

The sblidx Package

David Purton*

2026-04-04 v1.0

Abstract

sblidx provides a \LaTeX package for creating indices in a style recommended by the Society of Biblical Literature as outlined in the *Society of Biblical Literature Handbook of Style* and *Preparing Indices*.¹ This includes producing an index of ancient sources with the help of the `biblatex-sbl` package, an index of modern authors with the help of the `biblatex-sbl` package and an index of subjects. Page ranges are automatically compressed and when indexed entries appear in footnotes they are indicated with n. and nn.

Contents

1	Introduction	2
1.1	Usage	2
1.2	Bug Reports and Feature Requests	2
2	Package Options	3
3	Commands	4
4	Implementation	4
4.1	Custom Index Style File	4
4.2	Main Package	5
4.2.1	Define and Process Package Options	5
4.2.2	Set Up Indices	5
4.2.3	Page and Note Compression Code	10
4.2.4	Index Processing Code	17

*Email: dcpurton@marshwiggles.net

¹See *The SBL Handbook of Style: For Biblical Studies and Related Disciplines*, 2nd ed. (SBL Press, 2014); SBL Publications, *Preparing Indices*, https://www.sbl-site.org/wp-content/uploads/2024/05/Indexing_SBL.pdf.

1 Introduction

The Society of Biblical Literature has specific requirements for indexes. Typically there are separate indices for subjects, modern authors and ancient sources. And these different indices are laid out in a slightly different way. Additionally page ranges are compressed and index references in footnotes are indicated with n. and nn. This package attempts to handle these variations with as much automation as possible.

A subject index is automatically enabled and indices of modern authors and ancient sources can optionally be added. Modern authors are indexed when they are cited using the `biblatex-sbl` package and books of the Bible are indexed using the `bibleref-sbl` package. The different format of these three indices as required by the SBL is handled automatically.

1.1 Usage

A simple example document is shown in Figure 1 and the output is shown in Figure 2.

Note: You should load `sblidx` *after* you load `biblatex`.

1.2 Bug Reports and Feature Requests

Bug reports and feature requests can be made at the `sbltex` GitHub repository. See <https://github.com/dcpurton/sbltex>.

```
\documentclass{article}
\begin{filecontents}{\jobname.bib}
@book{talbert:1992,
  author = {Talbert, Charles H.},
  title = {Reading John},
  subtitle = {A Literary and Theological Commentary on the Fourth
             Gospel and the Johannine Epistles},
  location = {New York},
  publisher = {Crossroad},
  date = {1992}
}
\end{filecontents}
\usepackage[style=sbl]{biblatex}
\addbibresource{\jobname.bib}
\usepackage[subject, ancient sources, modern authors]{sblidx}
\begin{document}
A topic\index{topic}. A reference to \ibibleverse{John}(1:1)
\autocite{talbert:1992}. Another topic.\footnote{with an index
entry in the footnote.\index{another topic}.} And an example of a
cross reference in an index.\index{entry|see{topic}}
\SBLPrintIndices
\end{document}
```

Figure 1: Example L^AT_EX source showing subject, ancient sources and modern authors indices.

Ancient Sources Index		
New Testament		
John		
1:1		1
Modern Authors Index		
Talbert, Charles H.		1 n. 1
Subject Index		
another topic, 1 n. 2		
entry. <i>See</i> topic		
topic, 1		

Figure 2: Example output showing subject, ancient sources and modern authors indices.

2 Package Options

ancient_sources ancient sources = true | false Default: false

Enable an index of ancient sources. Bible books (including deuterocanonical books) are indexed in SBL style use the `bibleref-sbl`, `bibleref-parse` and `bibleref` packages.

ancient_sources_title ancient sources title = *<title>* Default: Ancient Sources Index

Set the title of the ancient sources index.

modern_authors modern authors = true | false Default: false

Enable an index of modern authors. Modern authors are indexed when cited using `biblatex-sbl`.

modern_authors_title modern authors title = *<title>* Default: Modern Authors Index

Set the title of the modern authors index.

subject subject = true | false Default: true

Enable an index of subjects. Subjects are indexed using `makeindex` and the standard `\index` macro. using `biblatex-sbl`.

subject_title subject title = *<title>* Default: Subject Index

Set the title of the subject index.

