BibLaTeX style extension for Software

Citation and References macros for BibLaTeX

Roberto Di Cosmo
roberto@dicosmo.org

Version 1.2-8
2025-12-26

Contents

1 Introduction

1.1 About

Software plays a significant role in modern research, and it must be properly acknowledged and referenced in scholarly works. To this end, specific bibliographic entries for describing *software*, *software versions*, *software modules* and *code fragments* have been designed by a dedicated task force at Inria in 2020 that brought together researchers from Computer Science and Applied Mathematics to discuss best practices for archiving and referencing software source code [2].

This package is a BibLaTeX *style extension* that adds support for these four *software entry types* to any other BibLaTeX style used in documents typeset in LaTeX. It is made up of the following key components: a references section style (`software.bbx`), a data model extension (`software.dbx`) and string localisation files (`<language>-software.lbx`)[1].

The distribution contains other material, for demonstration purposes, and for more advanced use.

[1] String localisations are provided for some languages: localisations for other languages are welcome, feel free to contribute them on the official repository, see Section *Contributing* below.

1.2 License

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License, version 1.3c[2]. The current maintainer is Roberto Di Cosmo (© 2020-2026).

1.3 History

When I decided to start the Software Heritage initiative in 2015, software in general and research software in particular was not yet a first class citizen in the scholarly world. The absence of support for properly citing software in a bibliography was just one of the many signs of this lack of recognition.

In order to properly reference a software project, and ensure that these references are stable enough to pass the test of time, it was necessary to build a *universal archive* for software source code, and to equip every software artifact with *intrinsic* identifiers.

Now that Software Heritage is providing the universal archive [1], with Software Heritage *intrinsic identifiers* (SWH-IDs) widely available [3], it is finally possible to propose proper bibliographic entries for software, at various levels of granularity, down to the line of code.

1.4 Acknowledgments

Thanks to the Inria working group members for their precious feedback and contribution to the desing of the software bibliography entries: Pierre Alliez, Benjamin Guedj, Alain Girault, Morane Gruenpeter, Mohand-Said Hacid, Arnaud Legrand, Xavier Leroy, Nicolas Rougier and Manuel Serrano.

2 Software entries

There are four entry types, corresponding to different granularities in the identification of the (part of) software artifacts that one whishes to cite. They are listed below in order of granularity.

2.1 software

Computer software.

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

Optional fields: `abstract`, `addendum`, `date`, `doi`, `eprint`, `eprintclass`, `eprinttype`, `file`, `hal_id`, `hal_version`, `institution`, `license`, `month`, `note`, `organization`, `publisher`, `related`, `relatedtype`, `relatedstring`, `repository`, `swhid`, `urldate`, `version`

2.2 softwareversion

A specific version of a software. Inherits values of missing fields from the entry mentioned in the `crossref` field.

Required fields: `author` / `editor`, `title`, `url`, `version`, `year`

[2] http://www.latex-project.org/lppl.txt

Optional fields: abstract, addendum, crossref, date, doi, eprint, eprintclass, eprinttype, file, hal_id, hal_version, institution, introducedin, license, month, note, organization, publisher, related, relatedtype, relatedstring, repository, swhid, subtitle, urldate

2.3 softwaremodule

A specific module of a larger software project. Inherits values of missing fields from the entry mentioned in the crossref field.

Required fields: author, subtitle, url, year

Optional fields: abstract, addendum, crossref, date, doi, eprint, eprintclass, eprinttype, editor, file, hal_id, hal_version, institution, introducedin, license, month, note, organization, publisher, related, relatedtype, relatedstring, repository, swhid, title, urldate, version

2.4 codefragment

A code fragment (e.g. a specific algorithm in a program or library). Inherits values of missing fields from the entry mentioned in the crossref field.

Required fields: url

Optional fields: author, abstract, addendum, crossref, date, doi, eprint, eprintclass, eprinttype, file, hal_id, hal_version, institution, introducedin, license, month, note, organization, publisher, related, relatedtype, relatedstring, repository, swhid, subtitle, title, urldate, version, year

The **softwareversion**, **softwaremodule** and **codefragment** entries can inherit the missing fields from another entry designated by the *crossref* field, which is expected to be higher in the granularity hierarchy: **softwareversion** may inherit from a **software** entry, **softwaremodule** may inherit from a **softwareversion** or a **software** entry, and **codefragment** may inherit from all other entries.

