`sref.sty`: Semantic Cross-Referencing in LaTeX*

Michael Kohlhase
Jacobs University, Bremen
`http://kwarc.info/kohlhase`

December 12, 2013

Abstract

The `sref` package is part of the STEX collection, a version of TEX/LATEX that allows to markup TEX/LATEX documents semantically without leaving the document format, essentially turning TEX/LATEX into a document format for mathematical knowledge management (MKM).

The `sref` package supplies an for semantic cross-referencing over multiple documents.

Contents

---

*Version v1.1 (last revised 2013/10/15)

1 Introduction

The automatic computation of cross-references is one of the traditional strong points of LaTeX. However, cross-referencing is limited to labels in the current document only. Cross-referencing between multiple documents in a jointly developed document collection is not easy to achieve in the LaTeX processing model, which reads files sequentially and lacks a path concept.

The `sref` package is mainly aimed at package developers. It supplies the internal macros that can be used to make document structuring elements cross-referencable. The general strategy here is to equip the document structuring
id   macros with an `id` key, so that the author can specify meaningful ones, but to let the transformation give default ones if the author did not. The value of the `id` key can also be used for cross-referencing like the `\label`/`\ref` mechanism in LaTeX. We implement an independent referencing mechanism, since the referencing model is geared more towards referencing text fragments than text fragment labels like section numbers. Therefore we let the referenced fragments define the reference text much like the `\autoref` macro from `\hpyerref`.

2 The User Interface

This package is currently mainly meaningful in the context of the sTeX collection, since all cross-referencable macros and environments must be extended to know about their referencing mechanism. We explain the user interface in Section 2.2. To port the functionality to other LaTeX classes and packages, they have to be upgraded via the API in Section 2.3.

2.1 Package Options

extrefs   The `sref` package has the `extrefs` package option, which can be set to activate
showmeta   multifile support (see Section 2.4). If the `showmeta` is set, then the metadata keys are shown (see [Koh12] for details and customization options).

2.2 Cross-Referencing

\sref   The `\sref{⟨id⟩}` macro is the main cross-referencing macro, see Figure 1 for an example. Depending on the whether macro or environment marking up the respective document fragment carries the key/value pair `id=⟨id⟩` the cross-reference will expand to "Section 2.1" or "this remark", both carrying hyper-references. The `\sref` macro takes optional arguments allows to customize its behavior the full form of `\sref` is: `\sref[⟨text⟩]{⟨id⟩}[⟨fallback⟩]`, where ⟨link⟩ can be used to specify a link text that overrides the auto-generated one and ⟨fallback⟩ gives the fallback text when the label ⟨id⟩ is not defined.
\sreflabel   The `\sreflabel`[1] macro is a variant to the `\label` macro provided by LaTeX

---

[1]It would have been more natural to name the macro `slabel`, but this is overwritten by other packages without warning.

```
\mysection[id=foo]{#2}
... \sref{foo} ...
... \sref[this section]{foo} ...
... \sref{foo}[above] ...
```

Example 1: Semantic Crossreferencing

proper. It takes two arguments, the first one is a classification (used in `\sref`) and the second one the identifier.

\srefs The `\srefl{`⟨*id¹*⟩`}{`⟨*id²*⟩`}` is a variant it `\sref`, only that it allows to reference two semantic objects and expands to "⟨*reference¹*⟩ and ⟨*reference²*⟩".

\srefl `\srefl{`⟨*id¹*⟩`}{`⟨*idⁿ*⟩`}` is similar, but for ranges; it expands to "⟨*reference¹*⟩ to ⟨*referenceⁿ*⟩". Its use should be restricted to cases, where the types of objects references are homogenous.

\spageref Finally, there is a variant `\spageref` that only outputs the page number of the referenced object. It can be used in cases where no hyper-referencing is present.