3 Commands

<hr/> <hr/> <code>\SBLFootnoteIndex</code>	<code>\SBLFootnotes [<i><name></i>] {<i><entry></i>}</code>						
	<p><code>\index</code> is defined to this macro within footnotes so that note numbers are automatically included unless some other formatting is requested by the user. <i><name></i> is the name of the raw index file and <i><entry></i> should be formatted according to syntax required by <code>makeindex</code>. In general this macro should not be called directly as it automatically includes the current value of the <code>footnote</code> counter.</p>						
<hr/> <hr/> <code>\SBLPageWithNote</code>	<code>\SBLPageWithNote {<i><comma separated list of notes></i>} {<i><page></i>}</code>						
	<p>Output page with note number using n. or nn. depending on how many notes there are. When <code>\index</code> appears in a footnote, this command is called automatically using the <code>makeindex \index{gnat SBLPageWithNote{\thefootnote}}</code> syntax.</p>						
<hr/> <hr/> <code>\SBLPrintIndices</code>	<code>\SBLPrintIndices</code>						
	<p>Print the indices. This command prints one or more indices depending on which are enabled with the <code>ancient sources</code>, <code>modern authors</code> and <code>subject</code> options.</p>						
<hr/> <hr/> <code>\SBLProcessIndexPages</code>	<code>\SBLProcessIndexPages {<i><list of pages></i>}</code>						
	<p>Process a list of pages produced by <code>makeindex</code>, compressing page ranges and note ranges as needed according to SBL style. This function is called by the required custom index style file <code>sblidx.ist</code>.</p>						
<hr/> <hr/> <code>\SBLSetMaxRomanPage</code>	<code>\SetMaxRomanPage {<i><page></i>}</code>						
	<p>Set the maximum Roman page number (as an integer, not in Roman format) in the document in case automatic detection fails. This should be set before the indices are printed.</p>						
<hr/> <hr/> <code>\see</code>	<code>\see {<i><index entry></i>} {<i><page number></i>}</code>						
<hr/> <hr/> <code>\seealso</code>	<code>\seealso {<i><index entry></i>} {<i><page number></i>}</code>						
	<p><code>\see</code> and <code>\seealso</code> are used for index cross-references. These are slightly redefined from the default as SBL style requires capital letters for <i>See</i> and <i>See also</i>. They are called in the standard way for the <code>\index</code> command:</p>						
	<table><tr><td>Page 2: <code>\index{at}</code></td><td> </td><td>at, 2</td></tr><tr><td>Page 2: <code>\index{at!bat see{bat, at}}</code></td><td> </td><td>bat. <i>See</i> bat, at</td></tr></table>	Page 2: <code>\index{at}</code>		at, 2	Page 2: <code>\index{at!bat see{bat, at}}</code>		bat. <i>See</i> bat, at
Page 2: <code>\index{at}</code>		at, 2					
Page 2: <code>\index{at!bat see{bat, at}}</code>		bat. <i>See</i> bat, at					

4 Implementation

4.1 Custom Index Style File

`1` `{*indexstyle}`

Custom index style file which processes a list of indexed pages with the macro `\ProcessIndexPages` and removes the default comma delimiter before the first page number for an entry. SBL only uses a comma for the Subject Index.

```

2 delim_0 "\\SBLProcessIndexPages{"
3 delim_1 "\\SBLProcessIndexPages{"
4 delim_2 "\\SBLProcessIndexPages{"
5 delim_t "}"
6 </indexstyle>