3 Field description

The field description is based on the biblatex documentation

3.1 Data fields

abstract field (literal). 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). Additional information to be printed at the end of the entry. The printing of this field can be controlled by the addendum bibliography option.

author list (name). The authors of the title.

date [biblatex only] field (date). The date of creation or release in ISO format.

editor list (name). The coordinator(s) of large modular software projects.

file field (verbatim). A link to download a copy of the work.

doi field (verbatim). The Digital Object Identifier of the work.

eprint [biblatex only] field (verbatim). An electronic identifier of the work. This field can be used to accommodate electronic identifiers different from the ones that have a dedicate field in this style.

eprinttype [biblatex only] field (verbatim). The type of eprint identifier, e. g., the name of the archive, repository, service, or system the eprint field refers to. Will be typeset by default as a prefix of the content of the eprint field.

eprintclass [biblatex only] field (verbatim). 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.

`hal_id` **[not in biblatex standard styles]** field (verbatim). A digital identifier for the software record including its description and metadata on HAL.

`hal_version` **[not in biblatex standard styles]** field (verbatim). The version of the HAL software record designated by `hal_id`.

license [not in biblatex standard styles] list (literal). The license/s of the title in SPDX format.

month field (literal). The month of creation or release. In BibLaTeX, this must be an integer, not an ordinal or a string. For compatibility with BibTeX, one can also use the three letter abbreviations *jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec*, which must be given without any braces or quotes.

note field (literal). Release note of the cited version.

institution list (literal). The institution(s) that took part in the software project.

introducedin [not in biblatex standard styles] field (literal). If this is a software module or fragment, the version of the containing project where it has been first introduced.

organization list (literal). The organization(s) that took part in the software project.

publisher list (literal). The name(s) of the publisher(s) of the *qualified* software record.

related [biblatex only] field (separated values). Citation keys of other entries which have a relationship to this entry.

relatedtype [biblatex only] field (identifier).

relatedstring [biblatex only] field (literal).

repository [not in biblatex standard styles] field (uri). The url of the code repository (e.g on GitHub, GitLab).

swhid [not in biblatex standard styles] field (verbatim). The identifier of the digital object (a.k.a the software artifact itself). The intrinsic identifier of the item is an swh-id (swh:cnt for a content, swh:dir for a directory, swh:rev for a revision, swh:rel for a release, etc.). See the SWH-ID specification.

subtitle field (literal). The title of a component of the software artifact.

title field (literal). The title of the software artifact.

url field (uri). The url of a reference resource (e.g the project's official webpage).

urldate field (date). The access date of the address specified in the url field.

version field (literal). The revision number of a piece of software, a manual, etc.

year field (literal). The year of creation or release.

3.2 Special fields

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.

4 Examples

Here are a few example of use of the proposed entries.

4.1 software and softwareversion

This is an example description of a software release using a single `@softwareversion` entry.

```
@softwareversion {delebecque:hal-02090402-condensed,
  title = {Scilab},
  author = {Delebecque, Fran{\c c}ois and Gomez, Claude and Goursat, Maurice
    and Nikoukhah, Ramine and Steer, Serge and Chancelier, Jean-Philippe},
  url = {https://www.scilab.org/},
  date = {1994-01},
  file = {https://hal.inria.fr/hal-02090402/file/scilab-1.1.tar.gz},
  institution = {Inria},
  license = {Scilab license},
  hal_id = {hal-02090402},
  hal_version = {v1},
  swhid = {swh:1:dir:1ba0b67b5d0c8f10961d878d91ae9d6e499d746a;
           origin=https://hal.archives-ouvertes.fr/hal-02090402},
  version = {1.1},
  note = {First Scilab version. It was distributed by anonymous ftp.},
  repository= {https://github.com/scilab/scilab},
  abstract = {Software for Numerical Computation freely distributed.}
}
```