\sref@page@label It uses the macro `\sref@page@label` for styling the page reference. Redefining this will allows to customize this. The default setting is

```
\newcommand\sref@page@label[1]{p.~{#1}}
```

2.3 An API for Package Authors

To make use of the `sref` package, the package must define the document structuring infrastructure using the `sref` internal macros. The STEX packages already does this, so we make an example here for a slightly upgraded sectioning command in Figure 2. The first three lines define the keys for the keyval attribute of the `\mysection` command using the infrastructure supplied by the `omd` package [Koh12] (remember the `\RequirePackage{metakeys}`). The first two just initialize the keys to save the key values in internal macros, and the `\metasetkeys` activates the keys when reading the keyval argument. The `\srefaddidkey` macro

\srefaddidkey is a variant of `\addmetakey` macro supplied by the `sref` package that sets up the

\sref@id keys to set the `\sref@id` register for later use by the `sref` infrastructure. Note that the `\srefaddidkey` macro uses the `prefix` key to systematically construct prefixed identifiers. This can be useful in particular for sectioning commands.

```
\addmetakey{sec}{short}
\addmetakey[black]{sec}{color}
\srefaddidkey[prefix=sec.]{sec}
\newcommand\mysection[2][]{\metasetkeys{#1}\sref@target\color{\sec@color}
\section[\sec@short]{#2}\sref@label@id{Section \thesection}}
```

Example 2: A slightly upgraded sectioning command

In this situation, the `\mysection` macro processes the optional argument with

\sref@target `\metasetkeys` and then sets the color of the section. The `\sref@target` sets

3

up the hypertarget for the `hyperref` package to use. Then we use the regular `\section` command, and we use the `\sref@label@id` macro to define the label that the `\sref` macro will use for cross-referencing.

Note that the use of the straight use of the label "Section", which will be written into the auxiliary files is bad practice since it is not configurable. It would be much better to make it configurable via a presentation macro like `\my@section@label` in Figure 3. Then translators or even the user could redefine the `\my@section@label` to adapt them to their needs.

```
\newcommand\my@section@label[1]{Section~{#1}}
\newcommand\mysection[2][]{\metasetkeys{#1}\sref@target\color{\sec@color}
\section[\sec@short]{#2}\sref@label@id{\my@section@label\thesection}}
```

Example 3: A Sectioning Command with Configurable Label

2.4 Inter-Document Cross-Referencing

`sref.sty` provides inter-document cross-referencing. The use case is simple: we want to have a document collection (e.g. a book with conference proceedings), but also want to be able format the individual documents separately, and still have meaningful cross-references. To show off the possibilities, let us assume that we have a book with two separate papers, which we put into separate directories `idc` and `scr` to minimize interference between the authors Jane Doe and John Bull. To achieve this, we would set up paper driver files `main.tex` like the one \makeextrefs in Figure 4 in the two directories. These use the `\makeextrefs` macro, which causes the `sref` package to generate a *external references file* `main.refs`. Note that the `\makeextrefs` macro reads the previous `main.refs` file so that forward-referencing is possible (in the pass after a reference was labeled).

```
\documentclass{article}
\usepackage[extrefs]{sref}
\makeextrefs{idc}
\inputrefs{scr}{../scr/main}
\extrefstyle{scr}{\cite[\protect{\theextref}]{Doe09}}
\title{Inter-Document Crossreferencing}
\author{John Bull\\...}
\begin{document}\maketitle\input{paper}\end{document}
```

Example 4: A document driver `idc/main.tex` for a paper

The external references file can be read by other documents; in Figure 4, we \inputrefs read the references file of Jane Doe's paper via the `\inputrefs` macro. This \extref allows John Bull to use[2] references like `\extref{scr}{foo}` to reference document fragments in Jane Doe's paper she has labeled with the *reference prefix* `\sreflabel{foo}` (assuming that she has added `\makeextrefs{scr}` in the

---

[2]Note that the external references file is updated every time LATEX is run, so that references may be off by one version.

preamble of her paper). Note that just as the `\sref` macro `\extref` takes an optional first argument that allows to specify the link text. Here, John Bull uses the `\extrefstyle` macro to specify how the externa references are to be formatted, in this case he decided to use a LaTeX citation. Generally, first argument of the `\extrefstyle` macro is the reference prefix which should be configured, and the second is the format, where the `\theextref` macro expands to the cross-reference. In this case, John chose to use a bibTeX citation (he has an entry `Doe09` in his database) for the reference to the external paper.

