexu`
is a Unicode-enabled implementation of BibTeX which supports UTF-8. Note that `bibtexu` is not actively supported by Biblatex and has not been tested as backend in any way. Biber is the recommended backend.

See § 2.4.3 for further instructions concerning the encoding of

`bib` files. This option is only available at load-time as internally, the code chooses completely different paths depending on the backend. This means that you can't set the backend with, for example,

`\ExecuteBibliogra`
in the preamble.

style=_{⟨file⟩}

default: **numeric**

Loads the bibliography style **file.bbx** and

the citation style **file.cbx**. See § 3.3 for an overview of the standard styles.

bibstyle=_{⟨file⟩}

default: **numeric**

Loads the bibliography style **file.bbx**.

See § 3.3.2 for an overview of the standard bibliography styles.

citestyle=_{⟨file⟩}

default: **numeric**

Loads the citation style **file.cbx**. See

§ 3.3.1 for an overview of the standard citation styles.

natbib=true,
false

default: **false**

Loads compatibility module which provides aliases for the citation commands of the

natbib package. See § 3.7.9 for details.

`mcite=true`,
`false`
default: `false`

Loads a citation module which provides `mcite` /
`mciteplus` -like citation commands.
See § 3.7.10 for details.

3.1.2 Preamble Options

3.1.2.1 General

The following options may be used in the optional argument to
`\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.

sorting=nty,nyt,
nyvt,
anyt,
anyvt,
ynt,
ydnt,
none,
debug, *<name>*

default: nty

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

nty

Sort by name, title, year.

nyt

Sort by name, year, title.

nyvt

Sort by name, year, volume, title.

anyt

Sort by alphabetic label, name, year, title.

anyvt

Sort by alphabetic label, name, year, volume, title.

ynt

Sort by year, name, title.

ydnt

Sort by year (descending), name, title.

none

Do not sort at all. All entries are processed in citation order.

debug

Sort by entry key. This is intended for debugging only.

<name>

Use *<name>*, as defined with (§ 4.5.5)

\DeclareSort

Biber only

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

nty

default (). 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 § 4.5.5 on how to adapt the predefined schemes or define new ones.

sortcase=true,
false
true

default:

Whether or not to sort the bibliography and the list of shorthands case-sensitively. Note that case-sensitive sorting is only supported

bibtex8

by the and Biber back-ends. Sorting is always case-insensitive with legacy BibTeX. See the

backend option for details.

`sortupper=true`,
`false`

default: `true`

Biber only

`--sortupper`

This option corresponds to Biber's

command-line option. It has no effect with any other backend. If enabled, the bibliography is sorted in 'uppercase before lowercase' order. Disabling this option means 'lowercase before uppercase' order.

`sortlocale`
`=auto`, $\langle locale \rangle$

Biber only

This option sets the global sorting locale. Every sorting scheme inherits this locale if none is specified using the $\langle locale \rangle$ option to

`\printbibliography`
`auto`

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. Biber will map Babel/Polyglossia language identifiers into sensible locale identifiers (see the 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 Biber makes of this is fine for you.

`sortlos=biblos`,
`los`

default:

BibTeX only

The sorting order of the list of shorthands. The following choices are available:

**bib
los**

Sort according to the sorting order of the bibliography.

Sort by shorthand.

The sorting order of shorthands with Biber is more flexible and is

set with the **sorting** option to the

\printbiblist

command.

related=true,
false
true

default:

Biber only

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

sortcites=true,
false
false

default:

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 global sorting order of the bibliography. This feature works with all citation styles.

maxnames

= **3**
<integer> default:

A threshold affecting all lists of names (**author**,

editor, etc.). If a list exceeds this threshold, i. e., if it holds more than **<integer>** names, it is automatically truncated according

to the setting of the **minnames** option.
maxnames is the master option which sets
both **maxbibnames** and
maxcitenames.

minnames

= $\langle integer \rangle$ default: **1**

A limit affecting all lists of names (**author**,
editor, etc.). If a list holds more than
 $\langle maxnames \rangle$ names, it is automatically truncated to $\langle minnames \rangle$ names.
The $\langle minnames \rangle$ value must be smaller than or equal to $\langle maxnames \rangle$.

minnames is the master option which sets
both **minbibnames** and
mincitenames.

maxbibnames

= $\langle integer \rangle$ default: $\langle maxnames \rangle$

Similar to **maxnames** but affects only the
bibliography.

minbibnames

=
⟨integer⟩

default: ⟨minnames⟩

Similar to minnames but affects only the bibliography.

maxcitenames

=
⟨integer⟩

default: ⟨maxnames⟩

Similar to maxnames but affects only the citations in the document body.

mincitenames

=
⟨integer⟩

default: ⟨minnames⟩

Similar to minnames but affects only the citations in the document body.

maxitems

=
⟨integer⟩

default: 3

Similar to maxnames, but affecting all literal lists (publisher, location, etc.).

minitems

= $\langle integer \rangle$ default: 1

Similar to minnames, but affecting all literal lists (publisher, location, etc.).

autocite

= plain, inline, footnote, superscript,

This option controls the behavior of the \autocite command discussed in § 3.7.4. The plain option makes \autocite behave like \cite, inline makes it behave like \parencite, footnote makes it behave like \footcite, and

superscript makes
 it behave like \supercite. The
 options plain, inline,
 and footnote are always available, the
 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.

autopunct=true,
 false
 true
 default:

This option controls whether the citation commands
 scan ahead for punctuation marks. See § 3.7 and

\DeclareAutoPunct
 in § 4.7.5 for details.

language=autobib,
 autocite,
 auto,
 <language>

autobib
 default:

This option controls multilingual support. When

set to autobib,

autocite auto,

Biblatex will try to get the main document language from the

babel/polyglossia

babel/polyglo

package (and fall back to English if

is not available). It is also possible to select the document language manually.

In this case, the autolang option below

will have no effect. Please refer to table 2 for a list of supported languages and

the corresponding identifiers. autobib

switches the language for each entry in the bibliography using the

langid field and the language environment

specified by the autolang option.

autocite switches the language for each

citation using the langid field and the language envi-

ronment specified by the autolang

option. auto is a shorthand to

set both autobib and

autocite. The default is to switch languages automatically only for bibliography entries

(autobib).

`clearlang=true,`
`false`

default: `true`

If this option is enabled, Biblatex will automatically

clear the language
field of all entries whose language matches the

`babelpolyglossia`

language of the document (or the language specified explic-

itly with the language op-

tion) in order to omit redundant language specifications. The
language mappings required by this feature are provided by the

`\DeclareRedundant`

command from § [4.9.1](#).

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

default: none

This option controls which **babel** language environment¹⁸

is used if the **babel/polyglossi**
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. This option is for switching languages on a per-entry basis within the bibliography. The possible choices are:

none Disable this feature, i. e., do not use any language environment at all.

¹⁸ **polyglossia** under-

stands the **babel** language environments too

and so this option controls both the **babel** and

polyglossia language environments.

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

babelpolyglossia
and Biblalex provide for the respective language, and translate key
terms such as 'editor' and 'volume'. The extra definitions include
localizations of the date format, of ordinals, and similar things.

other* Enclose the entry in an
otherlanguage*
environment. Please note that Biblalex treats
otherlanguage*
otherlanguage
like
but other packages may make a distinction in this case.

langname
polyglossia
only. Enclose the entry in a

‘`language`’

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 select-

ing a language variant). When using `babel`,

this option does the same as the option value. `other`

`block=none`,
`space`,
`par`,
`nbpar`,
`ragged`
default: `none`

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 dis-
allows page breaks at block boundaries and within an entry.

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

The **\newblockpunct**
command may also be redefined directly to achieve different results, see § 3.9.1. 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 Sup-
port both footnotes and endnotes, i. e.,
\mkbibfootnote
\mkbibendnote
will generate footnotes and
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

hyperref
automatically detects if the 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 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.

refsection

backrefstyle

=none, three, two, two+, three+, all+ default:

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 localization string

two 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 localization 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 (default),
setonly,
memonly,
setormem,
setandmem,
memandset,
setplusmem
@set

This option controls how back references to 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
false

default:

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.

`loadfiles=true`,
`false`

default: `false`

This option controls whether external files requested by way of the `\printfile` command are loaded. See also § [3.11.8](#) and

`\printfile` in § [4.4.1](#). Note that this feature is disabled by default for performance reasons.

refsection
=none, default: none
part,
chapter,
section,
subsection

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 § 3.6.5 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.

`refsegment`

`=none`, default: `none`

`part`,

`chapter`,

`section`,

`subsection`

`refsection`

Similar to the `refsection` option but starts a new reference segment. This is equivalent to the

`\newrefsegment`

command, see § 3.6.6 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.

`citereset=none`,
`part`,
`chapter`,
`section`,
`subsection`

default: `none`

`\citereset`

This option automatically executes the command from § 3.7.8 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
command. \subsection

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

abbreviate
=true, default: true
false

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

date=year, default: comp
short,
long,
terse,
comp,
iso8601

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

year Use only years, for example:

2010

2010–2012

short Use the short format with verbose ranges, for example:

01/01/2010

21/01/2010–30/01/2010

01/21/2010–01/30/2010

long

Use the long format with verbose ranges, for example:

1st January 2010
21st January 2010–30th January 2010
January 21, 2010–January 30, 2010

terse

Use the short format with compact ranges, for example:

21–30/01/2010
01/21–01/30/2010

comp

Use the long format with compact ranges, for example:

21st–30th January 2010
January 21–30, 2010

iso8601

Use extended ISO-8601 format

(**yyyy-mm-dd**),
for example:

2010-01-01
2010-01-21/2010-01-30

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 in all short formats may be controlled separately with the

datezeros package option.

dateLabel=year,
short,
long,
terse,
comp,
iso8601
year

default:

date
Similar to the option but con-
trols the format of the date field selected with
\DeclareLabeldate

origdate=year,
short,
long,
terse,
comp,
iso8601
default: comp

Similar to the date option but controls the format of the
origdate.

eventdate=year,
short,
long,
terse,
comp,
iso8601
comp

default:

date
Similar to the option but controls the format of the
eventdate.

urldate=year,
short,
long,
terse,
comp,
iso8601

default: short

Similar to the date option but controls the format of the

urldate.

all dates
=year, short,
long, terse,
comp, iso8601

Sets all of the above date options to the same value.

datezeros=true,
false

default: true

This option controls whether short and
terse dates are printed with leading zeros.

dateabbrev
=true, default: true
false

This option controls whether long and
comp dates are printed with long or abbrev-
iated month names. The option is similar to the generic

abbreviate option but specific
to the date formatting.

defernumbers
=true, default: false
false

In contrast to standard LaTeX, the numeric labels generated by this pack-
age 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 dis-
cussed in § 4.2.4) are assigned the first time an entry is printed in any bibliography.

See § 3.12.5 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

`\printbibliograph`

macro, see § 3.6.2), 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`
`abs`

default:

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 § 3.11.7 for details on support for arXiv and electronic publishing information.

texencoding

=auto, *<encoding>* default: **auto**

tex
Specifies the encoding of the file. This option affects the data transferred from the backend to Biblatex. When using Biber, 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=`
`bibencoding`
`bibencoding=`
`auto`.

`bibencoding`

`=auto`, `<encoding>` default: `auto`

Specifies the encoding of the `bib` files. When using Biber, 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 the one
 of the `tex` file, you need to specify it explicitly.

By default, Bibl_{at}ex assumes that the `tex`
 file and the `bib` file use the same encoding

`(bibencoding = auto`

). Note that some backends only support a limited number of encodings. See § 2.4.3 for further instructions.

`safeinputenc`

`=true, false`
`false`

default:

Biber only

If this option is enabled, Biblatex will automatically force

`texencoding =`

`asciiinputenc` if the

`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 § 2.4.3.4 for an explanation of why you may

want to enable this option.

`bibwarn=true,`

`false`

`true`

default:

By default, Biblatex will report warnings issued by the backend concerning the data

bib in the **bib** file as LaTeX warnings. Use this option to suppress such warnings.

mincrossrefs

= $\langle integer \rangle$ default: **2**

Sets the minimum number of cross references to $\langle integer \rangle$ when requesting a backend run.¹⁹ Note that when using the BibTeX backend, this package option merely affects the format of certain requests written to the transcript file. It will not have any effect if the editor or compiler script launching BibTeX does not include dedicated Biblatex support or if BibTeX is manually launched from the command-line.²⁰ See § 2.4.4 for

details. This option also affects the handling of the **xref** field. See the field description in § 2.2.3 as well as § 2.4.1 for details.

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**, default: **true**
false

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

¹⁹If 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 (also Biber) and not specific to Biblatex. See the description of the

crossref field in § 2.2.3 for further information.

²⁰As of this writing, no LaTeX editors or compiler scripts with dedicated Biblatex support are known, but this will hopefully change in the future.

`url=true`, default: `true`
`false`

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`, default: `true`
`false`

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
`firstinits` options, which you may want to adapt, there is normally no need to set them explicitly.

pagetracker

=true, default: false
false,
page,
spread

This option controls the page tracker which is required by the

\ifsamepage and
\iffirstonpage

tests from § 4.6.2. The possible choices are:

true Enable the tracker in automatic mode. This is like
spread if LaTeX is in twoside mode,
and like page otherwise.

false Disable the tracker.
page Enable the tracker in page mode. In this mode, tracking
works on a per-page basis.

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, default: false
false,
context

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, default: false
false,
context,
strict,
constrict

This option controls the citation tracker which is required by the

\ifciteseen and

`\ifentryseen` 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`, default: `false`
`false`,
`context`,
`strict`,
`constrict`

This option controls the ‘ibidem’ tracker which is required by the

`\ifciteibid` 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.²¹

`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.

²¹For 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.

opcittracker

=true, default: false

false,
context,

strict,
constrict

This option controls the ‘opcit’ tracker which is required by the

\ifopcit

test from § 4.6.2. This fea-

ture is similar to the ‘ibidem’ tracker, except that it tracks citations on a

per-author/editor basis, i.e., \ifopcit

true

will yield 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.

loccittracker
=true, default: **false**
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**

true will yield 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, default: false

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 unambigu-

ous 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, default: true
false

This option controls the parenthesis tracker which keeps track of nested parentheses and brackets. This information is used

by \parenttext and

\brackettext from

§ 3.7.5, \mkbibparens

and \mkbibbrackets

from § 4.10.4 and \bibopenparen,

\bibcloseparen,

\bibopenbracket,

\bibclosebracket

(also § 4.10.4).

maxparens

= <integer> default: 3

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

`firstinits=true,`
`false`

default: `false`

When enabled, all first and middle names will be rendered as initials. The option will affect the

`\iffirstinits`

test from § 4.6.2.

`sortfirstinits`
`=true,` default: `false`
`false`

When enabled, sorting names will only use their initials. This is separate from Biber only

`firstinits` in case users want to show only inits but sort on full names, for example.

`terseinits=true,`
`false`
default: `false`

This option controls the format of 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. With Biber, the option works by redefining some macros which Biber only control the format of initials. See § 3.12.4 for details.

`labelalpha=true`,
`false`
default: `false`

Whether or not to provide the special fields

`labelalpha` and
`extraalpha`, see § 4.2.4

for details. With Biber, this option is also setttable on a per-type basis. See also [Biber only](#)

`maxalphanames`
and `minalphanames`.
`extra*` disambigua-

tion counters and what they track.

`maxalphanames`

= $\langle integer \rangle$ default: 3 [Biber only](#)

Similar to the `maxnames` option but cus-

tomizes the format of the `labelalpha` field.

`minalphanames`

= $\langle integer \rangle$ default: 1 [Biber only](#)

Similar to the `minnames` option but customizes the format of the `labelalpha` field.

`labelnumber`
`=true`, default: `false`
`false`

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

`labeltitle`
`=true`,
`false`
 default: `false`

Biber only

Whether or not to provide the special field `extratitle`, see § 4.2.4 for details. Note that the special field

`labeltitle` is always provided and this option controls rather whether `labeltitle` is used to generate `extratitle` information. This option is also settable on a per-type basis. Table 5 summarises the various `extra*` disambiguation counters and what they track.

labeltitleyear

=true, default: false
false

Biber only

Whether or not to provide the special field

extratitleyear,

see § 4.2.4 for details. Note that the special field

labeltitle

is always provided and this option controls rather whether

labeltitle

is used to generate

extratitleyear

information. This option is also settable on a per-type basis. Table 5 summarises the

extra*

various
they track.

disambiguation counters and what

labeldate=true,
false
false

default:

Whether or not to provide the special fields

labelyear,

labelmonth,

labelday

and

`extrayear`, see § 4.2.4 for details. With Biber, this option is also settable on a per-type basis. Table 5 summarises [Biber only](#)

`extra*`
the various disambiguation counters and what they track.

`singletitle`

`=true`, default: `false`
`false`

Whether or not to provide the data required by the
`\ifsingletitle`

test, see § 4.6.2 for details. With Biber, this option is also settable on a per-type basis. [Biber only](#)

`uniquename`
`=true`,
`false`,
`init`,
`full`,
`allinit`,
`allfull`,
`mininit`,
`minfull`

default: `false`

[Biber only](#)

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 names using initials only.
full Disambiguate names using initials or full names, as required.
allinit Similar to **init** but disambiguates all names in the **labelname** list, beyond **maxnames** / **minnames**, **unique** list.
allfull Similar to **full** but disambiguates all names in the **labelname** list, beyond **maxnames** /

minnames,
unique~~list~~.

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 op-

tion will also affect unique~~list~~,

the \ifsingletitle

test, and the extrayear field. See
§ 4.11.4 for further details and practical examples.

unique~~list~~=true,
false,
minyear

default: false Biber only

Whether or not to update the unique~~list~~
counter, see § 4.6.2 for details. This feature will disambiguate the

labelname
list if it has be-
come ambiguous after maxnames /
minnames
truncation. Essen-
tially, it overrides maxnames /
minnames
on a per-field basis. This
option is also settable on a per-type basis. The possible choices are:

true labelname
Disambiguate the
list.

false
Disable this feature.

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

labeldate =
true.

Note that the uniquelist op-

tion will also affect the \ifsingletitle

extrayear field. See
test and the
§ 4.11.4 for further details and practical examples. See § 4.11.4 for further details and
practical examples.

Option	Enabled field	Enabled counter	Counter tracks
la- bel- pha la- bel- date	la- bel- pha la- bel- year	ex- tra- pha ex- tra- year	la- bel- pha la- bel- year name+la- bel- year
la- bel- tle	— — —	ex- tra- tle	la- bel- name+la- bel- tle
la- bel- tle	— — year	ex- tra- tle	la- bel- name+la- bel- year

Table 5: Disambiguation counters

3.1.3 Entry Options

Entry options are package options which can be adjusted on a per-entry basis in the `options` field from § 2.2.3.

3.1.3.1 Preamble/Type/Entry Options

The following options are settable on a per-entry basis 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 behavior 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

author is printed after the title in this case. See also § 3.5. With Biber, this option is also settable on a per-type basis. [Biber only](#)

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. With Biber, this option is also settable on a per-type basis. [Biber only](#)

usetranslator

=true, default: false
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
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: With Biber, [Biber only](#) this option is also settable on a per-type basis.

@Book{ . . . ,

options = {u
author = {.
translator = {.

translator

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

useprefix=true,
false
false

default:

Whether the name prefix (von, van, of, da, de, della, etc.) is considered when printing the last name in citations. This also affects the sorting and formatting of the bibliography as well as the generation of certain types of labels. If this option is enabled, Biblatex always precedes the last name with the prefix. For example, Ludwig van Beethoven would be cited as “Beethoven” and alphabetized as “Beethoven, Ludwig van” by default. If this option is enabled, he is cited as “van Beethoven” and alphabetized as “Van Beethoven, Ludwig” instead. With Biber, this option is also settable on a per-type basis.

Biber only

indexing
=true, false,
citebib
indexing

The 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 only settable on a per-entry basis in the `options` field. They are not available globally.

`skipbib=true`,
`false`
default: `false`

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

Biber only

`skiplos=true`,
`false`
default: `false`

If this option is enabled, the entry is excluded from the list of shorthands. It is still included in the bibliography and it may also be cited by shorthand. With Biber, this option is also settable on a per-type basis.

Biber only

`skiplab=true`,
`false`
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. With Biber, this option is also settable on a per-type basis.

Biber only

`dataonly=true,`
`false`
default: `false`

Setting this option is equivalent to `uniquename=`
`false, uniquelist=`
`false, skipbib,`
`skiplos,` and
`skiplab`. It is not required for normal
operation. Use it with care. With Biber, this option is also settable on a per-type basis. [Biber only](#)

`labelnamefield`
=
<fieldname>

Specifies the field to consider first when looking for a

`labelname` candi-
date. It is essentially prepended to the search list created by

`\DeclareLabelname`
for just this entry.

`labeltitlefield`
=
<fieldname>

Specifies the field to consider first when looking for a

labeltitle

date. It is essentially prepended to the search list created by candi-

\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 [Deprecated](#)

the standard LaTeX document classes.

openbib

is similar to block=par.

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

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`
`[<entrytype, ...>] { <key=value, ...> }`

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 `<entrytype>` argument specifies the entry type. The `<entrytype>` 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 Bibl_{at}ex 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.

`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.

`numeric-comp`

A compact variant of the `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 op-

tions at load time:

```
autocite =
inline, labelnumber =
true
```

It also provides the

`subentry` option.

alphabetic This style implements an alphabetic citation scheme similar to the

alpha.bst

style of traditional

BibTeX. The alphabetic 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 pack-

age options at load time:

```
autocite=
inline, labelalpha=
true.
```

alphabetic-verb

A verbose variant of the

alphabetic

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, labeldate=
```

```
true,uniquename=
full,uniqueelist=
true.
```

authoryear

authoryear-comp

A compact variant of the

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,labeldate=
true,uniquename=
full,uniqueelist=
true.
```

authoryear

authoryear-ibid

A variant of the

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, labeldate=
true, uniquename=
full, uniquelist=
true, 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

```
autocite =
inline, labeldate=
true, uniquename=
full, uniquelist=
```

time:

```

true, ibidtracker=
constrict,
pagetracker =
true, sortcites=
true

```

. This style also provides an additional preamble option called **ibidpage**. See the style example for details.

authortitle 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

```

autocite =
footnote,
uniquename =
full, uniquelist=
true

```

authortitle-comp

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 op-

```

autocite =

```

tions at load time:

```
footnote,
sortcites =
true,uniquename=
full,uniqueelist=
true.
```

`authortitle-ibid` 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 op-

tions at load time:

```
autocite =
footnote,
uniquename =
full,uniqueelist=
true,ibidtracker=
constrict,
pagetracker =
```

`true`. This style also provides an additional preamble option
called `ibidpage`. See the style example for
details.

`authortitle-ibid` A style combining the features of `authortitle-`

and `authortitle-ibid`.
The style will set the following package options at load

`time: autocite =`
`footnote,`
`uniquename =`
`full,uniqueelist=`
`true,ibidtracker=`
`constrict,`
`pagetracker =`
`true,sortcites=`
`true`. This style also provides an additional preamble option
called `ibidpage`. See the style example for
details.

`authortitle-terse` 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.**

authortitle-
authortitle-terse
shorttitle

This style will make use of the 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.**

`authortitle-ticomp`

A style combining the features of

`authortitle-`

and

`authortitle-terse`

`authortitle-`

In other words: a variant of the

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 op-

tions 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 op-

tions at load time:

```
autocite =
footnote,
citetracker =
context
citepages.
```

. This style also provides an additional preamble option called See the style example for details.

[verbose-ibid](#) A variant of the **verbose** 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,ibidtracke
constrict,
pagetracker =
true.
```

This style also provides additional pream-

ble options called **ibidpage** and **citepages**. See the style example for details.

verbose
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

verbose 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 op-

tions 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-note` 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, page tracker = true**. This style also provides additional pream-

ble 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

shorthand in a special way. If the 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 op-

tions at load time: **autocite = footnote, citetracker = context, ibidtracker = constrict, idemtracker = constrict, opcittracker = context, loccittrac context**. This style also provides additional

preamble options called **ibidpage, strict, citepages**, and

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

verbose-trad1.

than to 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 op-

tions 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

author-title-comp
author-title
will load the 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.

shorthand
Note that the 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 op-

subentry
tion called 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.

alphabetic This style prints an alphabetic label similar to the **alpha.**

bst style of traditional BibTeX. It is intended for use in conjunction with an alphabetic citation style. Note that the **shorthand** field overrides the

default label. The style will set the following package options at load

time: `labelalpha =
true, sorting =
anyt.`

authoryear This style differs from the other styles in that the publication date is not printed towards the end of the entry but rather after the author/editor. It is intended for use in conjunction with an author-year citation style. Recurring author and editor names are replaced by a dash unless the entry is the first one on the current page or double-page spread. This style provides an additional preamble option called

`dashed` which controls this feature. It also provided
a preamble option called `mergedate`.

See the style example for details. The style will set the following package options

at load time: `labeldate =
true, sorting =
nyt, pagetracker=
true, mergedate=
true.`

authortitle This style does not print any label at all. It is intended for use in conjunction with an author-title citation style. Recurring author and editor names are replaced by a dash unless the entry is the first one on the current page or double-page spread. This style also provides an additional preamble option called

`dashed` which controls this feature. See the style
example for details. The style will set the following package options at load

time: pagetracker=
true.

verbose This style is similar to the authortitle style. It also provides an additional preamble option called

dashed. See the style example for details. The style will set the following package options at load time:

pagetracker =
true.

reading This special bibliography style is designed for personal reading lists, annotated bibliographies, and similar applications. It optionally includes

the fields annotation,
abstract,
library, and file

in the bibliography. If desired, it also adds various kinds of short headers to the bibliography. This style also provides the additional

preamble options entryhead,

entrykey,
annotation,
abstract,
library, and file

which control whether or not the corresponding items are printed in the bibliography. See the style example for details. See also

§ 3.11.8. The style will set the following package options at load

time: `loadfiles =
true,entryhead=
true,entrykey=
true,annotation=
true,abstract=
true,library =
true, file =
true.`

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 pack-

age 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 op-

tions 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 mech-

anism 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 localization 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:

```
@Book{key1,
  related
  relatedtype = {
  ...
}
```

```
@Book{key2,
```

}

Here we specify that entry **key1** is a

reprint of entry **key2**. In the biblio-

graphy driver for **Book** entries, when

`\usebibmacro{rela`

is called for entry **key1**:

- If the localization string “**reprintof**”

is defined, it is printed in the **relatedstring**

format. If this formatting directive is undefined, the string is printed in the

relatedstring:de format.

- If the **related:reprint** macro is defined, it is used to format the information con-

tained in entry **key2**, otherwise the

related:default macro is used

- If the **related:reprint** format is defined, it is used to format both the localization string and data. If this

format is not defined, then the
format is used instead.

related

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

```
@Book{key1,  
  related  
  relatedtype = {  
    .  
    .  
    .  
  }  
}
```

```
@Book{key2,  
  related  
  relatedtype = {  
    .  
    .  
    .  
  }  
}
```

```
@Book{key3,  
  related  
  relatedtype = {  
    .  
    .  
    .  
  }  
}
```

```

    . . .
}

```

Multiple relations to the same entry are also possible:

```

@MVBook{key1,
  related
  relatedtype = {
    . . .
  }
@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 `relatedoptions` from § 3.1.2.1.

You can use the

to set options on the related entry data clone. This is useful if you need to override

`related`
`relatedoptions`

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 **uniqueelist=false**. In such a case, you can set

relatedoptions
skiplab, skipbib to just

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**,
translator fields,
 and in this order. Which fields are considered also depends on the
 setting of the **useauthor**,
useeditor, and
usetranslator
 options. If all three of them 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 **useauthor**,
useeditor, and
usetranslator
 are disabled, Biblatex will fall back to the
sorttitle and
title fields as a last resort. The remaining items

are, in various order: the **sortyear** field, if defined, or the first four digits of the **year** field otherwise; the **sorttitle** field, if defined, or the **title** field otherwise; the **volume** field, which is padded to four digits with leading zeros, or the string **0000** otherwise. Note that the sorting schemes shown in appendix [C.2](#) include an additional item: **labelalpha** is the label used by ‘alphabetic’ bibliography styles. Strictly speaking, the string used for sorting is **labelalpha + 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 **presort** and **sortkey** fields, if defined. The next item considered is the **sortyear** or the **year** field, depending on availability. The **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 **ydn**t scheme is similar in concept but sorts the

year in descending order. As with the **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

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 **sortkey**,
sortname, or
sorttitle is usually not required.

The 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. Also note that using Biber instead of legacy BibTeX is strongly recommended.

3.6 Bibliography Commands

3.6.1 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.

sion from the filename. Also note that the $\langle resource \rangle$ is a single resource. Invoke

`\addbibresource`

multiple times to add more resources, for example:

```
\addbibresource{b  
\addbibresource{b  
\addbibresource[l  
\addbibresource[l
```

Since the $\langle resource \rangle$ 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 $\langle options \rangle$ are available:

`label=` $\langle identifier \rangle$

Assigns a label to a resource. The $\langle identifier \rangle$ may be used in place of the full resource name in the optional argument of

`refsection` (see § 3.6.5).

`location=` $\langle location \rangle$

default: `local`

The location of the resource. The $\langle location \rangle$ may be ei-

ther `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=` _{$\langle type \rangle$} default: `file`
`file`

The type of resource. Currently, the only supported type is

`datatype=` _{$\langle datatype \rangle$}
default: `bibtex`

The data type (format) of the resource. The following formats are currently supported:

`bibtex`

BibTeX
for-
mat.

`ris`

Re-
search
In-
for-
ma-
tion
Sys-
tems
(RIS)
for-
mat.²²
Note
that
an

ID

tag
is

²² [http://en.wikipedia.org/wiki/RIS_\(file_format\)](http://en.wikipedia.org/wiki/RIS_(file_format))

zoterordfxml

re-
quired
in
all
RIS
records.
The
ID

value
cor-
re-
sponds
to
the
en-
try
key.
Sup-
port
for
this
for-
mat
is
ex-
per-
i-
men-
tal.

Zotero [Biber only](#)
RDF/XML
for-
mat.
Sup-
port
for
this
for-
mat
is
ex-
per-
i-
men-
tal.
Re-
fer

to
the
Biber
man-
ual
for
de-
tails.

endnotexml

End-
Note
XML
for-
mat.
Sup-
port
for
this
for-
mat
is
ex-
per-
i-
men-
tal.
Re-
fer
to
the
Biber
man-
ual
for
de-
tails.

[Biber only](#)

\addglobalbib

[*options*] {*resource*}

\addbibresour

This command differs from `\addbibresource` in that the *resource* 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

refsection (§ 3.6.5) is used to specify alternative resources which replace the default resource list. Any global resources are added to all reference sections.

\addsectionbib
[*options*] { *resource* }

This command differs from **\addbibresource** in that the resource *options* are registered but the *resource* not added to any resource list. This is only required for resources which 1) are given exclusively in the op-

tional argument of **refsection** (§ 3.6.5) and 2) require options different from the default settings. In this case,

\addsectionbib is employed to qualify the *resource* prior to using it by setting the appropriate

label *options* in the preamble. The **label** option may be useful to assign a short name to the resource.

\bibliography
{ *bibfile, ...* }

Deprecated

The legacy command for adding bibliographic resources, supported for backwards compatibility. Like

\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

bib files. The **.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

`\addbibresource`
instead.

3.6.2 The Bibliography

`\printbibliography`
[$\langle key=value, \dots \rangle$]

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

`env = $\langle name \rangle$` de-
fault: `bibliography/show`

The ‘high-level’ layout of the bibliography and the list of shorthands is controlled by environments defined with

`\defbibenvironment`

This option selects an environment. The $\langle name \rangle$ corresponds to the identifier used when defining the environment with

`\defbibenvironment`

By default, the `\printbibliography`

command uses the identifier `bibliography`;

`\printbiblist`

`shorthands`. See also
uses §§ 3.6.4 and 3.6.8.

heading=_{⟨name⟩} de-
fault: bibliography/sho

The bibliography and the list of shorthands typically have a chapter or section heading. This option selects the heading *⟨name⟩*, as defined with

\defbibheading

By default, the **\printbibliogra**

command uses the heading **bibliography**;

\printbiblist
shorthands

uses §§ 3.6.4 and 3.6.8. See also

title=_{⟨text⟩}

This option overrides the default title provided by the heading selected with the

heading option, if supported by the heading definition. See § 3.6.8 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 § 3.6.9 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 § 3.6.9 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 § 3.6.5 for details and § 3.11.3 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

refsegment environment are assigned to segment 0. See § 3.6.6 for details and § 3.11.3 for usage examples. Remember that segments within a section are numbered local to the section so the segment you request will be the *n*th segment in the requested (or currently active enclosing) section.

sorting=_{⟨name⟩}

Biber only

Sort the bibliography according to a built-in or user specified sorting scheme defined with

\DeclareSortingScheme

If omitted, the global sorting scheme specified with package

sorting option is used. This option facilitates multiple bibliography lists within a refsection, each of which can be sorted differently.

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.

notkeyword=_{⟨keyword⟩}

Print only entries whose **keywords** field does not include *⟨keyword⟩*. This option may be used multiple times.

category=_{⟨category⟩}

Print only entries assigned to category *⟨category⟩*. This option may be used multiple times.

notcategory=_{⟨category⟩}

Print only entries not assigned to category *⟨category⟩*. This option may be used multiple times.

filter=_{⟨name⟩}

Filter the entries with filter *⟨name⟩*, as defined with **\defbibfilter**.
See § 3.6.10 for details.

check=_{⟨name⟩}

Filter the entries with check *⟨name⟩*, as defined with **\defbibcheck**.
See § 3.6.10 for details.

prefixnumbers

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 current bibliography. The option assigns the $\langle string \rangle$ as a prefix to all entries

in the respective bibliography. For example, if the $\langle string \rangle$ 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 $\langle string \rangle$ is available to styles

in the **prefixnumber** field of all affected entries. See § 4.2.4.2 for details.

resetnumbers= $\langle true, false, number \rangle$

This option applies to numerical citation/bibliography styles only and requires that

the **defernumbers** 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:

resetnumbers=10

to reset numbering to the specified number to aid numbering continuity across documents. Use this option with care as Biblatex can not guarantee unique labels globally if they are reset manually.

`omitnumbers=true,`
`false`

This option applies to numerical citation/bibliography styles only and requires that

the `defernumbers`
option from § 3.1.2.1 be enabled globally. If enabled, 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).

`\bibbysection`

`[
 ⟨key=value, ...⟩
]`

This 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. Note that

`\bibbysection`

starts looking for references in section `1`. It will ignore references given

outside of `refsection`

environments since they are assigned to section 0. See § 3.11.3 for usage examples. The options are a subset of those supported by

`\printbibliography`

`envheading`,
Valid options are

`prenote`, `postnote`.

The global sorting scheme specified with package

sorting

option is used for all sections.

\bibbysegment

[*<key=value, ...>*]

This command automatically loops over all references segments. This is equivalent to giving one

\printbibliograph

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

refsegment

given outside of

environments since they are assigned to segment 0. See § [3.11.3](#) for usage examples. The options are a subset of those supported by

\printbibliograph

envheading

Valid options are

prenote, postnote

The global sorting scheme specified with package

sorting

option is used for all segments.

`\bibbcategory`

`[
 $\langle key=value, \dots \rangle$
]`

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 § 3.11.3 for usage examples. The options are a subset of those supported by

`\printbibliography`

Valid options are `env`, `prenote`,

`postnote`,

`section`.

`heading`

Note that

is not available with this command. The name of the current category is automatically used as the heading name. This is equivalent to pass-

ing `heading= $\langle category \rangle$` to

`\printbibliography`

and implies that there must be a matching heading definition for every category. The global sorting scheme specified with package

`sorting`

option is used for all categories.

`\printbibheading`

`[
 $\langle key=value, \dots \rangle$
]`

This command prints a bibliography heading defined with

`\defbibheading`

It takes one optional argument, which is a list of options given in $\langle key \rangle$ notation. The options are a small subset of those supported by

`\printbibliography`

Valid options are `heading` and

`title`

`bibliography`. By default, this command uses the heading `bibliography`. See

§ 3.6.8 for details. Also see §§ 3.11.3 and 3.11.4 for usage examples.

3.6.3 The List of Shorthands

This section applies only to BibTeX. When using Biber, the list of shorthands is just a special case of a bibliography list. See § 3.6.4. BibTeX only

`shorthand`

If any entry includes a `biblatex` field, automatically builds a list of shorthands which may be printed in addition to the regular bibliography. The following command prints the list of shorthands.

`\printshorthands`

`[
 $\langle key=value, \dots \rangle$
]`

This command prints the list of shorthands. It takes one optional argument, which is a list of options given in $\langle key \rangle$

$\langle value \rangle$ notation. Valid options are all options supported by

`\printbibliography`

`prefixnumbers`,
(§ 3.6.2) except

`resetnumbers`, and

`omitnumbers`. If there

`refsection`
are any environments in the document, the list of shorthands will be local to these

environments; see § 3.6.5 for details. By default, this command uses the heading

`shorthands`. See § 3.6.8 for
details.

The `sorting` option differs from

`\printbibliography`

in that if omitted, the default is to sort by shorthand.

3.6.4 Bibliography Lists

This section applies only to Biber. It is a generalisation of the shorthands facility Biber only available in earlier versions and with BibTeX. When using BibTeX as the backend, please refer to section § 3.6.3.

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`
`[$\langle key=value, \dots \rangle$] { $\langle <biblistname>$ }`

This command prints a bibliography list. It takes an op-

ditional argument, which is a list of options given in $\langle key \rangle$ notation. Valid options are all options supported by

\backslash printbibliography

prefixnumbers, (3.6.2) except

resetnumbers, and

omitnumbers. If there

are any refsection environments in the document, the bibliography list will be local to these environments; see 3.6.5 for details. By default, this command uses the heading

biblist. See 3.6.8 for details.