The same information can also be represented using a `@software` / `@softwareversion` pair that factors out the general information in the `@software` entry, so for other versions only the changes need to be added in a new `@softwareversion` entry:

```
@software {delebecque:hal-02090402,
  title = {Scilab},
  author = {Delebecque, Fran{\c c}ois and Gomez, Claude and Goursat, Maurice
    and Nikoukhah, Ramine and Steer, Serge and Chancelier, Jean-Philippe},
  date = {1994},
  institution = {Inria},
  license = {Scilab license},
  hal_id = {hal-02090402},
  hal_version = {v1},
  url = {https://www.scilab.org/},
  abstract = {Software for Numerical Computation freely distributed.},
  repository= {https://github.com/scilab/scilab},
}

@softwareversion {delebecque:hal-02090402v1,
  version = {1.1},
  date = {1994-01},
  file = {https://hal.inria.fr/hal-02090402/file/scilab-1.1.tar.gz},
  swhid = {swh:1:dir:1ba0b67b5d0c8f10961d878d91ae9d6e499d746a;
           origin=https://hal.archives-ouvertes.fr/hal-02090402},
  note = {First Scilab version. It was distributed by anonymous ftp.},
  crossref = {delebecque:hal-02090402}
}
```

4.2 softwaremodule

For highly modular software projects, like CGAL, one may need to reference specifically a particular module, that has distinguished authors, and may heve been introduced in the project at a later time.

The following example uses the informations in the existing BibTeX entries for CGAL that currently refer to the user manual, to create the corresponding software entries.

```
@software {cgal,
 title = {The Computational Geometry Algorithms Library},
 author = {{The CGAL Project}},
 editor = {{CGAL Editorial Board}},
 date = {1996},
 url = {https://cgal.org/}
}

@softwareversion{cgal:5-0-2,
 crossref = {cgal},
 version = {{5.0.2}},
 url = {https://docs.cgal.org/5.02},
 date = {2020},
 swhid = {swh:1:rel:636541bbf6c77863908eae744610a3d91fa58855;
          origin=https://github.com/CGAL/cgal/}
```

```
}

@softwaremodule{cgal:lp-gi-20a,
 crossref = {cgal:5-0-2},
 author = {Menelaos Karavelas},
 subtitle = {{2D} Voronoi Diagram Adaptor},
 license = {GPL},
 introducedin = {cgal:3-1},
 url = {https://doc.cgal.org/5.0.2/Manual/packages.html#PkgVoronoiDiagram2},
}
```

Of course, it is always be possible to use only one entry to get an equivalent result; here one would use just `@softwaremodule` with all the needed data fields as follows:

```
@softwaremodule{cgal:lp-gi-20a-condensed,
 title = {The Computational Geometry Algorithms Library},
 subtitle = {{2D} Voronoi Diagram Adaptor},
 author = {Menelaos Karavelas},
 editor = {{CGAL Editorial Board}},
 license = {GPL},
 version = {{5.0.2}},
 introducedin = {cgal:3-1},
 date = {2020},
 swhid = {swh:1:rel:636541bbf6c77863908eae744610a3d91fa58855;
          origin=https://github.com/CGAL/cgal/},
 url = {https://doc.cgal.org/5.0.2/Manual/packages.html#PkgVoronoiDiagram2},
}
```

4.3 codefragment

Finally, if one wants to have a particular code fragment appear in the bibliography, we can do this as follows:

```
@software {parmap,
  title = {The Parmap library},
  author = {Di Cosmo, Roberto and Marco Danelutto},
  date = {2012},
  institution = {{Inria} and {University of Paris} and {University of Pisa}},
  license = {LGPL-2.0},
  url = {https://rdicosmo.github.io/parmap/},
  repository= {https://github.com/rdicosmo/parmap},
}

@softwareversion {parmap-1.1.1,
  crossref = {parmap},
  date = {2020},
  version = {1.1.1},
  swhid = {swh:1:rel:373e2604d96de4ab1d505190b654c5c4045db773;
```

```
     origin=https://github.com/rdicosmo/parmap;
     visit=swh:1:snp:2a6c348c53eb77d458f24c9cbcecaf92e3c45615},
}

@codefragment {simplemapper,
  subtitle = {Core mapping routine},
  swhid = {swh:1:cnt:43a6b232768017b03da934ba22d9cc3f2726a6c5;
     origin=https://github.com/rdicosmo/parmap;
     visit=swh:1:snp:2a6c348c53eb77d458f24c9cbcecaf92e3c45615;
     anchor=swh:1:rel:373e2604d96de4ab1d505190b654c5c4045db773;
     path=/src/parmap.ml;
     lines=192-228},
  crossref = {parmap-1.1.1}
}
```