As the content of the respective paper is input from a file `paper.tex` in the individual papers, we can re-use these in the book. To do this we set up a book driver file like the one in Figure 5. This one does not use the `extrefs` option, so the references are written to the `.aux` file. Furthermore `\extref` is redefined to act like `\sref` disregarding the first required argument. Thus all references work like they should.

```
\documentclass{book}
\usepackage{sref}
\title{Cross-Referencing in {\LaTeX}}
\author{Elder Reseacher}
\begin{document}
\maketitle
\chapter{Semantic Crossreferencing (Jane Doe, ...)}
\input{scr/paper}\newpage
...
\chapter{Inter-Document Crossreferencing (John Bull, ...)}
\input{idc/paper}\newpage
\end{document}
```

Example 5: A document driver for the book assembling the papers

This example has been carried through (without the separation of chapters in to subdirectories) in the files accompanying the source distribution of the `sref` package. They are used for testing the package.

2.5 Semantic Versions of Commonly used Referencing Commands

The `sref` package defines semantically referencable versions of commonly used LaTeX environments and command sequences.[3]

sequation   The `sequation` environment takes an optional key/value argument that allows to specify an identifier and unifies the behavior of the `equation` (if an `id` key is given) and `displaymath` (else) environments. So the markup

---

[3]This section will be extended by need, so if you miss some semantic environment, please contact the package author, or (better) file an issue at [sTeX])

```
 A semantic equation with id
 \begin{sequation}[id=foo]
   e^{mc}=-1
 \end{sequation}
 and another one without id
 \begin{sequation}
   e^{mc}=-1
 \end{sequation}
 now, we reference the first equation: {\sref{foo}}
```

yields the result:

A semantic equation with id

$$e^{mc} = -1 \tag{1}$$

and another one without id

$$e^{mc} = -1$$

now, we reference the first equation: equation (1)

Example 6: Semantic Equation

2.6 Semantic Citations

bibT$_{\text{E}}$X [Pat] and bibL^{A}T$_{\text{E}}$X [Leh10] provide a semi-semantic way of referencing literature. If we look at the current practice of citing from an RDF standpoint [LS99]which views links as subject/predicate/object triples, then the treatment of the predicate and object are semantic, but the subject is hinted at by mere juxtaposition in the text. The `sref` package helps out here via the macro for short \withcite subjects (in the second argument) that are postfixed by the citation (key in the first argument). For instance the occurrence at the beginning of this paragraph was created by

```
\withcite{Patashnik:b88}{bib\TeX}
```

withcitation The general case is covered by the `withcitation` environment for long subjects. \citeit In the latter, the citation can be placed by the . For instance, the second sentence was marked up as

```
If we look at the
\begin{withcitation}{LasSwi:rdf99}
   current practice of citing from an RDF standpoint which views links as
   subject/predicate/object triples,
\end{withcitation}
then the treatment of the predicate ...
```

The advantage of this treatment is that the meaning of the reference is fully marked up and can be taken advantage of in the OMDoctransformation, from which RDF triples can then be harvested for a linked open data treatment.

3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTeX TRAC [sTeX].

1. currently only the `\sref` macro has a fallback argument. The others `\srefs` and `\srefl` and their external variants should also have them, but I am not clear what the adequate invocation pattern would be.