The $\langle biblistname \rangle$ is a mandatory argument which names the bibliography list. This name is used to identify:

- The bibliography driver used to print the list entries

\backslash DeclareBibliography

- A filter declared with used to filter the entries returned from Biber

- The default bib environment to use

- The default sorting scheme name to use if the

sorting option is not set

Note that the sorting option differs from

\backslash printbibliography

in that if omitted, the default is to sort use the sorting scheme named after the bibliography list and only then to fall back to the global sorting scheme is this is not defined.

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[...`

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 § 3.6.8)
- A bibliography list filter (See § 4.5.6)
- 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:

`\DeclareBibliography \printfield{jou`

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

`\printbiblist[tit shortjournal`

Since 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 .bbl`

field in the

- 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 global sorting scheme since no

sorting
`\printbiblist`

and no scheme exists with the name `shortjournal`

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.

3.6.5 Bibliography Sections

`refsection`

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 doc-

ument 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 § 3.11.3 for usage examples.

```
\beginrefsection[⟨resource, ...⟩]  
\end{refsection}
```

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 resource list, as specified with

`\addbibresource`

in the preamble. If the argument is provided, it replaces the default resource list. Global resources specified with

`\addglobalbib`

`refsection`

are always considered.

environments may not be nested, but you may use

`refsegment`

environments

`refsection`

within a

to subdi-

`section`

vide it into segments. Use the

option of

`\printbibliograph`

to select a section when printing the bibliography, and the corresponding option of

`\printbiblist`

when printing bibliography lists.

Bibliography sections are num-

bered starting at **1**. The number of the current section is also written to the transcript file. All citations given out-

`refsection`

side a

environment

are assigned to section 0.

If

`\printbibliograph`

`refsection`

is used within a

it will automatically select the current section.

The

`section`

option

is

not required in this case. This also applies to

`\printbiblist`.

`\newrefsection`

`[
 ⟨resource, ...⟩
]`

`refsection`

This command is similar to the `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.

3.6.6 Bibliography Segments

`refsegment`

The `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 docu-

ment divisions such as `\chapter`

`\section` and they will typically be used per chapter or section. even though See the

`refsegment` package option in

§ 3.1.2.1 for a way to automate this. Also see § 3.11.3 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 docu-

ment. **refsegment** envi-

ronments 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 ref-

erence 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 envi-

environment, the current segment is not selected automatically if

`\printbibliography`
`refsegment`

is used within a
environment.

`\newrefsegment` This com-
`refsegment`

mand is similar to the

environment except that it is a stand-alone command rather than an environ-
ment. 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.

3.6.7 Bibliography Categories

Bibliography categories allow you to split the bibliography into multiple parts dedi-
cated to different topics or different types of references, for example primary and
secondary sources. See § 3.11.4 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`

`{\langle category \rangle}{\langle key \rangle}`

Assigns a `\langle key \rangle` to a `\langle category \rangle`, 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 `\langle key \rangle` may be a single entry key or a comma-separated list of keys. The assignment is global.

3.6.8 Bibliography Headings and Environments

`\defbibenvironment`

`{\langle name \rangle}{\langle begin code \rangle}{\langle end code \rangle}{\langle item code \rangle}`

This command defines bibliography environments. The `\langle name \rangle`

is an identifier passed to the `env` option of

`\printbibliography`

`\printbiblist`

and
when selecting the environment. The `\langle begin code \rangle` is LaTeX code to be executed at the beginning of the environment; the `\langle end code \rangle` is executed at the end of the environment; the `\langle item code \rangle` 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

environment:

```
\list
{\setlength{
\setlength{
\setlength{
\setlength{
{\endlist}
{\item}
```

As seen in the above example, usage of

is roughly similar to

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

`\printbibliograph`
or `\printbibheading`
and `\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

`\printbibliograph`
or `\printbiblist`

`title` 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` ar-

gument 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{bi
  \chapter*{#1}%
  \markboth{#1}{#
```

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

`bibliography`

`\printbiblio`
This is the default heading used by
`heading` if the `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`

`\chapter*` class, i.e., it uses 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 un-
numbered section heading which is not included in the table of con-
tents. The string used in the heading also depends on the document

class. With `book`-like classes the localization string
`bibliography` is

used, with other classes it is
(see § 4.9.2). See also §§ 3.12.1 and 3.12.2 for class-specific hints.

references

subbibliography

bibliography

Similar to

but one sectioning level lower. This heading definition

uses `\section*` in-

stead of `\chapter*`

with a book-like class and

`\subsection*` in-

stead of `\section*` otherwise.

bibintoc

bibliography

Similar to

above but adds an entry to the table of contents.

subbibintoc

subbibliography

Similar to

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 localization string

shorthands instead of
bibliography
references (see § 4.9.2). See
or also §§ 3.12.1 and 3.12.2 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 head-
ing which is also added to the table of contents.

3.6.9 Bibliography Notes

\defbibnote{*<name>*}{*<text>*}

Defines the bibliography note *<name>*, 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 § 3.12.6.

3.6.10 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

(\dots) , 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=_{⟨keyword⟩}

Matches all entries whose **keywords** field includes _{⟨keyword⟩}. If the _{⟨keyword⟩} contains spaces, it needs to be wrapped in braces.

category=_{⟨category⟩}

Matches all entries assigned to _{⟨category⟩} with

\addtocategory.

Here is an example of a filter expression:

```
\defbibfilter{example}
{ type=book or
  and keyword=abc
  and not keyword
}
```

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 `\ifbool`expr
 command from the `etoolbox`
`etoolbox`
 package. See the
 manual for details about the syntax. The syntax of the
`\ifthenelse` command

`ifthen`
 from the `ifthen` package, which has been employed
 in older versions of Bibl_{at}ex, is still supported. This is the same test using
`ifthen` -like syntax:

```
\defbibfilter{example}{
  \iftype{book}
  \and \keyword{a}
  \and \not \keyword{b}
}
```

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`

`{\langle name \rangle}{\langle code \rangle}`

Defines the custom bibliography filter `\langle name \rangle`, to be

used via the `check` option of

`\printbibliograph`

`\defbibcheck` is simi-

lar in concept to `\defbibfilter`

but much more low-level. Rather than a high-level expression, the `\langle code \rangle` 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 `\langle code \rangle` is executed. Issuing the

command `\skipentry` in the `\langle code \rangle` will cause the current entry to be skipped. For example, the following filter

will only output entries with an `abstract` field:

```
\defbibcheck{abst
```

```
  \iffieldundef{a
```

```
  .  
  .  
  .  
\printbibliograph
```

The following check will exclude all entries published before the year 2000:

```
\defbibcheck{rece
```

```
\iffielddint{yea
{\ifnumless{\
{\skipentr
{}}
{\skipentry}}
```

See the author guide, in particular §§ [4.6.2](#) and [4.6.3](#), for further details.

3.6.11 *Dynamic Entry Sets*

@set

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 $\langle key \rangle$:

\defbibentryset

```
{\langle key \rangle}{\langle key1,key2,key3,... \rangle}
```

Biber only

The $\langle key \rangle$ is the entry key of the set, which is used like any other entry key when referring to the set. The $\langle key \rangle$ 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

\nocite

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 $\langle key \rangle$ are ig-

nored and work like $\langle key \rangle$. Dynamic entry sets defined in the document body are local to the enclosing

environment, if any. Otherwise, they are assigned to reference section 0. Those defined in the preamble are assigned to reference section 0. Note that dynamic entry sets require Biber. They will not work with any other backend. See § 3.11.5 for further details.

3.7 Citation Commands

All citation commands generally take one mandatory and two optional arguments. The $\langle prenote \rangle$ is text to be printed at the beginning of the citation. This is usually a notice such as ‘see’ or ‘compare’. The $\langle postnote \rangle$ 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

$\backslash cite[see][]\{key\}$.

The $\langle key \rangle$ 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

$\backslash bib$ file. In sum, all basic citations commands listed further down have the following syntax:

$\backslash command[\langle prenote \rangle][\langle postnote \rangle]\{ \langle keys \rangle \} \langle punctuation \rangle$

If the $\backslash autopunct$ package option from § 3.1.2.1 is enabled, they will scan ahead for any $\langle punctuation \rangle$ immediately following their last argument. This is useful to avoid spurious punctuation marks after citations. This feature is configured with

$\backslash DeclareAutoPunct$ see § 4.7.5 for details.

3.7.1 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.

`\cite` $\left[\left\langle \textit{prenote} \right\rangle \right] \left[\left\langle \textit{postnote} \right\rangle \right] \left\{ \left\langle \textit{key} \right\rangle \right\}$
`\Cite` $\left[\left\langle \textit{prenote} \right\rangle \right] \left[\left\langle \textit{postnote} \right\rangle \right] \left\{ \left\langle \textit{key} \right\rangle \right\}$

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 other-

wise. `\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` $\left[\left\langle \textit{prenote} \right\rangle \right] \left[\left\langle \textit{postnote} \right\rangle \right] \left\{ \left\langle \textit{key} \right\rangle \right\}$
`\Parencite` $\left[\left\langle \textit{prenote} \right\rangle \right] \left[\left\langle \textit{postnote} \right\rangle \right] \left\{ \left\langle \textit{key} \right\rangle \right\}$

These commands 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` [*<prenote>*] [*<postnote>*] { *<key>* }

`\footcitetext`
[*<prenote>*] [*<postnote>*] { *<key>* }

`\cite`

These commands 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`

`\footcite` differs from `\footcitetext` in that it uses

`\footnotetext`

instead of `\footnote`.

3.7.2 Style-specific Commands

The following additional citation commands are only provided by some of the citation styles which ship with this package.

`\textcite` [*<prenote>*] [*<postnote>*] { *<key>* }

`\Textcite` [*<prenote>*] [*<postnote>*] { *<key>* }

These citation commands are provided by all styles that ship with this package. They 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` [*<prenote>*] [*<postnote>*] {*<key>*}
`\Smartcite` [*<prenote>*] [*<postnote>*] {*<key>*}

Like `\parencite` in a footnote
 and like `\footcite` in the body.

`\cite*` [*<prenote>*] [*<postnote>*] {*<key>*}

This command is provided by all author-year and author-title styles. It 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`
`\multicitedelim`
 instead of
 as citation delimiter. Note that any `\prenote` and `\postnote` arguments are ignored.

`\supercite`
 If they are given,
 will discard them and issue a warning message.

3.7.3 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 com-

mands like `\parencite`
`\footcite`
 and

The citation style provides a multicite definition with

`\DeclareMultiCite`

(see § 4.3.1). The following example illustrates the syntax of multicite commands:

`\parencites[35]{k`

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]{k`

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 c`

`\parencites (Compa`
`\parencites (See) (`

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] {k`
`\parencites [35] {k`

By default, this package provides the following multicite commands which correspond to regular commands from §§ 3.7.1 and 3.7.2:

`\cites (<multiprenote>) (<multi postnote>) [<prenote>] [<postnote>]`
`\Cites (<multiprenote>) (<multi postnote>) [<prenote>] [<postnote>]`

The multicite version of `\cite` and
`\Cite`, respectively.

`\parencites` ($\langle \text{multiprenote} \rangle$) ($\langle \text{multipostnote} \rangle$) [$\langle \text{pre} \rangle$]
`\Parencites` ($\langle \text{multiprenote} \rangle$) ($\langle \text{multipostnote} \rangle$) [$\langle \text{pre} \rangle$]

The multicite version of `\parencite`
 and `\Parencite`, respectively.

`\footcites` ($\langle \text{multiprenote} \rangle$) ($\langle \text{multipostnote} \rangle$) [$\langle \text{pre} \rangle$]
`\footcitetexts`
 ($\langle \text{multiprenote} \rangle$) ($\langle \text{multipostnote} \rangle$) [$\langle \text{prenote} \rangle$] [$\langle \text{postnote} \rangle$] { $\langle \text{key} \rangle$ }

The multicite version of `\footcite`
 and `\footcitetext`, respectively.

`\smartcites` ($\langle \text{multiprenote} \rangle$) ($\langle \text{multipostnote} \rangle$) [$\langle \text{pre} \rangle$]
`\Smartcites` ($\langle \text{multiprenote} \rangle$) ($\langle \text{multipostnote} \rangle$) [$\langle \text{pre} \rangle$]

The multicite version of `\smartcite`
 and `\Smartcite`, respectively.

`\textcites () () [` `\Textcites () () [`

The multicite version of `\textcite`
and `\Textcite`, respectively.

`\supercites () () [`

The multicite version of `\supercite`.
This command is only provided by the numeric styles.

3.7.4 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 super-

scripts) possible. The `\autocite` command is built on top of backend commands like `\parencite` and `\footcite`. The citation style pro-

vides an `\autocite` definition with `\DeclareAutoCiteC`
 (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).

`\autocite` [*<prenote>*] [*<postnote>*] {*<key>*}

`\Autocite` [*<prenote>*] [*<postnote>*] {*<key>*}

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.

`\autocite*` $\left[\langle \text{prenote} \rangle \right] \left[\langle \text{postnote} \rangle \right] \left\{ \langle \text{key} \rangle \right\}$
`\Autocite*` $\left[\langle \text{prenote} \rangle \right] \left[\langle \text{postnote} \rangle \right] \left\{ \langle \text{key} \rangle \right\}$

`\autocite`
 The starred variants of 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*`.

`\autocites` $\left(\langle \text{multiprenote} \rangle \right) \left(\langle \text{multipostnote} \rangle \right) \left[\langle \text{pre} \right]$
`\Autocites` $\left(\langle \text{multiprenote} \rangle \right) \left(\langle \text{multipostnote} \rangle \right) \left[\langle \text{pre} \right]$

This is the multicite version of `\autocite`.

It also detects and moves punctuation if required. Note that there is no star-

red variant.

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.

3.7.5 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.

`\citeauthor` [*⟨prenote⟩*] [*⟨postnote⟩*] {*⟨key⟩*}

`\citeauthor*` [*⟨prenote⟩*] [*⟨postnote⟩*] {*⟨key⟩*}

`\Citeauthor` [*⟨prenote⟩*] [*⟨postnote⟩*] {*⟨key⟩*}

`\Citeauthor*` [*⟨prenote⟩*] [*⟨postnote⟩*] {*⟨key⟩*}

These commands print the authors. Strictly speaking, it prints the

`\labelname` list, which may be the
`author`, the `editor`,
`translator`.
 or the `\Citeauthor` is sim-

ilar 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.

`\citetitle`[*<prenote>*][*<postnote>*]{*<key>*}
`\citetitle*`[*<prenote>*][*<postnote>*]{*<key>*}

This command prints the title. It will use the abridged title in the
`shorttitle` field, if available.

Otherwise it falls back to the full title found in the
field. The starred variant always prints the full title.

`\citeyear`[*<prenote>*][*<postnote>*]{*<key>*}
`\citeyear*`[*<prenote>*][*<postnote>*]{*<key>*}

This command prints the year (`year` field or year com-

ponent of `date`). The starred variant includes the
`extrayear` information, if any.

`\citedate` $\left[\langle \textit{prenote} \rangle \right] \left[\langle \textit{postnote} \rangle \right] \{ \langle \textit{key} \rangle \}$
`\citedate*` $\left[\langle \textit{prenote} \rangle \right] \left[\langle \textit{postnote} \rangle \right] \{ \langle \textit{key} \rangle \}$

This command prints the full date (**date**)
or **year**). The starred variant includes the
extrayear information, if any.

`\citeurl` $\left[\langle \textit{prenote} \rangle \right] \left[\langle \textit{postnote} \rangle \right] \{ \langle \textit{key} \rangle \}$

This command prints the **url** field.

`\parentext` $\{ \langle \textit{text} \rangle \}$

This command wraps the $\langle \textit{text} \rangle$ in context sensitive parentheses.

`\brackettext` $\{ \langle \textit{text} \rangle \}$

This command wraps the $\langle \textit{text} \rangle$ in context sensitive brackets.

3.7.6 Special Commands

The following special commands are also provided by the core of Biblalex.

`\nocite` $\{ \langle \textit{key} \rangle \}$
`\nocite` $\{ * \}$

This command is similar to the standard LaTeX `\nocite` command. It adds the $\langle \textit{key} \rangle$ to the bibliography without printing a citation. If the $\langle \textit{key} \rangle$ is an asterisk, all

entries available in the **bib** file are added to the bibliography.

Like all other citation commands, **\nocite** commands in the document body are local to the enclosing

refsection environment,

if any. In contrast to standard LaTeX, **\nocite** may also be used in the document preamble. In this case, the references are assigned to reference section 0.

\fullcite[*<prenote>*] [*<postnote>*] { *<key>* }

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.

\footfullcite
[*<prenote>*] [*<postnote>*] { *<key>* }

Similar to **\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

\volcitedelim

(§ 4.10.1).

\volcites (*⟨multiprenote⟩*) (*⟨multipostnote⟩*) [*⟨prenote⟩*]
 . . . [*⟨prenote⟩*] { *⟨volume⟩* } [*⟨page⟩*] { *⟨key⟩* }

\Volcites (*⟨multiprenote⟩*) (*⟨multipostnote⟩*) [*⟨prenote⟩*]
 . . . [*⟨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` ($\langle\text{multiprenote}\rangle$) ($\langle\text{multipostnote}\rangle$) [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

`\Pvolcites` ($\langle\text{multiprenote}\rangle$) ($\langle\text{multipostnote}\rangle$) [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

`\pvolcite` ($\langle\text{multiprenote}\rangle$) ($\langle\text{multipostnote}\rangle$) [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

`\Pvolcite` ($\langle\text{multiprenote}\rangle$) ($\langle\text{multipostnote}\rangle$) [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

The multicite version of `\pvolcite`

and `\Pvolcite`, respectively.

`\fvolcite` [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

`\ftvolcite` [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

Similar to `\volcite` but

based on `\footcite` and

`\footcitetext`, respectively.

`\fvolcites` ($\langle\text{multiprenote}\rangle$) ($\langle\text{multipostnote}\rangle$) [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

... [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

`\Fvolcites` ($\langle\text{multiprenote}\rangle$) ($\langle\text{multipostnote}\rangle$) [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

... [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

The multicite version of `\fvolcite`
and `\Fvolcite`, respectively.

`\svolcite` [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

`\Svolcite` [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

Similar to `\volcite` but based on `\smartcite`.

`\svolcites` ($\langle\text{multiprenote}\rangle$) ($\langle\text{multipostnote}\rangle$) [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

... [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

`\Svolcites` ($\langle\text{multiprenote}\rangle$) ($\langle\text{multipostnote}\rangle$) [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

... [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

The multicite version of `\svolcite`
and `\Svolcite`, respectively.

`\tvolcite` [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

`\Tvolcite` [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

Similar to `\volcite` but based on
`\textcite`.

`\tvolcites` ($\langle \text{multiprenote} \rangle$) ($\langle \text{multipostnote} \rangle$) [$\langle \text{prenote} \rangle$] { $\langle \text{volume} \rangle$ } [$\langle \text{page} \rangle$] { $\langle \text{key} \rangle$ }
`\Tvolcites` ($\langle \text{multiprenote} \rangle$) ($\langle \text{multipostnote} \rangle$) [$\langle \text{prenote} \rangle$] { $\langle \text{volume} \rangle$ } [$\langle \text{page} \rangle$] { $\langle \text{key} \rangle$ }

The multicite version of `\tvolcite`
 and `\Tvolcite`, respectively.

`\avolcite` [$\langle \text{prenote} \rangle$] { $\langle \text{volume} \rangle$ } [$\langle \text{page} \rangle$] { $\langle \text{key} \rangle$ }
`\Avolcite` [$\langle \text{prenote} \rangle$] { $\langle \text{volume} \rangle$ } [$\langle \text{page} \rangle$] { $\langle \text{key} \rangle$ }

Similar to `\volcite` but based on
`\autocite`.

`\avolcites` ($\langle\text{multiprenote}\rangle$) ($\langle\text{multipostnote}\rangle$) [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }
`\Avolcites` ($\langle\text{multiprenote}\rangle$) ($\langle\text{multipostnote}\rangle$) [$\langle\text{prenote}\rangle$] { $\langle\text{volume}\rangle$ } [$\langle\text{page}\rangle$] { $\langle\text{key}\rangle$ }

The multicite version of `\avolcite`
 and `\Avolcite`, respectively.

`\notecite` [$\langle\text{prenote}\rangle$] [$\langle\text{postnote}\rangle$] { $\langle\text{key}\rangle$ }
`\Notecite` [$\langle\text{prenote}\rangle$] [$\langle\text{postnote}\rangle$] { $\langle\text{key}\rangle$ }

These commands print the $\langle\text{prenote}\rangle$ and $\langle\text{postnote}\rangle$ arguments but no citation. In-

stead, a `\nocite` command is issued for every $\langle\text{key}\rangle$. 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 $\langle\text{postnote}\rangle$ formatting and the implicit

`\nocite` 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 Bibl_{at}ex's punctuation tracker).

`\pnotecite` [$\langle prenote \rangle$] [$\langle postnote \rangle$] { $\langle key \rangle$ }
`\Pnotecite` [$\langle prenote \rangle$] [$\langle postnote \rangle$] { $\langle key \rangle$ }

`\notecite`
 Similar to `\notecite` but the notes are printed in parentheses.

`\fnotecite` [$\langle prenote \rangle$] [$\langle postnote \rangle$] { $\langle key \rangle$ }

`\notecite`
 Similar to `\notecite` but the notes are printed in a footnote.

3.7.7 Low-level Commands

The following commands are also provided by the core of Biblatex. They grant access to all lists and fields at a lower level.

`\citename` [$\langle prenote \rangle$] [$\langle postnote \rangle$] { $\langle key \rangle$ } [$\langle format \rangle$]

The $\langle format \rangle$ is a formatting directive defined with

`\DeclareNameFormat`

Formatting directives are discussed in § 4.4.2. If this optional argument is omitted, this command falls back to the format

`citename`. The last argument is the name of a $\langle name list \rangle$, in the sense explained in § 2.2.

`\citelist` [$\langle prenote \rangle$] [$\langle postnote \rangle$] { $\langle key \rangle$ } [$\langle format \rangle$]

The $\langle format \rangle$ 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}`

The *format* is a formatting directive defined with `\DeclareFieldForm`

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.

3.7.8 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

`\InitializeCitation`

command from § 4.3.1. It also resets the internal citation trackers of this package. The reset will affect the

`\ifciteseen`,

`\ifentryseen`,

`\ifciteibid`,

`\ifciteidem`

and tests discussed in § 4.6.2. When used inside a

`refsection` environment, the reset of the citation tracker is local to the current

refsection
citereset environment.
package

Also see the
option in § 3.1.2.1.

`\citereset*` Similar to
`\citereset` but only exe-
cutes the style's initialization hook, without resetting the internal citation
trackers.

`\mancite` Use this command to mark manually in-
serted 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
in the same context as the manual citation, e.g., if the citation is given

in a footnote, include

in the footnote. The
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

`\ifciteibid` and
`\ifciteidem` tests dis-
cussed in § 4.6.2.

`\pno` This command forces a single page prefix in the $\langle postnote \rangle$ argument to a
citation command. See § 3.12.3 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 § 3.12.3 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 § 3.12.3 for further details and usage instructions. Note that this command is only available locally in citations and the bibliography.

\psq In the *<postnote>* argument to a citation command, this command indicates a range of two pages where only the starting page is given. See § 3.12.3 for further details and usage instructions. The suffix printed is the localization string

sequens, see § 4.9.2. The spacing inserted between the suffix and the page number may be modified by redefining the macro

\sqspace. The default is an unbreakable interword space. Note that this command is only available locally in citations and the bibliography.

\psqq Similar to **\psq** but indicates an open-ended page range. See § 3.12.3 for further details and usage instructions. The suffix printed

is the localization string **sequentes**, see § 4.9.2. This command is only available locally in citations and the bibliography.

\RN{*<integer>*}

This command prints an integer as an uppercase Roman numeral. The formatting applied to the numeral may be modified by redefining the macro

\RNfont.

\Rn{*<integer>*}

Similar to **\RN** but prints a lowercase Roman numeral. The

formatting applied to the numeral may be modified by redefining the macro

`\Rnfont`.

3.7.9 **natbib** *Compatibility Commands*

The `natbib` package option loads a

`natbib` compatibility module. The module defines aliases for the citation commands provided by the

`natbib` package. This includes aliases

for the core citation commands `\citet`

and `\citep` as well as the vari-

ants `\citealt` and

`\citealp`. The starred variants of these commands, which print the full author list, are also supported. The

`\cite` command, which is handled in a particular way by

`natbib`, is not treated in a special way. The text com-

mands (`\citeauthor`,

`\citeyear`, etc.) are also

supported, as are all commands which capitalize the name prefix

(`\Citet`, `\Citep`,

`\Citeauthor`, etc.). Alias-
`\defcitealias`,
`\citetalias`, and
`\citepalias` is pos-

sible as well. Note that the compatibility commands will not emulate the
citation format of the `natbib` package. They

merely alias `natbib`'s commands to function-
ally equivalent facilities of the Biblatex package. The citation format de-
pends on the main citation style. However, the compatibility style will adapt

`\nameyear delim`
`natbib`
to match the default style of the `natbib` package.

3.7.10 `mcite`-like Citation Commands

Biber only

The `mcite` package option loads a spe-
cial citation module which provides `mcite`/

`mciteplus`-like citation com-
mands. Strictly speaking, what the module provides are wrappers for the commands
of the main citation style. For example, the following command:

`\mcite{key1, setA,`

is essentially equivalent to this:

`\defbibentryset{
\defbibentryset{
\cite{key1,setA,k`

The `\mcite` command will work with any style since
the `\cite` backend command is controlled by the main ci-

tation style as usual. The `mcite` module provides wrappers
for the standard commands in §§ 3.7.1 and 3.7.2. See table 7 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`
will execute:

`\defbibentryset{
\cite*[pre][post]`

Note that the `mcite` module is not a compati-
bility 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

<i>Standard Command</i>	mcite <i>-like Command</i>
<code>\cite</code>	<code>\mcite</code>
<code>\Cite</code>	<code>\Mcite</code>
<code>\parencite</code>	<code>\mparencite</code>
<code>\Parencite</code>	<code>\MParencite</code>
<code>\footcite</code>	<code>\mfootcite</code>
<code>\footcite-text</code>	<code>\mfootcite-text</code>
<code>\textcite</code>	<code>\mtextcite</code>
<code>\Textcite</code>	<code>\Mtextcite</code>
<code>\su-percite</code>	<code>\msu-percite</code>

Table 6: **mcite** *-like commands*

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,*sa`

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,*glas`

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 activat-

ing its `subentry` option, it is even pos-

sible to refer to set members. See table 7 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.

3.8 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.

`\DefineBibliographySt`

`{\langle language \rangle}{\langle definitions \rangle}`

This command is used to define localization strings. The `\langle language \rangle` must be a language name known to the

`babel`/`polyglossia`

packages, i. e., one of the identifiers listed in table 2 on page 78. The `\langle definitions \rangle` are

`\langle key \rangle` `\langle value \rangle` pairs which assign an expression to an identifier:

`\DefineBibliograp`

Input	Output	Comment
<code>\mcite{set1},*glashow,</code>	<code>\mcite{set1},*glashow,</code>	Defining and citing the set
<code>\mcite{set1}</code>	<code>\mcite{set1}</code>	Subsequent citation of the set
<code>\cite{set1}\cite</code>	<code>\cite{set1}\cite</code>	Regular works as usual
<code>\mcite{set1},*glashow,</code>	<code>\mcite{set1},*glashow,</code>	Redundant, but permissible
<code>\mcite{glashow}</code>	<code>\mcite{glashow}</code>	With a set number
<code>\cite{weinberg}\cite</code>	<code>\cite{weinberg}\cite</code>	Regular works as well
<code>berg}</code>	<code>berg}</code>	

mcite *-like syntax (sample out-*
put with **style** **=**
numeric *and*
subentry *option)*

```

bibliography      =
shorthands        =
editor            =
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 Bibl_atex 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

```

\DeclareBibliogra
\DefineBibliograp

```

overrides both the full and the abbreviated version of the string. See § 4.9.1 for further details.

```

\DefineBibliographyEx
{ } { }

```

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

```

babelpolyglossia

```

packages. The *code*, which may be arbitrary LaTeX code, will usually consist of redefinitions of the formatting commands from § 3.9.2.

`\UndefineBibliography`

```
{<language>} {<code>}
```

This command is used to restore the original definition of any commands modified with

`\DefineBibliograph`

If a redefined command is included in § 3.9.2, there is no need to restore its previous definition since these commands are adapted by all language modules anyway.

`\DefineHyphenationExc`

```
{<language>} {<text>}
```

`\hyphenation`

This is a LaTeX frontend to TeX's

command which defines hyphenation exceptions. The *<language>* must be a language name known to the

`babelpolyglossia`

packages. The *<text>* is a whitespace-separated list of words. Hyphenation points are marked with a dash:

```
\DefineHyphenatio  
hy-phen-ation e  
}
```

`\NewBibliographyString` `{\langle key \rangle}`

This command declares new localization strings, i.e., it initializes a new `\langle key \rangle` to be used in the `\langle definitions \rangle` of

`\DefineBibliography`

The `\langle key \rangle` argument may also be a comma-separated list of key names. The keys listed in § 4.9.2 are defined by default.

3.9 Formatting Commands

The commands and facilities presented in this section may be used to adapt the format of citations and the bibliography.

3.9.1 Generic Commands and Hooks

The commands in this section may be redefined with

`\renewcommand`

in the document preamble. 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`
`\BiberOnlyMark`
The punctuation printed between bibliography subentries of an entry set. The default definition is a semicolon and a space.

`\bibnamedelima`
`\BiberOnlyMark`
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 (§ 3.9.3). Please refer to § 3.12.4 for further details.

`\bibnamedelimb`
`\BiberOnlyMark`

This delimiter is inserted between the elements which make up a name part where

`\bibnamedelim`

does not apply. The default definition is an interword space penalized by the value of the

`\lownamepenalty`

counter (§ 3.9.3). Please refer to § 3.12.4 for further details.

`\bibnamedelimc`

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 (§ 3.9.3). Please refer to § 3.12.4 for further details.

`\bibnamedelimd`

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 (§ 3.9.3). Please refer to § 3.12.4 for further details.

`\bibnamedelimi`

`\BiberOnlyMark`

`\bibnamedelima`

This delimiter replaces 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

\BiberOnlyMark

The punctuation inserted after initials unless
\bibinitthyphendel

applies. The default definition is a period

\adddot). Please refer to § 3.12.4 for further details.

\bibinitdelim

\BiberOnlyMark

The spacing inserted between multiple initials unless
\bibinitthyphendel

applies. The default definition is an unbreakable interword space. Please refer to § 3.12.4 for further details.

\bibinitthyphendelim

\BiberOnlyMark

The punctuation inserted between the initials of hyphenated name parts, replacing

\bibinitperiod

\bibinitdelim

and

The default definition is a period followed by an unbreakable hyphen. Please refer to § 3.12.4 for further details.

`\bibindexnamedelim`

`\bibnamedelim`

Replaces
in the index.

`\bibindexnamedelim`

`\bibnamedelim`

Replaces
in the index.

`\bibindexnamedelim`

`\bibnamedelim`

Replaces
in the index.

`\bibindexnamedelim`

`\bibnamedelim`

Replaces
in the index.

`\bibindexnamedelim`

`\bibnamedelim`

Replaces
in the index.

`\bibindexinitperiod`

`\bibinitperiod`

Replaces
in the index.

`\bibindexinitdelim`

`\bibinitdelim`

Replaces
in the index.

`\bibindexinithyphende`

Replaces

in the index.

`\bibinithyphende`

`\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

`\bibnamedeli`

This command should be used with

as a reversed-name separator in formatting directives for name lists. Please refer to § 3.12.4 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 alphabetiz-

ing 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.

`\subtitledpunct` The separator printed between the fields `title` and `subtitle`, `booktitle` and `booksubtitle`, 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.

`\intitelpunct` The separator between the word “in” and the following title in entry types such as `@article`, `@inbook`, `@incollect` 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”, `\intitelpunct` should still insert a space.

`\bibpagespunct` The separator
`pages`
 printed before the field. The default is a comma plus
 an interword space.

`\bibpagerefspunct`
`pageref` field.
 The separator printed before the 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.²³

`\finalnamedelim` The de-

`\multinamedel`
 limiter printed instead of
 before the final name in a name list. The default is the localized term ‘and’, separated
 by interword spaces. Here is an example:

Michel Goossens,
 Edward Jones and

²³Note 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`.

The comma in the first example is the
 whereas the string ‘and’ in both examples is the

`\multinamede`
`\finalnamedelim`
`\finalandcomma`

See also
 in § 3.9.2.

`\revsdnamedelim`

An extra delimiter printed after the first name in a name list if the first name is reversed. 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, an

In this example, the comma after ‘Edward’ is the

`\revsdnamedelim`
`\finalnamede`

whereas the string ‘and’ is the
 printed in addition to the former.

`\andothersdelim`

The delimiter printed before the localization string

`\andothers`, if a name list like
 author editor is
 truncated. The default is an interword space.

`\multilistdelim`

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.

`\finallistdelim` The de-

limiter printed instead of before the final item in a literal list. The default is the localized term ‘and’, separated by interword spaces. See

`\finalnamedelim` for further explanation.

`\andmoredelim` The delimiter

printed before the localization string ‘andmore’,

if a literal list like publisher or

location is truncated. The default is an interword space.

`\multicitedelim` The de-

limiter printed between citations if multiple entry keys are passed to a single citation command. The default is a semicolon plus an interword space.

`\supercitedelim` Simi-

lar to `\multicitedelim`,

but used by the `\supercite` command only. The default is a comma.

`\compcitedelim` Similar to
`\multicitedelim`,
but used by certain citation styles when ‘compressing’ multiple citations. The default definition is a comma plus an interword space.

`\textcitedelim` Similar to
`\multicitedelim`,
`\textcite`
but used by and related
commands (§ 3.7.2). 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 localized term ‘and’, separated by interword spaces. See also

`\finalandcomma`
`\finalandsemicolon`
and
in § 3.9.2.

`\nametitledelim` The de-
limiter printed between the author/editor and the title by author-title and some
verbose citation styles. The default definition is a comma plus an interword space.

`\nameyeardelim` The delimiter
printed between the author/editor and the year by author-year citation styles. 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 ci-
tation styles) if the number of authors/editors exceeds the

maxalphanames

author /

threshold or the 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.

`\sortalphaothers`

`\BiberOnlyMark`

`\labelalphaothers`

Similar to

but used in the sorting process. Setting it to a different value is advisable if the latter contains formatting commands, for example:

`\renewcommand*{\l`

`\renewcommand*{\s`

If `\sortalphaothers`

`\labelalphaothers`

is not redefined, it defaults to

`\prenotedelim`

The delimiter

printed after the `\prenote` argument of a citation command. See § 3.7 for details. The default is an interword space.

`\postnotedelim` The delimiter
 printed before the `\postnote` argument of a citation command. See § 3.7 for details.
 The default is a comma plus an interword space.

`\mkbibnamelast`
 $\{ \langle \textit{text} \rangle \}$ This command, which takes one argument, is used to format
 the last name of all authors, editors, translators, etc.

`\mkbibnamefirst`
 $\{ \langle \textit{text} \rangle \}$ Similar to `\mkbibnamelast`
 but intended for the first name.

`\mkbibnameprefix`
 $\{ \langle \textit{text} \rangle \}$ Similar to `\mkbibnamelast`
 but intended for the name prefix.

`\mkbibnameaffix`
 $\{ \langle \textit{text} \rangle \}$ Similar to `\mkbibnamelast`
 but intended for the name affix.

`\relatedpunct` The separator
`relatedtype`
 between the bibliography localization string and the data from
 the first related entry. Here is an example with
`\relatedpunct`
 set to a dash:

A. Smith. Title.

`\relateddelim` The 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:

Donald E. Knuth.
Wesley, 1984-1986
Vol. A: The TEXbo
Vol. B: TEX: The
Vol. C: The METAF
Vol. D: METAFONT:
Vol. E: Computer

3.9.2 Language-specific Commands

The commands in this section are language specific. When redefining them, you need to wrap the new definition in a

`\DeclareBibliogra`
`.lbx`
command (in an file) or a
`\DefineBibliograp`

command (user documents), see § 3.8 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.

`\bibrangessep`
`\BiberOnlyMark`

The language specific separator to be used between multiple ranges.

`\bibdatedash` The language specific dash to be used for date ranges.

`\mkbibdate`long 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 `\mkbibdate`long but using the language specific short date 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:

Michel Goossens,

`\finalandcomma`

is the comma before the word ‘and’. See also

`\multinamedelim`,
`\finalnamedelim`,
`\textcitedelim`,
`\revsdnamedelim`

and
in § 3.9.1.

`\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:

Goossens, Mittelb

`\finalandsemicolon`

is the semicolon before the word ‘and’. See also

`\textcitedelim`

in § 3.9.1.

`\mkbibordinal{ }`

integer

This command, which takes an integer as its argument, prints an ordinal number.

`\mkbibmascord{ }`

integer

`\mkbibordinal`,

Similar to

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’.

3.9.3 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 in-

serted 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 in-

serted 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 localization modules, holds the penalty used in short or abbreviated localization 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).²⁴

`highnamepenalty`

This counter holds a penalty affecting line-breaking in names. Please refer to §§ 3.12.4 and 3.9.1 for explanation. The counter is initialized to

`\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

²⁴The default values assigned to `abbrvpenalty`,

`lownamepenalty`, and
`highnamepenalty`

are deliberately very low to prevent overfull boxes. This implies that you will hardly notice any 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 `\raggedright`

to prevent suboptimal linebreaks. In this case, even the fairly low default penalties will make a visible difference.

highnamepenalty

breakpoints), set it to ‘infinite’ (10 000 or higher).

lownamepenalty

Similar to

highnamepenalty

Please refer to §§ 3.12.4 and 3.9.1 for explanation. The counter is initialized to half the

\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.

3.9.4 All-purpose Commands

The commands in this section are all-purpose text commands which are generally available, not only in citations and the bibliography.