```

4.2 Main Package

```

7 <*package>
8 <@=sblidx>
9 \NeedsTeXFormat{LaTeX2e}
10 \ProvidesExplPackage{sblidx}{2026-04-04}{1.0}
11 {Society of Biblical Literature Indices (DCP)}

```

Load imakeidx and idxlayout with appropriate options. Most layout is controlled with idxlayout, but imakeidx provides an interface for setting the index title and the convenience of running makeindex inline.

```

12 \RequirePackage { imakeidx }
13 \RequirePackage { idxlayout }
14 \idxlayout
15 {
16   , columnsep      = 0.5in
17   , font           = small
18   , hangindent    = 0.25in
19   , initsep       = \bigskipamount
20   , itemlayout    = relhang
21   , subindent     = 0.25in
22   , subsubindent  = 0.5in
23   , todoc
24 }

```

4.2.1 Define and Process Package Options

```

25 \keys_define:nm { sblidx }
26 {
27   , ancient~sources      .bool_set:N = \l__sblidx_ancient_sources_bool
28   , ancient~sources~title .tl_set:N  = \l__sblidx_ancient_sources_title_tl
29   , ancient~sources~title .initial:n = Ancient~Sources~Index
30   , modern~authors      .bool_set:N = \l__sblidx_modern_authors_bool
31   , modern~authors~title .tl_set:N  = \l__sblidx_modern_authors_title_tl
32   , modern~authors~title .initial:n  = Modern~Authors~Index
33   , subject             .bool_set:N = \l__sblidx_subject_bool
34   , subject             .initial:n  = true
35   , subject~title      .tl_set:N  = \l__sblidx_subject_title_tl
36   , subject~title      .initial:n  = Subject~Index
37 }
38 \ProcessKeyOptions

```

4.2.2 Set Up Indices

The delimiter between the indexed entry and the first page is dependent on the index type, so it is set dynamically in \SBLPrintIndices.

```

39 \tl_const:Nn \c__sblidx_delim_tl { , \c_space_tl }
40 \tl_new:N \l__sblidx_delim_first_tl
41 \tl_const:Nn \c__sblidx_delim_see_tl { . \c_space_tl }

```

Set up code and call `\makeindex` for each required index.

```
42 \bool_if:NT \l__sblidx_subject_bool
43 {
44   \makeindex
45   [
46     , title = \l__sblidx_subject_title_tl
47     , options = -s ~ sblidx.ist ~ -q
48   ]
49 }

50 \bool_if:NT \l__sblidx_ancient_sources_bool
51 {
52   \RequirePackage { bibleref-parse }
53   \RequirePackage { bibleref-sbl }
54   \makeindex
55   [
56     name = \jobname-ancient-sources ,
57     title = \l__sblidx_ancient_sources_title_tl ,
58     options = -s ~ sblidx.ist ~ -q
59   ]
60   \cs_set_nopar:Npn \biblerefindex
61   {
62     \index [ \jobname-ancient-sources ]
63   }
64 }

65 \bool_if:NT \l__sblidx_modern_authors_bool
66 {
67   \RequirePackage { biblatex }
68   \ExecuteBibliographyOptions { indexing = cite }
69   \makeindex
70   [
71     name = \jobname-modern-authors ,
72     title = \l__sblidx_modern_authors_title_tl ,
73     options = -s ~ sblidx.ist ~ -q
74   ]
75   \DeclareIndexNameFormat { default }
76   {
77     \usebibmacro { index:name }
78     { \index [ \jobname-modern-authors ] }
79     { \namepartfamily }
80     { \namepartgiven }
81     { \namepartprefix }
82     { \namepartsuffix }
83   }
84 }
```

Set up `\footnote` so that `\index` includes the footnote number when indexing an entry in a footnote.