Of course, it is always be possible to use only one entry to get an equivalent result; here one would use just `@codefragment` with all the needed data fields as follows:

```
@codefragment {simplemapper-condensed,
  title = {The Parmap library},
  author = {Di Cosmo, Roberto and Marco Danelutto},
  date = {2020},
  institution = {{Inria} and {University of Paris} and {University of Pisa}},
  license = {LGPL-2.0},
  url = {https://rdicosmo.github.io/parmap/},
  repository= {https://github.com/rdicosmo/parmap},
  version = {1.1.1},
  subtitle = {Core mapping routine},
  swhid = {swh:1:cnt:43a6b232768017b03da934ba22d9cc3f2726a6c5;
     origin=https://github.com/rdicosmo/parmap;
     visit=swh:1:snp:2a6c348c53eb77d458f24c9cbcecaf92e3c45615;
     anchor=swh:1:rel:373e2604d96de4ab1d505190b654c5c4045db773;
     path=/src/parmap.ml;
     lines=192-228}
}
```

5 Use

This package can be used as a standalone on the fly extension, or to produce full bibliographic styles that extend pre-existing styles.

5.1 Use as an *on the fly* extension

The simplest way to use this package is to follow the example given in the `sample-use-sty.tex` that shows how one can *extend on the fly* any existing BibLaTeX style by just doing the following:

- pass the `datamodel=software` option to the `biblatex` package as in

```
\usepackage[datamodel=software]{biblatex}
```

- load the software biblatex style using

```
\usepackage{software-biblatex}
```

- set software specific bibliography options using the macro `\ExecuteBibliographyOptions`; the options with their default values (all shown below) are as in

```
\ExecuteBibliographyOptions{
  halid=true,
  swhid=true,
  shortswhid=false,
  swlabels=true,
  vcs=true,
  license=true,
  addendum=true}
```

  Note that all options default to `true` except `shortswhid` which defaults to `false`.

This is quite handy to add support for software entries in a single article, as it is enough to add `software.dbx`, `software.bbx`, `<language>-software.lbx` and `software-biblatex.sty` to make it work.

5.2 Generate extended styles

When a more systematic use is foreseen, as for institution-wide reports, or conference and journal proceedings, it is more appropriate to generate a new biblatex style that includes support for the software entries right away.

The following simple mechanism is provided for this use case:

- add to the `stublist` file the names of all the existing styles that must be extended
- run `make biblatex-styles` to produce new style files, with an added `+sw` suffix, for each of the existing style
- install the newly generated files in the standard path where BibLaTeX files are found

The stock `stublist` file contains the names of all the standard BibLaTeX styles. If this approach is followed, then one can load directly the extended file, and the software specific bibliography options become available when loading the BibLaTeX package directly. See the `sample.tex` file for a working example.

5.3 Installation

This package may become available in standard distributions like TEXLive as `biblatex-software`. To install manually, you can download it from CTAN and then, put the relevant files in your texmf tree, usually:

```
<texmf>/tex/latex/biblatex-software/software-biblatex.sty
<texmf>/tex/latex/biblatex-software/software.bbx
<texmf>/tex/latex/biblatex-software/software.dbx
<texmf>/tex/latex/biblatex-software/<language>-software.lbx
```

5.4 Package options

The following options are available to control the typesetting of software related entries.

`swlabels=true|false`

Software is a special research output, distinct from publications, hence software entries in a bibliography are distinguished by a special label by default. This behaviour can be disabled by setting this option to `false`.

`license=true|false`

This option controls whether license information for the software entry is typeset. The default is `true`.

`halid=true|false`

This option controls the inclusion of the identifier on the HAL repository of the metadata record for the software described in the entry. The default is `true`.

`swhid=true|false`

This option controls the inclusion of the identifier on the Software Heritage archive (SWHID) of the source code of the software described in the entry. The default is `true`.