\bibellipsis

An ellipsis symbol with

brackets: ‘[...]’.

\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

babelpolyglossia

packages.

\hyphenate

A conditional hyphen. In contrast to the

standard

command, this one allows hyphenation in the rest of the word.

It is similar to the

shorthand provided by some language modules of the

babelpolyglossia

packages.

\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

babelpolyglossia

packages.

\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

babelpolyglossia

packages.

\nohyphenation

A generic switch which suppresses hyphenation locally. Its scope should normally be confined to a group.

\textnohyphenation

{_{*text*}}

\nohyphenation

Similar to

but restricted to the *text* argument.

`\mknumalph{}`

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 for-

matting directives for the `extrayear`
and `extraalpha` fields.

`\mkbibacro{}`

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{}`

Automatically converts the `<character>` to its uppercase form if Biblatex’s punctuation tracker would capitalize a localization string at the current location. This command is robust. It is useful for conditional capitalization of certain strings in an entry. Note that the `<character>` argument is a single character given in lowercase. For example:

`\autocap{s}pecial`

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 `<character>` argument as follows:

`\autocap{\ ’e}diti`

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 `<character>` argument:

`\autocap{\oe}uvre`

This will yield ‘Œuvres’ or ‘œuvres’.

3.10 Language-specific Notes

The facilities discussed in this section are specific to certain localization modules.

3.10.1 American

The American localization module uses

from § 4.7.5 to enable ‘American-style’ punctuation. If this feature is enabled, all trailing commas and periods after

will be moved inside the quotes. If you want to disable this feature, use

as follows:

```
\uspunctuat
\mkbibquote
\stdpunctuation
\DefineBibliograph
\stdpunctuation
}
```

By default, the ‘American punctuation’ feature is enabled by the

american localization module only. The above code is only required if you want American localization 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 american, british, australian, english etc. rather than when loading the babelpolyglossia

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.

3.10.2 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 localization mod-

ule does not use the localization string ‘**and**’ but a special internal ‘smart and’ command. The behavior of this command is controlled by the **smartand** counter.

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 localization module ignores

and **\finalnamedelim**
\finallistdelim.

\forceE Use this command in **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 prevent confusing BibTeX. Here is an example:

```
author = {Edward
author = {Edward
```

Note that the initial letter of the respective name component is given as an argument to **\forceE** and that the entire construct is wrapped in an additional pair of curly braces.

\forceY Similar to **\forceE** but enforces ‘y’.

3.10.3 Greek

The Greek localization 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, provided that you use Biber as a backend, 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.

3.10.4 Russian

Like the Greek localization module, the Russian module also requires UTF-8 support. It will not work with any other encoding.

3.11 Usage Notes

The following sections give a basic overview of the Biblatex package and discuss some typical usage scenarios.

3.11.1 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:

```
\documentclass{...}
\usepackage[...]{...}
\addbibresource{...}
\begin{document}
\cite{...}
...
\printbibliography
\end{document}
```

With traditional BibTeX, the `\bibliography` command serves two purposes. It marks the location of the bibliography

and it also specifies the file(s). The file extension is omitted. With Biblatex, resources are specified in the preamble with

```
\addbibresource{...}
...
\printbibliography
```

using the full name with the suffix. The bibliography is printed using the

\printbibliography

command which may be used multiple times (see § 3.6 for details). The document body may contain any number of citation commands (§ 3.7). 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:

3.11.1.1 Biber

1. Run **latex 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 **biber example.bcf**. Biber will retrieve the data from

bibfile.bib and

write it to the auxiliary file **example.**

bb1 in a format which can be processed by Biblatex.

3. Run **latex example.tex**. Biblatex will read the data from

example.bbl

and print all citations as well as the bibliography.

3.11.1.2 BibTeX

1. Run `latex example.tex`

If the file contains any citations, Bibl_{at}ex will request the respective data from BibTeX by writing commands to the auxiliary file

`example.aux`

2. Run `bibtex example.aux`

BibTeX will retrieve the data from `bibfile.bib` and

write it to the auxiliary file `example.bbl` in a format which can be processed by Bibl_{at}ex.

3. Run `latex example.tex`

Bibl_{at}ex will read the data from `example.bbl`

and print all citations as well as the bibliography.

Whenever a reference to a work which has not been cited before is added, this procedure must be repeated. This is also the case if the last reference to a work which has been cited before is removed because some citation labels may change in this case. In contrast to traditional BibTeX, there is normally no need to run

latex twice after running the backend as far as the handling of bibliographic data is concerned.²⁵

Note that when using BibTeX as the backend this only applies to BibTeX only

the most basic case. Using the xref field or the entryset field may require an additional LaTeX/BibTeX/LaTeX cycle. Some other facilities provided by Biblatex

latex may also require an additional run to get certain references and the page tracking right. In this case, the usual warning messages such as “There were undefined references” and “Label(s) may have changed. Rerun to get cross-references right” will be printed.

3.11.2 Auxiliary Files

3.11.2.1 Biber

The Biblatex package uses one auxiliary bcf file only. Even if there are citation commands in a file included via

\include, you only need to run Biber

on the main bcf file. All information Biber needs is in the

bcf file, including information about all refsections if using multiple

refsection environments (see § 3.11.3).

3.11.2.2 BibTeX

By default, the Biblatex package uses the main aux file only. Even if there are citation commands in a file in-

cluded via \include, which

defernumbers

²⁵That is, unless the package option is enabled. See § 4.1

has its own **aux** file, you only need to run
 BibTeX on the main **aux** file. If you are using
refsection environments in a
 document (see § 3.11.3) Bibl_{at}ex will create one additional **aux** file for
 every **refsection** environment.

In this case, you also need to run **bibtex** on each
 additional **aux** file. The name of the additional **aux**
-blx

files is the base name of the main input file with the string
 and a running number appended at the end. The Bibl_{at}ex package issues a warning
 listing the files which require an additional BibTeX run. With the basic example
 presented in § 3.11.1, it would issue the following warning:

Package bibl_{at}ex
(bibl_{at}ex)
(bibl_{at}ex)
refsection

If the input file contained three
 environments, the warning would read as follows:

Package bibl_{at}ex
(bibl_{at}ex)
(bibl_{at}ex)

(biblatex)
(biblatex)

aux

Apart from these files, Biblatex uses an additional

bib

file with the same suffix to pass certain control parameters to BibTeX. In the example above, this file would be named

example-blx.

bib

. In the event of a file name conflict, you can change the suffix by redefining the macro

\blxauxsuffix

in the document preamble. When using Biber, Biblatex writes a control file named

example.bcf

and ignores

\blxauxsuffix.

bib

There is also no auxiliary file in this case.

Note that Biblatex will not overwrite any files it did not create. All auxiliary files created automatically by this package start with a special signature line. Before

aux

overwriting a file (excluding the main file, which is managed by LaTeX), Biblatex inspects the first line of the file to make sure that there is no file name conflict. If the file in question is missing the signature line, Biblatex will immediately issue an error message and abort before opening the output stream. In this case you should delete any spurious files accidentally left in the working directory. If the error persists, there may be a file name conflict with a file found in one of the TeX installation trees. Since the installation trees usually do not

aux

contain any files and the string

-blx

is

bib

fairly exotic in the name of a file, this is rather unlikely but theoretically possible. If you find out that this is indeed the case, you should redefine

`\bblauxsuffix`
permanently in the Biblalex configuration file,
`biblalex.cfg.`

3.11.3 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 `\chapter` with its own bibliography.

Note that with the BibTeX backend, Biblalex creates one additional `aux` file for every `refsection` environment.

These files have to be processed by BibTeX as well, see § 3.11.2 for details.

```
\documentclass{...}
\usepackage{biblalex}
\addbibresource{...}
\begin{document}
\chapter{...}
\begin{refsection}
...
\printbibliography
```

```

\end{refsection}
\chapter{...}
\begin{refsection}
...
\printbibliograph
\end{refsection}
\end{document}

```

If `\printbibliograph` 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 `\printbibliograph` to select a reference section, as shown in the next example.

```

\documentclass{...}
\usepackage{bibla}
\defbibheading{su

```

```

\section*{Refer
\addbibresource{.
\begin{document}
\chapter{...}
\begin{refsection
. . .
\end{refsection}
\chapter{...}
\begin{refsection
. . .
\end{refsection}
\printbibheading
\printbibliograph
\printbibliograph
\end{document}

```

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 begin-

ning of every

environment, using the standard command. The identifier used is the string

`refsection:` followed by

the number of the respective environment. The number of the current section is accessible via the

`refsection` counter.

When using the `section` option of

`\printbibliograph`

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` pack-

age and using `\nameref` instead of

`\ref`:

`\usepackage{namer`

`\defbibheading{su`

`\section*{\name`

`\printbibliogra`

Since giving one command for each part of a subdivided bibliogra-

phy is tedious, Biblatex provides a shorthand. The

`\bibbysection`

command automatically loops over all reference sections. This is equivalent to giving one

`\printbibliograph`

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[hea`

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. Note that when using BibTeX as the backend,

`refsegment` environments

`aux` do not require additional files. The next example could also be given in § 3.11.4 because, visually, it creates one global bibliography subdivided into multiple segments.

```
\documentclass{..
```

```
\usepackage{bibla  
\defbibheading{su  
    \section*{Refer  
\addbibresource{.  
\begin{document}  
\chapter{...}  
\begin{refsegment  
.  
.  
.  
\end{refsegment}  
\chapter{...}  
\begin{refsegment  
.  
.  
.  
\end{refsegment}  
\printbibheading  
\printbibliograph  
\printbibliograph
```

`\end{document}`

The use of `refsegment` is sim-

ilar to `refsection` and there

is also a corresponding `segment` option for

`\printbibliograph`

The Biblatex package automatically sets a label at the beginning of every

`refsegment` environment

using the string `refsegment:`

followed by the number of the respective `refsegment`

environment as an identifier. There is a matching

`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:

`\printbibheading`

`\bibbysegment[hea`

`\printbibliog`

This is equivalent to giving one

command for every segment in the current

`refsection`.

3.11.4 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

type and nottype filters of
`\printbibliography`

The next example also demonstrates how to generate matching subheadings for the two parts of the bibliography.

```
\documentclass{. . .}
\usepackage{bibla}
\addbibresource{. . .}
\begin{document}
  \printbibheading
  \printbibliography

  \printbibliography

\end{document}
```

You may also use more than two subdivisions:

```
\printbibliograph
\printbibliograph
\printbibliograph
```

It is even possible to give a chain of different types of filters:

```
\printbibliograph
```

This would print all works cited in reference section 2

whose entry type is **@book** and whose

keywords field includes the keyword 'abc' but not 'xyz'. When using bibliography filters in conjunction with a numeric style, see § [3.12.5](#). If you need complex filters

with conditional expressions, use the **filter** option in conjunction with a custom filter defined with

```
\defbibfilter
```

See § [3.6.10](#) for details on custom filters.

```
\documentclass{. .
\usepackage{bibla
\addbibresource{.
\begin{document}
. . .
```

```

\printbibheading
\printbibliograph

\printbibliograph

\end{document}

```

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:

```

\printbibliograph
\printbibliograph

```

Since 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 **keywords**

field to each entry in the **bib** file. These keywords may

then be used as targets for the **keyword** and

notkeyword filters, as shown

above. It may be a good idea to add such keywords right away while building

a `bib` file.

```
@Book{key,  
  keywords  
  . . .
```

=

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

`bib` file.

```
\documentclass{. .  
\usepackage{bibla  
\DeclareBibliogra  
\DeclareBibliogra  
\addtocategory{pr  
\addtocategory{se  
\addbibresource{.  
\begin{document}  
  . . .  
\printbibheading
```



```
\printbibliograph
```

```
\printbibliograph
```

```
\end{document}
```

In this case it would also be sufficient to use one category only:

```
\printbibliograph  
\printbibliograph
```

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

```
\printbibliograph
```

command for every category, in the order in which they were declared.

```
\documentclass{..
```

```
\usepackage{bibla
```

```
\DeclareBibliogra
```

```

\DeclareBibliography
\addtocategory{pr
\addtocategory{se
\defbibheading{pr
\defbibheading{se
\addbibresource{.
\begin{document}
.
\printbibheading
\bibbycategory
\end{document}

```

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 equiva-

lent to passing

```
heading=
```

 $\langle category \rangle$ to

\printbibliography

and implies that you need to provide a matching heading for every category.

3.11.5 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 en-

try sets. Static entry sets are defined in the **bib** file like any other entry. Dynamic entry sets are defined with

\defbibentryset

(§ 3.6.11) 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.

3.11.5.1 Static entry sets

Static entry sets are defined in the **bib** file like any other entry. When using Biber as the backend, 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:

```
@Set{set1,  
entryset = {key  
}
```

Entries may be part of a set in one document/refsection and stand-alone references in

another one, depending on the presence of the **@set** entry. If the

@set entry is cited, the set members are grouped automatically. If not, they will work like any regular entry.

When using BibTeX as the backend, which has no native support for entry sets, setting up entry sets involves more work.

BibTeX requires **entryset**

and **crossref** fields to be used in a special way. The members of the set are given in

the **entryset** field of the

@set entry. The **@set** entry also requires a

crossref field which points to the first key

in the **entryset** field. In addition to that,

all members of the set require **entryset**

fields which are reverse pointers to the entry key of the **@set** head entry:

```
@Set{set1,  
    entryset = {key  
    crossref = {key  
}  
@Article{key1,
```

```

        entryset = {set
author          = {...
title          = {...
        }
@InCollection{key
        entryset = {set
author          = {...
title          = {...
        }
@Article{key3,
        entryset = {set
author          = {...
title          = {...
        }

```

Note that citing any set member will automatically load the entire set with BibTeX. If you want to refer to an item as part of a set in one document/refsection and as a stand-alone reference in another one, you need two distinct entries with BibTeX.

3.11.5.2 *Dynamic entry sets*

Biber only

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

`\defbibentryset`

command from § 3.6.11:

`\defbibentryset{s`

Dynamic entry sets 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. Note that dynamic entry sets require Biber. They will not work with any other backend.

3.11.6 *Data Containers*

Biber only

The `@xdata` entry type serves as a data container holding one or more fields. These fields may be inherited by other entries using

the `xdata` field. `@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:

`@XData{hup,`
`publisher = {H`

```

    location      = {C
}
@Book{... ,
    author        = {
    title         = {
    date          = {
    xdata         = {h
}

```

Using a separated list of keys in its **xdata** field, an en-

try may inherit data from several **@xdata** entries.

Cascading **@xdata** entries are supported as well, i. e.,

an **@xdata** entry may reference one or more other

@xdata entries:

```

@XData{macmillan:
    publisher      = {M

```

```

}
@XData{macmillan:
  location      = {N
}
@XData{macmillan,
  xdata         = {m
}
@Book{... ,
  author        = {
  title         = {
  date          = {
  xdata         = {m
}

```

See also §§ [2.1.1](#) and [2.2.3](#).

3.11.7 Electronic Publishing Information

The Biblatex package provides three fields for electronic

publishing information:

eprint,

`eprinttype`, and
`eprintclass`. The
`eprint` field is a verbatim field simi-

lar to `doi` which holds the identifier of the item. The

`eprinttype` field holds the
resource name, i.e., the name of the site or electronic archive. The op-

tional `eprintclass`
field is intended for additional information specific to the resource indi-

cated by the `eprinttype`
field. This could be a section, a path, classification information, etc. If the

`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”:

```
eprint = {ide
eprinttype = {Res
```

The standard styles feature dedicated support for a few online archives. For arXiv

references, put the identifier in the `eprint`

field and the string `arxiv` in the
`eprinttype` field:

eprint = {mat
eprinttype = {arx

For papers which use the new identifier scheme (April 2007 and later) add the primary classification in the

eprintclass field:

eprint = {10
eprinttype = {ar
eprintclass = {cs

There are two aliases which ease the integration of arXiv entries.

archiveprefix
eprinttype;
is treated as an alias for
primaryclass is an
alias for eprintclass.

If hyperlinks are enabled, the eprint identifier will be transformed into a link to

arxiv.org. See the package option

arxiv in § 3.1.2.1 for further details.

For JSTOR references, put the stable JSTOR num-

ber in the eprint field

and the string `jstor` in the `eprinttype` field:

```
eprint = {num  
eprinttype = {jst
```

When using JSTOR's export feature to export citations in BibTeX format, JSTOR uses

the `url` field by default (where the `<number>` is a unique and stable identifier):

```
url = {http://www
```

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 "JSTOR: `<number>`" format which also supports hyperlinks. The `<number>`

becomes a clickable link if `hyperref` support is enabled.

For PubMed references, put the stable PubMed identifier in the `eprint` field

and the string `pubmed` in the `eprinttype` field. This means

that:

```
url = {http://www
```

becomes:

```
eprint = {pmi
eprinttype = {pub
```

and the standard styles will print “PMID: $\langle pmid \rangle$ ” instead of the lengthy 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 `eprint` field and the string `hdl` in the `eprinttype` field:

```
eprint = {han
eprinttype = {hdl
```

For Google Books references, put Google’s identifier in the `eprint` field and the string `googlebooks` in the `eprinttype` field. This means that, for example:

```
url = {http://boo
```

would become:

```
eprint = {XXu
eprinttype = {goo
```

and the standard styles would print “Google Books:

XXu4AkRVBBoC”

instead of the full URL. If hyperref support is enabled, the identifier will be a clickable link to Google Books.²⁶

eprint

Note that **eprint** is a verbatim field. Always give the identifier in its unmodified form. For example, there is no need to replace

with ****. Also see § 4.11.2 on how to add dedicated support for other eprint resources.

3.11.8 External Abstracts and Annotations

Styles which print the fields **abstract** and/or **annotation** may support an alternative way of adding abstracts or annotations to the bibliography. Instead of

bib including the text in the file, it may also be stored in an external LaTeX file. For example, instead of saying

```
@Article{key1,  
  abstract  
}
```

²⁶Note that the Google Books 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 ID on Google Books. You may prefer to use the

url in this case.

in the `bib` file, you may create a file named `bibabstract-key1.tex` and put the abstract in this file:

```
This is an abstract
\endinput
```

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
\bibannotationprefix
```

and

Note that this feature needs to be enabled explicitly by setting the pack-

age 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.

3.12 Hints and Caveats

This section provides additional usage hints and addresses some common problems and potential misconceptions.

3.12.1 Usage with KOMA-Script Classes

When using Biblatex in conjunction with one

of the `scrbook`,

`scrreprt`, or

`scrartcl` classes, the headings

`bibliography`

and `biblist`

from § 3.6.8 are responsive to the bibliography-related options of these classes.²⁷ You can override the de-

²⁷This applies to the traditional syntax of these options (`bibtotoc` and `bibtotocnumbered`),

as well as to the `<key>=value` syntax introduced in KOMA-Script 3.x, i.e., to

`bibliography =`

`nottotoc`, `bibliography =`

`totoc`, and

`bibliography =`

`totocnumbered`,

`toc = bibliograph`

`graph` and `toc = bib-`

fault headings by using the `heading` option of

`\printbibliograph`
`\printbibheading`
`\printbiblist`.

See §§ 3.6.2, 3.6.4, 3.6.8 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:

`\ifkomabibtoc`
 $\{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ if the class would add the bibliography to the table of contents, and to $\langle false \rangle$ otherwise.

`\ifkomabibtocnumbered`
 $\{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ if the class would add the bibliography to the table of contents as a numbered section, and to $\langle false \rangle$ otherwise. If this test yields $\langle true \rangle$,

`\ifkomabibtoc`
will always yield $\langle true \rangle$ as well, but not vice versa.

`liographynum-`
`bered`

options as well as their aliases are detected as well. In any case, the options must be set globally in the optional argument to

`\documentclass`.

3.12.2 Usage with the Memoir Class

When using Biblatex with the `memoir` class, most class facilities for adapting the bibliography have no effect. Use the corresponding facilities of this package instead (§§ 3.6.2, 3.6.8,

3.6.9). Instead of redefining `memoir`'s

`\bibsection`,

use the `heading` option of

`\printbibliography`

and `\defbibheading`

(§§ 3.6.2 and 3.6.8). Instead of `\prebibhook`

and `\postbibhook`,

use the `prenote` and

`postnote` options of

`\printbibliography`

and `\defbibnote`

(§§ 3.6.2 and 3.6.9). All default headings are adapted at load-time such that they blend well with the default layout of this class. The default headings

`bibliography`

and `biblist` (§ 3.6.8) are

also responsive to memoir's
`\bibintoc` and
`\nobibintoc` switches.
`\bibitemsep`
 The length register

is used by Biblatex in a way similar to memoir
 (§ 3.9.3). This section also introduces some additional length reg-

isters which correspond to memoir's
`\biblistextra`.
`\setbiblabel`
 Lastly,

does not map to a single facility of the 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 memoir class is detected, Biblatex will also provide the following additional test which may be useful in custom heading definitions:

`\ifmemoirbibintoc`
 $\{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ or $\langle false \rangle$, depending on memoir's
`\bibintoc` and
`\nobibintoc` switches.

This is a LaTeX frontend to memoir's

\ifnobibintoc

test. Note that the logic of the test is reversed.

3.12.3 Page Numbers in Citations

If the $\langle postnote \rangle$ argument to a citation command is a page number or page range, 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, 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 de-

pends on the **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 $\langle postnote \rangle$. All commands used in this argument must therefore be robust or prefixed with

\protect. Here are a few examples of $\langle postnote \rangle$ arguments which will be correctly recognized as a single number, a range of numbers, or a list of numbers, respectively:

```
\cite[25]{key}
\cite[vii]{key}
\cite[XIV]{key}
\cite[34 - - 38]{key}
\cite[iv - - x]{key}
\cite[185/86]{key}
\cite[XI \& XV]{k
```

`\cite[3, 5, 7]{ke`
`\cite[vii--x; 5,`

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 § 3.7.8. 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]{k`

There is also a `\ppno` command which forces a range prefix

as well as a `\nopp` command which suppresses all prefixes:

`\cite[\ppno~27a--`
`\cite[\nopp 25]{k`

These commands may be used anywhere in the *⟨postnote⟩* 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`
`\cite[VII, \pno~3`

`\cite[see][\ppno~`

There are also two auxiliary commands 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`

`\cite[\ppno~55~sq`

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]{ke`

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:

\DeclareFieldForm

The $\langle postnote \rangle$ 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.

3.12.4 Name Parts and Name Spacing

The Biblatex package gives users and style authors very fine-grained control of name spacing and the line-breaking behavior of names, especially when they are using Biber as the backend. The commands discussed in the following are documented in §§ 3.9.1 and 4.10.1. This section is meant to give an overview of how they are put together. A note on terminology: a name *part* is a basic part of the name, for example the first or the last name. Each part of a name may be a single name or it may be composed of multiple names. For example, the name part ‘first name’ may be composed of a first and a middle name. The latter are referred to as name *elements* in this section. Let’s consider a simple name first: “John Edward Doe”. This name is composed of the following parts:

First	John Edward
Prefix	—
Last	Doe
Suffix	—

The spacing, punctuation and line-breaking behavior of names is controlled by six macros:

a= \bibnamedelima

Inserted
by
the
back-
end
after
the
first
el-
e-
ment
of
ev-
ery
name
part
if
that
el-
e-
ment
is
less
than
three
char-
ac-
ters
long
and
be-
fore
the
last
el-
e-
ment
of
ev-
ery
name
part.

b= \bibnamedelimb

Inserted
by
the
back-
end
be-
tween
all
el-
e-
ments

This is how the delimiters are employed:

John Edward Doe
Doe, John Edward

Initials in the **bib** file get a special delimiter:

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:

First Charles-Jean Étienne Gustave Nicolas
Prefix de
Last La Vallée Poussin
Suffix —

The delimiters:

Charles-Jean Étienne

Note that **\bibnamedelim**

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

\bibnamedelimc

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:

a= \bibinitperiod

b= \bibinitdelim

c= \bibinitthyphende

This is how they are employed:

J. E. Doe
K. - H. Mustermann

3.12.5 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 **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

defernumbers

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.

3.12.6 Active Characters in Bibliography Headings

Packages using active characters, such as **babel**,
polyglossia,
csquotes, or
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 charac-

“

ter is the Ascii quote , which is used by various language modules of the

babel/polyglossia

packages. If shorthands such as " <

"a and are used in the argument to `\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 moving

`\defbibheading`
after `\begin{document}`

Alternatively, you may use `\babel`'s

`\shorthandon` and
`\shorthando`

commands to temporarily make the shorthands active in the preamble. The above also applies to bibliography notes and the

`\defbibnote` command.

3.12.7 Grouping in Reference Sections and Segments

All LaTeX environments enclosed in `\begin`

`\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 malfunc-

tions 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 form a group, but otherwise works as usual. As far as Bibl_{at}ex 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 §§ 3.6.5 and 3.6.6 for details.

4 Author Guide

This part of the manual documents the author interface of the Bibl_{at}ex package. The author guide covers everything you need to know in order to write new citation and bibliography styles or localization 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 Bibl_{at}ex, we will have a look at some of its fundamental concepts. The Bibl_{at}ex package uses

`bbl` auxiliary files in a special way. Most notably, the `bbl` file is used differently and when using BibT_{ex} as the backend, there is only one

bst

file which implements a structured data interface rather than exporting printable data. 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**

bbl 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

\cite citation labels used by

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

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 bbl file in the final format. The bbl file is read in and processed like any printable section of the document. For example, consider the following entry in a bib file:

```
@Book{companion,
  author      = {Mi
  title       = {Th
  publisher    = {Ad
  address     = {Re
  year        = {19
}
```

With the plain.bst style, BibTeX exports this entry to the bbl file as follows:

```
\bibitem{companion
Michel Goossens,
```

`\newblock {\em Th`
`\newblock Addison`

By default, LaTeX generates numeric citation labels, hence

`\bibitem` writes lines such as the
`aux` file:

`\bibcite{companio`

Implementing a different citation style implies that more data has to be transferred via

the `aux` file. With the `natbib` pack-
`aux` age, for example, the file contains lines like this one:

`\bibcite{companio`
`and Samarin}}}`

The 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 `jurabib` package which also makes all
bibliographic data available in citations:

`\bibcite{companio`
`{}}{\{0}}{\}{book}{`
`\bibbdsep{} 199`

{Goossens}{Mich
{}}\Bibbstasep
LaTeX Companion
Addison-Wesley\

In this case, the contents of the entire thebibliogra

environment are effectively transferred via the aux file. The data is

read from the bbl file, written to the aux file, read

back from the aux file and then kept in memory. The bibliography

itself is still generated as the bbl file is read in. The Biblatex package

would also be forced to cycle all data through the 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 Biblatex takes a different approach. First of all, the document structure is slightly different. Instead of using

\bibliography

in the document body, database files are specified in the preamble with

\addbibresource,

\bibliographystyle

is omitted entirely (all features are controlled by pack-

age options), and the bibliography is printed using

```
\printbibliography
\documentclass{...}
\usepackage[...]{...}
\addbibresource{...}
\begin{document}
\cite{...}
\printbibliography
\end{document}
```

In order to streamline the whole procedure, Bibl_{at}ex essentially em-

ploy_s the `bbl` file like an `aux` file, rendering `\bibcite` obsolete. We then get the following procedure:

1. Run `latex`: The first step is similar to the traditional procedure described above:

```
\bibstyle
\bibdata
aux
```

and commands are written to the `aux` file (BibTeX back-

end) or `bcbf` file (Biber backend), along with `\citation` commands for all citations. We then wait for the backend to supply the required data. With

BibTeX as a backend, since Biblatex uses a special `bst` file which implements its data interface on the BibTeX end, the

`\bibstyle` command is always

`\bibstyle{biblatex}`

2. Run `biber` or `bibtex`:

The backend supplies those entries from the `bib` file which were

requested by the `\citation` commands in the auxiliary file. However, it does not write a printable

bibliography to the `bbl` file, but rather a structured representation of the bibliographic data. Just like an `aux` file, this

`bbl` file does not print anything when read in. It merely puts data in memory.

3. Run `latex`: Starting with the second run, the

`bbl` file is processed right at the beginning of the document

body, just like an `aux` file. From this point on, all bibliographic data is available in memory so that all citations can be printed right away.²⁸

The citation commands have access to the complete bibliographic data, not

²⁸If the `defernumbers` package option is enabled Biblatex uses an algorithm similar to the traditional procedure to generate numeric labels.

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:

```
@Book{companion,  
  author      = {Mi  
  title       = {Th  
  publisher    = {Ad  
  address      = {Re  
  year        = {19  
}
```

With Biblatex and the Biber backend, this entry is essentially exported in the following format:

```
\entry{companion}  
  \labelname{auth  
    {\uniquename=  
    {\uniquename=  
    {\uniquename=
```

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.

```

    }
    \name{author}{3
        {{uniquename=
        {{uniquename=
        {{uniquename=
    }
    \list{publisher
        {Addison-Wesl
    }
    \list{location}
        {Reading, Mas
    }
    \field{title}{T
    \field{year}{19
\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 bib-

liography 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:

```
\ProvidesFile{example}
\defbibenvironment{bibliography}
{
  . . .
}
\defbibenvironment{bibliography}
```

```

{ . . . }
{ . . . }
{ . . . }
\InitializeBibliography
\DeclareBibliography
\DeclareBibliography
\DeclareBibliography
. . .
\DeclareBibliography
\endinput

```

The main structure of a bibliography style file consists of the following commands:

```

\RequireBibliographyStyle
{ style }

```

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

```

\InitializeBibliography

$$\{ \langle code \rangle \}$$

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.

\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

$$\{ \langle alias \rangle \} \{ \langle entrytype \rangle \}$$

If a bibliography driver covers more than one entry type, this command may be used to define an alias. This command is optional. The $\langle entrytype \rangle$ argument may also be an asterisk. In this case, the alias serves as a fallback which is used if no specific driver for the entry type has been defined.

`\DeclareBibliographyC`

`{\langle key \rangle} [\langle value \rangle] {\langle code \rangle}`

This command defines additional preamble options in `\langle key \rangle = \langle value \rangle` format. The `\langle key \rangle` is the option key. The `\langle code \rangle` is arbitrary TeX code to be executed whenever the option is used. The value passed to the option is passed on to the `\langle code \rangle` as

#1. The optional `\langle value \rangle` is a default value to be used if the bare key is given without any value. This is useful for boolean switches. For example, with a definition like the following:

`\DeclareBibliogra`

giving ‘`somekey`’ without a value is equivalent

to ‘`somekey=true`’.

`\DeclareEntryOption`

`{\langle key \rangle} [\langle value \rangle] {\langle code \rangle}`

Similar to `\DeclareBibliogr`

but defines options which are settable on a per-entry basis in the

`options`

field from § 2.2.3. The `\langle code \rangle` is executed whenever Bibl_{at}ex 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`
`\printbibliography`
`bibliography`.

By default,

uses the environment

Here is a definition suitable for a bibliography style which does not print any labels in the bibliography:

```
\defbibenvironment
{
  \list
  {}
  {\setlength{
    \setlength{
    \setlength{
    \setlength{
  {\endlist}
  {\item}
```

This definition employs a

environment with hanging

indentation, using the

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` biblio-
graphy styles use a definition similar to this example.

```
\defbibenviromen  
{\list  
  {\printfield  
  {\setlength{  
    \setlength{  
    \setlength{  
    \addtolengt  
    \setlength{  
    \setlength{  
    \renewcomma
```

`{\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

`numeric` §§ 4.10.4 and 4.10.5 for details. The bibliography style uses the definition given above. The

`alphabetic` style is similar,

except that `labelnumber`

is replaced by `labelalpha` and

`labelnumberwidth`
`labelalphawidth`.
by

Bibliography lists are handled in a similar way.

`\printbiblist`

uses the environment named after the bibliography list by default (when using BibTeX,

`\printshorthands`
`shorthand`

always uses the environment). 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.

```

\defbibenviromen
  {\list
    {\printfield
      {\setlength{
        \setlength{
          \setlength{
            \addtolengt
              \setlength{
                \setlength{
                  \renewcomma
                    {\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 Biblatex's data types, see § 2.2.

```
\DeclareBibliogra
  \printnames{aut
  \newunit\newblo
  \printfield{tit
  \newunit\newblo
  \printlist{publ
  \newunit
  \printlist{loca
  \newunit
  \printfield{yea
  \finentry}
```

The standard bibliography styles employ two bibliography

macros **begentry** and **finentry**:

```
\DeclareBibliogra
```

`\usebibmacro{be}`
`\usebibmacro{fi`

with the default definitions

`\newbibmacro*{beg`
`\newbibmacro*{fin`

Use of these macros is recommended for easy hooks into the beginning and end of the driver.

Returning to the driver for the **book** entrytype above, there is still one piece missing: the formatting directives used

by `\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

`\emph` command:

`\DeclareFieldForm`

List formats are slightly more complex. After splitting up the list into individual items, 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.

```
\DeclareListForma
#1%
\ifthenelse{\va
{\addcomma\sp
{}}}
```

Formatting directives for names are similar to those for literal lists, but the individual items in the list are names which are automatically dissected into their components. The list formatting directive is executed once for each name in the list and the components of the name are passed to the formatting directive as separate arguments.

For example, **#1** is the last name and **#3** is the first name. Here is a simplified example:

```
\DeclareNameForma
\ifthenelse{\va
{#1%
\ifblank{#3}
{\ifblank{#3}
#1}%
\ifthenelse{\va
{\addcomma\sp
```


{ } }

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 component of a name is not available, the corresponding argument will be blank. As with directives for literal lists, the separator to be inserted between the individual items in the 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 Biblatex to pass data to bibliography drivers

and citation commands. They are not used in **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 **bib** file.

4.2.4.1 Generic Fields

entrykey field (string)

The entry key of an item in the **bib** file. This is the string used by

Biblatex and the backend to identify an entry in the **bib** file.

childentrykey 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

entrykey field holds the entry key of the parent. The entry key of the child entry being cited is provided in the

childentrykey field. This field is only available when citing a subentry of an entry set.

Holds the prefix coming before ‘date’ of the date field name chosen by

`\DeclareLabeldate`
`eventdate`

For example, if the label date field is

`date-labels source`

then

`\DeclareLabeldate`
`date`

will be ‘event’. In case

selects the date field, then

`date-labels source`

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

`date-labels source`

can be used in constructing references to the field which

`\DeclareLabeldate`
`\DeclareLabeldate`

chooses. Since

can also select literal strings for fallbacks, if

`date-labels source`

`labeldate`

is undefined, then either the

`\DeclareLabeldate`

package option is set to false or

chose a literal string instead of a date field. Bear in mind that

`\DeclareLabeldate`

can also be used to select non-date fields as a fallback and so

`date-labels source`

might contain a field name. So, in summary, the rules are

```

\iffielddundef{dat
{
% labeldate p
}
}
\iffielddundef
{
% datelab
}
}
% datelab
}
}

```

`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

`@setentrytype` entry to citation commands. This implies that the `childentrytype` 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)

Biber only

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 com-

parison with `\iffieldequals`

`namehash` later (see §§ 4.6.1 and 4.6.2). The

is derived from the truncated **labelname**
list, i. e., it is responsive to **maxnames**
and **minnames**. See also
hash and **fullhash**.

fullhash field (string)

A hash string which uniquely identifies the
labelname list. This field
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**.

pageref list (literal)

backref
If the 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. With BibTeX, this field is also used internally instead of
sortinit **hash**.

`sortinit` hash field (string)

Biber only

With Biber, 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 § 3.9.3).

`clonesourcekey` field (string)

Biber only

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)

When using BibTeX as the backend, a label similar to the labels gener-

ated by the `alpha.bst` style

of traditional BibTeX. This default label consists of initials drawn from the

`labelname` list plus the last two dig-

its 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. With Biber, users can specify a template used to construct the alphabetic label (see § 4.5.4) and the default template mirrors

Biber only

the format mentioned for bibtex above. A complete ‘alphabetic’ label consists

of the fields

plus

Note

that the

and

fields need to be requested with the package option

(§ 3.1.2.3). See

also as well as

`\labelalpha`
`extraalpha`
`\labelalphaothers`

in § 3.9.1.

`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

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 `\labelalphaother`
`extra*`

in § 3.9.1. Table 5 summarises the various
 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`
`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 re-
 sponsive to the `useauthor`,
`useeditor`, and
`usetranslator`
 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. With Biber, this field may be customized. See § 4.5.9 for details.

Biber only

`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

`labelnumber`, option see § 3.1.2.3 for details. Also see the package option

`defernumbers` in § 3.1.2.1.

`prefixnumber` field (literal)

If the `prefixnumbers`

option of `\printbibliography`

has been set in order to prefix all entries in a subbibliography with a fixed string, this string is available in the

`prefixnumber`

field of all affected entries. If no prefix has been set, the

`prefixnumber`

field of the respective entry is undefined. See the

`prefixnumbers`

option of `\printbibliography`

`shorthand`

in § 3.6.2 for details. If the field is defined, Biblatex does not assign the prefix to the

prefixnumber

field of the respective entry. In this case, the

prefixnumber

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

Biber only

ing

may be customized. See § 4.5.9 for details. Note that

extratitle

the 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**,
extratitle
 the
 field holds an integer which may be converted to a letter with
\mknumalph 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

labeltitle
 with the
 field). This field is undefined if there is only one work with the same
labeltitle by the same
labelname in the bibliography.

extratitle
 Note that the
 field needs to be requested with the package option
labeltitle, see § 3.1.2.3 for
labeltitle.
 details. See also

extra*
 Table 5 summarises the various
 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**
labelyear, the
 and with the same

extratitleyear
 field holds an integer which may be converted to a letter with

`\mknumalph` 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 conjunc-

tion 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`

`extra*` disambigua-

Table 5 summarises the various tion counters and what they track.

`labelyear` field (literal)

The publication year, as specified in the `date` 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

\labeldate, see § 3.1.2.3 for details. See also **extrayear**. With

Biber, the source date field for this may be customized. See § 4.5.9 for details.