4 Implementation

The `sref` package generates two files: the LATEX package (all the code between ⟨*package⟩ and ⟨/package⟩) and the LATEXML bindings (between ⟨*ltxml⟩ and ⟨/ltxml⟩). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

We first set up header information for the LATEXML binding file.

```
1 ⟨*ltxml⟩
2 package LaTeXML::Package::Pool;
3 use strict;
4 use LaTeXML::Package;
5 ⟨/ltxml⟩
```

4.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).[1]

```
6 ⟨*package⟩
7 \DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
8 \newif\ifextrefs\extrefsfalse
9 \DeclareOption{extrefs}{\extrefstrue}
10 \ProcessOptions
11 ⟨/package⟩
12 ⟨*ltxml⟩
13 DeclareOption('extrefs','');
14 ⟨/ltxml⟩
```

Then we need to set up the packages by requiring the `metakeys` package [Koh12] to be loaded (in the right version).

```
15 ⟨*package⟩
16 \RequirePackage{metakeys}
17 \RequirePackage{xspace}
18 ⟨/package⟩
19 ⟨*ltxml⟩
20 RequirePackage('metakeys');
21 ⟨/ltxml⟩
```

4.2 Crossreferencing

The following user-level macros just use the `\sref@hlink` macros in various ways for internal referencing.[2]

`\sref`

EdN:1

EdN:2

---

[1] EDNOTE: need an implementation for LATEXML
[2] EDNOTE: they need implementation in LaTeXML, the ones here only are stubs to make the error messages shut up.

```
22 ⟨∗package⟩
23 \def\sref{\@ifnextchar[{\@sref}{\@sref[]}}
24 \def\@sref[#1]#2{\@ifnextchar[{\@@sref[#1]{#2}}{\@sref[#1]{#2}[]}}
25 \def\@@sref[#1]#2[#3]{\@ifundefined{sref@part}{\sref@hlink[#1]{#2}{#3}}{\sref@hlink[#1]{\sref@p
26 ⟨/package⟩
27 ⟨∗ltxml⟩
28 sub withhash {'#';}
29 DefConstructor('\sref[]{}[]',
30  "<omdoc:oref href='&withhash()#2'/>");
31 ⟨/ltxml⟩
```

\srefs

```
32 ⟨∗package⟩
33 \newcommand\srefs[3][]{%
34 \def\@test{#1}\ifx\@test\@empty\sref{#2} and \sref{#3}\else #1\fi}
35 ⟨/package⟩
36 ⟨∗ltxml⟩
37 DefConstructor('\srefs[]{}',
38  "<omdoc:oref href='&withhash()#2'/>");
39 ⟨/ltxml⟩
```

\srefl

```
40 ⟨∗package⟩
41 \newcommand\srefl[3][]{%
42 \def\@test{#1}\ifx\@test\@empty\sref{#2} to \sref{#3}\else #1\fi}
43 ⟨/package⟩
44 ⟨∗ltxml⟩
45 DefConstructor('\srefl[]{}',
46  "<omdoc:oref href='&withhash()#2'/>");
47 ⟨/ltxml⟩
```

EdN:3   3

\spageref

```
48 ⟨∗package⟩
49 \newcommand\spageref[1]{%
50 \@ifundefined{sref@part}{\sref@pageref{#1}}{\sref@pageref{\sref@part @#1}}}
51 ⟨/package⟩
52 ⟨∗ltxml⟩
53 DefConstructor('\spageref{}',
54  "<omdoc:oref href='&withhash()#1'/>");
55 ⟨/ltxml⟩
```

4.3   An API for Package Authors

We find out whether the hyperref package is loaded, since we may want to use
it for cross-references, for which we set up some internal macros that gracefully
degrade if hyperref is not loaded.