`shortswhid=true|false`

This option controls the way the SWHID is displayed. Setting it to true will include only the core part of the SWHID in the printed version, and keep the full SWHID, with all contextual information, in the hyperlink. The default is `false`.

`vcs=true|false`

This option controls the inclusion of the url of the code hosting platform where the software described in the enttry is developed. The default is `true`.

`addendum=true|false`

This option controls whether the addendum field is typeset in software entries. The addendum field can be used to include additional information at the end of a bibliography entry. The default is `true`.

`crossrefexpansion=verbose|conditional`

This option controls how crossref entries are displayed in the bibliography. When a child entry references a parent entry via the `crossref` field, this option determines the formatting behavior:

- `verbose` (default): Always show the full entry with all fields, even if the parent entry is also cited in the bibliography.
- `conditional`: Show a concise format when the parent entry is cited in the bibliography, displaying only fields that differ from the parent along with a citation reference to the parent. If the parent is not cited, show the full entry.

5.5 Entry Relationships and Inheritance

Software bibliography entries can reference parent entries using the `crossref` field, creating inheritance chains where child entries automatically inherit fields from their parents. This enables concise bibliography formatting by showing only the differences between child and parent entries.

5.5.1 Supported Inheritance Chains

The following inheritance chains are supported:

- **2-level**: Software → Version
- **3-level**: Software → Version → Module
- **3-level**: Software → Version → Fragment
- **4-level**: Software → Version → Module → Fragment

The inheritance mechanism works as follows: when a child entry has a `crossref` field pointing to a parent entry, any fields not explicitly set in the child entry are automatically inherited from the parent. The system then compares the child's fields with the parent's fields to determine what to display in the bibliography.

5.5.2 Field Comparison and Concise Format

When `crossrefexpansion=conditional` is set and the parent entry is cited in the bibliography, the system displays only fields that differ from the parent, along with a citation reference to the parent entry. The comparison is performed for the following fields:

- Title, author, editor
- Date (year, month, day components)

- Version
- Institution
- URL and HAL ID
- License
- Entry-specific fields: `swhid`, `repository`, `introducedin`

Fields that match the parent entry are automatically omitted to reduce repetition in the bibliography.

Different entry types use different relationship text in the concise format:

- Versions: “version X of [citation]”
- Modules: “part of [citation]”
- Fragments: “from [citation]”

5.5.3 Examples

The following examples demonstrate crossref functionality using real-world entries from the `biblio.bib` file.

Example 1: Software → Version

Consider the Scilab software and its version:

```
@software {delebecque:hal-02090402,
  title = {Scilab},
  author = {Delebecque, Fran{\c c}ois and Gomez, Claude and Goursat, Maurice
    and Nikoukhah, Ramine and Steer, Serge and Chancelier, Jean-Philippe},
  date = {1994},
  institution = {Inria},
  license = {Scilab license},
  hal_id = {hal-02090402},
  hal_version = {v1},
  url = {https://www.scilab.org/},
  repository = {https://github.com/scilab/scilab},
  abstract = {Software for Numerical Computation freely distributed.}
}

@softwareversion {delebecque:hal-02090402v1,
  version = {1.1},
  date = {1994-01},
  file = {https://hal.inria.fr/hal-02090402/file/scilab-1.1.tar.gz},
  swhid = {swh:1:dir:1ba0b67b5d0c8f10961d878d91ae9d6e499d746a;
           origin=https://hal.archives-ouvertes.fr/hal-02090402},
  note = {First Scilab version. It was distributed by anonymous ftp.},
  crossref = {delebecque:hal-02090402}
}
```

When both entries are cited and `crossrefexpansion=conditional` is set, the version entry will be displayed in concise format showing only the differing fields (version, date, file, swhid, note) along with a reference to the parent entry.