Biber only

labelmonth field (datepart)

The publication month, as specified in the **date** or the **month** field, for use in author-year labels. Note

that the **labelmonth** field needs to be requested with the package option

\labeldate, see § 3.1.2.3 for details. With Biber, the source date field for this may be customized. See § 4.5.9 for details.

Biber only

labelday field (datepart)

The publication day, as specified in the **date**, for use in author-year labels. Note that the

labelday field needs to be requested with the package option

\labeldate, see § 3.1.2.3 for details. With Biber, the source date field for this may be customized. See § 4.5.9 for details.

Biber only

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 for-

matted 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 con-

sists of the fields **labelyear**

plus **extrayear**. Note

that the **labelyear**

and **extrayear**

fields need to be requested with the package option

labeldate, see § 3.1.2.3 for de-

tails. See also **labelyear**. Table 5

extra* summarises the various disambiguation counters and what they track.

4.2.4.3 Date Component Fields

See table 8 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:

```
\iffielddundef{ori
```

This will tell you if the corresponding date is defined at all. This test:

```
\iffielddundef{ori
```

will tell you if the corresponding date is defined and a (fully specified) range. This test:

\iffieldequalstr{

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 100 for further examples.

day field (datepart)

date

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.

month field (datepart)

month

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.

year field (datepart)

year

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.

endday field (datepart)

date

If the date specification in the **date** field is a range, this field holds the day component of the end date.

endmonth field (datepart)

date

If the date specification in the **date** field is a range, this field holds the month component of the end date.

Field	Value (Example)	Data Interface
date	1988	<div>day</div> <div>month</div> <div>year</div> <div>endday</div> <div>endmonth</div> <div>endyear</div>
date	1997/	<div>day</div> <div>month</div> <div>year</div> <div>endday</div> <div>endmonth</div> <div>endyear</div>
urldate	2009-01-31	<div>urlday</div> <div>urlmonth</div> <div>urlyear</div> <div>urlendda</div>

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.

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.

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 range.

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.

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` com-

ponent indicates an open ended range.

`urlday` field (datepart)

This field holds the day component of the `urldate` field.

`urldmonth` 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.

`urlendday` field (datepart)

If the date specification in the `urldate` field is a range, this field holds the day component of the end date.

`urlendmonth` field (datepart)

If the date specification in the `urldate` field is a range, this field holds the month component of the end date.

`urlendyear` 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 de-

`url`endyear component
`url`date 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 § 3.7.5.

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{exa
\DeclareCiteComma
\DeclareCiteComma
\DeclareCiteComma
\DeclareCiteComma
\endinput
  
```

`\RequireCitationStyle`

`{
 $\langle style \rangle$
}`

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.`

`\InitializeCitationSt`

`{
 $\langle code \rangle$
}`

Specifies arbitrary $\langle code \rangle$ 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

§ 3.7.8 is used. The `\citereset`

command also resets the internal citation trackers of this package. The reset will

affect the `\ifciteseen`,

`\ifentryseen`,

`\ifciteibid`,

`\ifciteidem`

and 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

{
 $\langle code \rangle$
}

Specifies arbitrary $\langle code \rangle$ required for a partial reset of the citation style. This hook

will be executed every time the

command from § 3.7.8 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

command also resets the internal ‘ibidem’ and ‘idem’ trackers of this package. The reset will

affect the

and

discussed in § 4.6.2.

\mancite

\mancite

\ifciteibid

\ifciteidem

tests

\DeclareCiteCommand

{
 $\langle command \rangle$ } [
 $\langle wrapper \rangle$] {
 $\langle precode \rangle$ } {
 $\langle loopcode \rangle$ } {
 $\langle sepcode \rangle$ }

\DeclareCiteCommand*

{
 $\langle command \rangle$ } [
 $\langle wrapper \rangle$] {
 $\langle precode \rangle$ } {
 $\langle loopcode \rangle$ } {
 $\langle sepcode \rangle$ }

This is the core command used to define all citation commands. It takes one optional and five mandatory arguments. The $\langle command \rangle$ is the command to be defined, for ex-

ample

. If the optional $\langle wrapper \rangle$ argument is given, the entire citation will be passed to the $\langle wrapper \rangle$ as an argument, i. e., the wrapper command must take one mandatory argument.²⁹ The $\langle precode \rangle$ is arbitrary code to

\cite

²⁹Typical examples of wrapper commands are

and

\mkbibparens

\mkbibfootnote

be executed at the beginning of the citation. It will typically handle the $\langle prenote \rangle$

argument which is available in the **prenote** field. It may also be used to initialize macros required by the $\langle loopcode \rangle$. The $\langle loopcode \rangle$ is arbitrary code to be executed for each entry key passed to the $\langle command \rangle$. This is the core code which prints the citation labels or any other data. The $\langle sepcode \rangle$ is arbitrary code to be executed after each iteration of the $\langle loopcode \rangle$. It will only be executed if a list of entry keys is passed to the $\langle command \rangle$. The $\langle sepcode \rangle$ will usually insert some kind of separator, such as a comma or a semicolon. The $\langle postcode \rangle$ is arbitrary code to be executed at the end of the citation. The $\langle postcode \rangle$ will typically handle the $\langle postnote \rangle$ argument which is available in

the **postnote** field.³⁰ The starred variant of

\DeclareCiteCommand defines a starred $\langle command \rangle$. For example,

\DeclareCiteCommand
\cite*
would define ³¹

\DeclareMultiCiteComm
{ $\langle command \rangle$ **}** **[** $\langle wrapper \rangle$ **]** **{** $\langle cite \rangle$ **}** **{** $\langle delimiter \rangle$ **}**

This command defines ‘multicite’ commands (§ 3.7.3). The $\langle command \rangle$ is the multicite command to be defined, for example

\cites. It is automatically made robust. Multicite commands are built on top of backend commands defined with

³⁰The bibliographic data available to the $\langle loopcode \rangle$ is the data of the entry currently being processed. In addition to that, the data of the first entry is available to the $\langle precode \rangle$ and the data of the last one is available to the $\langle postcode \rangle$. ‘First’ and ‘last’ refer to the order in which the citations are printed. If

sortcites the package option is active, this is the order of the list after sorting. Note that no bibliographic data is available to the $\langle sepcode \rangle$.

³¹Note that the regular variant of **\DeclareCiteComm** defines a starred version of the $\langle command \rangle$ 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.

\DeclareCiteCommand

and the $\langle cite \rangle$ argument specifies the name of the backend command to be used. Note that the wrapper of the backend command (i. e., the $\langle wrapper \rangle$ argument passed to

\DeclareCiteCommand

is ignored. Use the optional $\langle wrapper \rangle$ argument to specify an alternative wrapper. The $\langle delimiter \rangle$ 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:

```
\DeclareMultiCite
{\cite}{\
```

```
\DeclareMultiCite
{\parenci
```

```
\DeclareMultiCite
{\footcit
```

\DeclareAutoCiteCommand

```
{\langle name \rangle} [\langle position \rangle] {\langle cite \rangle} {\langle multicite \rangle}
```

\autocite

This command provides definitions for the

and commands from § 3.7.4. The definitions are enabled with the

`\autocite` package option from § 3.1.2.1. The $\langle name \rangle$ is an identifier which serves as the value passed to the package option. The autocite commands are built on top of backend com-

mands like `\parencite`

and `\parencites`.

The arguments $\langle cite \rangle$ and $\langle multicite \rangle$ specify the backend commands to use. The $\langle cite \rangle$ argument refers to

`\autocite` and $\langle multicite \rangle$ refers

to `\autocites`. The $\langle position \rangle$ argument controls the handling of any punctuation marks after the cita-

tion. Possible values are `l`, `r`, `f`, `r`. means that the punctuation is placed to the right of the citation, i.e., it will not be moved

around. `l` means that any punctuation after the citation is moved to

the left of the citation. `f` is like `r` in a footnote and like `l`

otherwise. This argument is optional and defaults to `r`. See also

`\DeclareAutoPunct`

`autopunct` package

option in § 3.1.2.1. The following examples are real definitions taken from

`biblatex.def`:

`\DeclareAutoCiteC`

`\DeclareAutoCiteC`
`\DeclareAutoCiteC`
`\DeclareAutoCiteC`

A definition provided in the document preamble can be subsequently adopted with the following: (see § 3.2.2).

`\ExecuteBibliogra`

4.3.2 *Special Fields*

The following fields are used by Biblalex 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.

prenote field (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.

postnote field (literal)

The *⟨postnote⟩* argument passed to a citation command. This field is specific to citations and not available in the bibliography. If the *⟨postnote⟩* argument is missing or empty, this field is undefined.

multiprenote field (literal)

The *⟨multiprenote⟩* argument passed to a multicite command. This field is specific to citations and not available in the bibliography. If the *⟨multiprenote⟩* argument is missing or empty, this field is undefined.

multipostnote field (literal)

The *⟨multipostnote⟩* argument passed to a multicite command. This field is specific to citations and not available in the bibliography. If the *⟨multipostnote⟩* argument is missing or empty, this field is undefined.

postpunct field (punctuation command)

The trailing punctuation argument implicitly passed to a citation command. This field is specific to citations and not available in the bibliography. If the character following a given citation command is not specified in

`\DeclareAutoPunct`

(§ 4.7.5), this field is undefined.

4.4 Data Interface

The data interface are the facilities used to format and print all bibliographic data. These facilities are available in both bibliography and citation styles.

4.4.1 Data Commands

This section introduces the main data interface of the Biblatex package. These are the commands doing most of the work, i. e., they actually print the data provided in lists and fields.

`\printfield[<format>]{<field>}`

This command prints a *<field>* using the formatting directive *<format>*, as defined with

`\DeclareFieldForm`

If a type-specific *<format>* has been declared, the type-specific formatting directive takes precedence over the generic one. If the *<field>* is undefined, nothing is printed. If the *<format>* is omitted,

`\printfield`

tries using the name of the field as a format name. For example, if the

`title`

field is to be printed and the *<format>* is not specified, it will try to use the field format `title`.³²

³²In other words,

`\printfield{title}` is equivalent to `\printfield[title]{title}`.

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. Note

that **\printfield**

provides the name of the field currently being processed in

\currentfield

for use in field formatting directives.

\printlist[*<format>*] [*<start>*–*<stop>*] { *<literal list>* }

This command loops over all items in a *<literal list>*, starting at item number *<start>* and stopping at item number *<stop>*, including *<start>* and *<stop>* (all lists are numbered starting at 1). Each item is printed using the formatting directive *<format>*, as defined with

\DeclareListFormat

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, nothing is printed. If the *<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 *<start>* argument defaults to 1; *<stop>* defaults to the total number of items in the list. If the total number is greater than *<maxitems>*, *<stop>* defaults to *<minitems>* (see § 3.1.2.1). See

\printnames for further

details. Note that **\printlist**

provides the name of the literal list currently being processed in

\currentlist

for use in list formatting directives.

`\printnames [<format>] [<start> - <stop>] { <name list>`

This command loops over all items in a *<name list>*, starting at item number *<start>* and stopping at item number *<stop>*, including *<start>* and *<stop>* (all lists are numbered starting at 1). Each item is printed using the formatting directive *<format>*, as defined with

`\DeclareNameFormat`

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, nothing is printed. If the *<format>* is omitted,

`\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

default

are undefined, it falls back to as a last resort. The *<start>* argument defaults to 1; *<stop>* defaults to the total number of items in the list. If the total number is greater than *<maxnames>*, *<stop>* defaults to *<minnames>* (see § 3.1.2.1). 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 *<start>* and *<stop>* may be omitted, hence the following arguments are all valid:

```
\printnames [ . . . ] [
\printnames [ . . . ] [
\printnames [ . . . ] [
```

If you want to override *<maxnames>* and *<minnames>* and force printing of the entire

listtotal

list, you may refer to the counter in the second optional argument:

`\printnames[...][`

Whenever `\printnames`

and `\printlist` process
a list, information concerning the current state is accessible by way of

`listtotal`
four counters: the
counter holds the total number of items in the current

`listcount`
list,
holds the number of the item currently being processed,

`liststart` is the *⟨start⟩* argument

passed to `\printnames`

or `\printlist`,

`liststop` is the *⟨stop⟩* argu-

ment. These counters are intended for use in list formatting di-

`listtotal`
rectives.

may also be used in the second optional argument to

`\printnames` and

`\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 `author` counter holds the total

author list. These counters are similar to `listtotal` except that they may also be used independently of list formatting directives. There are also `maxnames` and `minnames` as well as `maxitems` and `minitems` counters which hold the values of the corresponding package options. See § 4.10.5 for a complete list of such internal counters. Note that

`\printnames` provides the name of the name list currently being processed in `\currentname` for use in name formatting directives.

`\printtext[<format>]{<text>}`

This command prints *<text>*, which may be printable text or arbitrary code generating printable text. It clears the punctuation buffer before inserting *<text>* and informs Biblatex that printable text has been inserted. This ensures that all

preceding and following `\newblock` and `\newunit` commands have the desired effect. `\printfield` and `\printnames` as well as `\bibstring` and its compan-

ion 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 $\langle format \rangle$ argument specifies a field formatting directive to be used to format $\langle text \rangle$. 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 $\langle format \rangle$ has been declared, the type-specific formatting directive takes precedence over the generic one. If the $\langle format \rangle$ is omitted, the $\langle text \rangle$ is printed as is. See also § 4.11.7 for some practical hints.

$\backslash printfile[\langle format \rangle] \{ \langle file \rangle \}$

$\backslash printtext$

This command is similar to

except that the second argument is a file name rather than literal text. The $\langle file \rangle$ argument must be the name of a valid LaTeX file found in TeX's search

path. $\backslash printfile$ will use

$\backslash input$ to load this $\langle file \rangle$. If there is no such file,

$\backslash printfile$ does nothing. The

optional $\langle format \rangle$ argument specifies a field formatting directive to be applied to the $\langle file \rangle$. If a type-specific $\langle format \rangle$ has been declared, the type-specific formatting directive takes precedence over the generic one. If the $\langle format \rangle$ is omitted, the $\langle file \rangle$ is printed as is. Note that this feature needs to be enabled explicitly by setting the pack-

age option $\backslash loadfiles$ from § 3.1.2.1.

By default, $\backslash printfile$ will not input any files.

$\backslash printdate$ This command prints the

date of the entry, as specified in the fields $\backslash date$ or

$\backslash month$, $\backslash year$. 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.

`\printdatelabel` Simi-

lar to `\printdate` but prints the

`\DeclareLabel`

date field determined by The date format is controlled by the package option `date` from § 3.1.2.1. Additional formatting may be applied by adjusting the field format

`date` label (§ 4.10.4).

`\printdateextralabel`

Similar to `\printdate` label

but incorporates the **extrayear** field in the date specification. This is useful for bibliography styles designed for author-year citations.

\printurldate Similar to

\printdate but prints the

urldate of the entry. The date format is

urldate controlled by the package option from § 3.1.2.1. Additional formatting may be applied by adjusting the field format

urldate (§ 4.10.4).

\printorigdate Similar to

\printdate but prints the

origdate of the entry. The date format is

origdate controlled by the package option from § 3.1.2.1. Additional formatting may be applied by adjusting the field format

origdate (§ 4.10.4).

\printeventdate

Similar to **\printdate**

but prints the **eventdate**

of the entry. The date format is controlled by the package option

eventdate from § 3.1.2.1.
Additional formatting may be applied by adjusting the field format

eventdate (§ 4.10.4).

\indexfield[*format*]{*field*}

\printfield

This command is similar to

except that the *field* is not printed but added to the index using the formatting directive *format*, as defined with

\DeclareIndexField

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

\indexfield
omitted,
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

default
formats are undefined, it falls back to
as a last resort.

\indexlist[*format*][*start*–*stop*]{*literal list*}

\printlist

This command is similar to

except that the items in the list are not printed but added to the index using the formatting directive *format*, as defined with

\DeclareIndexList

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*

This command is similar to **\printnames**

except that the items in the list are not printed but added to the index using the formatting directive *format*, as defined with

\DeclareIndexName

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.

\entrydata {*key*} {*code*}

\entrydata* {*key*} {*code*}

Data commands like **\printfield**

normally use the data of the entry currently being processed. You may use

\entrydata to switch contexts

locally. The *key* is the entry key of the entry to use locally. The *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 re-

source names prefixed with the string ‘`saved`’.

This is useful when comparing two data sets. For example, inside the

`\code` argument, the `author` field holds the author of entry `<key>` and the author of the enclosing entry is avail-

able as `savedauthor`.

The `author` counter holds the number of

names in the `author` field of `<key>`; the

`savedauthor` counter

refers to the author count of the enclosing entry.

`\entryset{<precode>}{<postcode>}`

This command is intended for use in bibliography drivers handling

`@set` entries. It will loop over all members of the set, as

indicated by the `entryset` field, and

execute the appropriate driver for the respective set member. This is similar

to executing the `\usedriver`

command from § 4.6.4 for each set member. The `<precode>` is arbitrary code to be executed prior to processing each item in the set. The `<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.

`\DeclareFieldInputHar`

`{\langle field \rangle}{\langle code \rangle}`

This command can be used to define a data input handler for `\langle field \rangle` when

it is read from the `.bbl`. Within the `\langle code \rangle`, the macro

`\NewValue` contains the value of the

field. For example, to ignore the field when it appears, you could do

`\DeclareFieldInput`

`\DeclareSource`

Generally, you would want to use

(see § 4.5.2) 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 `\langle code \rangle` can be arbitrary TeX.

`\DeclareListInputHand`

`{\langle list \rangle}{\langle code \rangle}`

`\DeclareFieldInput`

As but for lists. Within the `\langle code \rangle`, the macro

`\NewValue` contains the value of

the list and `\NewCount` contains the number of items in the list.

`\DeclareNameInputHand`

`{\langle name \rangle}{\langle code \rangle}`

`\DeclareFieldInput`

As

but

for names.

Within

the

`\langle code \rangle`,

the

macro

`\NewValue`

contains the value

`\NewCount`

of the name,

contains the number of individual names in the name and

`\NewOption`

contains any

`.bbl`

per-name options passed in the

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`

`[\langle entrytype, ... \rangle]{\langle format \rangle}{\langle code \rangle}`

`\DeclareFieldFormat`

`*{\langle format \rangle}{\langle code \rangle}`

Defines the field format `\langle format \rangle`. This formatting directive is arbitrary `\langle code \rangle` to be

executed by `\printfield`.

The value of the field will be passed to the `\langle code \rangle` as its first and only argument.

The name of the field currently being processed is available to the $\langle code \rangle$ as

$\backslash currentfield$.

If an $\langle entrytype \rangle$ is specified, the format is specific to that type. The $\langle entrytype \rangle$ 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.

$\backslash DeclareListFormat$

$[\langle entrytype, ... \rangle] \{ \langle format \rangle \} \{ \langle code \rangle \}$

$\backslash DeclareListFormat$

$* \{ \langle format \rangle \} \{ \langle code \rangle \}$

Defines the literal list format $\langle format \rangle$. This formatting directive is arbitrary $\langle code \rangle$ to be executed for every item in a list processed by

$\backslash printlist$.

The current item will be passed to the $\langle code \rangle$ as its first and only argument. The name of the literal list currently being processed is available to the $\langle code \rangle$ as

$\backslash currentlist$.

If an $\langle entrytype \rangle$ is specified, the format is specific to that type. The $\langle entrytype \rangle$ 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

$listcount$

is

smaller than or equal to

$liststop$.

The starred variant of this command is similar to the regular version, except that all type-specific formats are cleared.

\DeclareNameFormat

[$\langle entrytype, \dots \rangle$] { $\langle format \rangle$ } { $\langle code \rangle$ }

\DeclareNameFormat

* { $\langle format \rangle$ } { $\langle code \rangle$ }

Defines the name list format $\langle format \rangle$. This formatting directive is arbitrary $\langle code \rangle$ to be executed for every name in a list processed by

\printnames

. If an $\langle entrytype \rangle$ is specified, the format is specific to that type. The $\langle entrytype \rangle$ argument may be a comma-separated list of values. The individual parts of a name will be passed to the $\langle code \rangle$ as separate arguments. These arguments are as follows:

#1

The last names. If a name consists of a single part only (for example, ‘Aristotle’), this part will be treated as the last name.

#2

The last names, given as initials.

#3

The first names. This argument also includes all middle names.

#4

The first names, given as initials.

#5

The 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.

#6

The name prefixes, given as initials.

#7

The name affixes, for example ‘junior’, ‘senior’, ‘der Jüngere’, ‘der Ältere’, etc. Note that name affixes are referred to as the ‘junior part’ of the name in the BibTeX file format documentation.

#8

The name affixes, given as initials.

If a certain part of a name is not available, the corresponding argument will be

empty, hence you may use `\ifblank` tests to check for the individual parts of a name. The name of the name list currently being processed is available to the `\currentname` 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 § 3.12.4. The starred

variant of this command is similar to the regular version, except that all type-specific formats are cleared.

`\DeclareIndexFieldFormat`

`[$\langle entrytype, \dots \rangle$]{ $\langle format \rangle$ }{ $\langle code \rangle$ }`

`\DeclareIndexFieldFormat*`

`{ $\langle format \rangle$ }{ $\langle code \rangle$ }`

Defines the field format $\langle format \rangle$. This formatting directive is arbitrary $\langle code \rangle$ to be

executed by `\indexfield`.

The value of the field will be passed to the $\langle code \rangle$ as its first and only argument.

The name of the field currently being processed is available to the $\langle code \rangle$ as

`\currentfield`.

If an $\langle entrytype \rangle$ is specified, the format is specific to that type. The $\langle entrytype \rangle$ argument may be a comma-separated list of values. This command is similar to

`\DeclareFieldFormat`

except that the data handled by the $\langle code \rangle$ is not intended to be printed but written to the index. Note that

`\indexfield` will execute
`\index`

the `<code>` as is, i. e., the `<code>` must include
 or a similar command. The starred variant of this command is similar to the regular
 version, except that all type-specific formats are cleared.

`\DeclareIndexListForm`

`[<entrytype, ...>]{<format>}{<code>}`

`\DeclareIndexListForm`

`*{<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 cur-
 rent 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

`\DeclareListForma`

except that the data handled by the `<code>` is not in-
 tended to be printed but written to the index. Note that

`\indexlist` will execute
`\index`

the `<code>` as is, i. e., the `<code>` must include
 or a similar command. The starred variant of this command is similar to the regular
 version, except that all type-specific formats are cleared.

`\DeclareIndexNameForm`

`[<entrytype, ...>] { <format> } { <code> }`

`\DeclareIndexNameForm`

`* { <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

`\indexnames`. The name of the name list currently being processed is available to the `<code>` as

`\currentname`. If an `<entrytype>` is specified, the format is specific to that type. The `<entrytype>` argument may be a comma-separated list of values. The parts of the name will be passed to the `<code>` as separate arguments. This command is very similar to

`\DeclareNameForm`

except that the data handled by the `<code>` is not intended to be printed but written to the index. Note that

`\indexnames` will execute `\index`

the `<code>` as is, i. e., the `<code>` must include or a similar command. The starred variant of this command is similar to the regular version, except that all type-specific formats are cleared.

`\DeclareFieldAlias`

`[<entry type>] { <alias> } [<format entry type>] { <format> }`

Declares `<alias>` to be an alias for the field format `<format>`. If an `<entrytype>` is specified, the alias is specific to that type. The `<format entry type>` is the entry type of the backend format. This is only required when declaring an alias for a type-specific formatting directive.

\DeclareListAlias

[*entry type*] { *alias* } [*format entry type*] { *format* }

Declares *alias* to be an alias for the literal list format *format*. If an *entrytype* is specified, the alias is specific to that type. The *format entry type* is the entry type of the backend format. This is only required when declaring an alias for a type-specific formatting directive.

\DeclareNameAlias

[*entry type*] { *alias* } [*format entry type*] { *format* }

Declares *alias* to be an alias for the name list format *format*. If an *entrytype* is specified, the alias is specific to that type. The *format entry type* is the entry type of the backend format. This is only required when declaring an alias for a type-specific formatting directive.

\DeclareIndexFieldAlias

[*entry type*] { *alias* } [*format entry type*] { *format* }

Declares *alias* to be an alias for the field format *format*. If an *entrytype* is specified, the alias is specific to that type. The *format entry type* is the entry type of the backend format. This is only required when declaring an alias for a type-specific formatting directive.

\DeclareIndexListAlias

[*entry type*] { *alias* } [*format entry type*] { *format* }

Declares *alias* to be an alias for the literal list format *format*. If an *entrytype* is specified, the alias is specific to that type. The *format entry type* is the entry type of the backend format. This is only required when declaring an alias for a type-specific formatting directive.

\DeclareIndexNameAlias

[$\langle entry type \rangle$] { $\langle alias \rangle$ } [$\langle format entry type \rangle$] { $\langle format \rangle$ }

Declares $\langle alias \rangle$ to be an alias for the name list format $\langle format \rangle$. If an $\langle entrytype \rangle$ is specified, the alias is specific to that type. The $\langle format entry type \rangle$ is the entry type of the backend format. This is only required when declaring an alias for a type-specific formatting directive.

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, localization strings to prefix the related data
- Macros to extract and print the related data
- Formats to format the localization 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 localization string identifier, then the resulting localized string is printed before the data from the related entries. Secondly, if there is a macro called

`related:` $\langle relatedtype \rangle$, 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 localized string and related entry data. If there is no related type

related format is used.

relatedstring

If an entry contains this field, then if value of the field resolves to a localization string identifier, the localization key value specified is printed before data from the related entries. If the field does not specify a localization 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 op-

related from § 3.1.2.1. The related information entry data from the related entries is included via a

\usebibmacro{rela

call. Standard styles call this macro towards the end of each driver. Style authors should ensure the existence of (or take note of existing) localization strings which are useful as values for the

relatedtype field, such as

translationof
translatedas.

or perhaps

A plural variant can be identified with the localization key `<relatedtype>`. 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>`.

`biblatex`.

The file `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 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 mappings can be defined at different "levels" which are applied in a defined order. See the Biblatex manual regarding these macros:

`user` -level maps defined with
`\DeclareSourcemap`
`user` -level maps defined in the Biber config file (see Biber

docs)→

style

-level

maps

defined

with

`\DeclareStyleSource`

`driver`

-level

maps

defined

with

`\DeclareDriverSource`

`\DeclareSourcemap`

`{
 $\langle specification \rangle$
}`

Biber only

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

`\maps`

directives

The $\langle specification \rangle$ is an unlimited list of which specify containers for mappings rules applying to a particular data source type (§ 3.6.1). Spaces, tabs, and line endings may be used freely to visually arrange the $\langle specification \rangle$. 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{ *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.

datatype=*bibtex*,
biblatex,
ris,
zoterord,
endnotex

default: **bibtex**

Data source type to which the contained **\map** directives apply (§ 3.6.1).

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{ *<restrictions,steps>* }

A container for an ordered set of **\step**s, 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.

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 spec-

ified, the default is the parent **\maps** element option

value. The short form **overwrite**

is equivalent to **overwrite=**
true.

\perdatasource
{ *<datasource>* }

Restricts all **\step**s 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`
`\map`
restrictions are allowed within a element.

`\pertype{<entrytype>}`

`\step` `\map`
Restricts all s in this
element to entries of the named *<entrytype>*. Multiple

`\pertype`
`\map`
restrictions are allowed within a element.

`\step`

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 options:

`typesource=<entrytype>`
`typetarget=<entrytype>`
`fieldsource=<entryfield>`
`fieldtarget=<entryfield>`
`match=<regexp>`

notmatch=⟨regexp⟩

replace=⟨regexp⟩

fieldset=⟨entryfield⟩

fieldvalue=⟨string⟩

append=true,
false

default: false

final=true, false
default: false

null=true, false
default: false

origfield=true,
false

default: false

origfieldval=true
false

default: false

`origentrytype`
`=true`, default: `false`
`false`

For all boolean `\step` options, the
short form `option` is equivalent to
`option = true`. The

following rules for a mapping step apply:

• If `entrynull` is set, processing
`\map` of the 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.

• Change the `typesource`
`typetarget` to the
`final`
`<entrytype>`, if defined. If `is`

`true` then if the `<entrytype>` of the entry is not
`typesource`, process-
`\map` ing of the parent immediately terminates.

• Change the `fieldsource`

⟨entryfield⟩ to fieldtarget,
final

if defined. If
true
is then if there is no
fieldsource
map

⟨entryfield⟩ in the entry, processing of the parent
immediately terminates.

match
•If is defined but
replace is not, only apply the step

fieldsource
if the match
⟨entryfield⟩ matches the regular expression

notmatch
(logic is reversed if you use
instead)³³. You may use capture parenthesis as usual and refer to these (\$1...\$9)

fieldvalue
in later speci-
fications. 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
match
⟨entryfield⟩ if and
replace are defined.

³³Regular expressions are full Perl 5.16 regular expressions. This means you may need to deal with special characters, see examples.

fieldset is defined, then its value is $\langle entryfield \rangle$ which will be set to a value specified by further options.

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 $\langle entryfield \rangle$. The value to set is specified by a mandatory one and only one of the following options:

◦ **fieldvalue** — The

fieldset $\langle entryfield \rangle$ is set to the

fieldvalue $\langle string \rangle$

◦ **null fieldset** — The $\langle entryfield \rangle$ is ignored, as if it did not exist in the datasource

◦ **origentrytype**

— The **fieldset** $\langle entryfield \rangle$ is set to the most recently mentioned

typesource $\langle entrytype \rangle$ name

◦ **origfield**

— The **fieldset** $\langle entryfield \rangle$ is set to the most recently mentioned

fieldsource

<entryfield> name

origfieldval

fieldset

— The *<entryfield>* is set to the most recently mentioned

fieldsource

value

With BibTeX and RIS datasources, you may specify the

pseudo-field entrykey for

fieldsource

which is the citation key of the entry. Naturally, this ‘field’ cannot

be changed (used as fieldset,

fieldtarget or changed

using replace).

\DeclareStyleSourcemap

{ *<specification>* }

Biber only

This command sets the source mappings used by a style. Such mappings are conceptually separate from user mappings defined with

\DeclareSourcemap

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

`\DeclareDriverSou`

This command is for use in style files and can only be used once—subsequent uses will overwrite earlier definitions.

`\DeclareDriverSource`

`[<datatype=driver>]{<specification>}`

Biber only

This command sets the driver default source mappings for the specified *<driver>*. Such mappings are conceptually separate from user mappings defined with

`\DeclareSourcemap`

`\DeclareStyle`

and style mapping defined with

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

`\DeclareSourcemap`

`\DeclareStyleS`

and style mappings (

These defaults are described in Appendix § A. The *<specification>* is identical to that for

`\DeclareSourcemap`

`\maps`

but without the

elements: the

`\map`

<specification> is just a list of

elements since each

`\DeclareDriverSou`

only applies to one datatype driver. See the default definitions in Appendix § A for examples.

Here are some data source mapping examples:

```

\DeclareSourcemap
  \maps[datatype=
    \map{
      \perdatasou
      \perdatasou
      \step[field
      \step[field
      \step[field
    }
  }
}

```

This would add a **keywords**
note
 field with value 'keyw1, keyw2' and set the
 field to the entry key to all entries which are found in either the
examples1.bib

examples2.bib
or
files.

```
\DeclareSourcemap
  \maps[datatype=
    \map{
      \step[field
        \step[field
      }
    }
  }
}
```

Copy the `title` field to the `note` field
unless the `note` field already exists.

```
\DeclareSourcemap
  \maps[datatype=
    \map{
      \step[types
```

```

        \step[field
    }
}
}
}

```

Any **chat** entrytypes would become customa entrytypes and would automatically have a **type** field set to 'chat' unless the **type** field already exists in the entry (because **overwrite** is false by default).

This mapping applies only to entries of type **@chat** since the first step has **final** set and so if the **typesource** does not match

the entry entrytype, processing of this **\map** immediately terminates.

```

\DeclareSourcemap
\maps[datatype=

```

```

\map{
  \perdatasou
  \pertype{ar
  \pertype{bo
  \step[fiel
  \step[fiel
}
}
}

```

Any entries of entrytype **@article** or **@book** from the **examples.bi**
 datasource would have their **abstract**
 fields removed and a **note** field added with value 'Auto-created this field'.

```

\DeclareSourcemap
\maps[datatype=

```

```

        \map{
            \step[field]
            \step[field]
            \step[field]
        }
    }
}

```

This removes **abstract** fields from any
conductor entry, changes fields to
namea fields and changes **gps** fields to
usera fields.

```

\DeclareSourcemap
\maps[datatype=
    \map{
        \step[field]
    }
]

```

```

        \step[field]
        \step[field]
    }
}
}

```

Applies only to entries with **pubmed**
 fields and maps **pubmedid**
 fields to **eprint** fields, sets the
eprinttype field to 'pub-
userid medid' and also sets the field to the string 'Some
 string of things'.

```

\DeclareSourcemap
\maps[datatype=
\map{
\step[field]

```


matc
repl

}

}

}

series

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.

```
\DeclareSourcemap
\maps[datatype=
\map{
\step[fiel
matc
fina
\step[fiel
```

```

    }
  }
}

```

maintitle

Here, if for an entry, the field matches a particular regular expression, we set a special keyword so we can, for example, make a references section just for certain items.

```

\DeclareSourcemap
  \maps[datatype=
    \map{
      \step[fiel
      \step[fiel
    }
  }
}

```

lista

If an entry has a field which matches regular expression 'regexp', then it is removed.

```

\DeclareSourcemap

```

```

\maps[datatype=
  \map[overwrit
    \step[fiel
    \step[fiel
    \step[fiel
  }
}
}

```

For any entry with an **author** field,
 try to set **sortname** to the
 same as **author**. If this fails because
sortname already exists, stop, other-
sortname
 wise truncate **sortname** to just the first
 name in the name list.

```

\DeclareSourcemap
\maps[datatype=

```

```

\map{
  \step[field
    matc
    repl
  \step[field
    matc
    repl
}
}
}

```

Here, we use multiple match/replace for the same field to regularise some in-

constant name variants. Bear in mind that `\step` pro-

cessing 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 `\regex` 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 \TeX , 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:

```
\DeclareSourcemap
\maps[datatype=
\map{
\step[field
matc
repl
\step[field
matc
repl
}
}
}
```

Here, we have used the hexadecimal escape sequence ‘\x20’ in place of literal spaces in the replacement strings.

```
\DeclareSourcemap
\maps[datatype=
\map[overwrit
\step[fiel
\step[fiel
\step[fiel
\step[fiel
matc
repl
\step[fiel
matc
repl
}
```

```
}
}
```

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.

```
\DeclareSourcemap
  \maps[datatype=
    \map[overwrit
      \step[field
        \step[field
          \step[field
            \step[field
              \step[field
                \step[field
```

```

    }
  }
}

```

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.

```

\DeclareSourcemap
  \maps[datatype=
    \map[overwrit
      \step[field

```



```

\step[field
\step[field
\step[field
\step[field
\step[field
\step[field
\step[field
\step[field
\step[field
\step[field
}
}
}

```

This example assumes you have created a field called

autourl using the datamodel macros from § 4.5.3 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 lastname from the author, assuming a 'last, first' 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=
    \map{
      \step[field
        \step[entry
      }
    }
  }
}
```

Any entry with a title field matching 'A Title' will be completely ignored.

4.5.3 Data Model Specification

\BiberOnlyMark

The data model which Biblatex uses consists of four main elements:

- 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 backends. The data model for the BibTeX backend is hard-coded and cannot be changed. Changing the data model is only supported for the Biber 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 § 4.5.2).

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**
 → **'style option'.dbx**

‘citetstyle option
 ‘bibstyle option
 and
 →
 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:

\DeclareDatamodelEntrytypes
 [*options*] { *entrytypes* } Biber only

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.

skipout=true,
 false
 default: false

This entrytype is not output to the .bbl. Typically used for special entrytypes which are processed and consumed by the backend such as

@xdata.

\DeclareDatamodelFieldtypes
 [*options*] { *fields* } Biber only

Declares the comma-separated list of $\langle fields \rangle$ to be valid fields in the data model with associated comma-separated $\langle options \rangle$. The $\langle type \rangle$ and $\langle datatype \rangle$ options are mandatory. All valid $\langle options \rangle$ are:

type= $\langle field\ type \rangle$

Set the type of the field as described in § 2.2.1. Is typically ‘field’ or ‘list’.

format= $\langle field\ format \rangle$

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 **xsvsep** and defaults to the expected comma surrounded by optional whitespace.

datatype= $\langle field\ datatype \rangle$

Set the datatype of the field as described in § 2.2.1. For example, ‘name’ or ‘literal’.

nullok=true,
false
false

default:

The field is allowed to be defined but empty.

skipout=true,
false
false

default:

.bbl
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.

`label=true` default: `false`
`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.

`\DeclareDatamodelEntry`
`[<entrytypes>] { <fields> }` Biber only

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.

`\DeclareDatamodelCons`
`[<entrytypes>] { <specification> }` Biber only

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:

`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.

datatype=integer,
isbn,
issn,
ismn,
date,
pattern

For constraints of type $\langle data \rangle$, constrain field values to be the given datatype.

rangemin= $\langle num \rangle$

For constraints of $\langle type \rangle$ ‘data’ and $\langle datatype \rangle$ ‘integer’, constrain field values to be at least $\langle num \rangle$.

rangemax= $\langle num \rangle$

For constraints of $\langle type \rangle$ ‘data’ and $\langle datatype \rangle$ ‘integer’, constrain field values to be at most $\langle num \rangle$.

pattern= $\langle patt \rangle$

For constraints of $\langle type \rangle$ ‘data’ and $\langle datatype \rangle$ ‘pattern’, constrain field values to match regular expression pattern $\langle patt \rangle$. It is best to wrap any regular expression in

the macro **\regexp**, see § 4.5.2.

A **\constraint** macro
may contain any of the following:

\constraintfieldsor
{ *<fields>* **}**

For constraints of *<type>* 'mandatory', specifies
that an entry must contain a boolean OR of the

\constraintfield_s.

\constraintfieldsxor
{ *<fields>* **}**

For constraints of *<type>* 'mandatory', specifies that
an entry must contain a boolean XOR of the

\constraintfield_s.