---

[3]EDNOTE: it is not clear what we want in the LaTeXML implementation of spageref

9

```
56 ⟨∗package⟩
57 \newif\ifhref\hreffalse
58 \AtBeginDocument{\@ifpackageloaded{hyperref}{\hreftrue}{\hreffalse}}
59 \newcommand\sref@href@ifh[2]{\ifhref\href{#1}{#2}\else#2\fi}
60 \newcommand\sref@hlink@ifh[2]{\ifhref\hyperlink{#1}{#2}\else#2\fi}
61 \newcommand\sref@target@ifh[2]{\ifhref\hypertarget{#1}{#2}\else#2\fi}
```

Then we provide some macros for sTeX-specific crossreferencing

\sref@target The next macro uses this and makes an target from the current `sref@id` declared by a `id` key.

```
62 \def\sref@target%
63 {\ifx\sref@id\@empty\else%
64 \edef\@target{sref@\@ifundefined{sref@part}{}{\sref@part @}\sref@id @target}
65 \sref@target@ifh\@target{}\fi}
```

The next two macros are used for setting labels, it is mainly used for enabling forward references, to do this, it is written into ⟨*jobname*⟩.`aux` or ⟨*jobname*⟩.`refs`.

\@sref@def This macro stores the value of its last argument in a custom macro for reference.

```
66 \newcommand\@sref@def[3]{\expandafter\gdef\csname sref@#1@#2\endcsname{#3}}
```

The next step is to set up a file to which the references are written, this is normally the .`aux` file, but if the `extref` option is set, we have to use an .`ref` file.

```
67 \ifextrefs\newwrite\refs@file\else\def\refs@file{\@auxout}\fi
```

\sref@def This macro writes an `\@sref@def` command to the current aux file and also executes it.

```
68 \newcommand\sref@def[3]{%\@sref@def{#1}{#2}{#3}%
69 \protected@write\refs@file{}{\string\@sref@def{#1}{#2}{#3}}}
```

\srefaddidkey `\srefaddidkey[`⟨*keyval*⟩`]{`⟨*group*⟩`}` extends the metadata keys of the group ⟨*group*⟩ with an `id` key. In the optional key/value pairs in ⟨*keyval*⟩ the `prefix` key can be used to specify a prefix. Note that the `id` key defined by `\srefaddidkey[`⟨*keyval*⟩`]{`⟨*group*⟩`}` not only defines `\sref@id`, which is used for referencing by the `sref` package, but also `\`⟨*group*⟩`@id`, which is used for showing metadata via the `showmeta` option of the `metakeys` package.

```
70 \addmetakey{srefaddidkey}{prefix}
71 \newcommand\srefaddidkey[2][]{\metasetkeys{srefaddidkey}{#1}%
72 \@metakeys@ext@clear@keys{#2}{sref@id}{}% id cannot have a default
73 \metakeys@ext@clear@keys{#2}{id}{}%
74 \metakeys@ext@showkeys{#2}{id}%
75 \define@key{#2}{id}{\edef\sref@id{\srefaddidkey@prefix ##1}%
76 \expandafter\edef\csname #2@id\endcsname{\srefaddidkey@prefix ##1}}}
77 ⟨/package⟩
```

4.4 Inter-Document Crossreferencing

\makeextrefs

```
78 ⟨*package⟩
79 \newcommand\makeextrefs[1]{\gdef\sref@part{#1}%
80 \makeatletter
81 \IfFileExists{\jobname.refs}{\input{\jobname.refs}}{}%
82 \immediate\openout\refs@file=\jobname.refs
83 \makeatother}
84 ⟨/package⟩
85 ⟨ltxml⟩DefConstructor('\makeextrefs{}','');
```

\sref@label   The \sref@label macro writes a label definition to the auxfile.

```
86 ⟨*package⟩
87 \newcommand\sref@label[2]{%
88 \sref@def{\@ifundefined{sref@part}{}{\sref@part @}#2}{page}{\thepage}%
89 \sref@def{\@ifundefined{sref@part}{}{\sref@part @}#2}{label}{#1}}
90 ⟨/package⟩
```