`__sblidx_set_up_footnotes:` Footnote set up code.

```
85 \cs_new_protected:Nn \__sblidx_set_up_footnotes:
86 {
87   \cs_set_eq:NN \@sblidx@index \index
88   \bool_if:NT \l__sblidx_ancient_sources_bool
```

```

89     {
90       \cs_set_nopar:Npn \bvidxpgformat
91       { SBLPageWithNote { \thefootnote } }
92     }
93     \cs_set_nopar:Npn \index { \SBLFootnoteIndex }
94   }

```

(End of definition for `__sblidx_set_up_footnotes:`.)

For new footnote code, the set up can be injected with hooks. Otherwise the `footnote` command must be redefined. Note that in the latter case if you or some other package again redefines `\footnote` after loading `sblidx` automatically including note numbers with indexed entries in footnotes will not work.

```

95 \IfDocumentMetadataTF
96 {
97   \hook_gput_code:nmn { fntext } { sblidx } { \__sblidx_set_up_footnotes: }
98 }
99 {
100  \cs_set_eq:NN \__sblidx_footnote: \footnote
101  \RenewDocumentCommand { \footnote } { o+m }
102  {
103    \group_begin:
104      \__sblidx_set_up_footnotes:
105      \tl_if_novalue:nTF {#1}
106      { \__sblidx_footnote: {#2} }
107      { \__sblidx_footnote: [#1] {#2} }
108    \group_end:
109  }
110 }

```

`\see` SBL want *See* and *See also* to begin a new sentence so redefine `\see` and `\seealso` to capitalise the first letter of *See*.

```

111 \cs_set_nopar:Npn \see #1 #2
112 {
113   \emph { \text_titlecase_first:n { \seename } }
114   \c_space_tl #1
115 }
116 \cs_set_nopar:Npn \seealso #1 #2
117 {
118   \emph { \text_titlecase_first:n { \alsoname } }
119   \c_space_tl #1
120 }

```

(End of definition for `\see` and `\seealso`. These functions are documented on page 4.)

`\SBLFootnoteIndex` [`<name>`] {`<entry>`}

`\index` is defined to this macro within footnotes so that note numbers are automatically included unless some other formatting is requested by the user. `<name>` is the name of the raw index file and `<entry>` should be formatted according to syntax required by `makeindex`.

```

121 \NewDocumentCommand { \SBLFootnoteIndex } { om }
122 {
123   \tl_if_novalue:nTF {#1}

```

```

124     {
125       \str_if_in:nnTF {#2} { | }
126       { \@sblidx@index {#2} }
127       {
128         \@sblidx@index
129         { #2 | SBLPageWithNote { \thefootnote } }
130       }
131     }
132     {
133       \str_if_in:nnTF {#2} { | }
134       { \@sblidx@index [#1] {#2} }
135       {
136         \@sblidx@index [ #1 ]
137         { #2 | SBLPageWithNote { \thefootnote } }
138       }
139     }
140   }

```

(End of definition for `\SBLFootnoteIndex`. This function is documented on page 4.)

`\SBLPageWithNote` `{⟨comma separated list of notes⟩} {⟨page⟩}`

Output page with note number using `n.` or `nn.` depending on how many notes there are. When `\index` appears in a footnote, this command is called automatically using the `makeindex \index{gnat|SBLPageWithNote{\thefootnote}}` syntax. It allows for a comma separated list of notes because the command `\SBLProcessIndexPages` combines indexed entries in footnotes on the same page into one command with the format `\SBLPageWithNote{x,y,z}{page}`.

```

141 \cs_new_protected:Npn \SBLPageWithNote #1 #2
142   {
143     #2
144     \c_space_tl
145     \int_compare:nNnTF { \clist_count:n {#1} } > { \c_one_int }
146     { nn. }
147     { n. }
148     \nobreakspace
149     \__sblidx_compress_note_list:n {#1}
150   }

```

(End of definition for `\SBLPageWithNote`. This function is documented on page 4.)