\antecedent[*<quantifier=quantspec>*] **{** *<fields>* **}**

For constraints of *<type>* 'conditional', specifies a quantified set of

\constraintfield_s

\consequent

which must be satisfied before the

of the constraint is checked. *<quantspec>* should have one of the following values:

quantifier=all,
one,
none

Specifies how many of the **\constraintfie**

\antecedent

inside the

have to be present to satisfy the antecedent of the conditional constraint.

`\consequent[<quantifier=quantspec>]{<fields>}`

For constraints of *<type>* ‘conditional’, specifies a quantified set of

`\constraintfields`

`\antecedent`

which must be satisfied if the preceding

of the constraint was satisfied. *<quantspec>* should have one of the following values:

`quantifier=all`,
`one`,
`none`

`\constraintfi`

Specifies how many of the

`\consequent`

inside the

have to be present to satisfy the consequent of the conditional constraint.

`\constraintfield` `{<field>}`

For constraints of *<type>* ‘data’, the constraint applies to this *<field>*. For constraints of *<type>* ‘mandatory’, the entry must contain this *<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 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

`biblatex-dm.cfg`

file:

`\ResetDatamodelEntryt`

Clear all data model entrytype information.

`\ResetDatamodelFields`

Clear all data model field information.

`\ResetDatamodelEntryt`

Clear all data model fields for entrytypes information.

`\ResetDatamodelConst`

Clear all data model fields Constraints information.

Here is an example of a simple data model. Refer to the core Biblatex file

`blx-dm.def` for the default data
model specification.

`\ResetDatamodelEn`

`\ResetDatamodelFi`

`\ResetDatamodelEn`

`\ResetDatamodelCo`

`\DeclareDatamodel`

`\DeclareDatamodel`

`\DeclareDatamodel`

```

\DeclareDatamodel
\DeclareDatamodel
\DeclareDatamodel
    \constraint[typ
        \constraintfi
    }
    \constraint[typ
        \constraintfi
        \constraintfi
            \constraint
            \constraint
        }
    }
}
\DeclareDatamodel

```

```

    \constraint[typ
      \antecedent[q
        \constraint
      }
    \consequent[q
      \constraint
      \constraint
    }
  }
}

```

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

4.5.4 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. Label customisation requires Biber and will not work with any other backend.

`\DeclareLabelalphaTem`

`[<entrytype, ...>]{<specification>}`

Biber only

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

`\literal` commands 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 $\langle field \rangle$ 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 the options which take substrings of the $\langle field \rangle$ to use in the

`\labelelement`

are applied to the lastname of every name in a name list (see below).

`final` = `true` or `false`
default: `false`

`\field`

This option marks a `\field` directive as the final one in the $\langle specification \rangle$. If the $\langle field \rangle$ is non-empty, then this field is used for the label and the remainder of the $\langle specification \rangle$ will

be ignored. The short form `final` is equivalent to

`final=true`

`lowercase` = `true` or `false`
default: `false`

default:

Forces the label part derived from the field to lowercase. By default, the case is taken from the field source and not modified.

`strwidth` = $\langle integer \rangle$ default: `1`

The number of characters of the $\langle field \rangle$ to use. For name list fields,

if `useprefix` = `true`, the first character of the name prefix is prepended to the substring.

strside=left,
right

default: left

The side of the string from which to take the

strwidth number of characters.

padside=left,
right

default: right

Side to pad the label part when using the

padchar option. Only for use with fixed-

strwidth width label strings (

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-

strwidth width label strings (

uppercase=true,
false

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

default: false

Use a variable width, left-side substring of characters from the *<field>* (each lastname in name list fields). 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, if

useprefix =

true

, the first character of the name prefix is prepended to the substring.

`varwidthnorm=true`
`false`

default: `false`

`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`
`true`.

`varwidthlist=true`
`false`

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

`varwidth`. For fields, this is equivalent to . 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, if
`useprefix` =
`true`, the first character of the name prefix is prepended
to the substring.

`strwidthmax` = $\langle integer \rangle$
`varwidth`

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.

`strfixedcount`
= $\langle integer \rangle$ default: 1

When using `varwidthnorm`,
`strfixedcount`
there must be at least
disambiguated substrings with the same, maximal length to trigger the forcing of all
disambiguated substrings to this same maximal length.

`compound` = `true`,
`false`
`false`
default:

For static (non-`varwidth`) disambiguation, treat lastname name components sepa-
rated by whitespace or hyphens (compound last 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`
`compound` =
and ‘B’ with

`false`. The short form
compound is equivalent
to compound =
`true`.

`ifnames=` $\langle integer \rangle$

Only use this `\field` specification if it is a name list

field with `ifnames` names in it. This allows a

`\label element`

to be conditionalised on name length (see below).

`names=` $\langle integer \rangle$

By default, for name list fields, the number of
names considered when building the label obeys the

`maxalphanames`,
`minalphanames`

truncation. This option can be used to override this with an explicit upper limit of
how many names to consider. This is useful if you have some very long name lists
which might result in a very long label.

form=original,
translated,
romanised,
uniform
original

default:

Specifies the script variant of a field, if the datasource you are using supports this.

lang=‘language
specifier’
babelpolyglossi

Specifies the

language variant of a field if the datasource you are using supports this. The default, if no *lang* option is set, this means to use the field variant in the datasource which has no explicit language defined.

\literal{ *characters* }

Insert the literal *characters* into the label at this point.

Note that the template for labels can be defined per-type and you should be aware of this when using the automatically disambiguated label functionality. Disambiguation is not per-type as this might lead to ambiguity due to different label formats for different types being isolated from each others disambiguation process. Normally, you will want to use very different label formats for different types to make the type obvious by the label.

Here are some examples. The default global Biblatex alphabetic label template

is defined below. Firstly, **shorthand**
final = true and
so if there is a **shorthand**

field, it is used as the label and nothing more of the template is con-

sidered. Next, the `label` field is used as the first label element if it exists. Otherwise, if there is only one

name (`ifnames = 1`)

in the `labelname` list, then three characters from the left side of the lastname in the

`labelname` are used as the first

`labelname` label element. If the `labelname` has

more than one name in it, one character from the left side of each lastname is used as the first label element. The second label element consists of 2 characters from the

right side of the `year` field.

```
\DeclareLabelalph
\labelelement{
  \field[final]
  \field{label}
  \field[strwid
  \field[strwid
}
\labelelement{
```

```

        \field[strwid
    }
}

```

To get an idea of how the label automatic disambiguation works, consider the following author lists:

```

Agassi, Chang, La
Agassi, Connors,
Agassi, Courier,
Borg, Connors, Ed
Borg, Connors, Em

```

Assuming a template declaration such as:

```

\DeclareLabelalph
    \label element{
        \field[varwid
    }
}

```

Then the labels would be:

Agassi, Chang, La
Agassi, Connors,
Agassi, Courier,
Borg, Connors, Ed
Borg, Connors, Em

With normalised variable width labels defined:

```
\DeclareLabelalph  
  \label element{  
    \field[varwid  
  }  
}
```

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:

Agassi, Chang, La
Agassi, Connors,
Agassi, Courier,

Borg, Connors, Ed
Borg, Connors, Em

With a restriction to two characters for the name components of the label element defined like this:

```
\DeclareLabelalph  
  \label element{  
    \field[varwid  
  }  
}
```

This would be the result (note that the individual lastname label parts are no longer unambiguous):

Agassi, Chang, La
Agassi, Connors,
Agassi, Courier,
Borg, Connors, Ed
Borg, Connors, Em

Alternatively, you could choose to disambiguate the name lists as a whole with:

```
\DeclareLabelalph
```

```

\label{element{
  \field[varwid
}
}

```

Which would result in:

Agassi, Chang, La
 Agassi, Connors,
 Agassi, Courier,
 Borg, Connors, Ed
 Borg, Connors, Em

Perhaps you only want to consider at most two names for label generation but disambiguate at the whole name list level:

```

\DeclareLabelalph
\label{element{
  \field[varwid
}
}

```

Which would result in:

Agassi, Chang, La
Agassi, Connors,
Agassi, Courier,
Borg, Connors, Ed
Borg, Connors, Em

`\labelalphao`

In this last example, you can see

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:

```
\DeclareLabelalph  
  \labelelement{  
    \field[varwid  
  }  
  \labelelement{  
    \literal{-}
```

```

    }
    \label{element{
        \field{strwid
    }
}

```

Which would result in:

```

Agassi, Chang, La
Agassi, Connors,
Agassi, Courier,
Borg, Connors, Ed
Borg, Connors, Em

```

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:

```

\DeclareLabelalph
    \label{element{
        \literal{>}
    }
}

```

```

    }
    \label{element{
        \literal{\%}
    }
    \label{element{
        \field[strwid
    }
    \label{element{
        \field[strwid
    }
    \label{element{
        \field[strwid
    }
}

```

which given:

```
@Book{test,
```

```

author      = {XX
title       = {T}
year        = {20
}

```

would resulting a label looking like this:

```
[>%YY__&&T07]
```

4.5.5 *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. Note that custom sorting schemes require Bibex. They will not work with any other backend.

```

\DeclareSortingScheme
[ <options> ] { <name> } { <specification> }

```

Biber only

Defines the sorting scheme `<name>`. The `<name>` is the identifier

passed to the `sorting` option (§ 3.1.2.1) when selecting the sorting scheme. The

```

\DeclareSortingSc

```

command supports the following optional arguments:

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.

\sort directives

The *⟨specification⟩* is an undelimited list of which specify the elements to be considered in the sorting process. 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.

\sort{_{⟨elements⟩}**}**

Specifies the elements considered in the sorting process. The

⟨elements⟩ are an undelimited list of **\field**,

\literal, and

\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 **\sort** directive. If it is undefined, the next element is evaluated. Since literal strings are always defined, any

\literal commands should be the

sole or the last element in a **\sort** directive. The

\sort command supports the following optional arguments:

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

\DeclareSortingSc

and also the global setting. See also the discussion of the global sorting locale option

sortlocale

in § 3.1.2.1.

direction=ascendi

descend

default: ascending

The sort direction, which may be either ascending

or descending. The default is ascending order.

final=true

default:

false

\sort

This option marks a directive as the final one in the *<specification>*. If one of the *<elements>* is available, the remainder of the

<specification> will be ignored. The short form *final* is

equivalent to *final=true*.

sortcase=true

false

Whether or not to sort case-sensitively. The default setting depends on the global

sortcase

option.

`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.

`\field[<key=value, ...>]{<field>}`

The `\field` element adds a *<field>* to the sorting specification. If the *<field>* is undefined, the element is skipped. The

`\field` command supports the following optional arguments:

`padside=left,
right`

default: `left`

Pads a field on the `left` or `right`

side using `padchar` so that its width is

`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. Padding is particularly useful with numeric fields. For example, the command

`\field[padding=le`

`volume` will pad the field with leading zeros to a width

of two characters. This way, volumes are sorted by numeric value (`01/`

`02/11/12`) rather than in alphabetic order

(`1/11/12/2`).

`padding=` $\langle integer \rangle$ default: `4`

The target width in characters.

`padding=` $\langle character \rangle$ default: `0`

The character to be used when padding the field.

`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

`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.

`strwidth=` _{$\langle integer \rangle$} default: 4

The number of characters to match.

`\literal{}` _{$\langle string \rangle$}

The `\literal` element adds a literal $\langle string \rangle$ to the sorting specification. This is useful as a fallback if some fields are not available.

`\citeorder` 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 sorting scheme: `none`

```
\DeclareSortingScheme
  \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
```

```

\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).

Here are 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 editor, the translator,
or the title. Given
this specification, the sorting routine will use the first element which

```

is available and continue with the title. Note

```

that the options useauthor,
useeditor, and
usetranslator

```

are considered automatically in the sorting process:

```

\DeclareSortingScheme
\sort{

```

```

        \field{author}
        \field{editor}
        \field{transl}
    }
    \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**,
sortkey
etc. Since the **sortkey** field
specifies the master sort key, it needs to override all other ele-
ments 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:

```
\DeclareSortingScheme
  \sort{
    \field{presort}
  }
  \sort[final]{
    \field{sortkey}
  }
  \sort{
    \field{sortname}
    \field{author}
    \field{editor}
```

```

        \field{transl
        \field{sortti
        \field{title}
    }
\sort{
    \field{sortti
    \field{title}
}
\sort{
    \field{sortye
    \field{year}
}
}
}

```

Finally, here is an example of a sorting scheme which overrides the global sorting locale and additionally overrides again when sorting by the

origtitle field. Note the use in the scheme-level override of a babel/polyglossia language name instead of a real lo-

cale identifier. Biber will map this to a suitable, real locale identifier (in this case,

`sv_SE`):

```
\DeclareSortingScheme
  \sort{
    \field{sortna}
    \field{author}
    \field{editor}
    \field{transl}
    \field{sortti}
    \field{title}
  }
  \sort[locale=de]
    \field{origti}
  }
}
```

\DeclareSortExclusion

$\{ \langle \textit{entrytype}, \dots \rangle \} \{ \langle \textit{field}, \dots \rangle \}$

Biber only

Specifies fields to be excluded from sorting on a per-type basis. The $\langle \textit{entrytype} \rangle$ argument and the $\langle \textit{field} \rangle$ argument may be a comma-separated list of values. A blank $\langle \textit{field} \rangle$ argument will clear all exclusions for this $\langle \textit{entrytype} \rangle$. This command may only be used in the preamble.

\DeclarePresort

$[\langle \textit{entrytype}, \dots \rangle] \{ \langle \textit{string} \rangle \}$

Biber only

Specifies a string to be used to automatically populate the

presort

field of entries with-

out a **presort**

field. The

presort

may be defined globally or on a

per-type basis. If the optional $\langle \textit{entrytype} \rangle$ argument is given, the $\langle \textit{string} \rangle$ applies to the respective entry type. If not, it serves as the global default value. Specifying an $\langle \textit{entrytype} \rangle$ in conjunction with a blank $\langle \textit{string} \rangle$ will clear the type-specific setting. The $\langle \textit{entrytype} \rangle$ argument may be a comma-separated list of values. This command may only be used in the preamble.

4.5.6 Bibliography List Filters

Biber only

When using customisable bibliography lists (See § 3.6.4), usually one wants

to return in the **.bb1** 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 re-

turned by Biber in the **.bb1** to contain only those en-

tries which have a **shorthand**

field. This is accomplished by defining a bibliography list filter using the

\DeclareBiblistFi

command. This differs from the filters defined using

`\defbibfilter`
`\defbibfilt`
(see § 3.6.10) since the filters defined by

`.bbl`
run inside Biblatex after the `.bbl` has been generated. In

addition, bibliography lists in the `.bbl` do not contain entry data, only the citation keys for the entries and so no filtering by Biblatex using

`\defbibfilter`
is possible for bibliography lists.

`\DeclareBiblistFilter`

`{\langle name \rangle}{\langle specification \rangle}`

Biber only

Defines a bibliography list filter with `\langle name \rangle`. The `\langle specification \rangle` con-

sists of one or more `\filter` or

`\filteror` macros, all of which must be satisfied for the entry to pass the filter:

`\filter{\langle type \rangle}{\langle filter \rangle}`

Filter entries according to the `\langle type \rangle` and `\langle filter \rangle`. `\langle type \rangle` can be one of:

`entrytype`
`type/nottype` Entry is/is not of `\langle filter \rangle`

`subtype`
`subtype/notsubtype` Entry is/is not of `\langle filter \rangle`

keyword/notkeyword Entry has/does not have

keyword $\langle filter \rangle$

field/notfield Entry has/does not have a field called $\langle filter \rangle$

$\backslash filter or \{ \} \{ \}$

$\backslash filter$

A wrapper around one or more commands specifying that they form a disjunctive set, i.e. any one of the $\langle filters \rangle$ must be satisfied.

Fields in the datamodel which are marked as ‘Label fields’ (see § 4.5.3) 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 shorthand field:

```
\DeclareBibliographyField{
  \filter[type=fi
}
```

4.5.7 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.

$\backslash DeclareNoinit$
 $\{ \}$ $\langle specification \rangle$

Biber only

Defines regular expressions to strip from names before generating initials. The

$\langle specification \rangle$ is an undelimited list of $\backslash noinit$

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{⟨regex⟩}`

Any number of `\noinit` commands can be given each of which specifies to remove the *⟨regex⟩* 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

`\regex` macro as shown—this will pass the expression through to BibTeX correctly.

If there is no specification, BibTeX will default to:

`\DeclareNoinit`

```
\DeclareNoinit{
% strip lowerca
\noinit{\regex
% strip some co
\noinit{\regex
}
```

This BibTeX default strips a couple of diacritics and also strips lowercase prefixes from names before generating initials.

4.5.8 Fine Tuning Sorting

It can be useful to fine tune sorting so that it ignores certain parts of particular fields.

`\DeclareNosort`

`{ <specification> }`

Biber only

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 simply a convenience grouping of semantically similar fields from which you might want to remove a regexp. Table 9 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.

`\DeclareNosort`

If there is no specification, Biber will default to:

```
\DeclareNosort{
% strip prefixes
\nosort{type_name}
% strip some diacritics
\nosort{type_name}
```

}

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:

```
\DeclareNosort{
  \nosort{title}{
}
```

Or if you wanted to ignore “The” at the beginning of any title field:

```
\DeclareNosort{
  \nosort{type_title}{
}
```

4.5.9 *Special Fields*

Some of the automatically generated fields from § 4.2.4.2 may be customized. Note that this requires Biber.

```
\DeclareLabelname
[<entrytype, ...>] {<specification>}
```

Biber only

Defines the fields to consider when generating the

```
labelname field (see § 4.2.4.2).
\field
```

The *<specification>* is an ordered list of

commands. The fields are checked in the order listed and the first field which is

available will be used as

This is the default definition:

`labelname`

```
\DeclareLabelname
  \field{shortaut
  \field{author}
  \field{shortedi
  \field{editor}
  \field{translat
}
```

`labelname`

The `labelname` field may be customized globally or on a per-type basis. If the optional `<entrytype>` argument is given, the specification applies to the respective entry type. If not, it is applied globally. The `<entrytype>` argument may be a comma-separated list of values. This command may only be used in the preamble.

`\DeclareLabeldate`

`[<entrytype, ...>] { <specification> }`

Biber only

Defines the date components to consider when generat-

ing `labelyear`,

`labelmonth`

and

`labelday` fields (see § 4.2.4.2). The

`\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 `labelyear`,

`labelmonth` and

`labelday` fields. Note that the

`\field` items do not have to be datatype '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`

`\literal` suitably. Note also that a command will always be used when found and so this should always be the last thing

`\literal` in the list. If the value of a

command is a valid localization 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{eventdat
```

```

\field{origdate
\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 **labelyear**,

labelmonth and **labelday** fields may be customized

globally or on a per-type basis. If the optional *⟨entrytype⟩* argument is given, the specification applies to the respective entry type. If not, it is applied globally. The *⟨entrytype⟩* 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`

`[<entrytype, ...>]{<specification>}`

Biber only

Defines the fields to consider when generating the

`labeltitle` field (see § 4.2.4.2).

The *<specification>* 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

This is the default definition:

`labeltitle`.

```
\DeclareLabeltitl
  \field{shorttit
  \field{title}
}
```

The `labeltitle` field may be

customized globally or on a per-type basis. If the optional *<entrytype>* argument is given, the specification applies to the respective entry type. If not, it is applied globally. The *<entrytype>* argument may be a comma-separated list of values. This command may only be used in the preamble.

4.5.10 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. Note that customized data inheritance requires Biber. It will not work with any other backend. A note on terminology: the *child* or *target*

is the entry with the `crossref` field, the

parent or *source* is the entry the field points to. The child inherits data from the parent.

\DefaultInheritance

[*<exceptions>*] { *<options>* }

Biber only

Configures the default inheritance behavior. This command may only be used in the preamble. The default behavior may be customized by setting the following *<options>*:

all=true default: true
false

Whether or not to inherit all fields from the parent by default.

means that the child entry inherits all fields from the parent, unless a more specific inheritance rule has been set up with

\DeclareDataInher

If an inheritance rule is defined for a field, data inheritance is controlled

by that rule. **all** = false

means that no data is inherited from the parent by default. Each field to be inherited requires an explicit inheritance rule set up with

\DeclareDataInher

all=true.

The package default is

override=true
false
false

default:

Whether or not to overwrite target fields with source fields if both are de-

defined. This applies both to automatic inheritance and to explicit inheritance rules.

The package default is **override=false**, i. e., existing fields of the child entry are not overwritten.

The optional $\langle exceptions \rangle$ are an undelimited list of **\except** directives. Spaces, tabs, and line endings may be used freely to visually arrange the $\langle exceptions \rangle$. Blank lines are not permissible.

\except $\{ \langle source \rangle \} \{ \langle target \rangle \} \{ \langle options \rangle \}$

Sets the $\langle options \rangle$ for a specific $\langle source \rangle$ and $\langle target \rangle$ combination. The $\langle source \rangle$ and $\langle target \rangle$ arguments specify the parent and the child entry type. The asterisk matches all types and is permissible in either argument.

\DeclareDataInheritance
 $\{ \langle source, ... \rangle \} \{ \langle target, ... \rangle \} \{ \langle rules \rangle \}$ Biber only

Declares inheritance rules. The $\langle source \rangle$ and $\langle target \rangle$ 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 $\langle rules \rangle$

are an undelimited list of **\inherit** and/or **\noinherit** directives. Spaces, tabs, and line endings may be used freely to visually arrange the $\langle rules \rangle$. Blank lines are not permissible. This command may only be used in the preamble.

\inherit $[\langle option \rangle] \{ \langle source \rangle \} \{ \langle target \rangle \}$

Defines an inheritance rule by mapping a $\langle source \rangle$ field to a $\langle target \rangle$ field. The $\langle option \rangle$ is the **override** option explained

above. When set locally, it takes precedence over any global options set with

```
\DefaultInheritance
\noinherit{<source>}
```

Unconditionally prevents inheritance of the `<source>` field.

```
\ResetDataInheritance
\BiberOnlyMark
```

```
\DeclareDataInheritance
```

Clears all inheritance rules defined with `\DeclareDataInheritance`.
This command may only be used in the preamble.

Here are some practical examples:

```
\DefaultInheritance
```

This example shows how to configure the default inheritance behavior. The above settings are the package defaults.

```
\DefaultInheritance
  \except{*}{online}
]{all=true,override}
```

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
```

```

\inherit{title}
\inherit{subtit
\inherit{titlea
}

```

So far we have looked at setting up the defaults. For example,

```
all = true
```

means that the

publisher

field of a source en-

```
publisher
```

try is copied to the

field of the target entry. In some cases, however, asym-

metric mappings are required. They are defined with

```
\DeclareDataInher

```

The above example sets up three typical rules for

```
@incollection
@collection

```

entries referencing a

```
title
```

We map the

```
booktitle
```

corresponding fields of
the target.

```
\DeclareDataInher
\inherit{author

```

```
\inherit{author
}
```

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 ei-

ther an **@inbook** or a

@bookinbook entry.

@bookinbook entry.

```
\DeclareDataInher
```

```
\noinherit{intr
```

```
}
```

```
}
```

introduction

@inbook or

field. It applies to all targets of type

@incollection,

regardless of the source entry type.

```
\DeclareDataInherit  
  \noinherit{abstract  
}
```

This rule, which applies to all entries, regardless of the source and target entry types,

prevents inheritance of the **abstract** field.

```
\DefaultInheritance  
\ResetDataInherit
```

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.

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.

```
\thefield{<field>}
```

Expands to the unformatted *<field>*. If the *<field>* is undefined, this command expands to an empty string.

`\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.

`\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.

`\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.

`\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` { $\langle literal list \rangle$ } { $\langle macro \rangle$ }
`\savelist*` { $\langle literal list \rangle$ } { $\langle macro \rangle$ }

Copies an unformatted $\langle literal list \rangle$ to a $\langle macro \rangle$. The regular variant of this command defines the $\langle macro \rangle$ globally, the starred one works locally.

`\savename` { $\langle name list \rangle$ } { $\langle macro \rangle$ }
`\savename*` { $\langle name list \rangle$ } { $\langle macro \rangle$ }

Copies an unformatted $\langle name list \rangle$ to a $\langle macro \rangle$. The regular variant of this command defines the $\langle macro \rangle$ globally, the starred one works locally.

`\savefieldcs`
{ $\langle field \rangle$ } { $\langle csname \rangle$ }
`\savefieldcs*`
{ $\langle field \rangle$ } { $\langle csname \rangle$ }

Similar to `\savefield`, but takes the control sequence name $\langle csname \rangle$ (without a leading backslash) as an argument, rather than a macro name.

`\savelistcs` { $\langle literal list \rangle$ } { $\langle csname \rangle$ }
`\savelistcs*`
{ $\langle literal list \rangle$ } { $\langle csname \rangle$ }

Similar to `\savelist`, but takes the

control sequence name $\langle csname \rangle$ (without a leading backslash) as an argument, rather than a macro name.

$\backslash savenames$ $\{ \langle name list \rangle \} \{ \langle csname \rangle \}$
 $\backslash savenames^*$
 $\{ \langle name list \rangle \} \{ \langle csname \rangle \}$

Similar to $\backslash savecsname$, but takes the control sequence name $\langle csname \rangle$ (without a leading backslash) as an argument, rather than a macro name.

$\backslash restorefield$
 $\{ \langle field \rangle \} \{ \langle macro \rangle \}$

Restores a $\langle field \rangle$ from a $\langle macro \rangle$ defined with $\backslash savefield$ before. The field is restored within a local scope.

$\backslash restorelist$
 $\{ \langle literal list \rangle \} \{ \langle macro \rangle \}$

Restores a $\langle literal list \rangle$ from a $\langle macro \rangle$ defined with $\backslash savelist$ before. The list is restored within a local scope.

$\backslash restorename$
 $\{ \langle name list \rangle \} \{ \langle macro \rangle \}$

Restores a $\langle name list \rangle$ from a $\langle macro \rangle$ 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.

\iffieldundef
{ } {*true*} {*false*}

Expands to *true* if the *field* is undefined, and to *false* otherwise.

\iflistundef
{ } {*true*} {*false*}

Expands to *true* if the *literal list* is undefined, and to *false* otherwise.

\ifnameundef
{ } {*true*} {*false*}

Expands to *true* if the *name list* is undefined, and to *false* otherwise.

\iffieldequal

$\{ \langle field\ 1 \rangle \} \{ \langle field\ 2 \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ if the values of $\langle field\ 1 \rangle$ and $\langle field\ 2 \rangle$ are equal, and to $\langle false \rangle$ otherwise.

\iflistsequal

$\{ \langle literal\ list\ 1 \rangle \} \{ \langle literal\ list\ 2 \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ if the values of $\langle literal\ list\ 1 \rangle$ and $\langle literal\ list\ 2 \rangle$ are equal, and to $\langle false \rangle$ otherwise.

\ifnamesequal

$\{ \langle name\ list\ 1 \rangle \} \{ \langle name\ list\ 2 \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ if the values of $\langle name\ list\ 1 \rangle$ and $\langle name\ list\ 2 \rangle$ are equal, and to $\langle false \rangle$ otherwise.

\iffieldequals

$\{ \langle field \rangle \} \{ \langle macro \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ if the value of the $\langle field \rangle$ is equal to the definition of $\langle macro \rangle$, and to $\langle false \rangle$ otherwise.

\iflistequals

$\{ \langle literal\ list \rangle \} \{ \langle macro \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ if the value of the $\langle literal\ list \rangle$ is equal to the definition of $\langle macro \rangle$, and to $\langle false \rangle$ otherwise.

\ifnameequal

$\{ \langle name list \rangle \} \{ \langle macro \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ if the value of the $\langle name list \rangle$ is equal to the definition of $\langle macro \rangle$, and to $\langle false \rangle$ otherwise.

\iffieldequalcs

$\{ \langle field \rangle \} \{ \langle csname \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Similar to $\backslash\iffieldequal$ but takes the control sequence name $\langle csname \rangle$ (without a leading backslash) as an argument, rather than a macro name.

\iflstequalcs

$\{ \langle literal list \rangle \} \{ \langle csname \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Similar to $\backslash\iflstequal$ but takes the control sequence name $\langle csname \rangle$ (without a leading backslash) as an argument, rather than a macro name.

\ifnameequalcs

$\{ \langle name list \rangle \} \{ \langle csname \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Similar to $\backslash\ifnameequal$ but takes the control sequence name $\langle csname \rangle$ (without a leading backslash) as an argument, rather than a macro name.

\iffieldequalstr

{*⟨field⟩*} {*⟨string⟩*} {*⟨true⟩*} {*⟨false⟩*}

Executes *⟨true⟩* if the value of the *⟨field⟩* is equal to *⟨string⟩*, and *⟨false⟩* otherwise. This command is robust.

\iffieldxref

{*⟨field⟩*} {*⟨true⟩*} {*⟨false⟩*}

If the **crossrefxref** field of an entry is defined, this command checks if the *⟨field⟩* is related to the cross-referenced parent entry. It executes *⟨true⟩* if the *⟨field⟩* of the child entry is equal to the corresponding *⟨field⟩* of the parent entry, and *⟨false⟩* otherwise.

If the **crossrefxref** field is undefined, it always executes *⟨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.

\iflistxref

{*⟨literal list⟩*} {*⟨true⟩*} {*⟨false⟩*}

Similar to **\iffieldxref** but checks if a *⟨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.

`\ifnamexref{<name list>}{<true>}{<false>}`

Similar to `\iffieldxref` but checks if a `<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`

`{<field>}{<true>}{<false>}`

Executes `<true>` if the current field is `<field>`, and `<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.

`\ifcurrentlist`

`{<literal list>}{<true>}{<false>}`

Executes `<true>` if the current list is `<literal list>`, and `<false>` otherwise. This command is robust. It is intended for use in list formatting directives and always executes `<false>` when used in any other context.

`\ifcurrentname`

`{<name list>}{<true>}{<false>}`

Executes `<true>` if the current list is `<name list>`, and `<false>` otherwise. This command is robust. It is intended for use in list formatting directives and always executes `<false>` when used in any other context.

\ifuseprefix

{ $\langle true \rangle$ } { $\langle false \rangle$ }

Expands to $\langle true \rangle$ if the **useprefix** option is enabled (either globally or for the current entry), and $\langle false \rangle$ otherwise. See § 3.1.3 for details on this option.

\ifuseauthor

{ $\langle true \rangle$ } { $\langle false \rangle$ }

Expands to $\langle true \rangle$ if the **useauthor** option is enabled (either globally or for the current entry), and $\langle false \rangle$ otherwise. See § 3.1.3 for details on this option.

\ifuseeditor

{ $\langle true \rangle$ } { $\langle false \rangle$ }

Expands to $\langle true \rangle$ if the **useeditor** option is enabled (either globally or for the current entry), and $\langle false \rangle$ otherwise. See § 3.1.3 for details on this option.

\ifusetranslator

{ $\langle true \rangle$ } { $\langle false \rangle$ }

Expands to $\langle true \rangle$ if the **usetranslator** option is enabled (either globally or for the current entry), and $\langle false \rangle$ otherwise. See § 3.1.3 for details on this option.

\ifsingletitle

{ $\langle true \rangle$ } { $\langle false \rangle$ }

Expands to $\langle true \rangle$ if there is only one work by the

labelname

name in the bibliography, and to $\langle false \rangle$ otherwise. If there is no

labelname

name at all for the entry, then this expands to $\langle true \rangle$ if there is only one work

labeltitle

with the title in the bibliography and $\langle false \rangle$ otherwise. If nei-

ther labelname

nor

labeltitle

are set for an entry, this will always expand to $\langle false \rangle$. Note that this feature needs to be enabled explicitly with the package option

singletitle.

\ifandothers

{ $\langle list \rangle$ } { $\langle true \rangle$ } { $\langle false \rangle$ }

Expands to $\langle true \rangle$ if the $\langle list \rangle$ is defined and has been truncated

in the bib file with the keyword ‘and

others’, and to $\langle false \rangle$ otherwise. The $\langle list \rangle$

may be a literal list or a name list.

\ifmorenames

$\{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ if the current name list has been or will be truncated, and to $\langle false \rangle$ otherwise. This command is intended for use in formatting directives for name lists. It will always expand to $\langle false \rangle$ 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

listtotal counter is larger than 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

listcount is smaller than or equal to liststop , see § 4.4.1 for details.

\ifmoreitems

$\{ \langle true \rangle \} \{ \langle false \rangle \}$

\ifmorenames

This command is similar to \ifmoreitems but checks the current literal list. It is intended for use in formatting directives for literal lists. It will always expand to $\langle false \rangle$ when used elsewhere.

\iffirstinits

$\{ \langle true \rangle \} \{ \langle false \rangle \}$

Expands to $\langle true \rangle$ or $\langle false \rangle$, depending on the state of the

firstinits

package option (see § 3.1.2.3). This command is intended for use in formatting directives for name lists.

\ifterseinits

{ } { }

Expands to $\langle true \rangle$ or $\langle false \rangle$, depending on the state of the

terseinits

package option (see § 3.1.2.3). This command is intended for use in formatting directives for name lists.

\ifentrytype

{ } { } { }

Executes $\langle true \rangle$ if the entry type of the entry currently being processed is $\langle type \rangle$, and $\langle false \rangle$ otherwise.

\ifkeyword{ } { } { }

Executes $\langle true \rangle$ if the $\langle keyword \rangle$ is found in the

keywords

field of the entry currently being processed, and $\langle false \rangle$ otherwise.

\ifentrykeyword

{ } { } { } { }

\ifkeyword

A variant of which takes an entry key as its first argument. This is useful for testing an entry other than the one currently processed.

`\ifcategory{ $\langle category \rangle$ }{ $\langle true \rangle$ }{ $\langle false \rangle$ }`

Executes $\langle true \rangle$ if the entry currently being processed has been assigned to a $\langle category \rangle$ with

`\addtocategory`,

and $\langle false \rangle$ otherwise.

`\ifentrycategory`

`{ $\langle entrykey \rangle$ }{ $\langle category \rangle$ }{ $\langle true \rangle$ }{ $\langle false \rangle$ }`

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{ $\langle true \rangle$ }{ $\langle false \rangle$ }`

Executes $\langle true \rangle$ if the entry currently being processed has been cited before, and $\langle false \rangle$ otherwise. This command is robust and intended for use in citation styles.

If there are any `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

`citetracker`. The behavior of this test depends on the mode the citation tracker is operating in, see § 3.1.2.3 for details. If the citation tracker is disabled, the test always yields $\langle false \rangle$. Also see the

`\citetrackertrue`
`\citetrackerfalse`

and switches in § 4.6.4.

\ifentryseen

$\{ \langle \text{entrykey} \rangle \} \{ \langle \text{true} \rangle \} \{ \langle \text{false} \rangle \}$

\ifciteseen

A variant of `\ifentryseen` which takes an entry key as its first argument. Since the $\langle \text{entrykey} \rangle$ is expanded prior to performing the test, it is possible to test for entry keys in a field such as

`xref`:

`\ifentryseen{\the`

`\ifentryseen`

Apart from the additional argument,

`\ifciteseen`.

behaves like

\ifentryinbib

$\{ \langle \text{entrykey} \rangle \} \{ \langle \text{true} \rangle \} \{ \langle \text{false} \rangle \}$

Executes $\langle \text{true} \rangle$ if the entry $\langle \text{entrykey} \rangle$ appears in the current bibliography, and $\langle \text{false} \rangle$ otherwise. This command is intended for use in bibliography styles.

\iffirstcitekey

$\{ \langle \text{true} \rangle \} \{ \langle \text{false} \rangle \}$

Executes $\langle \text{true} \rangle$ if the entry currently being processed is the first one in the citation list, and $\langle \text{false} \rangle$ otherwise. This command re-

lies on the `citecount`,
`citetotal`,

multicitecount
multicitetotal

and counters (§ 4.10.5) and thus is intended for use only in the `<loopcode>` of a citation command defined with

`\DeclareCiteCommand`

`\iflastcitekey`

`{<true>}{<false>}`

`\iffirstcitekey`

Similar

but executes `<true>` if the entry currently being processed is the last one in the citation list, and `<false>` otherwise.

`\ifciteibid{<true>}{<false>}`

Expands to `<true>` if the entry currently being processed is the same as the last one, and to `<false>` otherwise. This command is intended for use in citation styles.

`refsection`
If there are any environments in the document, the tracking is local to these environments. Note that the ‘ibidem’ tracker needs to be enabled explicitly with the package

`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 `<false>`. Also see the

`\citetrackertrue`
`\citetrackerfalse`

and switches in § 4.6.4.

`\ifciteidem{ }{ }`

Expands to `<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 `<false>` otherwise. This command is intended for use in citation styles. If there are

any `refsection` environments in the document, the tracking is local to these environments. Note that the ‘idem’ tracker needs to be enabled explicitly with the package op-

tion `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 `<false>`. Also see

`\citetrackertrue`
`\citetrackerfalse`
and
in § 4.6.4.

`\ifopcit{ }{ }`

`\ifciteibid`

This command is similar to `\ifciteidem` except that it expands to `<true>` if the entry currently being processed is the same as the last one *by this author or editor*. Note that the ‘opcit’ tracker needs to be enabled explicitly with the package option

`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 `<false>`. Also see the

`\citetrackertrue`
`\citetrackerfalse`
and
switches in § 4.6.4.

`\ifloccit{ }{ }`

`\ifopcit`

This command is similar to

except that it also compares the $\langle postnote \rangle$ arguments and expands to $\langle true \rangle$ only if they match and are numerical (in the sense of

`\ifnumerals`

from

`\ifloccit`

§ 4.6.2), i. e.,

will

`true`

yield `true` if the citation refers to the same page cited before.

Note that the ‘loccit’ tracker needs to be enabled explicitly with the package option

`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 $\langle false \rangle$. Also see the

`\citetrackertrue`

`\citetrackerfalse`

and

switches in § 4.6.4.

`\iffirstonpage`

`{ }{ }`

The behavior of this command is responsive to the package option

`pagetracker`.