\sreflabel   The \sreflabel macro is a semantic version of \label, it combines the categorization given in the first argument with LaTeX's \@currentlabel.

```
91 ⟨*package⟩
92 \newcommand\sreflabel[2]{\sref@label{#1 \@currentlabel}{#2}}
93 ⟨/package⟩
```

\sref@label@id   The \sref@label@id writes a label definition for the current \sref@id if it is defined.

```
94 ⟨*package⟩
95 \newcommand\sref@label@id[1]{\ifx\sref@id\@empty\else\sref@label{#1}{\sref@id}\fi}
96 ⟨/package⟩
```

EdN:4   Finally we come to the user visible macro \sref which is used for referencing.[4]

\sref@hlink   \sref@hlink[⟨alt⟩]{⟨label⟩}{⟨fallback⟩} creates an error message if the target specified by ⟨label⟩ is not defined (but uses ⟨fallback⟩ if provided), and otherwise generates a hyperlinked reference whose link text is ⟨alt⟩ (if the optional argument is given) and the label generated by object specified by ⟨label⟩ otherwise.

```
97 ⟨*package⟩
98 \newcommand\sref@hlink[3][]{\def\@test{#1}\def\sref@ifundef{#3}%
99 \@ifundefined{sref@#2@label}%
100 {\ifx\sref@ifundef\@empty%
101 \protect\G@refundefinedtrue\@latex@warning{reference #2 undefined}?#2?%
102 \else%
103 \protect\G@refundefinedtrue\@latex@warning{using fallback for undefined reference #2}#3%
104 \fi}%
105 {\sref@hlink@ifh{sref@#2@target}{\ifx\@test\@empty\@nameuse{sref@#2@label}\else #1\fi}}}
106 ⟨/package⟩
```

---

[4]EDNOTE: The LaTeXML does not take into account the optional argument yet.

`\sref@page@label` This macro styles a page reference.

```
107 ⟨*package⟩
108 \newcommand\sref@page@label[1]{p.~{#1}}
109 ⟨/package⟩
```

`\sref@pageref` The next macro creates an error message if the target is not defined, and otherwise generates a page reference.

```
110 ⟨*package⟩
111 \newcommand\sref@pageref[1]{\@ifundefined{sref@#1@page}%
112 {\protect\G@refundefinedtrue\@latex@warning{reference #1 undefined}\sref@page@label{??}}%
113 {\sref@hlink@ifh{sref@#1@target}{\sref@page@label{\@nameuse{sref@#1@page}}}}}
114 ⟨/package⟩
```

`\sref@href` The next macro creates an error message if the target is not defined, and otherwise generates a hyperlinked reference.

```
115 ⟨*package⟩
116 \newcommand\sref@href[3][]{\def\@test{#1}%
117 \@ifundefined{sref@#2@label}%
118 {\protect\G@refundefinedtrue\@latex@warning{reference #2 undefined}??}%
119 {\@ifundefined{sref@#3@URI}%
120 {\protect\G@refundefinedtrue\@latex@warning{external refs of type  #3 undefined}??}%
121 {\edef\@uri{\@nameuse{sref@#3@URI}.pdf\#sref@#2@target}
122 \edef\@label{\ifx\@test\@empty\@nameuse{sref@#2@label}\else #1\fi}
123 \sref@href@ifh\@uri\@label}}}
124 ⟨/package⟩
```

`\extref` The next macros use `\sref@href` with the respective prefix for external referencing if external references are used as indicated by the `extrefs` option; otherwise it
EdN:5 disregards the first required macro and uses internal referencing.[5]

```
125 ⟨*package⟩
126 \ifextrefs
127 \newcommand\extref[3][]{\def\theextref{\sref@href[#1]{#2@#3}{#2}}%
128 \csname doextref@#2\endcsname}
129 \else
130 \newcommand\extref[3][]{\sref[#1]{#3}}
131 \fi
132 ⟨/package⟩
133 ⟨*ltxml⟩
134 DefConstructor('\extref[]{}{}',
135   "<omdoc:oref href='#2@#3'/>");
136 DefConstructor('\theextref',''');
137 ⟨/ltxml⟩
```