Example 2: Software → Version → Module

The CGAL library demonstrates a 3-level inheritance chain:

```
@software {cgal,
  title = {The Computational Geometry Algorithms Library},
  author = {{The CGAL Project}},
  editor = {{CGAL Editorial Board}},
  date = {1996},
  url = {https://cgal.org/}
}

@softwareversion {cgal:5-0-2,
  crossref = {cgal},
  version = {{5.0.2}},
  url = {https://docs.cgal.org/5.02},
  date = {2020},
  swhid = {swh:1:rel:636541bbf6c77863908eae744610a3d91fa58855;
           origin=https://github.com/CGAL/cgal/}
}

@softwaremodule {cgal:lp-gi-20a,
  crossref = {cgal:5-0-2},
  author = {Menelaos Karavelas},
  subtitle = {{2D} Voronoi Diagram Adaptor},
  license = {GPL},
  introducedin = {cgal:3-1},
  url = {https://doc.cgal.org/5.0.2/Manual/packages.html#PkgVoronoiDiagram2}
}
```

The module entry inherits fields from the version entry, which in turn inherits from the base software entry. When all three are cited, the module will show only its specific fields (subtitle, author, license, introducedin, url) with a "part of [citation]" reference.

Example 3: Software → Version → Fragment

The Parmap library demonstrates a version with a code fragment:

```
@software {parmap,
  title = {The Parmap library},
  author = {Di Cosmo, Roberto and Marco Danelutto},
  date = {2012},
  institution = {{Inria} and {University of Paris} and {University of Pisa}},
  license = {LGPL-2.0},
  url = {https://rdicosmo.github.io/parmap/},
  repository = {https://github.com/rdicosmo/parmap}
}

@softwareversion {parmap-1.1.1,
  crossref = {parmap},
  date = {2020},
  version = {1.1.1},
```

```
  swhid = {swh:1:rel:373e2604d96de4ab1d505190b654c5c4045db773;
     origin=https://github.com/rdicosmo/parmap}
}

@codefragment {simplemapper,
  crossref = {parmap-1.1.1},
  subtitle = {Core mapping routine},
  swhid = {swh:1:cnt:43a6b232768017b03da934ba22d9cc3f2726a6c5;
     origin=https://github.com/rdicosmo/parmap;
     visit=swh:1:snp:2a6c348c53eb77d458f24c9cbcecaf92e3c45615;
     anchor=swh:1:rel:373e2604d96de4ab1d505190b654c5c4045db773;
     path=/src/parmap.ml;
     lines=192-228}
}
```

The code fragment inherits from the version entry, showing only its specific fields (subtitle, swhid with path and lines information) with a "from [citation]" reference.

5.5.4 Known Biber Warnings

When using multiline `swhid` fields with semicolons, Biber may emit "possible runaway string" warnings. These warnings are *false positives* and can be safely ignored.

The warnings occur because BibTeX's parser sees a semicolon followed by a newline in multiline `swhid` fields and becomes cautious, thinking the string might be unclosed. However, the fields are correctly formatted. For example:

```
swhid = {swh:1:dir:1ba0b67b5d0c8f10961d878d91ae9d6e499d746a;
         origin=https://hal.archives-ouvertes.fr/hal-02090402}
```

The `DeclareStyleSourcemap` in `software.bbx` normalizes these fields by removing whitespace, so the multiline format is handled correctly. The warnings do not affect the functionality or correctness of the bibliography output.

5.6 Adding support for additional software identifiers

It would not be reasonable to have a dedicated field for each of the many software releated identifiers that exist. If you want to create bibliographic records that use identifiers not natively supported by this package, you can use the standard BibLaTeX mechanism that uses the `eprint`, `eprinttype` and `eprintclass` fields.

The default formatting of these fields may be what you want, but if it's not the case, you can define your own format, as explained in the official BibLaTeX documentation.

As an example, this style already contains a specific formatting definition for the Astrophysics Source Code Library (ASCL) software records, via the following declaration in the `software.bbx` file:

```
\DeclareFieldFormat{eprint:ascl}{%
 \mkbibacro{ASCL}\addcolon\addspace%
```

```
  \ifhyperref
    {\href{https://ascl.net/#1}{%
         \(\langle\)ascl\addcolon\nolinkurl{#1}\(\rangle\)%
       \iffieldundef{eprintclass}
         {}
         {\addspace\texttt{\mkbibbrackets{\thefield{eprintclass}}}}}}
    {\(\langle\)ascl\addcolon\nolinkurl{#1}\(\rangle\)%
     \iffieldundef{eprintclass}
       {}
       {\addspace\texttt{\mkbibbrackets{\thefield{eprintclass}}}}}
}
```

If you want to adapt this very example to an identifier `foo` with resolver prefix `https://myfoo.org/`, just replace in the LaTeX code above `https://ascl.net/` with `https://myfoo.org/`, `ascl` with `foo` and `ASCL` with `FOO`.