If the

`page`

option is set to `page`, it expands to $\langle true \rangle$ if the current item is the first one on the page, and to $\langle false \rangle$ otherwise. If the option is

set to `spread`,

it expands to $\langle true \rangle$ if the current item is the first one on the double-page spread, and to $\langle false \rangle$ otherwise. If

the page tracker is disabled, this test always yields $\langle false \rangle$. Depending on the context, the ‘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 $\langle true \rangle$ even if there is a citation in the body text prior to the footnote. Also see the

$\backslash\text{pagetrackertrue}$
 $\backslash\text{pagetrackerfalse}$
 and
 switches in § 4.6.4.

$\backslash\text{ifsamepage}\{ \langle instance\ 1 \rangle \} \{ \langle instance\ 2 \rangle \} \{ \langle true \rangle \}$

This command expands to $\langle true \rangle$ if two instances of a reference are located on the same page or double-page spread, and to $\langle false \rangle$ 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 instcount counter, see § 4.10.5. The behavior of this command is responsive to the package op-

tion pagetracker .

If this option is set to spread ,

$\backslash\text{ifsamepage}$ is in fact an ‘if same spread’ test. If the page tracker is disabled, this test always yields $\langle false \rangle$. The arguments $\langle instance\ 1 \rangle$ and $\langle instance\ 2 \rangle$ are treated as integer expressions in the

sense of e-TeX’s $\backslash\text{numexpr}$. This implies that it is possible to make calculations within these arguments, for example:

$\backslash\text{ifsamepage}\{ \backslash\text{valu}$

Note that `\value` is not prefixed by

`\the` and that the subtraction is included in the second argument in the above example. If $\langle instance\ 1 \rangle$ or $\langle instance\ 2 \rangle$ is an invalid number (for example, a negative one), the test yields $\langle false \rangle$. 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` $\{ \langle string \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Executes $\langle true \rangle$ if the $\langle string \rangle$ is a positive integer, and $\langle false \rangle$ otherwise. This command is robust.

`\ifnumeral` $\{ \langle string \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Executes $\langle true \rangle$ if the $\langle string \rangle$ is an Arabic or Roman numeral, and $\langle false \rangle$ otherwise. This command is robust. See also

`\DeclareNumChars`
and `\NumCheckSetup`

in § 4.6.4.

`\ifnumerals` $\{ \langle string \rangle \} \{ \langle true \rangle \} \{ \langle false \rangle \}$

Executes $\langle true \rangle$ if the $\langle string \rangle$ is a range or a list of Arabic or Roman numerals, and $\langle false \rangle$ otherwise. This command is robust. In contrast to

`\ifnumeral`, it will also execute $\langle true \rangle$ with arguments like “52–58”, “14/15”, “1, 3, 5”, and so on. See also

`\DeclareNumChars`,

`\DeclareRangeChar`
`\DeclareRangeComm`
`\NumCheckSetup`

and
in § 4.6.4.

`\ifpages`{_{⟨string⟩}}{_{⟨true⟩}}{_{⟨false⟩}}

Similar to

`\ifnumerals`,
`\DeclarePageCor`

but also considers
from § 4.6.4.

`\iffieldint`{_{⟨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⟩}.

`\iffieldnum`{_{⟨field⟩}}{_{⟨true⟩}}{_{⟨false⟩}}

`\ifnumeral`, but uses

the value of a _{⟨field⟩} rather than a literal string in the test. If the _{⟨field⟩} is undefined,
it executes _{⟨false⟩}.

\iffielnums

{ $\langle field \rangle$ } { $\langle true \rangle$ } { $\langle false \rangle$ }

Similar to **\ifnumerals**, but uses the value of a $\langle field \rangle$ rather than a literal string in the test. If the $\langle field \rangle$ is undefined, it executes $\langle false \rangle$.

\iffieldpages

{ $\langle field \rangle$ } { $\langle true \rangle$ } { $\langle false \rangle$ }

Similar to **\ifpages**, but uses the value of a $\langle field \rangle$ rather than a literal string in the test. If the $\langle field \rangle$ is undefined, it executes $\langle false \rangle$.

\ifbibstring

{ $\langle string \rangle$ } { $\langle true \rangle$ } { $\langle false \rangle$ }

Expands to $\langle true \rangle$ if the $\langle string \rangle$ is a known localization key, and to $\langle false \rangle$ otherwise. The localization keys defined by default are listed in § 4.9.2. New ones may be defined with

\NewBibliographyS

\ifbibxstring

{ $\langle string \rangle$ } { $\langle true \rangle$ } { $\langle false \rangle$ }

Similar to **\ifbibstring**, but the $\langle string \rangle$ 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 localization 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`

`\DeclareBibliographyCommand`

executes a driver defined with

`\ifcitation`

any tests in the driver code will yield `\true`. See § 4.11.6 for a practical example.

`\ifbibliography`

`{ $\langle true \rangle$ }` `{ $\langle false \rangle$ }`

Expands to $\langle true \rangle$ when located in a bibliography, and to $\langle false \rangle$ otherwise. Note that this command is responsive to the outermost context in which it is used. For example, if a driver defined with

`\DeclareBibliogra`

`\DeclareCit`

executes a citation command defined with

any `\ifbibliography`

tests in the citation code will yield $\langle true \rangle$. See § 4.11.6 for a practical example.

`\ifnatbibmode`

`{ $\langle true \rangle$ }` `{ $\langle false \rangle$ }`

Expands to $\langle true \rangle$ or $\langle false \rangle$ depending on the `natbib` option from § 3.1.1.

`\ifciteindex`

`{ $\langle true \rangle$ }` `{ $\langle false \rangle$ }`

Expands to $\langle true \rangle$ or $\langle false \rangle$ depending on the `indexing` option from § 3.1.2.1.

`\ifbibindex`{ $\langle true \rangle$ }{ $\langle false \rangle$ }

Expands to $\langle true \rangle$ or $\langle false \rangle$ depending on the `indexing` option from § 3.1.2.1.

`\iffootnote`{ $\langle true \rangle$ }{ $\langle false \rangle$ }

Expands to $\langle true \rangle$ when located in a footnote, and to $\langle false \rangle$ otherwise. Note

that footnotes in `minipage` environments are considered to be part of the body text. This command will only expand to $\langle true \rangle$ in footnotes at 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 pack-

age option `citecounter`.

`context` 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` `\BiberOnly`

This counter refers to the `labelname`

list. It is set on a per-name basis. Its value is `0` if the last name is unique,

`1` if adding the other parts of the name (first name, prefix, suffix) 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, 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 **initallinit**, **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, **uniqueiname** will also be set to **0** for that name. This feature needs to be enabled explicitly with the package option **uniqueiname**. Note that the **uniqueiname** 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 \BiberOnly

This counter refers to the 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 **unique**list 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 automati-
 cally override **maxnames** /
minnames. This fea-
 ture 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**
ifthen pack-
 age. The syntax of the tests is slightly different in this case: the $\langle true \rangle$
 and $\langle false \rangle$ 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:

`\ifboolexpr{ (` `te`
`)` `an`
`or t`

`}`
`{ . . . }`
`{ . . . }`

`\ifthenelse{ } { } { }`

`ifthen` command which allows for complex tests
with boolean operators and grouping:

`\ifthenelse{ \ (`
`\)`
`\or`
`}`
`{ . . }`
`{ . . }`

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`

`{\langle name \rangle} [\langle arguments \rangle] [\langle optional \rangle] {\langle definition \rangle}`

`\newbibmacro*`

`{\langle name \rangle} [\langle arguments \rangle] [\langle optional \rangle] {\langle definition \rangle}`

`\usebibmacro`

Defines a macro to be executed via

later. The syntax of this command is very similar to

`\newcommand`

except that `\langle name \rangle` may contain characters such as numbers and punctuation marks and does not start with a backslash. The optional argument `\langle arguments \rangle` is an integer specifying the number of arguments taken by the macro. If `\langle optional \rangle` is given, it specifies a default value for the first argument of the macro, which automatically becomes an optional argument. In con-

trast to `\newcommand`,

`\newbibmacro` issues a warning message if the macro is already defined, and automatically falls back to

`\renewbibmacro`

As with `\newcommand`, the

regular variant of this command uses the `\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 `\par` tokens.

`\newbibmacro`

and

`\renewbibmacro`

are provided for convenience. Style authors are free to use

`\newcommand` or

`\def` instead. However, note that most shared definitions found

in `biblatex.def` are

defined with `\newbibmacro`, hence they must be used and modified accordingly.

`\renewbibmacro`

`{\langle name \rangle} [\langle arguments \rangle] [\langle optional \rangle] {\langle definition \rangle}`

`\renewbibmacro*`

`{\langle name \rangle} [\langle arguments \rangle] [\langle optional \rangle] {\langle definition \rangle}`

Similar to `\newbibmacro`

but redefines `\langle name \rangle`. In contrast to `\renewcommand`

`\renewbibmacro`

issues a warning message if the macro is undefined, and automatically falls back to

`\newbibmacro`.

`\providebibmacro`
`{\langle name \rangle} [\langle arguments \rangle] [\langle optional \rangle] {\langle definition \rangle}`

`\providebibmacro*`
`{\langle name \rangle} [\langle arguments \rangle] [\langle optional \rangle] {\langle definition \rangle}`

`\newbibmacro`
Similar to
but only defines `\langle name \rangle` if it is undefined. This command is similar in concept to
`\providecommand`.

`\usebibmacro{\langle name \rangle}`
`\usebibmacro*{\langle name \rangle}`

This command executes the macro `\langle name \rangle`, as defined with

`\newbibmacro`. If
the macro takes any arguments, they are simply appended after `\langle name \rangle`. The regular
variant of this command sanitizes `\langle name \rangle` while the starred variant does not.

`\savecommand{\langle command \rangle}`
`\restorecommand`
`{\langle command \rangle}`

These commands save and restore any `\langle command \rangle`, which must be a command name
starting with a backslash. Both commands work within a local scope. They are
mainly provided for use in localization 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 localization files.

`\savefieldformat`
`[<entry type>]{<format>}`
`\restorefieldformat`
`[<entry type>]{<format>}`

These commands save and restore the formatting directive *<format>*, as defined with

`\DeclareFieldForm`

Both commands work within a local scope. They are mainly provided for use in localization 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 localization 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 localization files.

\ifbibmacroundef

{*⟨name⟩*} {*⟨true⟩*} {*⟨false⟩*}

Expands to *⟨true⟩* if the bibliography macro *⟨name⟩* is undefined, and to *⟨false⟩* otherwise.

`\iffieldformatundef`

`[$\langle entry type \rangle$] { $\langle name \rangle$ } { $\langle true \rangle$ } { $\langle false \rangle$ }`

`\iflistformatundef`

`[$\langle entry type \rangle$] { $\langle name \rangle$ } { $\langle true \rangle$ } { $\langle false \rangle$ }`

`\ifnameformatundef`

`[$\langle entry type \rangle$] { $\langle name \rangle$ } { $\langle true \rangle$ } { $\langle false \rangle$ }`

Expands to $\langle true \rangle$ if the formatting directive $\langle format \rangle$ is undefined, and to $\langle false \rangle$ otherwise.

`\usedriver{ $\langle code \rangle$ }{ $\langle entrytype \rangle$ }`

Executes the bibliography driver for an $\langle entrytype \rangle$. Calling this command in the $\langle loopcode \rangle$ 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 $\langle code \rangle$ argument. This argument is executed inside

`\usedriver`

the group in which 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`

`{\langle name \rangle}{\langle text \rangle}`

A wrapper for `hyperref`'s

`\hypertarget`

command. The `\langle name \rangle` is the name of the anchor, the `\langle text \rangle` is arbitrary printable text or code which serves as an anchor. If there are

any `refsection` environments in the document, the `\langle name \rangle` is local to the current environ-

ment. If the `hyperref` package

option is disabled or the `hyperref`

package has not been loaded, this command will simply pass on its `\langle text \rangle` argument. See also the formatting directive

`\bibhypertarget`

in § 4.10.4.

`\bibhyperlink`

`{\langle name \rangle}{\langle text \rangle}`

A wrapper for `hyperref`'s

`\hyperlink`

command. The `\langle name \rangle` is the name of an anchor defined with

`\bibhypertarget`,

the $\langle text \rangle$ is arbitrary printable text or code to be transformed into a link. If

there are any **refsection** environments in the document, the $\langle name \rangle$ 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 $\langle text \rangle$ argument. See also the formatting direc-

tive **bibhyperlink**

in § 4.10.4.

\bibhyperref

$[\langle entrykey \rangle] \{ \langle text \rangle \}$

Transforms $\langle text \rangle$ into an internal link pointing to $\langle entrykey \rangle$ in the bibliography. If $\langle entrykey \rangle$ 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 $\langle entrykey \rangle$ 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 $\{ \langle true \rangle \} \{ \langle false \rangle \}$

hyperref

Expands to $\langle true \rangle$ if the package option is enabled (which implies that the

hyperref package has been loaded),

and to $\langle false \rangle$ 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 $\langle text \rangle$ starts with a control sequence, nothing is capitalized. This command is robust.

\MakeSentenceCase

$\{ \langle text \rangle \}$

\MakeSentenceCase*

$\{ \langle text \rangle \}$

Converts its $\langle text \rangle$ 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. It only con-

verts the $\langle text \rangle$ to sentence case if the **langid** field is undefined or if it holds a language declared with

\DeclareCaseLangs

(see below).³⁴ Otherwise, the $\langle text \rangle$ is not altered in any way. It is recommended to use

³⁴By default, converting to sentence case is enabled for the following language identifiers:

american, british,
canadian, en-
glish, australian,
newzealand

as the aliases **USenglish** as well

and **UKenglish**. Use

`\MakeSentenceCase`

rather than the regular variant in formatting directives. Both variants support the

`bib` files that anything wrapped in a pair of curly braces is not modified when changing the case. For example:

`\MakeSentenceCase`
`\MakeSentenceCase`

would yield:

An introduction t
An introduction t

`bib`

In files designed with traditional BibTeX in mind, it has been fairly common to only wrap single letters in braces to prevent case-changing:

`title = {An Intro`

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.

`\mkpageprefix`

`[⟨pagination⟩][⟨postpro⟩]{⟨text⟩}`

This command is intended for use in field formatting directives which format the page numbers in the `⟨postnote⟩` argument of citation commands and

`\DeclareCaseLangs`

to extend or change this list.

the **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 **pagination** or

bookpagination,

with **pagination** being the default. The spacing between the prefix and the $\langle text \rangle$ may be modified by

redefining **\ppspace**. The default is an unbreakable interword space. See §§ [2.3.10](#) and [3.12.3](#) for further details. See also

\DeclareNumChars,

\DeclareRangeChar

\DeclareRangeComm

and **\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:

\DeclareFieldForm

\DeclareFieldForm

The optional argument **pagination**

in the first example is omissible.

`\mkpagetotal`

`[
 \langle pagination \rangle
]
[
 \langle postpro \rangle
]
{
 \langle text \rangle
}`

This command is similar to

`\mkpageprefix` `pagetotal`

except that it is intended for the

field of bibliography entries, i.e., it will print “123 pages” rather than “page 123”. The optional \langle pagination \rangle argument defaults to

`bookpagination`

The spacing inserted between the pagination suffix and the \langle text \rangle may be modified

by redefining the macro

`\ppspace`

. 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:

`\DeclareFieldForm`

`bookpagination`

The optional argument
is omissible in this case.

`\mkcomprange`

`[
 \langle postpro \rangle
]
{
 \langle text \rangle
}`

`\mkcomprange*`

`[
 \langle postpro \rangle
]
{
 \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

$\backslash mkcomprange$ by ad-

justing the LaTeX counters \mincomprange ,

\maxcomprange , and

\mincompwidth ,

as illustrated in table 10. The default settings are 10,

100000, and 1, respectively. This means that the command tries to compress as much as possible by default.

$\backslash setcounter$ Use to adjust the parameters. The scanner recognizes

$\backslash bibrangedash$

and hyphens as range dashes. It will normalize the dash by replacing any number of consecutive hyphens with

$\backslash bibrangedash$.

$\backslash bibrangesep$ Lists of ranges delimited with

(Biber³⁵) or commas/semicolons (BibTeX) are also supported. With Biber, The scanner will normalise any comma or semicolons surrounded by optional space by replacing them with

$\backslash bibrangesep$.

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

³⁵Biber will always convert commas/semicolon multi-range separators into

$\backslash bibrangesep$ so that it can be controlled in the style.

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

$\backslash mkcomprange$ with

other formatting macros which also need to parse their $\langle text \rangle$ argument, such as

$\backslash mkpageprefix$.

Simply nesting these commands will not work as expected. Use the $\langle postpro \rangle$ argument to set up the processing chain as follows:

$\backslash DeclareFieldForm$

Note that $\backslash mkcomprange$

is executed first, using $\backslash mkpageprefix$

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:

$\backslash mkcomprange[\backslash mkp$
 $\backslash mkcomprange*[\backslash mk$

will output:

pp. 5, 123-9, 423
 p. 5, pp. 123-9,

`\mkfirstpage`

`[⟨postpro⟩]{⟨text⟩}`

`\mkfirstpage*`

`[⟨postpro⟩]{⟨text⟩}`

This command, which is intended for use in field formatting directives, will parse its *⟨text⟩* argument for page ranges and print the start page of the range only. The scanner recognizes

`\bibrangedash`

and hyphens as range dashes. Lists of ranges delimited with

`\bibrangessep`

(Biber³⁶) or commas/semicolons (BibTeX) 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 *⟨postpro⟩* argument specifies a macro to be used for post-processing the *⟨text⟩*. See

`\mkcomprange`

on how to use this argument. The starred version of this command differs from the regular one in the way the *⟨postpro⟩* argument is applied to a list of values. For example:

`\mkfirstpage[\mkp`
`\mkfirstpage*[\mk`

will output:

³⁶Biber will always convert commas/semicolon multi-range separators into `\bibrangessep` so that it can be controlled in the style.

pp. 5, 123, 423
p. 5, p. 123, p.

`\rangelen{<text>}`

This command will parse its argument as a range and return the length of the range.
It will return 0 for open-ended ranges.

```
\rangelen{5-10} r
\rangelen{-10} re
\rangelen{5-} ret
\rangelen{5} retu
```

This can be used as part of tests in styles which require, for example, ‘f’ as a suffix for ranges of only two pages, for example, when a page range ‘36-37’ should be printed as ‘36f’. This could be done using

`\ifnumcomp`:

`\ifnumcomp{\range`

`\DeclareNumChars`
`{` *<characters>* `}`

`\DeclareNumChars*`
`{` *<characters>* `}`

This command configures the `\ifnumeral`,

`\ifnumerals`, and

`\ifpages` tests from § 4.6.2. The setup will

also affect `\iffielddnum`,

`\iffielddnums`,

`\iffieldpages`

as well as `\mkpageprefix`

and `\mkpagetotal`.

The *<characters>* 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`

`{
 $\langle characters \rangle$
}`

`\DeclareRangeChars*`

`{
 $\langle characters \rangle$
}`

This command configures the

`\ifnumerals`

and

`\ifpages`

tests from § 4.6.2.

The setup will also affect

`\iffielddnums`

and

`\iffieldpages`

as well as

`\mkpageprefix`

and

`\mkpagetotal`

The $\langle characters \rangle$ 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:

`\DeclareRangeChar`

This means that strings like ‘3–5’, ‘35+’, ‘8/9’ and so on will be considered as a

`\ifnumerals`

range by

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 3.12.3 for further details.

`\DeclareRangeCommands`

`{
 \langle commands \rangle
}`

`\DeclareRangeCommands`

`{
 \langle commands \rangle
}`

`\DeclareRange`

This command is similar to

except that the \langle commands \rangle 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:

`\DeclareRangeComm`

See also §§ 2.3.10 and 3.12.3 for further details.

`\DeclarePageCommands`

`{
 \langle commands \rangle
}`

`\DeclarePageCommands>`

`{
 \langle commands \rangle
}`

`\DeclareRange`

This command is similar to

except that it only affects the `\ifpages`
 and `\iffieldpages`
 tests but not `\ifnumerals`
 and `\iffieldnums`.
 The default setting is:

`\DeclarePageComma`

`\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`,
`\iffieldnums`,
`\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 3.12.3 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

`babelpolyglossia`

languages 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`

`USenglish, UKeng`

`babelpolyglossia`

See the manuals and table 2 for a list of languages identifiers.

`\BibliographyWarning`

`{
 $\langle message \rangle$
}`

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 $\langle message \rangle$ 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.

`\RequireBiber`

`[
 $\langle severity \rangle$
]`

This command is intended for use in `cbx/bbx`

files and in the `@preamble` of

`bib`

files. It checks the selected backend and warns if it is not Biber. The optional $\langle severity \rangle$ argument is an integer specifying the severity. The value 1 triggers an informational message stating that Biber is recommended; 2 triggers

a warning stating that Biber is required and the style/`bib` file may not work properly; 3 triggers an error stating that Biber is strictly required and

the style/**bib** file will not work at all with any other backend. If

\RequireBiber

is used multiple times, the highest *<severity>* takes precedence.

cbxbbx files on the one hand and the

@preamble snippets of all

bib files on the other are tracked separately. If the optional *<severity>* argument is omitted, the default severity is 2 (warning).

\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

\DeclareCiteC

in the *<precode>* argument of

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`,

`\ifciteibid`, 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`

`\DeclareCiteC`

in the *⟨precode⟩* argument of

See § 4.3.1 for details. Note that these commands have no effect if tracking has been disabled globally.

`\backtrackerttrue`

`\backtrackertfalse`

`backref`

These commands activate or deactivate the 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

`\backtrackertfalse`

`\DeclareCiteC`

in the *⟨precode⟩* argument of

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 localization strings, see

\DeclareCapitalPu

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 in-

serting **\newblock** and

\newunit commands at suitable places and

\finentry at the very end (see § 4.2.3

for an example). See also § 4.11.7 for some practical hints.

\newblock Records the end of a block. This command does not print anything, it merely marks the end of the block. The block delimiter

\newblockpunct

will be inserted by a subsequent `\printtext`,
`\printfield`,
`\printlist`,
`\printnames`, or
`\bibstring` command. You may
`\newblock` at suitable places
 use without having to worry about spurious blocks. A new block will only be started by
`\printfield` (or
 the next similar) command if this command prints anything. See § 4.11.7 for further details.

`\newunit` Records the end of a unit and puts the default delim-
`\newunitpunct` iter
 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 `\printtext`,
`\printfield`,
`\printlist`,
`\printnames`, or
`\bibstring` command. You

may use `\newunit` after commands like `\printfield` without having to worry about spurious punctuation and whitespace. The buffer will only be inserted by the next `\printfield` or similar command if *both* fields are non-empty. This also applies to `\printtext`, `\printlist`, `\printnames`, and `\bibstring`. See § 4.11.7 for further details.

`\finentry` Inserts `\finentryp`
 This command should be used at the very end of every bibliography entry.

`\setunit{ }` *⟨punctuation⟩*
`\setunit*{ }` *⟨punctuation⟩*

The `\setunit` command is similar to `\newunit` except that it uses *⟨punctuation⟩* instead of `\newunitpunct`. The starred variant differs from the regular version in that it checks

if the last `\printtext`,
`\printfield`,
`\printlist`,
`\printnames`, or
`\bibstring` command did
 actually print anything. If not, it does nothing.

`\printunit{<punctuation>}`
`\printunit*{<punctuation>}`

The `\printunit` com-
 mand is similar to `\setunit`
 except that *<punctuation>* persists in the buffer until the
 next `\printtext`,
`\printfield`,
`\printlist`,
`\printnames`, or
`\bibstring` command.

\setpunctfont

{*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 *command* should be a text font com-

mand which takes one argument, such as **\emph** or

\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

\resetpunctfont.

If the **punctfont** package option is disabled, this command does nothing. Note that

the **\mkbibemph**,

\mkbibitalic

and **\mkbibbold** wrappers from

§ 4.10.4 incorporate this feature by default.

\resetpunctfont

This command resets the unit punctuation font defined with

\setpunctfont

before it takes effect. If the **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{ $\langle true \rangle$ }{ $\langle false \rangle$ }

Executes $\langle true \rangle$ if preceded by any punctuation mark except for an abbreviation dot, and $\langle false \rangle$ otherwise.

\ifterm{ $\langle true \rangle$ }{ $\langle false \rangle$ }

Executes $\langle true \rangle$ if preceded by a terminal punctuation mark, and $\langle false \rangle$ otherwise. A terminal punctuation mark is any punctuation mark which has been registered for automatic capitalization, either with

\DeclareCapitalPu

or by default, see § 4.7.5 for details. By default, this applies to periods, exclamation marks, and question marks.

\ifpunctmark
{ $\langle character \rangle$ }{ $\langle true \rangle$ }{ $\langle false \rangle$ }

Executes $\langle true \rangle$ if preceded by the punctuation mark $\langle character \rangle$, and $\langle false \rangle$ otherwise. The $\langle character \rangle$ 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

\DeclareNameForma

the $\langle character \rangle$ may also be an apostrophe.

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 Bibl_{at}ex switches languages. This behavior may be adjusted with

`\DeclarePunctuation`
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.

`\addnbthinspace`

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 §§ 3.9.3 and 4.10.3 for details.

`\addhighpenspace`

Adds a space penalized by the value of the

`highnamepenalty`

counter, see §§ 3.9.3 and 4.10.3 for details.

`\addlpthinspace`

Simi-

`\addlowpenspace`

lar to

but adds a breakable thin space.

`\addhpthinspace`

Simi-

`\addhighpenspace`

lar to

but adds a breakable thin space.

`\addabbrvspace`

Adds a space penalized by the value of the

abbrvpenalty

counter, see §§ 3.9.3 and 4.10.3 for details.

`\addabthinspace`

Simi-

lar to `\addabbrvspace`

but using a thin space.

`\adddotSPACE`

Executes

`\adddot`

and adds a space penalized by the value

of the `abbrvpenalty`

counter, see §§ 3.9.3 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 white-space, 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.

`\DeclareAutoPunct`

`{\langle characters \rangle}`

This command defines the punctuation marks to be considered by the citation commands as they scan ahead for punctuation. Note that $\langle characters \rangle$ is an undelimited list of characters. Valid $\langle characters \rangle$ are period, comma, semicolon, colon, exclamation and question mark. The default setting is:

`\DeclareAutoPunct`

This definition is restored automatically whenever the `autopunct` package-

`true`

age option is set to Executing

`\DeclareAutoPunct`

`autopunct=`

is equivalent to setting

`false`

, i. e., it disables this feature.

`\DeclareCapitalPunct`

`{\langle characters \rangle}`

When Biblatex inserts localization 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 localization strings to be capitalized if one of them precedes a string. Note that $\langle characters \rangle$ is an undelimited list of characters. Valid $\langle characters \rangle$ are period, comma, semicolon, colon, exclamation and question mark. The package default is:

`\DeclareCapitalPu`

Using `\DeclareCapitalP`

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

`\DefineBibliograph`
`\DeclareBibli`

(when used in the preamble) or

(when used in a localization module). See §§ 3.8 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).

`\DeclarePunctuationPa`
`{ } { }`
`\langle identifier \rangle \langle characters \rangle`

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:

`\DeclarePunctuati`

The `\langle identifier \rangle` 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

`\langle identifier \rangle` strings are `dot`, `comma`,
`semicolon`,
`colon`, `period`,

exclamquestion.

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:

`\DeclarePunctuati`
`\DeclarePunctuati`
`\DeclarePunctuati`
`\DeclarePunctuati`
`\DeclarePunctuati`
`\DeclarePunctuati`
`\DeclarePunctuati`

Since this feature is language specific,

must be used in the argument to

(when used in the preamble) or

(when used in a localization module). See §§ 3.8 and 4.9 for details. Note that some localization modules may use a setup which is different from the package default.³⁷

³⁷As of this writing, the `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`

`\DeclareQuotePunctuation`

Executing

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

`\DefineBibliography`

`\DeclareBibliography`

(when used in the preamble) or

(when used in a localization module). See §§ 3.8 and 4.9 for details. See also § 3.10.1.

`\uspunctuation`

A shorthand using the lower-level commands

`\DeclareQuotePunctuation`

`\DeclarePunctuation`

and

to activate ‘American-style’ punctuation. See § 3.10.1 for details. This shorthand is provided for convenience only. The effective settings are applied by the lower-level commands.

`\stdpunctuation` Un-

does 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 § 3.10.1 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 localization 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 localization 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 localization 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 localization 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 localized term.

\bibstring[*<wrapper>*]{*<key>*}

Prints the localization 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 localization 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.

\emph. This is intended for font commands such as

\biblstring[*<wrapper>*]{*<key>*}

Similar to **\bibstring**

but always prints the long string, ignoring the
abbreviate option.

`\bibstring[<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 `\bibsstring`
but the term is always capitalized.

`\bibucstring`

`[⟨wrapper⟩]{⟨key⟩}`

Similar to `\bibstring` but the
whole term is uppcased.

`\bibuc\string`

`[⟨wrapper⟩]{⟨key⟩}`

Similar to `\bib\string`
but the whole term is uppcased.

`\bibucsstring`

`[⟨wrapper⟩]{⟨key⟩}`

Similar to `\bibsstring`
but the whole term is uppcased.

`\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.

`\bibxstring{ <key> }`

`\bibstring`

A simplified but expandable version of

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 localization string is required in a test.

`\bibxstring`

`[<wrapper>]{<key>}`

Similar to `\bibxstring`
but always uses the long string, ignoring the
`abbreviate` option.

`\bibxsstring`

`[<wrapper>]{<key>}`

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 `<wrapper>` argument in the localization string commands above.

4.9 Localization Modules

A localization 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

`lbx`. The base name of the file must be a language name known to the
`babelpolyglossia`
`lbx` packages. The files may also be used to map

babelpolyglossia

language names to the backend modules of the Bibl_atex package. All localization 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 localization commands were already introduced in

§ 3.8. When used in **l_bx** files, however, the syntax of localization commands is different from the user syntax in the preamble and the configuration file. When used in localization files, there is no need to specify the *⟨language⟩* because the mapping of strings to a language is already provided by the name of the

l_bx file.

```
\DeclareBibliographyS  
{  
  ⟨definitions⟩  
}
```

This command is only available in **l_bx** files. It is used to define localization strings. The *⟨definitions⟩* consist of *⟨key⟩* **=** *⟨value⟩* 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 **l_bx** 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:

```
\DeclareBibliogra  
bibliography      =  
shorthands        =
```

```
editor
editors
}
```

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

```
abbreviate
```

the package option

(see § 3.1.2.1), Biblatex selects one expression when loading the

```
lbx
inherit
```

file. There is also a special key named 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:

```
\DeclareBibliography
inherit
january
}
```

The above examples are slightly simplified. Real localization files should use the punctuation and formatting commands discussed in §§ 4.7.3 and 3.9 instead of literal punctuation. Here is an excerpt from a real localization file:

```
bibliography
shorthands
```

editor
editors
byeditor
mathesis

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 Biblatex package will automatically capitalize the first word when required at the beginning of a sentence, see

`\DeclareCapitalPu`

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).

`\InheritBibliographyS`
`{language}`

This command is only available in `laTeX` files. It copies the localization strings for *language* to the current language, as specified by the name of the

`laTeX` file.

`\DeclareBibliographyE`
`{code}`

This command is only available in `laTeX` files. It is used to adapt language specific features such as the date format and ordinals. The *code*, which

may be arbitrary LaTeX code, will usually consist of redefinitions of the formatting commands from § 4.10.2.

\UndeclareBibliography
{_{*code*}}

This command is only available in **l_bx** files. It is used to restore any formatting commands modified with

\DeclareBibliogra

If a redefined command is included in § 4.10.2, there is no need to restore its previous definition since these commands are localized by all language modules anyway.

\InheritBibliographyE
{_{*language*}}

This command is only available in **l_bx** files. It copies the bibliography extras for *language* to the current language, as specified by the name of the

l_bx file.

\DeclareHyphenationEx
{_{*text*}}

This command corresponds to **\DefineHypher**

from § 3.8. The difference is that it is only available in **l_bx** files and that the *language* argument is omitted. The hyphenation exceptions will affect the

language of the **l_bx** file currently being processed.

\DeclareRedundantLang

```
{ \language, language, ... } { \langid, langid, ... }
```

This command provides the language mappings required by

the `clearlang` option from § 3.1.2.1. The `\language` is the string given in

the `language` field (with-

out the optional `lang` prefix); `\langid` is

`babelpolyglossia`'s language identifier, as given in the optional argument of

```
\usepackage
```

when loading `babel` or the argument of

```
\setdefaultlanguage
```

or `\setotherlanguage`

when using `polyglossia`.

```
ltx
```

This command may be used in `ltx` files or in the document preamble. Here are some examples:

```
\DeclareRedundant
```

```
\DeclareRedundant
```


`\DeclareRedundant` `canadian,`

Note that this feature needs to be enabled globally with the

`clearlang`

option from § 3.1.2.1. If it

is disabled, all mappings will be ignored. If the `\langid` parameter is blank, Biblatex will clear the mappings for the corresponding `\language`, i. e., the feature will be disabled for this `\language` only.

`\DeclareLanguageMapp`

`{\language}{\file}`

This command maps a `babelpolygloss`
`lbx`

language identifier to an file. The
`\language` must be a language name known to the

`babelpolyglossia`

package, i. e., one of the identifiers listed in table 2. The `\file` argument is the

`lbx` name of an alternative file without the

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.

`\NewBibliographyStrin`

`{\key}`

This command, which may be used in the preamble (including `cbx`

and **bbx** files) as well as in **lbx** files, declares new localization strings, i. e., it initializes a new $\langle key \rangle$ to be used in the $\langle definitions \rangle$ of

\DefineBibliograph

or **\DeclareBibliogra**

The $\langle key \rangle$ argument may also be a comma-separated list of key names. When used

in an **lbx**, the $\langle key \rangle$ 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 localization keys in this section are defined by default and covered by the localization files which come with Bibl_{at}ex. 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. Bibl_{at}ex 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 Bibl_{at}ex’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

\biblistname.

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

compiler The term ‘compiler’, referring to an editor whose task is to compile a work.

compilers The plural form of

founder The term ‘founder’, referring to a founding editor.

founders The plural form of

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

redactor The term ‘redactor’, referring to a secondary editor.

redactors The plural form of

reviser The term ‘reviser’, referring to a secondary editor.

revisers The plural form of

collaborator A term like ‘collaborator’, ‘collaboration’, ‘cooperator’, or ‘cooperation’, referring to a secondary editor.

collaborators The plural form of

translator The term ‘translator’.

translators The plural form of

commentator The term ‘commentator’, referring to the author of a commentary to a work.

commentators The plural form of

annotator The term ‘annotator’, referring to the author of annotations to a work.

annotators The plural form of

editor.

compiler.

founder.

continuator.

redactor.

reviser.

collaborator.

translator.

commentators.

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’.

editor
translator / **editortr**
Used if / are identical.

editortr The plural form of **editortr**.

editor
commentator / **editorco**
Used if / are identical.

editorco The plural form of **editorco**.

editor
annotator / **editoran**
Used if / are identical.

editoran The plural form of **editoran**.

editor
introduction / **editorin**
Used if / are identical.

editorin The plural form of **editorin**.

editor /
foreword are identical.

editorfo .
The plural form of

editor /
aftword are identical.

editoraf .
The plural form of

editortranslator /
Keys for
<role> combinations:

editor /
translator /
commentator are
identical.

editortrco .
The plural form of

editor /
translator /
annotator are identical.

editortran .
The plural form of

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.

editorcommentator
Keys for
(role) combinations:

editorcoin Used if editor,
commentator,
introduction are
identical.

editorcoin.
editorscoin The plural form of

editor,
editorcofo Used if
commentator,
foreword are identical.

editorcofo.
editorscofo The plural form of

editor,
editorcoaf Used if
commentator,
aftword are identical.

editorcoaf.
editorscoaf The plural form of

editorannotator,
Keys for
<role> combinations:

editor,
editoranin Used if
annotator,
introduction are
identical.

editoranin.
editorsanin The plural form of

editor,
editoranfo Used if
annotator,
foreword are identical.

editorsanfo The plural form of editoranfo.

editoranaf Used if editor/
annotator/
aftword
are identical.

editorsanaf The plural form of editoranaf.

editortranslator
commentator
Keys for /
binations: /*(role)* com-

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.

editorannotator,
commentator

Keys for

/⟨role⟩ com-

binations:

editortranin Used if

editor,
annotator,
commentator,
introduction

are

identical.

editorstranin The plural form of

editortranin.

editortranfo Used if

editor,
annotator,

commentator,
foreword

are identical.

editorstranfo The plural form of

editorstranfo

editortranaf

Used if

editor,
annotator,
commentator,
aftword

are identical.

editorstranaf The plural form of

editorstranaf

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

translatoran.

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
 translatorcoaf.
 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	translatoranaf	

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 <i><name></i> '.
byeditor	The expression 'edited by <i><name></i> '.
bycompiler	The expression 'compiled by <i><name></i> '.
byfounder	The expression 'founded by <i><name></i> '.
bycontinuator	The expression 'continued by <i><name></i> '.
byredactor	The expression 'redacted by <i><name></i> '.
byreviser	The expression 'revised by <i><name></i> '.
byreviewer	The expression 'reviewed by <i><name></i> '.
bycollaborator	An expression like 'in collaboration with <i><name></i> ' or 'in cooperation with <i><name></i> '.
bytranslator	The expression 'translated by <i><name></i> ' or 'translated from <i><language></i> by <i><name></i> '.
bycommentator	The expression 'commented by <i><name></i> '.
byannotator	The expression 'annotated by <i><name></i> '.

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.

editortranslator
Keys for
(*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 /
aftword are identical.

editor,commentator
Keys for
<role> combinations:

byeditorcoin Used if editor /
commentator /
introduction are
identical.

byeditorcofo Used if editor /
commentator /
foreword are identical.

byeditorcoaf Used if editor /
commentator /
aftword are identical.

editorannotator /
Keys for
<role> combinations:

byeditoranin Used if editor /
annotator /
introduction are
identical.

byeditoranfo Used if editor /
annotator /
foreword are identical.

byeditoranaf Used if editor,
annotator,
aftword are identical.