`\extpageref` The next macros use `\sref@pagref` with the respective prefix for external referencing if external references are used as indicated by the `extrefs` option; otherwise
EdN:6 it disregards the first required macro and uses internal referencing.[6]

---

[5]EDNOTE: This needs to be implemented on the LaTeXML side.
[6]EDNOTE: This needs to be implemented on the LaTeXML side.

```
138 ⟨∗package⟩
139 \ifextrefs
140 \newcommand\extpageref[3][]{\def\theextref{\sref@pageref{#2@#3}}%
141 \csname doextpageref@#2\endcsname}
142 \else
143 \newcommand\extpageref[3][]{\spageref{#3}}
144 \fi
145 ⟨/package⟩
146 ⟨∗ltxml⟩
147 DefConstructor('\extpageref[]{}{}',
148   "<omdoc:oref href='#2@#3'/>");
149 DefConstructor('\theextref','');
150 ⟨/ltxml⟩
```

\extrefstyle This user macro defines an internal macro that is used for internal styling; for instance \extrefstyle{foo}{\theextref in bar} defines the macro \doextref@foo which evaluates to ⟨*the reference*⟩ in bar. This is used in the \extref macro.

```
151 ⟨∗package⟩
152 \newcommand\extrefstyle[2]{\expandafter\gdef\csname doextref@#1\endcsname{#2}}
153 ⟨/package⟩
154 ⟨∗ltxml⟩
155 DefConstructor('\extrefstyle{}{}',"");
156 ⟨/ltxml⟩
```

\extpagerefstyle This is analogous to \extrefstyle

```
157 ⟨∗package⟩
158 \newcommand\extpagerefstyle[2]{\expandafter\gdef\csname doextpageref@#1\endcsname{#2}}
159 ⟨/package⟩
160 ⟨∗ltxml⟩
161 DefConstructor('\extrefstyle{}{}',"");
162 ⟨/ltxml⟩
```

\inputrefs If the external references file exists, it is read (under the protection of \makeatother) otherwise an error message is displayed.

```
163 ⟨∗package⟩
164 \newcommand\inputrefs[2]{%
165 \@namedef{sref@#1@URI}{#2}%
166 \extrefstyle{#1}{\theextref}\extpagerefstyle{#1}{\theextref}%
167 \makeatletter%
168 \IfFileExists{#2.refs}{\message{Reading external references: #2.refs}\input{#2.refs}}
169                {\PackageError{sref}{Reference file #2.refs does not exist}
170                                 {Maybe you have to run LaTeX on #2.tex first}}
171 \makeatother}
172 ⟨/package⟩
173 ⟨∗ltxml⟩
174 DefConstructor('\inputrefs{}{}','');
175 ⟨/ltxml⟩
```

4.5 Semantic Versions of Commonly used Referencing Commands

sequation

```
176 ⟨∗package⟩
177 \srefaddidkey{sequation}
178 \def\sref@sequation@heading{equation}
179 \newenvironment{sequation}[1][]{\metasetkeys{sequation}{#1}%
180 \ifx\sref@id\@empty\begin{displaymath}\else% no id, using equation*
181 \begin{equation}\sref@target\sref@label@id{\sref@sequation@heading~(\theequation)}\fi}
182 {\ifx\sref@id\@empty\end{displaymath}\else\end{equation}\fi}
183 ⟨/package⟩
184 ⟨∗ltxml⟩
185 DefEnvironment('{sequation} OptionalKeyVals',
186       "<ltx:equation "
187             .     "?&KeyVal(#1,'id')(xml:id='&KeyVal(#1,'id')' "
188             .     "refnum='#refnum')(xml:id='#id')>"
189       .   "<ltx:Math mode='display'>"
190       .     "<ltx:XMath>#body</ltx:XMath>"
191       .   "</ltx:Math>"
192     . "</ltx:equation>",
193       mode=>'display_math',
194       properties=> sub { RefStepCounter('equation') },
195       locked=>1);
196 ⟨/ltxml⟩
```