`\SBLPrintIndices` Print the indices. This command prints one or more indices depending on which are enabled with the `ancient sources`, `modern authors` and `subject` options.

```

151 \cs_new_protected:Npn \SBLPrintIndices
152   {
153     \__sblidx_set_up_hyperref:
154     \bool_if:NT \l__sblidx_ancient_sources_bool
155     {
156       \group_begin:
157       \idxlayout
158       {
159         subindent    = Opt ,
160         subsubindent = 0.25in ,
161         hangindent   = 0.25in

```



```

162     }
163     \cs_set_nopar:Npn \@idxitem
164     { \indexspace \bfseries }
165     \cs_set_nopar:Npn \subitem
166     { \indexspace \normalfont }
167     \tl_set:Nn \l__sblidx_delim_first_tl { \quad \hfill }
168     \printindex [ \jobname-ancient-sources ]
169     \group_end:
170   }
171   \bool_if:NT \l__sblidx_modern_authors_bool
172   {
173     \group_begin:
174     \tl_set:Nn \l__sblidx_delim_first_tl { \quad \hfill }
175     \printindex [ \jobname-modern-authors ]
176     \group_end:
177   }
178   \bool_if:NT \l__sblidx_subject_bool
179   {
180     \group_begin:
181     \tl_set:Nn \l__sblidx_delim_first_tl { , \c_space_tl }
182     \printindex
183     \group_end:
184   }
185 }

```

(End of definition for \SBLPrintIndices. This function is documented on page 4.)

\SBLProcessIndexPages {<list of pages>}

Process a list of pages produced by `makeindex`, compressing page ranges and note ranges as needed according to SBL style. This function is called by the required custom index style file `sblidx.ist`.

```

186 \cs_new_protected:Npn \SBLProcessIndexPages #1
187 {
188   \__sblidx_process_index_pages:n {#1}
189 }

```

(End of definition for \SBLProcessIndexPages. This function is documented on page 4.)

\SBLSetMaxRomanPage {<page>}

Set the maximum Roman page number (as an integer, not in Roman format) in the document in case automatic detection fails. This should be set before the indices are printed.

```

190 \cs_new_protected:Npn \SBLSetMaxRomanPage #1
191 {
192   \__sblidx_set_max_roman_page:n {#1}
193 }

```

(End of definition for \SBLSetMaxRomanPage. This function is documented on page 4.)

4.2.3 Page and Note Compression Code

The following code handles compression of page numbers and note numbers in indices according to SBL requirements.

```
\_sblidx_set_max_roman_page:n {<number>}
```

Set the maximum Roman page number used in the front matter. This is needed for incrementing page numbers.

```
194 \int_new:N \g__sblidx_max_roman_page_int
195 \int_gset:Nn \g__sblidx_max_roman_page_int { 1000 }

196 \cs_new_protected:Nn \_sblidx_set_max_roman_page:n
197 {
198   \int_gset:Nn \g__sblidx_max_roman_page_int {#1}
199 }
```

Although not completely reliable, try to set this automatically when `\mainmatter` is called.

```
200 \hook_gput_code:nnn { cmd / mainmatter / before } { sblidx }
201 {
202   \int_gset_eq:NN \g__sblidx_max_roman_page_int \c@page
203   \legacy_if:nT { @twoside }
204   {
205     \int_if_odd:nT { \c@page }
206     { \int_gincr:N \g__sblidx_max_roman_page_int }
207   }
208 }
```

(End of definition for `_sblidx_set_max_roman_page:n`.)

```
\_sblidx_compress_range:n {<number range>}
```