Keys for editor, translator
commentator /⟨role⟩ com-
binations:

byeditortrcoin Used if editor,
translator,
commentator,
introduction are
identical.

byeditortrcofo Used if editor,
translator,
commentator,
foreword are identical.

byeditortrcoaf Used if editor,
translator,
commentator,
aftword are identical.

editor translator
Keys for /
annotator /⟨role⟩ combinations:

byeditortranin Used if editor,
annotator,
commentator,
introduction are
identical.

byeditortranfo Used if editor,
annotator,
commentator,
foreword are identical.

byeditortranaf Used if editor,
annotator,
commentator,
aftword are identical.

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’.

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 /
aftword are identical.

Keys for translator /
commentator / $\langle role \rangle$
combinations:

bytranslatorcoin Used if translator /
commentator /
introduction are
identical.

bytranslatorcofo Used if translator,
commentator,
foreword are identical.

bytranslatorcoaf Used if translator,
commentator,
aftword 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,
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').

withcommentator	The expression 'with a commentary by <i><name></i> '.
withannotator	The expression 'with annotations by <i><name></i> '.
withinintroduction	The expression 'with an introduction by <i><name></i> '.
withforeword	The expression 'with a foreword by <i><name></i> '.
withafterword	The expression 'with an afterword by <i><name></i> '.

4.9.2.9 Supplementary Material

commentary	The term 'commentary'.
annotations	The term 'annotations'.
introduction	The term 'introduction'.
foreword	The term 'foreword'.
afterword	The term 'afterword'.

4.9.2.10 Publication Details

volume	The term 'volume', referring to a book.
volumes	The plural form of volume .
involumes	The term 'in', as used in expressions like 'in <i><number of volumes></i> volumes'.
jourvol	The term 'volume', referring to a journal.
jourser	The term 'series', referring to a journal.
book	The term 'book', referring to a document division.
part	The term 'part', referring to a part of a book or a periodical.
issue	The term 'issue', referring to a periodical.
newseries	The expression 'new series', referring to a journal.
oldseries	The expression 'old series', referring to a journal.
edition	The term 'edition'.
in	The term 'in', referring to the title of a work published as part of another one, e. g., ' <i><title of article></i> in <i><title of journal></i> '.
inseries	The term 'in', as used in expressions like 'volume <i><number></i> in <i><name of series></i> '.
ofseries	The term 'of', as used in expressions like 'volume <i><number></i> of <i><name of series></i> '.
number	The term 'number', referring to an issue of a journal.
chapter	The term 'chapter', referring to a chapter in a book.
version	The term 'version', referring to a revision number.
reprint	The term 'reprint'.
reprintof	The expression 'reprint of <i><title></i> '.

reprintas	The expression ‘reprinted as <i><title></i> ’.
reprintfrom	The expression ‘reprinted from <i><title></i> ’.
translationof	The expression ‘translation of <i><title></i> ’.
translationas	The expression ‘translated as <i><title></i> ’.
translationfrom	The expression ‘translated from [the] <i><language></i> ’.
reviewof	The expression ‘review of <i><title></i> ’.
origpubas	The expression ‘originally published as <i><title></i> ’.
origpubin	The expression ‘originally published in <i><year></i> ’.
astitle	The term ‘as’, as used in expressions like ‘published by <i><publisher></i> as <i><title></i> ’.
bypublisher	The term ‘by’, as used in expressions like ‘published by <i><publisher></i> ’.

4.9.2.11 Publication State

inpreparation	The expression ‘in preparation’ (the manuscript is being prepared for publication).
submitted	The expression ‘submitted’ (the manuscript has been submitted to a journal or conference).
forthcoming	The expression ‘forthcoming’ (the manuscript has been accepted by a press or journal).
inpress	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).
prepublished	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

page	The term ‘page’.	page
pages	The plural form of	pages
column	The term ‘column’, referring to a column on a page.	column
columns	The plural form of	columns
section	The term ‘section’, referring to a document division (usually abbreviated as §).	section
sections	The plural form of (usually abbreviated as §§).	sections
paragraph	The term ‘paragraph’ (i. e., a block of text, not to be confused with	paragraph
paragraphs	The plural form of	paragraphs
verse	The term ‘verse’ as used when referring to a work which is cited by verse numbers.	

verse
verses The plural form of

line The term 'line' as used when referring to a work which is cited by line numbers.

line
lines The plural form of

4.9.2.13 Types

The following keys are typically used in the
field of
@thesis,
@report, **@misc**,

and other entries:

mathesis An expression equivalent to the term 'Master's thesis'.

phdthesis The term 'PhD thesis', 'PhD dissertation', 'doctoral thesis', etc.

candthesis An expression equivalent to the term 'Candidate thesis'. Used for 'Candidate' degrees that have no clear equivalent to the Master's or doctoral level.

techreport The term 'technical report'.

resreport The term 'research report'.

software The term 'computer software'.

datacd The term 'data CD' or 'CD-ROM'.

audiocd The term 'audio CD'.

4.9.2.14 Miscellaneous

nodate The term to use in place of a date when there is no date for an entry e. g., 'n.d.'

and The term 'and', as used in a list of authors or editors, for example.

andothers The expression 'and others' or 'et alii', used to mark the truncation of a name list.

andothers
andmore Like but used to mark the truncation of a literal list.

4.9.2.15 Labels

The following strings are intended for use as labels, e. g., 'Address: *<url>*' or 'Abstract: *<abstract>*'.

url The term 'address' in the sense of an internet address.

urlfrom An expression like 'available from *<url>*' or 'available at *<url>*'.

urlseen	An expression like ‘accessed on <i><date></i> ’, ‘retrieved on <i><date></i> ’, ‘visited on <i><date></i> ’, referring to the access date of an online resource.
file	The term ‘file’.
library	The term ‘library’.
abstract	The term ‘abstract’.
annotation	The term ‘annotations’.

4.9.2.16 Citations

Traditional scholarly expressions used in citations:

idem	The term equivalent to the Latin ‘idem’ (‘the same [person]’).
idemsf	The feminine singular form of idem .
idems	The masculine singular form of idem .
idemsn	The neuter singular form of idem .
idempf	The feminine plural form of idem .
idempm	The masculine plural form of idem .
idempn	The neuter plural form of idem .
idempp	The plural form of idem suitable for a mixed gender list of names.
ibidem	The term equivalent to the Latin ‘ibidem’ (‘in the same place’).
opcit	The term equivalent to the Latin term ‘opere citato’ (‘[in] the work [already] cited’).
loccit	The term equivalent to the Latin term ‘loco citato’ (‘[at] the place [already] cited’).
confer	The term equivalent to the Latin ‘confer’ (‘compare’).
sequens	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’).
sequentes	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 sqq.’ or ‘25 ff.’).
passim	The term equivalent to the Latin ‘passim’ (‘throughout’, ‘here and there’, ‘scatteredly’).

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,

thiscite in: Book, pp. 45–61, 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’.
langfinnish	The language ‘Finnish’.
langfrench	The language ‘French’.
langgerman	The language ‘German’.
langgreek	The language ‘Greek’.
langitalian	The language ‘Italian’.
langlatin	The language ‘Latin’.
langnorwegian	The language ‘Norwegian’.
langpolish	The language ‘Polish’.
langportuguese	The language ‘Portuguese’.
langrussian	The language ‘Russian’.
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’.
fromczech	The expression ‘from [the] Czech’.
fromcroatian	The expression ‘from [the] Croatian’.
fromdanish	The expression ‘from [the] Danish’.
fromdutch	The expression ‘from [the] Dutch’.
fromenglish	The expression ‘from [the] English’.
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’.
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’.
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 localized 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 localized 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’.

4.10 Formatting Commands

This section corresponds to § 3.9 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 § 3.9.1 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
\mk...

that all commands starting with **\mk...** take one mandatory argument.

`\bibnamedelima`

`\BiberOnlyMark`

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 (§ 3.9.3). Please refer to § 3.12.4 for further details.

`\bibnamedelimb`

`\BiberOnlyMark`

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 (§ 3.9.3). Please refer to § 3.12.4 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

`useprefix =`

`true`.

The default definition is

`\addhighpenspace`,

i. e., a space penalized by the value of the

`highnamepenalty`

counter (§ 3.9.3). Please refer to § 3.12.4 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 (§ 3.9.3). Please refer to § 3.12.4 for further details.

`\bibnamedelimi`

`\BiberOnlyMark`

`\bibnamedelima`

This delimiter replaces 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`

`\BiberOnlyMark`

The punctuation inserted automatically by the backend after all initials unless

`\bibinithyphendel`

applies. The default definition is a period

`\adddot` (). Please refer to § 3.12.4 for further details.

`\bibinitdelim`

`\BiberOnlyMark`

The spacing inserted automatically by the backend between multiple initials unless

`\bibinithyphendel`

applies. The default definition is an unbreakable interword space. Please refer to § 3.12.4 for further details.

`\bibinithyphendelim`

`\BiberOnlyMark`

The punctuation inserted automatically by the backend between the initials of hyphenated name parts, replacing

`\bibinitperiod`

`\bibinitdelim`

and

The default definition is a period followed by an unbreakable hyphen. Please refer to § 3.12.4 for further details.

`\bibindexnamedelima`

`\bibnamedelima`

Replaces
in the index.

`\bibindexnamedelimb`

Replaces
in the index.

`\bibnamedelim`

`\bibindexnamedelim`

Replaces
in the index.

`\bibnamedelim`

`\bibindexnamedelim`

Replaces
in the index.

`\bibnamedelim`

`\bibindexnamedelim`

Replaces
in the index.

`\bibnamedelim`

`\bibindexinitperiod`

Replaces
in the index.

`\bibinitperiod`

`\bibindexinitdelim`

Replaces
in the index.

`\bibinitdelim`

`\bibindexinitthyphende`

Replaces
in the index.

`\bibinitthyphende`

`\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 § 3.12.4 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 alphabetiz-

ing in the bibliography (author or editor author, if the field is undefined). Use this separator instead of

`\newunitpunct`

at this location. The default is

i. e., it is not handled differently from regular unit punctuation but permits convenient reconfiguration.

`\newunitpunct`

`\subtitlepunct` The sep-

arator to be printed between the fields title and subtitle, booktitle and booksubtitle, as well as maintitle and

main subtitle. 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`, `@incollect`

etc. Use this separator instead of `\newunitpunct` at this location. The default definition is a colon plus an interword space.

`\bibpagespunct` The separator

to be printed before the `pages` field. Use this separator in-

stead of `\newunitpunct` at this location. The default is a comma plus an interword space.

`\bibpagerefspunct`

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.

`\revsdnamedelim`

The extra delimiter to be printed after the first name in a name list (in addition to

`\finalnamedelim`

if the first name is reversed. This command should be incorporated in all formatting directives for name lists.

`\andothersdelim`

The delimiter to be printed before the localization string

`andothers`, 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

`\multilistdelim` this command instead of before the final item in a literal list.

`\andmoredelim` The delimiter to be

printed before the localization string ‘`andmore`’,

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 `<sepcode>` argument passed to

`\DeclareCiteComma`

See § 4.3.1 for details.

`\supercitedelim` Simi-

lar to `\multinamedelim`,

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`,
`\textcite` and
but intended for
related commands (§ 3.7.2).

`\nametitledelim` The de-
limiter 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.

`\volcitedelim` The delimiter
to be printed between the volume portion and the page/text portion of
`\volcite` and related commands (§ 3.7.6).

`\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.

`\mkbibnamelast`

`{ } <text>` Formatting hook for the last name, to be used in all formatting directives for name lists.

`\mkbibnamefirst`

`{ } <text>` Similar to `\mkbibnamelast` but intended for the first name.

`\mkbibnameprefix`

`{ } <text>` Similar to `\mkbibnamelast` but intended for the name prefix.

`\mkbibnameaffix`

`{ } <text>` Similar to `\mkbibnamelast` but intended for the name affix.

`\relatedpunct`

The separator between the relatedtype bibliography localization 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.

4.10.2 Language-specific Commands

This section corresponds to § 3.9.2 in the user part of the manual. The commands discussed here are usually handled by the localization modules, but may also be redefined by users on a per-language basis. Note that all commands starting with

`\mk...` take one or more mandatory arguments.

`\bibrangedash` The language specific range dash.

`\bibdatedash` The language specific date range dash.

`\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.

`\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{integer}` Takes an integer argument and prints it as an ordinal number.

`\mkbibmascord{integer}`

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 § 3.9.3 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`.

`\biblabelsep` The horizontal
space between entries and their corresponding labels. Bibliography

styles which use `list` environments and print a la-

bel should set `\labelsep` to

`\biblabelsep` in the
definition of the respective environment.

`\bibitemsep` The vertical space
between the individual entries in the bibliography. Bib-

liography styles using `list` environments

should set `\itemsep` to

`\bibitemsep` in the
definition of the respective environment.

`\bibparsep` The vertical space be-
tween paragraphs within an entry in the bibliography. Bib-

liography styles using `list` environ-

ments should set `\parsep` to
`\bibparsep` in the definition of
the respective environment.

abbrvpenalty The penalty used by
`\addabbrvspace`,
`\addabthinspace`,
`\adddotspace`,
and
see § 4.7.4 for details.

lownamepenalty The penalty
used by `\addlowpenspace`
and `\addlpthinspace`,
see § 4.7.4 for details.

highnamepenalty The
penalty used by `\addhighpenspace`
and `\addhpthinspace`,
see § 4.7.4 for details.

biburlnumpenalty
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.

`biburlucpenalty`

Simi-

lar to `biburlnumpenalty`,

except that it will add a breakpoint after all uppercase letters.

`biburlcpenalty`

Simi-

lar to `biburlnumpenalty`,

except that it will add a breakpoint after all lowercase 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{ }`

A generic command which prints its argument as emphasized text. This is a simple wrapper around the standard

`\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 `\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 pack-

age.

also supports ‘American-style’ punctuation, see

`\DeclareQuotePunc`

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`

`\bibcloseparen`
and
must always be balanced.

`\bibopenbracket`

`\bibclosebracket`
<text>

`\mkbibbrackets`
Alternative syntax for
This will also work across groups. Note that

`\bibopenbracket`

`\bibclosebracket`
and
must always be balanced.

`\mkbibfootnote`

`{ }`
<text>

A generic command which prints its argument as a footnote. This is a wrapper around

`\footnote`
the standard LaTeX
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

`\bibfootnotewrapp`

macro introduced below.

`\mkbibfootnotetext`
`{⟨text⟩}`

Similar to `\mkbibfootnote`
but uses the
command. `\footnotetext`

`\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 `\endnote`

command provided by the `endnotes`

package as well as the `\pagenote`

command provided by the `pagenote`

package and the `memoir` class. If both com-

mands are available,
takes precedence. If no endnote support is available,

`\mkbibendnote`
`\footnote`
issues an error and falls back to

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`

`{
 $\langle text \rangle$
}`

Similar to `\mkbibendnote`

but uses the `\endnotetext` command. Please note that as of this writing, neither the

`pagenote` package nor the

`memoir` class provide a corresponding

`\pagenotetext`

command. In this case, `\mkbibendnote`

will issue an error and fall back to `\footnotetex`

`\bibfootnotewrapper`

`{
 $\langle text \rangle$
}`

An inner wrapper which encloses the $\langle text \rangle$ argument of

`\mkbibfootnote`

and `\mkbibfootnotetext`

For example, `\mkbibfootnote`
eventually boils down to this:

```
\footnote{\bibfoo
```

The wrapper ensures capitalization at the beginning of the note and adds a period at the end. The default definition is:

```
\newcommand{\bibf
```

If you don't want capitalization at the beginning or
a period at the end of the note, do not modify

```
\mkbibfootnote  
\bibfootnotewra  
but redefine  
instead.
```

```
\bibendnotewrapper  
{  
  <text>  
}
```

Similar in concept to `\bibfootnotewr`

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`

`{\integer}`

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 localization modules.

`\mkdatezeros`

`{\integer}`

This command strips leading zeros from a number or preserves them, depending on the `datezeros` package option (§ 3.1.2.1). It is intended for use in the definition of date formatting macros.

`\stripzeros`

`{\integer}`

This command strips leading zeros from a number. It is intended for date formatting and ordinals.

`shorthandwidth`

BibTeX only

A special field formatting directive which is used internally by Biblatex. When the bibliographic data is read from the

`bb\shorthand` file, Biblatex measures the values of all fields and sets the length

register `\shorthandwidth`

to the width of the widest shorthand (see § 4.10.5). In order to determine the correct width, the package considers two factors: the definition of

`\bibfont`

and this formatting directive. All styles should adjust this directive such that it corresponds to the format used in the list of shorthands.

`'labelfield'width`
`\BiberOnlyMark`

With Biber, for every field marked as a 'Label field' in the data model, a formatting directive is created as per

`shorthandwidth`
`shorthand`

above. Since is so marked in the default data model, this functionality is a superset of that described for

`shorthandwidth`.

`labelnumberwidth`

Similar to `shorthandwidth`,

but referring to the `labelnumber`

field and the length register `\labelnumberw`

Numeric styles should adjust this directive such that it corresponds to the format used in the bibliography.

`labelalphawidth`

Simi-

lar to `shorthandwidth`,

but referring to the `labelalpha`

`\labelalphawi`

field and the length register

Alphabetic styles should adjust this directive such that it corresponds to the format used in the bibliography.

`bibhyperref`

A special formatting directive

for use with `\printfield`

and `\printtext`. This directive

wraps its argument in a `\bibhyperref` command, see § 4.6.4 for details.

`bibhyperlink`

A special formatting directive for use with

`\printfield` and

`\printtext`. It wraps its argu-

ment in a `\bibhyperlink`

command, see § 4.6.4 for details. The `\langle name \rangle` argument passed to

`\bibhyperlink`

is the value of the `entrykey` field.

`bibhypertarget`

A special formatting directive for use with

`\printfield` and

`\printtext`. It wraps its argu-

ment in a `\bibhypertarget` command, see § 4.6.4 for details. The `\name` argument passed to

`\bibhypertarget` entrykey field.
is the value of the

`volcitepages` A special formatting directive which controls the format of the page/text portion in the argument of citation

commands like `\volcite`.

`volcitevolume` A special formatting directive which controls the format of the volume portion in the argument

of citation commands like `\volcite`.

`date` A special formatting directive which controls the format of

`\printdate` (§ 4.4.1). Note that the date format (long/short etc.) is controlled by the package option

`date` from § 3.1.2.1. This formatting directive only controls additional formatting such as fonts etc.

`datelabel` Similar to `date` but con-

trols the format of `\printdate` label

`urldate` Similar to `date` but controls the format of `\printurldate`.

`origdate` Similar to `date` but controls the format of `\printorigdate`.

`eventdate` Similar to `date` but controls the format of `\printeventdate`.

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.

`\shorthandwidth` BibTeX only
This length register indicates the width of the widest shorthand. Bibliography styles should incorporate this length in the definition of the list of shorthands, if applicable.

`\labelfield'width`
`\BiberOnlyMark`
With Biber, for every field marked as a 'Label field' in the data model, a length register is created as per `\shorthandwidth`.

shorthand
above. Since 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. Nu-
meric bibliography styles should incorporate this length in the definition of the
bibliography environment.

`\labelalphawidth`

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

extraalpha field.

`maxextrayear`

This counter holds the highest number found in any

extrayear field.

`refsection`

refsection This counter indicates the current
environment.

When queried in a bibliography heading, the counter returns the value of the

refsection option passed to
`\printbibliography`

refsegment This counter indicates the current

refsegment environment.
When queried in a bibliography heading, this counter returns the value of the

refsegment 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.

citetotal

This counter, which is only available in the `\loopcode` of a citation command defined with

\DeclareCiteComma

holds the total number of valid entry keys passed to the citation command.

citecount

This counter, which is only available in the `\loopcode` of a citation command defined with

\DeclareCiteComma

holds the number of the entry key currently being processed by the `\loopcode`.

multicitetotal

This counter

is similar to **citetotal** but only available in multicite commands. It holds the total number of citations passed to the multicite command. Note that each of these citations may consist of more than one entry key. This information is provided by the

citetotal counter.

multicitecount

This counter

is similar to **citecount** but only available in multicite commands. It holds the number of the citation currently being processed. Note that this citation may consist of more than one entry key. This infor-

mation is provided by the **citetotal**

and **citecount** counters.

listtotal

This counter holds the total number of items in the current list. It is intended for use in list formatting directives and does not hold a meaningful value when used anywhere else. As an exception, it may also be used in the second optional argument

to `\printnames` and
`\printlist`, see § 4.4.1

for details. For every list, there is also a counter by the same name which holds the total number of items in the corresponding list. For ex-

ample, the `author` counter holds the total

number of items in the `author` list. This applies to both name lists and literal lists. These counters are similar to

`listtotal` except that they may also be used independently of list formatting directives. For example, a bibliography style

might check the `editor` counter to decide whether or not to print the term “editor” or rather its plural form “editors” after the list of editors.

`listcount` This counter holds the number of the list item currently being processed. It is intended for use in list formatting directives and does not hold a meaningful value when used anywhere else.

`liststart` This counter holds the `⟨start⟩` argument passed to `\printnames` or `\printlist`. It is intended for use in list formatting directives and does not hold a meaningful value when used anywhere else.

`liststop` This counter holds the `⟨stop⟩` argument passed to `\printnames` or

\printlist. It is intended for use in list formatting directives and does not hold a meaningful value when used anywhere else.

\currentfield
The name of the field currently being processed by
\printfield. This information is only available locally in field formatting directives.

\currentlist
The name of the literal list currently being processed by
\printlist. This information is only available locally in list formatting directives.

\currentname
The name of the name list currently being processed by
\printnames. This information is only available locally in name formatting directives.

4.10.6 General Purpose Hooks

\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`

`{
 $\langle code \rangle$
}`

Appends the $\langle code \rangle$ to an internal hook executed at the beginning of the list of shorthands. The $\langle code \rangle$ is executed at the beginning of the list of shorthands, immediately after the $\langle begin code \rangle$ of

`\defbibenvironment`

This command may only be used in the preamble.

When using Biber, this is just an alias for:

`\AtBeginBiblist{ $\langle biblistname \rangle$ }`

`\AtBeginBiblist`

`{
 $\langle biblistname \rangle$
}{
 $\langle code \rangle$
}`

Biber only

Appends the $\langle code \rangle$ to an internal hook executed at the beginning of the bibliography list $\langle biblistname \rangle$. The $\langle code \rangle$ is executed at the beginning of the bibliography list, immediately after the $\langle begin code \rangle$ of

`\defbibenvironment`

This command may only be used in the preamble.

`\AtEveryBibitem`

`{
 $\langle code \rangle$
}`

Appends the $\langle code \rangle$ to an internal hook executed at the beginning of every item in the bibliography. The $\langle code \rangle$ is executed immediately after the $\langle item code \rangle$ 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 shorthands. 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.

When using Biber, this is just an alias for:

`\AtEveryBiblistit`

`\AtEveryBiblistitem`

`{
 ⟨biblistname⟩
}{
 ⟨code⟩
}`

Biber only

Appends the `⟨code⟩` to an internal hook executed at the beginning of every item in the bibliography list named `⟨biblistname⟩`. 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.

`\AtNextBibliography`

`{
 ⟨code⟩
}`

Similar to

`\AtBeginBibliogr`

`\printbiblog`

but only affecting the next

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.

\AtNextCitekey

{
 $\langle code \rangle$
}

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

{
 $\langle code \rangle$
}

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

[
 $\langle entrytype \rangle$
]{
 $\langle code \rangle$
}

Appends the $\langle code \rangle$ to an internal hook executed once for every entry as the

bb \l

bibliographic data is imported from the file. The $\langle entrytype \rangle$ is the entry type the $\langle code \rangle$ applies to. If it applies to all entry types, omit the optional argument. The $\langle code \rangle$ is executed immediately after the entry has been imported. This command may only be used in the preamble. Note that $\langle code \rangle$ may be executed multiple times for an entry. This occurs when the same entry is cited

in different

**refsection
sorting**

environments or the 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 `\precode` 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 § 3.11.5. 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,
\finentry

respectively) and append

at the end. That's it. The formatting is handled by the drivers for the entry types of the individual set members:

```
\DeclareBibliography
\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:

```
\DeclareBibliography
\entryset
{\printfield{
\setunit*{\a
}}%
\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{label

entrysetcount

you simply append the field:

\printfield{label

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:

\iffielddundefined{ep

{\printfield{ep

{\printfield[ep

eprinttype

If an

field is available, the above code tries to use the field format

`eprint:` $\langle eprinttype \rangle$. If this format is undefined, `\printfield`

`eprint` automatically falls back to the field format. There are two predefined field formats, the type-specific format

`eprint:arxiv`

and the fallback format `eprint:`

`\DeclareFieldForm`
`\DeclareFieldForm`

In other words, adding support for additional resources is as easy as defining

`eprint:` $\langle resource \rangle$
 a field format named $\langle resource \rangle$ where $\langle resource \rangle$ 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 § 3.11.8. 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*{ann`
`\iffieldundef{a`

```

        {\printfile[a
        {\printfield{
\newcommand*{\bib
\newbibmacro*{abs
    \iffieldundef{a
        {\printfile[a
        {\printfield{
\newcommand*{\bib

```

abstract
 If the annotation field is unde-
 fined, the above code tries to load the abstracts/annotations from an external file.

\printfile commands
 The also incorporate file name prefixes which may be redefined by users. Note that

\printfile
 loadfiles
 you must enable
 explicitly by setting the
 package option from § 3.1.2.1. This feature is disabled by default for performance
 reasons.

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.

4.11.4.1 Individual Names

(uniquename)

Let's start off with some **uniquename** examples. Consider the following data:

John Doe	2008
Edward Doe	2008
John Smith	2008

Jane Smith 2008

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 last 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 last name. In order to disambiguate the author's last 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

John Smith 2008

Jane Smith 2008

In order to illustrate the difference between

uniquename =

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 maxnames /

minnames,

uniquelist truncation point.

For example, given this data:

William Jones/Edw
John Doe
John Smith

and `maxnames = 1`,
`minnames = 1`,
`uniquename =`
`init/full`, we would get the

following names in citations:

Jones et al.
Doe
Smith

When disambiguating names, `uniquename =`
`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 `uniquename =`
`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 uniqueness =

allfull

which also disambiguates 'John

Smith' from 'Jane Smith':

Jones et al.

J. Doe

John Smith

The options uniqueness=

mininit/minfull

init/full

are similar to 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
Edward Doe/Willia
John Smith/Willia
Edward Smith/Alla

With `uniquename =
init/full`, we would get:

J. Doe and Jones
E. Doe and Jones
J. Smith and Edwa
E. Smith and John

With `uniquename =
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 last 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:

John Doe/William Edward Doe

With `maxnames = 1,`
`uniquename =`
`init/full`, we would get:

J. Doe et al. E. Doe

With `uniquename =`
`mininit/minfull:`

Doe et al. Doe

Because the lists differ in the 'et al.', the names are not disambiguated.

unique_{list}

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

unique_{list} option. Consider the following data:

Doe/Jones/Smith
Smith/Johnson/Doe
Smith/Doe/Edwards
Smith/Doe/Jones

Many author-year styles truncate long author/editor lists in citations. For example,

with maxnames=1 we would get:

Doe et al. 2005
Smith et al. 2005
Smith et al. 2005
Smith et al. 2005

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 **unique**list=

true, we would get:

Doe et al. 2005

Smith, Johnson et

Smith, Doe and Ed

Smith, Doe and Jo

The **unique**list

option overrides **maxnames** /

minnames on a per-entry ba-

sis. 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 **unique**name.

Consider the following data:

John Doe/Allan Jo
 John Doe/Edward J
 John Doe/Jane Smi
 John Doe/John Smi
 John Doe/John Edw
 John Doe/John Edw

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 =`

```
fulluniquelist=
true:
```

```
Doe, A. Johnson e
Doe, E. Johnson e
Doe, Jane Smith e
Doe, John Smith e
Doe, Edwards and
Doe, Edwards and
```

```
With uniquelist =
minyear
labelyear
```

, list disambiguation only happens if the visible list is identical to another visible list with the same

. 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 2000

With `maxnames = 1`
and `uniquelist =`
`true`, we would get:

Smith and Jones 2
Smith and Johnson

With `uniquelist =`
`minyear`:

Smith et al. 2000
Smith et al. 2001

With `uniquelist =`
`minyear`, it is not clear that the au-

thors 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/Garbl
Vogel/Beast/Tremb
Vogel/Beast/Acid/

With `maxnames = 3,`
`minnames = 1,`
`unique_list =`
`true`, we would get:

Vogel, Beast, Gar
Vogel, Beast, Tre
Vogel, Beast, Aci

With `unique_list =`
`minyear:`

Vogel, Beast, Gar
Vogel, Beast, Tre
Vogel et al. 2001

In the last citation, `unique` `list=`
`minyear` does not over-
`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

`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 Biblatex package provides two main programming interfaces for style authors. The

`\DeclareBibliography`
 command, which defines a handler for an en-

try type, is typically used in `bbx` files.

`\DeclareCiteCommand`

`cbx`
 which defines a new citation command, is typically used in files. However, in some cases it is convenient to mix these two interfaces. For example,

`\fullcite`
 the command prints a verbose citation similar to the full bibliography entry. It is essentially defined as follows:

```
\DeclareCiteCommand
{...}
{\usedriver{...}
{...}
{...}}
```

As you can see, the core code which prints the citations simply executes the bibliography driver defined with

```
\DeclareBibliogra
```

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:

```
\ProvidesFile{exa
\DeclareCiteCommand
{...}
{\usedriver{...}
{...}
{...}
\DeclareBibliogra
```

```
\DeclareBibliography
\DeclareBibliography
...
```

Another case in which mixing interfaces is helpful are styles using cross-references within the bibliography. For example, when printing an

@incollection
@collection

entry, the data inherited from the parent entry would be replaced by a short pointer to the respective parent entry:

[1] Audrey Author: *Title of article*. In: [2], pp. 134–165.

[2] Edward Editor, ed.: *Title of collection*. Publisher: Location, 1995.

One way to implement such cross-references within the bibliography is to

think of them as citations which use the value of the `xref` or `crossref` field as the entry key. Here is an example:

```
\ProvidesFile{example.tex}
\DeclareCiteCommand{\cite}{}{}{}{}% code for citation
\end{document}
```

```

\DeclareBibliography
  \iffielddundef{x
    {...}% code i
    {\bbx@xref{\t
  ...
}

```

When defining `\bbx@xref`, the `<precode>`, `<postcode>`, and `<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 `<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 `\iffieldxref`, `\iflistxref`, and `\ifnamexref` tests in § 4.6.2. The above could also be implemented using the `\entrydata` command from § 4.4.1:

```

\ProvidesFile{exa
\DeclareBibliogra
\iffielddundefined{x
{...}% code i
{\entrydata{\
% code for
}}}%
}

```

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{editi
\newunit

```

`\printfield{note}`

edition

If there is no field, this piece of code will not print:

Title. . Note

but rather:

Title. Note

because the unit punctuation tracker works asynchronously.

`\newunit` will not print the unit punctuation immediately. It merely records a unit boundary and puts

`\newunitpunct` on the punctuation buffer. This buffer will be handled by *subse-*

quent `\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{editi
\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

`\iffielddundef`

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):

```
\iffielddundef{tit
  {\iffielddundef{
    {\printfield
    {\printfield
    \iffielddund
    {}
    {. \print
  {\printfield{ti
  \iffielddundef{
    {}
    {. \printfie
  \iffielddundef{
```

}
}. \printfield

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
```

This code will yield the following result:

Title (. Series N

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`),
`\printtext`

always wrap the text in a

command. For the punctuation tracker to work as expected, it needs to know about all literal text inserted by a driver. This is what

`\printtext` is all about.

`\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

`\printfield{serie`

In the corrected example, we will use the punctuation tracker to handle that space.

`\printfield{title`

`\newunit`

`\printtext{() %`

`\printfield{serie`

`\setunit*{\addspa`

`\printfield{numbe`

`\printtext{) %`

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:

`\DeclareFieldForm`

`\printfield`

Field formats may be used with both

and `\printtext`, hence we can
use them to enclose several fields in a single pair of parentheses:

```
\printtext[parens  
  \printfield{ser  
  \setunit*{\adds  
  \printfield{num  
}%
```

We still need to handle cases in which there is no series information at all, so let's improve the code some more:

```
\iffielddundef{ser  
  {}  
  {\printtext[par  
    \printfield{  
    \setunit*{\a
```

\printfield{

One final hint: localization strings are not literal text as far as the punctuation tracker

is concerned. Since **\bibstring** and similar commands interface with the punctuation tracker, there is no need to wrap them in a **\printtext** command.

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:

...	text.	Location
...	text.	Location
...	text.	Location
...	text.	Publishe
...	text.	Location
...	text.	Publishe
...	text.	Year. Te

This problem can be solved with a rather convoluted set of

`\iflistundef` and
`\iffieldundef`

tests which check for all possible field combinations:

```
\iflistundef{loca
  {\iflistundef{p
    {\printfield
    {\printlist{
      \iffieldund
    }}
    {\print
  {\printlist{loc
    \iflistundef{p
      {\iffieldund
    }}
    {\print
```

```

{: \printlis
\iffieldund
{}
{, \print

```

The above could be written in a somewhat more readable way by employ-

ing `\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{locati
\setunit*{\addcol
\printlist{publis
\setunit*{\addcom
\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

```


...	text.	Location
...	text.	Location
...	text.	Publishe
...	text.	Location
...	text.	Publishe
...	text.	Year. Te

This can be handled by the following code:

```

\newunit
\printlist{locati
\iflistundef{publ
    {\setunit*{\add
    {\setunit*{\add
\printlist{publis
\setunit*{\addcom
\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 ‘Works Cited’ rather than ‘Bibliography’ or ‘References’ in the heading of the bibliography. Localization commands such as

`\DefineBibliograph`
`cbx`

from § 3.8 may indeed be used in

`bbx` files to handle such cases. However, overloading style files with translations is rather inconvenient. This is where

`\DeclareLanguageM`

from § 4.9.1 comes into play. This command maps an

`l`
`bx` file with alternative translations to a

`babelpolyglossia`

language. For example, you could create a file named

`french-humanities`

`l`
`bx` which provides French translations

adapted for use in the humanities and map it to the

`babelpolyglossia`

`french` language in the preamble or in the configuration file:

`\DeclareLanguageM`

If the document language is set to french,
french-humanities
lbr will replace french.
lbr.

Coming back to the MLA exam-
ple mentioned above, an MLA style may come with an
american-mla.
lbr file to provide strings which comply with the MLA style

guide. It would declare the following mapping in the cbr and/or
lbr file:

\DeclareLanguageM

lbr Since the alternative lbr file can inherit strings from the standard

american.lbr mod-

ule, american-mla.

lbr may be as short as this:

```
\ProvidesFile{ame
\InheritBibliogra
\DeclareBibliogra
```

```

inherit
bibliography
references
}
\endinput

```

`\lbx` Alternative files must ensure that the localization module is complete. They will typically 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`

`\DeclareLanguage`

Note that is not intended to handle language variants (e. g., American English vs. British English) or

`babelpolyglossia`
`USenglish`
 language aliases (e.g., `american`). For example,
 vs. `babelpolyglossia`
`USenglish`
 offers the `american` option which
 is similar to `USenglish`. Therefore, Bibla-
 tex ships with an `american`.
`lbx` file which simply inherits all data from
`american.lbx`
`english`.
 (which in turn gets the 'strings' from `lbx`). In other words, the mapping of language variants and
`babelpolyglossia`
 language aliases happens on the file level, the point being that Biblalex's language sup-
 port can be extended simply by adding additional `lbx` files. There is
 no need for centralized mapping. If you need support for, say, Portuguese (`babelpoly-`
 glossia: `portuges`), you create a file named
`portuges.lbx`. If

babelpolyglossia

offered an alias named `brasil`, you would

create `brasil.lbx` and

inherit the data from `portuges.`

`lbx`.

In contrast to that, the point of `\DeclareLanguageM`

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

`\printbibliograph`

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

`\defbibenviromme`

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 *⟨item code⟩* of

`\defbibenvironment`

as well as all driver code.

`\DeclareCite`

- All citation commands defined with `\DeclareCite` are processed in a group holding the complete citation code consisting of the `\precode`, `\sepcode`, `\loopcode`, and `\postcode` arguments. The `\loopcode` is enclosed in an additional group every time it is executed. If any `\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

`\DeclareCiteComm`

all ‘autocite’ and ‘multicite’ definitions imply an additional group.

`\printfile`,

`\printtext`,

`\printfield`,

`\printlist`, and

`\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

`\DeclareBibliogr`

the definitions must be global.

\aftergroup

Note that using `\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 *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 package requires a macro for internal use, it would typically

be called

or

rather than

`\@macro`. Here is a list of the prefixes used or recommended by Biblatex:

blx

All macros with names like

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 in their name in any styles.

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.

bbx

This is the recommended prefix for internal macros defined in bibliography styles.

cbx

This is the recommended prefix for internal macros defined in citation styles.

lbx

This is the recommended base prefix for internal macros defined in localization modules. The localization module should add a second prefix to specify the language. For example, an internal macro defined by the Spanish localization module would be named

```
\lbx@es@macro
```

Appendix

A Default Driver Source Mappings

These are the driver default source mappings. For drivers other than

bibtex and **ris**, they are highly experimental and subject to change (because the driver datatype itself is unstable or not well suited to bibliographic data).