seqnarray

```
197 ⟨∗package⟩
198 \newenvironment{seqnarray}[1][]%
199 {\metasetkeys{sequation}{#1}\begin{eqnarray*}\sref@target%
200 \sref@label@id{\sref@sequation@heading~(\theequation)}}
201 {\end{eqnarray*}}
202 ⟨/package⟩
203 ⟨∗ltxml⟩
204 DefMacro('\seqnarray OptionalKeyVals','\begin{eqnarray*}');
205 DefMacro('\endseqnarray','\end{eqnarray*}');
206 ⟨/ltxml⟩
```

4.6 Semantic Citations

withcite

```
207 ⟨∗package⟩
208 \newcommand\withcite[2]{#2~\cite{#1}}
209 ⟨/package⟩
210 ⟨∗ltxml⟩
211 DefMacro('\withcite{}{}','\begin{withcitation}{#1}#2\citeit\end{withcitation}');
212 ⟨/ltxml⟩
```

withcitation

```
213 ⟨∗package⟩
214 \newenvironment{withcitation}[1]{\def\citeit{\cite{#1}}}{}
215 ⟨/package⟩
216 ⟨∗ltxml⟩
217 DefConstructor('\citeit',"<omdoc:citation/> ",
218 afterConstruct => sub {
219   my ($document,$whatsit) = @_;
220   # LibXML acrobatics, since we can't talk about the xml:id prior to construction's end
221   # (and please do correct me if this is inaccurate)
222   my $node = $document->getNode;
223   my ($citenode) = $document->findnodes('preceding-sibling::omdoc:citation',$node);
224 my ($phrase_parent) = $document->findnodes('ancestor::ltx:text[@xml:id]',$node);
225   return unless (defined $phrase_parent) && (defined $citenode);
226   my $id = $phrase_parent->getAttribute('xml:id');
227   my $refs = $phrase_parent->getAttribute('citeit-refs');
228   $phrase_parent->removeAttribute('citeit-refs');
229   $citenode->setAttribute('for',$id);
230   $citenode->setAttribute('refs',$refs);
231 });#$
232 DefEnvironment('{withcitation}{}',
233   "<ltx:text citeit-refs='#1'>#body</ltx:text>");
234 ⟨/ltxml⟩
```

4.7 Finale

Finally, we need to terminate the file with a success mark for perl.

```
235 ⟨∗ltxml⟩
236 1;
237 ⟨/ltxml⟩
```

References

[Koh12]   Michael Kohlhase. `metakeys.sty`: *A generic framework for extensible Metadata in LaTeX*. Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2012. URL: `http://www.ctan.org/tex-archive/macros/latex/contrib/stex/metakeys/metakeys.pdf`.

[Leh10]   Philipp Lehmann. *The `biblatex` Package*. Tech. rep. CTAN: Comprehensive TeX Archive Network, 2010. URL: `http://www.ctan.org/tex-archive/macros/latex/exptl/biblatex/doc/biblatex.pdf`.

[LS99]   Ora Lassila and Ralph R. Swick. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation. World Wide Web Consortium (W3C), 1999. URL: `http://www.w3.org/TR/1999/REC-rdf-syntax`.

[Pat]   Oren Patashnik. *bibTeXing*. URL: `http://www.ctan.org/get/biblio/bibtex/contrib/doc/btxdoc.pdf` (visited on 12/14/2009).

[sTeX]   *Semantic Markup for LaTeX*. Project Homepage. URL: `http://trac.kwarc.info/sTeX/`.