6 Details

The detailed information for this style is contained in the example document and accompanying `.bib` files:

`software-biblatex.tex` This document.

`biblio.bib` An example bibliography showcasing all the software entries.

`sample-use-sty.tex` [3] This document exerces most useful feature of this style extension, using the `biblio.bib` entries.

`sample.tex` This document produces the same output as `sample-use-sty.tex`, but instead of extending on the fly and existing style, it assumes that an extended bibliographic style `numeric+sw` has been created starting from the standard `numeric` style.

`software.bbx` The `biblatex-software` references style.

`software.dbx` The `biblatex-software` data model additions.

`*.lbx` The `biblatex-software` localisation files.

`software-biblatex.sty` The `software-biblatex` LaTeX package for extending on the fly any preloaded Bib LaTeX style.

7 Contributing

This style extension is currently developed on a git-based repository at https://gitlab.inria.fr/gt-sw-citation/bibtex-sw-entry/. Contributions and bug reports are very welcome. In particular, translation of the localization strings for other languages are needed.

[3] `sample-use-sty.pdf` is also provided and is the typeset version of this LaTeX source file.

8 Revision history

bltx-v1.2-8 2025-12-26

Release 1.2-8: Complete crossref support for software entry types

bltx-v1.2-7 2025-12-21

Add addendum field support to software entry types

bltx-v1.2-6 2025-01-05

Updated URL for swMath links, and minor fixes.

bltx-v1.2-5 2022-08-02

Fix mishandling of SWHID short toggle when hyperref is not loaded

bltx-v1.2-4 2022-03-03

Add support for displaying short SWHID

bltx-v1.2-3 2021-08-20

Support backrefs.

bltx-v1.2-2 2020-06-27

Fix handling of related field; use date instead of year/month in examples; add swMATH definition

bltx-v1.2-1 2020-06-01

Fix mishandling of SWHIDs and HALids when hyperref is not loaded. Fix wrong origins in some SWHIDs in the examples. Improve ASCL example.

bltx-v1.2 2020-05-29

Bump version to 1.2 with clean support of multiline SWHIDs

bltx-v1.1 2020-04-29

Add support for the institution, organization, eprint, eprinttype and eprintclass fields Force urls output when they are the only reference available Updates to the documentation

bltx-v1.0 2020-04-25

First public release

bltx-v0.9 2020-04-25

Preparing for public release: Licence, Readme, update documentation, handle suggestions from the Working Group

bltx-v0.8 2020-04-09

Make the style usable as an extension, and keep possibility of generating extended styles

bltx-v0.7 2020-04-09

Move to diff model approach to be more portable

bltx-v0.6 2020-04-08

Standardise file names, make softwarebib.tex self contained, separate out sample.tex, update Makefile, use printdate macro

bltx-v0.5 2020-04-08

Added standard list format for licenses

bltx-v0.4 2020-04-07

Added repository and licence field

bltx-v0.3 2020-04-05

Biblatex style with first complete example

bltx-v0.2 2020-04-02

Biblatex style sent for review

bltx-v0.1 2020-04-02

First version of the biblatex style

References

[1] J.-F. Abramatic, R. Di Cosmo, and S. Zacchiroli. Building the universal archive of source code. *Commun. ACM*, 61(10):29–31, Sept. 2018.

[2] P. Alliez, R. Di Cosmo, B. Guedj, A. Girault, M.-S. Hacid, A. Legrand, and N. Rougier. Attributing and referencing (research) software: Best practices and outlook from inria. *Computing in Science and Engineering*, 22(1):39–52, Jan 2020. Available from https://hal.archives-ouvertes.fr/hal-02135891.

[3] R. Di Cosmo, M. Gruenpeter, and S. Zacchiroli. Referencing source code artifacts: a separate concern in software citation. *Computing in Science and Engineering*, 22(2):33–43, 2020.