A.1 **bibtex**

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.

```
\DeclareDriverSou
\map{
  \step[typesou
  \step[typesou
```

```

        \step[typesou
    }
\map{
    \step[typesou
    \step[fieldse
}
\map{
    \step[typesou
    \step[fieldse
}
\map{
    \step[typesou
    \step[fieldse
}
\map{

```

```

\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
}
}
}

```

A.2 **endnotexml**

The **endnotexml** driver for Biber uses the following mappings. This format is quite messy and not well suited to robust typesetting of bibliographic data. It is presented largely as an example of Biber's modular datasource abilities and also of how to map foreign data models into the Biblatex data model. Mappings from such messy and relatively simple data models are inevitably fragile and are of limited use.

```

\DeclareDriverSou

```

[illegible]

\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou

\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou

```
        \step[typesou  
        \step[typesou  
    }  
    \map{  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou
```

```
        \step[fieldso
        \step[fieldso
        \step[fieldso
        \step[fieldso
        \step[fieldso
    }
    \map{
        \pertype{Edit
        \step[fieldso
    }
    \map{
        \pertype{Elec
        \pertype{Jour
        \pertype{Maga
        \pertype{News
```



```

        \step[fieldso
    }
\map{
    \pertype{Pate
    \pertype{Repo
    \pertype{Gove
    \pertype{Lega
    \step[fieldso
}
\map{
    \pertype{Blog
    \pertype{Onli
    \pertype{Onli
    \pertype{Web
    \step[fieldso

```

```

}
\map{
  \pertype{Book
  \step[fieldso
}
\map{
  \pertype{Book
  \pertype{Elec
  \pertype{Manu
  \pertype{Unpu
  \step[fieldso
}
\map{
  \pertype{Conf
  \pertype{Conf

```

```

        \step[fieldso
    }
\map{
    \pertype{Elec
    \pertype{Jour
    \pertype{Maga
    \pertype{News
    \step[fieldso
}
\map{
    \pertype{Book
    \step[fieldso
}
\map{
    \pertype{Conf

```

```

\pertype{peri
\step[fieldso
}
}

```

A.3 ris

The `ris` driver reflects the fact that `ris` itself is a very simple and stable format. It is in fact so simple, it's hardly of any use for most Biblatex users. Again, here more as a proof of concept example.

```

\DeclareDriverSou
\map{
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou

```

\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou

```
        \step[typesou  
        \step[typesou  
    }  
    \map{  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou
```

\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso

```

\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
}
}

```

A.4 **zoterordfxml**

The **zoterordfxml** driver reads the Zotero³⁸ XML representation of its RDF format. From the Biblatex point of view, it suffers from a lack of the notion of a citation key and so is of limited use. The format may include a citation key in the future at which point the Biber driver and these default mappings will be updated. As with

endnotexml, the format is somewhat messy with, in particular, a very limited way of dealing with related entries which makes the driver setup rather ugly. As can be seen below, it is possible to map into a field prefixed with “BIBERCUSTOM” which is not a real field in the data model but can be detected in the Biber driver itself and used to process a source field which does not map naturally to any Biblatex data model field as it needs special treatment. This is messy but is due largely due to messy source data models.

```
\DeclareDriverSou
```

³⁸ www.zotero.org


```
\map{
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
  \step[typesou
```

\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou
\step[typesou

```
        \step[typesou  
        \step[typesou  
    }  
    \map{  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou  
        \step[fieldsou
```

\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso

\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso

\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso
\step[fieldso

```
\step[fieldso  
\step[fieldso  
\step[fieldso  
\step[fieldso  
\step[fieldso  
\step[fieldso  
\step[fieldso  
\step[fieldso  
      fieldta  
\step[fieldso  
      fieldta  
\step[fieldso  
      fieldta  
}
```



B Default Inheritance Setup

The following table shows the Biber cross-referencing rules defined by default. Please refer to §§ [2.4.1](#) and [4.5.10](#) for explanation.

Types		Fields	
Source	Target	Source	Target
*	*	ids	
		cross -	
		ref	
		xref	
		en -	
		try -	
		set	
		en -	
		try -	
		sub -	
		type	
		ex -	
		e -	
		cute	
		la -	
		bel	
		op -	
		tions	
		pre -	
		sort	
		re -	
		lated	
		re -	
		late -	
		dop -	
		tions	

Types		Fields	
Source	Target	Source	Target
mv-	inbook	au-	au-
book-	book-	thor	thor
book-	inbook	au-	bookau-
	supp-	thor	thor
	book		

Types		Fields	
Source	Target	Source	Target
mv-book	book	ti-main-	
book	inbook	tle	ti-
	book-	subtle	
	inbook	ti-main-	
	supp-	tle	sub-
	book	ti-ti-	
		tle	ade
		don	main-
		short	ti-
		ti-tlead-	
		tle	don
		sort-	
		ti-	
		tle	
		in-	
		dex-	
		ti-	
		tle	
		in-	
		dex-	
		sort-	
		ti-	
		tle	

Types		Fields	
Source	Target	Source	Target
mv-col-		ti-main-	
collec-		tle ti-	
lection		subtle	
tion	refer-	ti-main-	
mvref-	ence	tle sub-	
er-col-		ti-ti-	
ence	lec-	tle ad-	
	tion	don main-	
	inref-	short ti-	
	erence	ti-tlead-	
	supp-	tle don	
	col-	sort-	
	lec-	ti-	
	tion	tle	
		in-	
		dex-	
		ti-	
		tle	
		in-	
		dex-	
		sort-	
		ti-	
		tle	

Types		Fields	
Source	Target	Source	Target
mvp pro -		ti-main -	
ceed -		tle ti -	
ings		subtle	
	inpro -	ti-main -	
	ceed -	tle sub -	
	ings	ti-ti -	
		tleade	
		donmain -	
		shorti -	
		ti-tlead -	
		tle don	
		sort -	
		ti -	
		tle	
		in -	
		dex -	
		ti -	
		tle	
		in -	
		dex -	
		sort -	
		ti -	
		tle	

Types		Fields	
Source	Target	Source	Target
book	inbook	ti-book-	
	book-	tle	ti-
	inbook	subtle	
	supp-	ti-book-	
	book	tle	sub-
		ti-ti-	
		tle	ade
		don	book-
		short	ti-
		ti-tlead-	
		tle	don
		sort-	
		ti-	
		tle	
		in-	
		dex-	
		ti-	
		tle	
		in-	
		dex-	
		sort-	
		ti-	
		tle	

Types		Fields	
Source	Target	Source	Target
collection	in collection	title	book-
reference	lec-	subtle	ti-
er-	tion	ti-	book-
ence	in ref-	tle	sub-
	erence	ti-	ti-
	supp-	tle	ad-
	col-	don	book-
	lec-	short	ti-
	tion	ti-	tlead-
		tle	don
		sort-	
		ti-	
		tle	
		in-	
		dex-	
		ti-	
		tle	
		in-	
		dex-	
		sort-	
		ti-	
		tle	

Types		Fields	
Source	Target	Source	Target
proceedings	inproceedings	ti-book-titlesub-ti-book-titlesub-ti-titleslead-donbook-short-ti-titleslead-don-sort-ti-titlesin-dex-ti-titlesin-dex-sort-ti-titles	

Types		Fields	
Source	Target	Source	Target
pe-	arti-	ti-	journalt
ri-	cle	tle	tle
od-	supp-	sub	jour-
i-	peri-	ti-	nal-
cal	odical	tle	sub-
		short	ti-
		ti-	tle
		tle	
		sort-	
		ti-	
		tle	
		in-	
		dex-	
		ti-	
		tle	
		in-	
		dex-	
		sort-	
		ti-	
		tle	

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.

Option *Sorting scheme*

ntpre-sortsortsortvylear
sortnamei-year
mmtitle
thor
edti-
i-
tor
trans-
la-
tor
sort-
ti-
tle
ti-
tle

nytre-sorter
name-year
sorter
mm-tle
thor-ti-
ed-tle
i-
tor
trans-
la-
tor
sort-
ti-
tle
ti-
tle

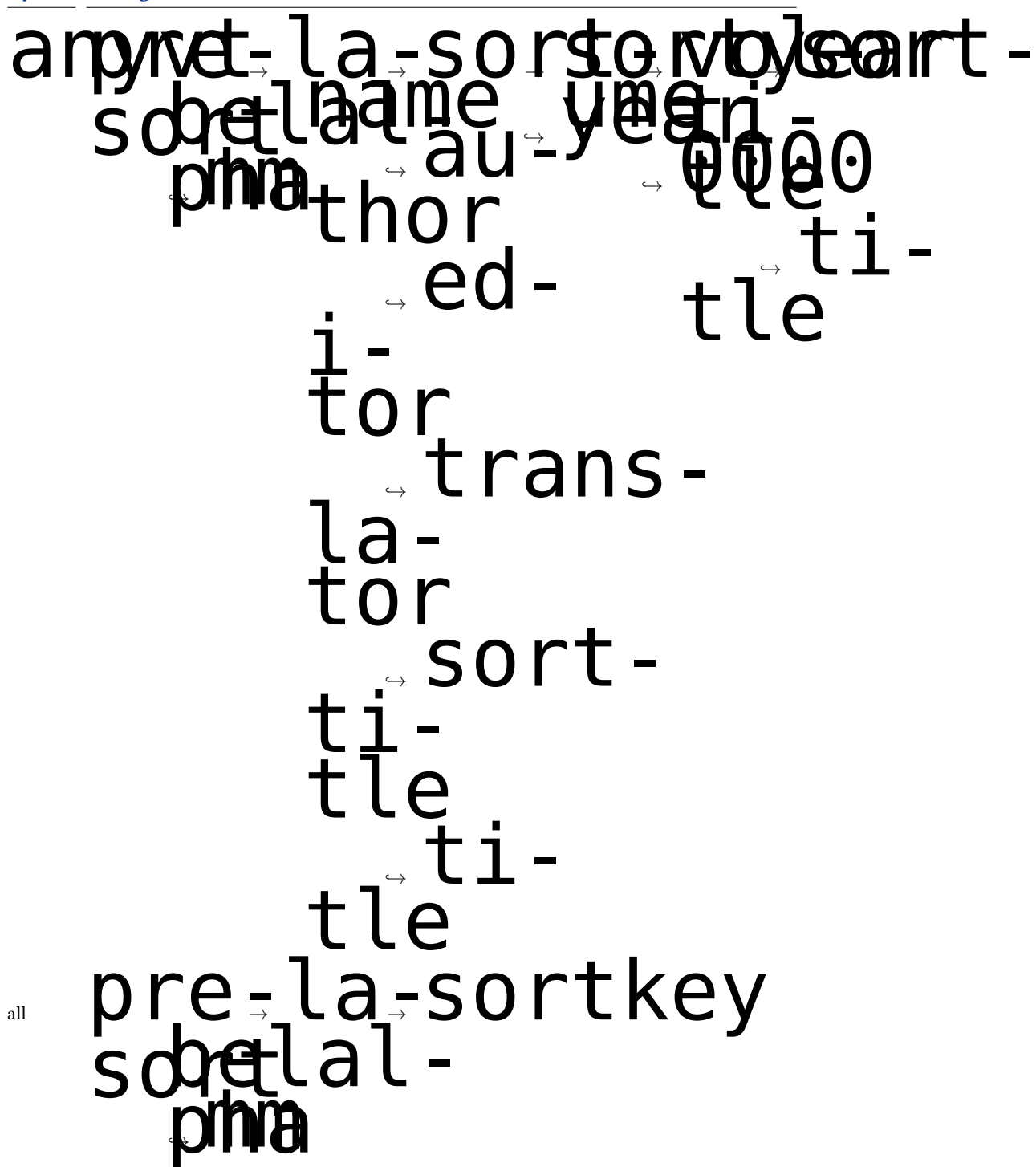
nyrte - sort to by sort -
name year
sort au - year
mm
thor
ed -
i -
tor
trans -
la -
tor
sort -
ti -
tle
ti -
tle
pre - sortkey
sort
mm

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.

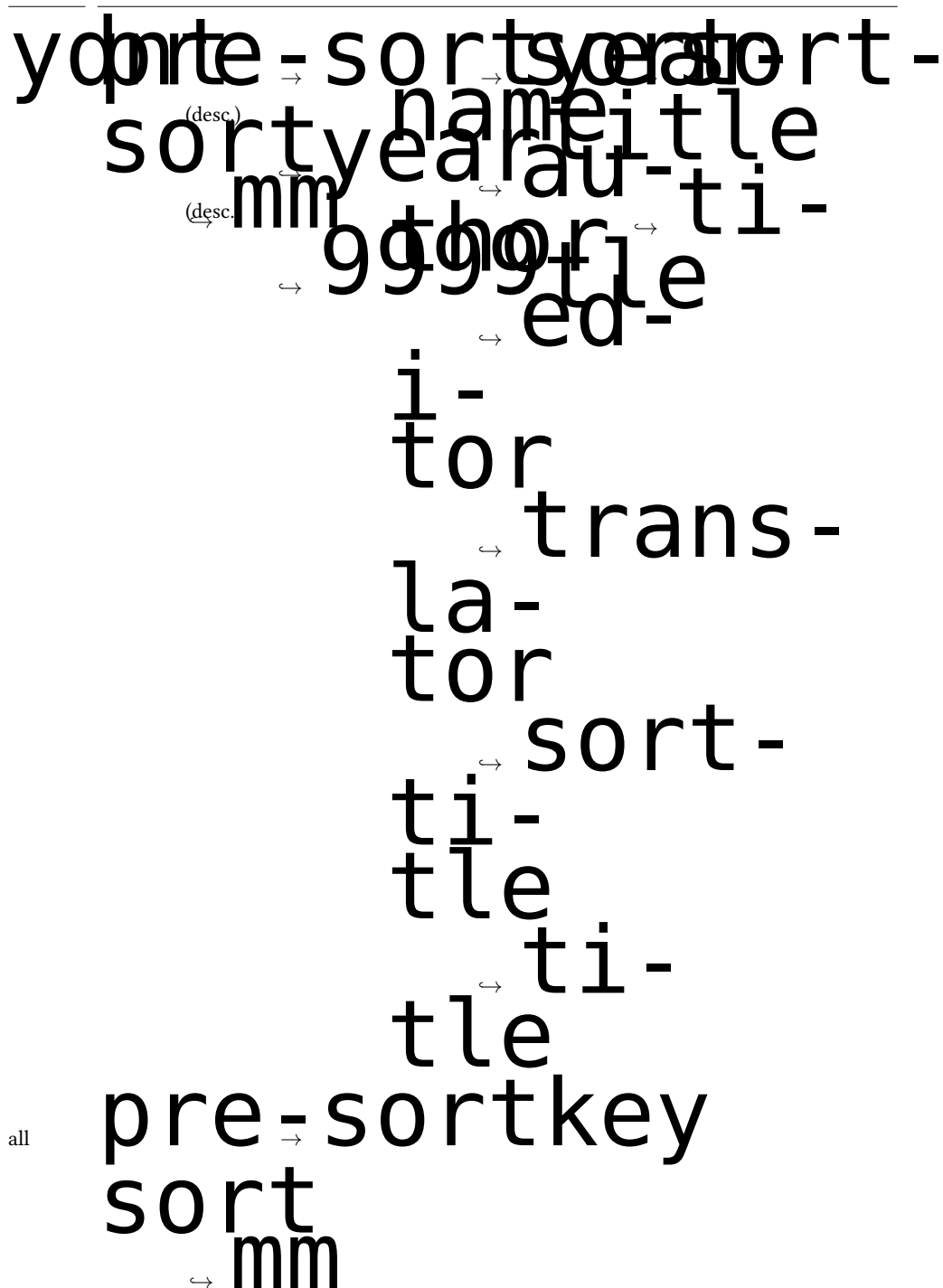
ampte-la-sorto-royal-
sorellame-yeame
phma au-tle 0000
thor-ti-
ed-tle
i-
tor
trans-
la-
tor
sort-
ti-
tle
ti-
tle



C.3 Chronological Schemes

The following table shows the chronological sorting schemes defined by default. Please refer to § 3.5 for explanation.

ymptre - sorts over short -
sortyear name title
mm9999 au - ti -
thorle
ed -
i -
tor
trans -
la -
tor
sort -
ti -
tle
ti -
tle



D Option Scope

The following table provides an overview of the scope (global/per-type/per-entry) of various package options.

<i>Option</i>	<i>Scope</i>			
	<i>Load-time</i>	<i>Global</i>	<i>Per-type</i>	<i>Per-entry</i>
abbreviate	.	.	-	-
all-dates	.	.	-	-
arxiv	.	.	-	-
au-tocite	.	.	-	-
autopunct	.	.	-	-
au-tolang	.	.	-	-
back-end	.	-	-	-
back-ref	.	.	-	-
back-ref-style	.	.	-	-

<i>Option</i>	<i>Scope</i>			
	<i>Load-time</i>	<i>Global</i>	<i>Per-type</i>	<i>Per-entry</i>
back- ref- style	.	.	-	-
biben- coding	.	.	-	-
bib- style	.	-	-	-
bib- warn	.	.	-	-
block	.	.	-	-
cite- counter	.	.	-	-
citere- set	.	.	-	-
citestyle		-	-	-
cite- tracker	.	.	-	-
clear- lang	.	.	-	-

Option	Scope			
	Load-time	Global	Per-type	Per-entry
data-	.	-	-	-
model				
dataonly		-	.	.
date	.	.	-	-
date-	.	.	-	-
abbrev				
dateze-	.	.	-	-
ros				
defer-	.	.	-	-
num-				
bers				
doi	.	.	-	-
eprint	.	.	-	-
event-	.	.	-	-
date				
firs-	.	.	-	-
tinits				
hyper-	.	.	-	-
ref				

Option	Scope			
	Load-time	Global	Per-type	Per-entry
ibid-tracker	.	.	-	-
idem-tracker	.	.	-	-
index-ing
isbn	.	.	-	-
label-alpha	.	.	.	-
label-name-field	-	-	-	.
label-name-form	-	-	-	.
label-name-lang	-	-	-	.
label-number	.	.	.	-

Option	Scope			
	Load-time	Global	Per-type	Per-entry
la- bel- ti- tle	.	.	.	-
la- bel- ti- tle- field	-	-	-	.
la- bel- ti- tle- form	-	-	-	.
la- bel- ti- tle- lang	-	-	-	.
la- bel- ti- tle- year	.	.	.	-
label- date	.	.	.	-

<i>Option</i>	<i>Scope</i>			
	<i>Load-time</i>	<i>Global</i>	<i>Per-type</i>	<i>Per-entry</i>
language	.	.	-	-
load-files	.	.	-	-
loc-cit-tracker	.	.	-	-
maxalphanames
maxbibnames
max-cite-names
max-items
max-names
max-parens	.	.	-	-
mcite	.	-	-	-

<i>Option</i>	<i>Scope</i>			
	<i>Load-time</i>	<i>Global</i>	<i>Per-type</i>	<i>Per-entry</i>
minimal- phanames
min- bib- names
mincite- names
min- cross- refs	.	.	-	-
minitems
min- names
natbib	.	-	-	-
note- type	.	.	-	-
opcit- tracker	.	.	-	-
open- bib	.	.	-	-

Option	Scope			
	Load-time	Global	Per-type	Per-entry
orig-	.	.	-	-
date				
page-	.	.	-	-
tracker				
paren-	.	.	-	-
tracker				
punct-	.	.	-	-
font				
ref-	.	.	-	-
sec-				
tion				
ref-	.	.	-	-
seg-				
ment				
safein-	.	.	-	-
putenc				
sin-	.	.	.	-
gleti-				
tle				

Option	Scope			
	Load-time	Global	Per-type	Per-entry
skip-	-	-	.	.
bib				
skiplab	-	-	.	.
skip-	-	-	.	.
los				
sort-	.	.	-	-
case				
sortcites	.	.	-	-
sort-	.	.	-	-
firs-				
tinits				
sort-	.	.	-	-
ing				
sort-	.	.	-	-
locale				
sort-	.	.	-	-
los				
sor-	.	.	-	-
tupper				
style	.	-	-	-

Option	Scope			
	Load-time	Global	Per-type	Per-entry
ter-	.	.	-	-
seinit				
texen-	.	.	-	-
coding				
uniquelist			.	.
unique-
name				
url-	.	.	-	-
date				
url	.	.	-	-
useau-
thor				
useed-
itor				
usepre-
fix				
use-
trans-				
lator				

E 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. If an entry in the revision history states that a feature has been *improved* or *extended*, this indicates a modification which either does not affect the syntax and behavior of the package or is syntactically backwards compatible (such as the addition of an optional argument to an existing command). Entries stating that a feature has been *modified*, *renamed*, or *removed* demand attention. They indicate a modification which may require changes to existing styles or documents in some, hopefully rare, cases. The numbers on the right indicate the relevant section of this manual.

2.9a 2014

resetnumbers now
allows passing a number to reset to 3.6.2

2.9 2014-02-25

Generalised shorthands facility 3.6.4 Biber only
Sorting locales can now be defined as part of a sorting scheme 4.5.5 Biber only

Added

sortinithash Biber only
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

Biber only

2.8 2013-10-21

New langidopts

... 2.2.3

Biber only

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

`postpunct`

Added

special field,

documented

`multiprenote`

and

`multipostnote`

special fields 4.3.2

Added

`\UseBibitemHook`,
`\AtEveryMultiCite`
`\AtNextMultiCite`,
`\UseEveryCiteHook`
`\UseEveryCitekeyH`
`\UseEveryMultiCite`
`\UseNextCiteHook`,
`\UseNextCitekeyHo`
`\UseNextMultiCite`
`\DeferNextCitekey`

4.10.6

`\textcite`

Fixed

and related

commands in the numeric and verbose styles 3.7.2

`\volcite`

Added multicite variants of
related commands

and

3.7.6

Added

`\finalandsemicolon`

3.9.2

Added citation delimiter

`\textcitedelim`
`\textcite`

for commands to styles 4.10.1 and related

Updated Russian localization (Oleg Domanov)

Fixed Brazilian and Finnish localization

2.6 2013-04-30

`\printunit`

Added . . . 4.7.1

Added field

`clonesourcekey`

4.2.4.1

Biber only

New options for

`\DeclareLabelalph`

4.5.4

Biber only

Added

`\DeclareLabeldate`

and retired

`\DeclareLabelyear`

4.5.9

Biber only

`nodate`

Added localization string 4.9.2.14

`\rangelen`

Added 4.6.4

Added starred variants of

`\citeauthor`

and

`\Citeauthor`

3.7.5

Restored original `url` format. Added
`urlfrom` localization key 4.9.2.15

Added
`\AtNextBibliograph`
 4.10.6

Fixed related entry processing to allow nested and cyclic related entries

Added Croatian localization (Ivo Pletikosić)

Added Polish localization (Anastasia Kandulina, Yuriy Chernyshov)

Fixed Catalan localization

Added smart “of” for titles to Catalan and French localization

Misc bug fixes

2.5 2013-01-10

Made `url` work as a localization 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 4.5.2 Biber only

Updated Catalan localization (Sebastià Vila-Marta)

2.4 2012-11-28

Added
`relatedoptions`
 field 4.5.1 Biber only

Added
`\DeclareStyleSource`
 4.5.2 Biber only

Renamed
`\DeclareDefaultSource`
 to
`\DeclareDriverSource`
 4.5.2 Biber only

Documented

`\DeclareFieldInput`
`\DeclareListInput`
and
`\DeclareNameInput`

Added Czech localization (Michal Hoftich)

Updated Catalan localization (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 4.5.2 Biber only

Extended auxiliary indexing macros

Added support for plural localization strings with

`relatedtype` 4.5.1 Biber only

Added `\csfield` and `\usefield` 4.6.1

Added starred variant of

`\usebibmacro`
4.6.4

Added
`\ifbibmacroun`
`\iffieldformatunc`
`\iflistformatunde`
and
`\ifnameformatunde`
4.6.4

Added Catalan localization (Sebastià Vila-Marta)

Misc bug fixes

2.2 2012-08-17

Misc bug fixes

Added
`\revsdnamepunct`
3.9.1

Added
`\ifterseinits`
4.6.2

2.1 2012-08-01

Misc bug fixes

Updated Norwegian localization (Håkon Malmedal)

Increased data model auto-loading possibilities 4.5.3 Biber only

2.0 2012-07-01

Misc bug fixes

Generalised `singletitle`
test a little 4.6.2 Biber only

Added new special field

`extratitleyear`
4.2.4 Biber only

Customisable data model 4.5.3 Biber only

Added	<code>\DeclareDefaultSort</code>		
	4.5.2		Biber only
	<code>labeltitle</code>	option	
Added			Biber only
	3.1.2.3		
Added new special field	<code>extratitle</code>	...	4.2.4
	<code>labeltitle</code>		Biber only
Made special field			
customisable	4.2.4	Biber only
Removed field	<code>reprinttitle</code>		
	3.4		Biber only
Added related entry feature	3.4	Biber only
Added	<code>\DeclareNoinit</code>		
	4.5.7		Biber only
Added	<code>\DeclareNosort</code>		
	4.5.8		Biber only
	<code>sorting</code>	option for	
Added	<code>\printbibliograph</code>		
and	<code>\printshorthands</code>		
	3.6.2		Biber only
	<code>ids</code>	field for citekey aliasing	2.2
Added			Biber only

Added

sortfirstinits

option 3.1.2.3 Biber only

Added data stream modification feature 4.5.2 Biber only

Added customisable labels feature 4.5.4 Biber only

Added **\citeyear*** and
\citedate* 3.7.5

1.7 2011-11-13

Added **xdata** containers 3.11.6 Biber only

Added special entry type **xdata** 2.1.1 Biber only

Added special field **xdata** 2.2.3 Biber only

Support

maxnamesminnames

globally/per-type/per-entry 3.1.2.1 Biber only

Support

maxbibnamesminbib

globally/per-type/per-entry 3.1.2.1 Biber only

Support

maxcitenamesminc

globally/per-type/per-entry 3.1.2.1 Biber only

Support

maxitemsminitems

globally/per-type/per-entry 3.1.2.1 Biber only

Support

maxalphanamesmina

globally/per-type/per-entry 3.1.2.3 Biber only

Support **uniquelist**
globally/per-type/per-entry 3.1.2.3 Biber only

Support **uniquename**
globally/per-type/per-entry 3.1.2.3 Biber only

Added **\textcite** and
\textcites to all
verbose citation styles 3.3.1

Added special field formats **date**,
urldate,
origdate,
eventdate 4.10.4

Added **\mkcomprange***
4.6.4

Added **\mkfirstpage***
4.6.4

Added **\volcitedelim**
4.10.1

Added missing documentation for
\ifentrytype
4.6.2

Added

\mkbibneutord

3.9.2

Added counter

biburlnumpenalty

4.10.3

Added counter

biburlucpenalty

4.10.3

Added counter

biburlllcpenalty

4.10.3

bookpart,

Added localization keys

issue,
forthcoming

4.9.2

lang . . . and

Added some

from . . . localization keys 4.9.2

Expanded documentation D

Added support for Russian (Oleg Domanov)

Updated support for Dutch (Pieter Belmans)

textcase

Fixed compatibility issue with
package

Fixed some bugs

Added special field

sortshorthand

2.2.3

Biber only

Revised options

maxnames,minnames

3.1.2.1

Options

maxcitenames,mincitenames

now supported by backend 3.1.2.1 Biber only

Options

maxbibnames,minbibnames

now supported by backend 3.1.2.1 Biber only

Added options

maxalphanames,minalphanames

3.1.2.3

Biber only

Removed local options

maxnames,minnames

from

\printbibliograph

3.6.2

Removed local options

maxitems,minitems

from

\printbibliograph

3.6.2

Removed local options

maxnames/minnames
from
`\bibbysection`

3.6.2

Removed local options

maxitems/minitems
from
`\bibbysection`

3.6.2

Removed local options

maxnames/minnames
from
`\bibbysegment`

3.6.2

Removed local options

maxitems/minitems
from
`\bibbysegment`

3.6.2

Removed local options

maxnames/minnames
from
`\bibbycategory`

3.6.2

Removed local options

maxitems/minitems
from
`\bibbycategory`

3.6.2

Removed local options

maxnames/minnames
from
`\printshorthands`

3.6.4

Removed local options

maxitems/minitems
from
`\printshorthands`

3.6.4

Added special field format

`volcitevolume`

3.7.6

Added special field format

`volcitepages`

3.7.6

hash

Added special field

..... 4.2.4.1

Biber only

Added

`\mkcomprange`

4.6.4

Added

`\mkfirstpage`

4.6.4

Removed

`\mkpagefirst`

4.6.4

Fixed some bugs

1.5a 2011-06-17

Fixed some bugs

1.5 2011-06-08

Added option

`uniquename= mininit/minfull`

3.1.2.3

Biber only

Added option

`uniquelist= minyear`

3.1.2.3

Biber only

Updated documentation of

`uniquename`

counter4.6.2

Biber only

Updated documentation of

`uniquelist`

counter4.6.2

Biber only

Expanded documentation for

`uniquename/unique`

options

4.11.4

Biber only

`reviser`

2.3.6

Added editorial role

Added localization keys **reviser**,
revisers,
byreviser 4.9.2

Added bibliography heading **none** 3.6.8

Fixed some **memoir** compatibility issues

1.4c 2011-05-12

Fixed some bugs

1.4b 2011-04-12

Fixed some bugs

1.4a 2011-04-06

Enable **uniquename** and
uniquelist in all
authortitle styles
3.3.1

Enable **uniquename** and
uniquelist in all
authoryear styles 3.3.1

Fixed some bugs

1.4 2011-03-31

Added package option

uniquelist . . . 3.1.2.3 Biber only

Added special counter

`uniqueilist` . . . 4.6.2 Biber only

Revised and improved package option

`uniqueiname` . . . 3.1.2.3 Biber only

Revised and improved special counter

`uniqueiname` . . . 4.6.2 Biber only

Added

`\bibnamedelii` Biber only
3.9.1

Added

`\bibindexnamedelii`
3.9.1

Added

`\bibindexnamedelii`
3.9.1

Added

`\bibindexnamedelii`
3.9.1

Added

`\bibindexnamedelii`
3.9.1

Added

`\bibindexnamedelii`
3.9.1

Added

`\bibindexinitperii`
3.9.1

Added
3.9.1 `\bibindexinitdel`

Added
3.9.1 `\bibindexinithyp`

Fixed conflict with some AMS classes

1.3a 2011-03-18

Fixed some bugs

1.3 2011-03-14

Support `@thesis` with
`isbn` 2.1.1

Updated `terseinits` option
3.1.2.1

Allow macros in argument to
`\addbibresource`
and friends 3.6.1

Allow macros in argument to
`\bibliography`
3.6.1

Introducing experimental support for Zotero RDF/XML 3.6.1 Biber only

Introducing experimental support for EndNote XML 3.6.1 Biber only

Added option
`citecounter` 3.1.2.3

Added
4.6.2 `citecounter`

Added	<code>\smartcite</code>	and	
	<code>\Smartcite</code>	3.7.2
Added	<code>\smartcites</code>	and	
	<code>\Smartcites</code>		3.7.3
Added	<code>\svolcite</code>	and	
	<code>\Svolcite</code>	3.7.6
Added	<code>\bibnamedelim</code>		Biber only
		3.9.1	
Added	<code>\bibnamedelim</code>		Biber only
		3.9.1	
Added	<code>\bibnamedelim</code>		
		3.9.1	
Added	<code>\bibnamedelim</code>		
		3.9.1	
Added	<code>\bibinitperiod</code>		Biber only
		3.9.1	
Added	<code>\bibinitdelim</code>		Biber only
		3.9.1	

Added
`\bibinithyphendel` 3.9.1 Biber only

Expanded documentation 3.12.4

Added $\langle position \rangle$ parameter to
`\DeclareAutoCiteC` 4.3.1

1.2a 2011-02-13

Fix in
`\mkbibmonth` 4.10.4

1.2 2011-02-12

Added entry type
`@mvbook` 2.1.1

Added entry type
`@mvcollection` 2.1.1

Added entry type
`@mvproceedings` 2.1.1

Added entry type
`@mvreference` 2.1.1

Introducing remote resources 3.6.1 Biber only

Introducing experimental RIS support 3.6.1 Biber only

Added
`\addbibresource` 3.6.1

`\bibliography`
now deprecated 3.6.1

`\bibliography*`

replaced by

`\addglobalbib`

3.6.1

Added

`\addsectionbib`

3.6.1

Updated and expanded documentation 2.4.1

`crossref`

Introducing smart
inheritance

data

2.4.1.2

Biber only

`crossref`

Introducing
interface

configuration

4.5.10

Biber only

Added

`\DefaultInheritance`

4.5.10

Biber only

Added

`\DeclareDataInheritance`

4.5.10

Biber only

Added

`\ResetDataInheritance`

4.5.10

Biber only

`\ifkeyword`

Added

. . . 4.6.2

Added

`\ifentrykeyword`

4.6.2

`\ifcategory`

Added

4.6.2

Added
`\ifentrycategory`

4.6.2

Added
`\ifdriver` 4.6.2

Added
`\forcsvfield`
4.6.4

Extended
`\mkpageprefix`
4.6.4

Extended
`\mkpagetotal`
4.6.4

Extended
`\mkpagefirst`
4.6.4

Added localization key
`inpreparation`
4.9.2

Rearranged manual slightly, moving some tables to the appendix

1.1b 2011-02-04

Added option
`texencoding` 3.1.2.1 Biber only

Added option
`safeinputenc` 3.1.2.1 Biber only

Expanded documentation 2.4.3.4

Improved **mergedate** option of
bibliography style

authoryear . . . 3.3.2

Removed **pass** option of

\DeclareSortingScheme
4.5.5 Biber only

Fixed some bugs

1.1a 2011-01-08

Added unsupported entry type **@bibnote**
2.1.3

Added **\bibliography***
3.6.1

Fixed some bugs

1.1 2011-01-05

Added option

maxbibnames 3.1.2.1

Added option

minbibnames 3.1.2.1

Added option

maxcitenames
3.1.2.1

Added option

mincitenames
3.1.2.1

Fixed **idemtracker=strict** and **idemtracker=constrict** 3.1.2.3

Added option **mergedate** to bibliography style

authoryear . . . 3.3.2

Added support for

prefixnumbers

to bibliography style

alphabetic . . . 3.3.2

Made option **useprefix** setttable on a per-type basis 3.1.3 Biber only

Made option **useauthor** setttable on a per-type basis 3.1.3 Biber only

Made option **useeditor** setttable on a per-type basis 3.1.3 Biber only

Made option

usetranslator setttable on a per-type basis 3.1.3 Biber only

Made option **skipbib** setttable on a per-type basis 3.1.3 Biber only

Made option **skiplos** settable on a per-type basis 3.1.3 Biber only

Made option **skiplab** settable on a per-type basis 3.1.3 Biber only

Made option **dataonly** settable on a per-type basis 3.1.3 Biber only

Made option **labelalpha** settable on a per-type basis 3.1.2.3 Biber only

Made option **labelnumber** settable on a per-type basis 3.1.2.3

Made option **labelyear** settable on a per-type basis 3.1.2.3 Biber only

Made option **singletitle** settable on a per-type basis 3.1.2.3 Biber only

Made option **uniquename** settable on a per-type basis 3.1.2.3 Biber only

Made option **indexing** settable on a per-type basis 3.1.2.1

Made option **indexing** settable on a per-entry basis 3.1.2.1

Extended **\ExecuteBibliography** 3.2.2

Added `\citedate` 3.7.5
 Improved static entry sets 3.11.5 Biber only
 Introducing dynamic entry sets 3.11.5 Biber only

Added `\defbibentryset` 3.6.11 Biber only

Added option `mcite` 3.1.1 Biber only

Added `mcite` / `mciteplus` -like commands 3.7.10 Biber only

Added `\sortalphaothers` 3.9.1 Biber only

Extended `\DeclareNameFormat` 4.4.2

Extended `\DeclareIndexName` 4.4.2

Extended `\DeclareListFormat` 4.4.2

Extended `\DeclareIndexList` 4.4.2

Extended `\DeclareFieldForm` 4.4.2

Extended
`\DeclareIndexField`
4.4.2

Added
`\DeclareNameFormat`
4.4.2

Added
`\DeclareIndexName`
4.4.2

Added
`\DeclareListFormat`
4.4.2

Added
`\DeclareIndexList`
4.4.2

Added
`\DeclareFieldForm`
4.4.2

Added
`\DeclareIndexField`
4.4.2

Introducing configurable sorting schemes Biber only

Added
`\DeclareSortingSc`
4.5.5 Biber only

Added
`\DeclarePresort`
4.5.5 Biber only

Added
`\DeclareSortExclu`
4.5.5 Biber only

Added
4.5.9 `\DeclareLabelname` Biber only

Added
4.5.9 `\DeclareLabelyear` Biber only

Improved special field
4.2.4 `labelname` Biber only

Improved special field
4.2.4 `labelyear` Biber only

Added
4.4.1 `\entrydata*`

Added
4.6.4 `\RequireBiber`

Added option `check` to
3.6.2 `\printbibliograph`

Added option `check` to
3.6.4 `\printshorthands`

Added
3.6.10 `\defbibcheck`

Updated support for Portuguese (José Carlos Santos)

Fixed conflict with `titletoc`

Fixed some bugs

1.0 2010-11-19

First officially stable release

Renamed option

bibencoding=
inputenc to
bibencoding=
auto 3.1.2.1

Made

bibencoding=
auto the package default 3.1.2.1

Added option

backend=
bibtexu 3.1.2.1

Slightly updated documentation 2.4.3

Updated support for Dutch (Alexander van Loon)

Updated support for Italian (Andrea Marchitelli)

type_name	author
	after-
	word
	annota-
	tor
	bookau-
	thor
	commen-
	tator
	editor
	editora
	editorb
	editorc
	fore-
	word
	holder
	intro-
	duction
	namea
	nameb
	namec
	shortau-
	thor
	shorted-
	itor

<i>Input</i>	<i>Output</i>
mincomprange=10	min
11-15	11-15
111-115	111-5
1111-1115	1111-5
maxcomprange=1000	max
1111-1115	1111-5
1111-1155	1111-55
1111-1555	1111-1555
mincompwidth=1	min
1111-1115	1111-15
1111-1155	1111-55
1111-1555	1111-555

Table 10: **\mkcomprange**
setup