Compress a number range according to SBL style. Non-Arabic numbers and non-ranges are returned unaltered.

```
209 \int_new:N \l__sblidx_range_start_int
210 \int_new:N \l__sblidx_range_end_int
211 \int_new:N \l__sblidx_range_length_int
212 \int_new:N \l__sblidx_range_end_position_int
213 \seq_new:N \l__sblidx_range_seq
214 \tl_new:N \l__sblidx_compressed_range_tl
215 \tl_new:N \l__sblidx_compressed_range_end_tl
216 \tl_new:N \l__sblidx_range_start_tl
217 \tl_new:N \l__sblidx_range_end_tl

218 \regex_const:Nn \c__sblidx_range_regex { \A ( .*? ) -- ( .*? ) \Z }
219 \regex_const:Nn \c__sblidx_number_range_regex { \A ( \d+ ) -- ( \d+ ) \Z }

220 \cs_new_protected:Nn \_sblidx_compress_range:n
221 {
222   \_sblidx_get_compressed_range_end:n {#1}
223   \tl_if_empty:NTF \l__sblidx_compressed_range_end_tl
224     {#1}
225     {
226       \tl_set:Ne \l__sblidx_compressed_range_tl
227       {
```

```

228         \l__sblidx_range_start_tl
229         --
230         \l__sblidx_compressed_range_end_tl
231     }
232     \tl_use:N \l__sblidx_compressed_range_tl
233 }
234 }

```

\@@_get_compressed_range_end:n finds the compressed end value for an Arabic number range and stores it in \l_@@_compressed_range_end_tl.

```

235 \cs_new_protected:Nn \__sblidx_get_compressed_range_end:n
236 {
237     \tl_clear:N \l__sblidx_compressed_range_end_tl

```

Only Arabic number ranges are compressed

```

238     \seq_set_regex_extract_once:NNn
239     \l__sblidx_range_seq \c__sblidx_number_range_regex {#1}
240     \int_compare:nNnTF
241     { \seq_count:N \l__sblidx_range_seq } = { 3 }
242     {
243         \int_set:Nn \l__sblidx_range_start_int
244         { \seq_item:Nn \l__sblidx_range_seq { 2 } }
245         \int_set:Nn \l__sblidx_range_end_int
246         { \seq_item:Nn \l__sblidx_range_seq { 3 } }

```

First number must be greater than 100.

```

247     \int_compare:nNnT
248     { \l__sblidx_range_start_int } > { 100 }
249     {

```

First number must not be divisible by 100.

```

250     \int_compare:nNnT
251     {
252         \int_mod:nn
253         { \l__sblidx_range_start_int }
254         { 100 }
255     }
256     >
257     { \c_zero_int }
258     {

```

Pages must be the same number of digits.

```

259     \int_set:Nn \l__sblidx_range_length_int
260     {
261         \tl_count:e { \int_use:N \l__sblidx_range_start_int }
262     }
263     \int_compare:nNnT
264     { \l__sblidx_range_length_int }
265     =
266     {
267         \tl_count:e { \int_use:N \l__sblidx_range_end_int }
268     }
269     {

```

Main compression code. Step through each digit of the start number in the range until the corresponding digit in the end number is different *or* there are only two digits left. The remaining digits of the end number is the compressed value.

```

270         \tl_set:No \l__sblidx_range_start_tl
271         { \int_use:N \l__sblidx_range_start_int }
272         \tl_set:No \l__sblidx_range_end_tl
273         { \int_use:N \l__sblidx_range_end_int }
274         \int_zero:N \l__sblidx_range_end_position_int
275         \tl_map_inline:Nn \l__sblidx_range_start_tl
276         {
277             \int_incr:N \l__sblidx_range_end_position_int
278             \int_compare:nNnT
279             { \l__sblidx_range_end_position_int + 1 }
280             =
281             { \l__sblidx_range_length_int }
282             {
283                 \tl_set:Ne
284                 \l__sblidx_compressed_range_end_tl
285                 {
286                     \tl_range:Nnn
287                     \l__sblidx_range_end_tl
288                     { \l__sblidx_range_end_position_int }
289                     { \l__sblidx_range_length_int }
290                 }
291                 \tl_map_break:
292             }
293         \tl_if_eq:neF
294         {##1}
295         {
296             \tl_item:Nn
297             \l__sblidx_range_end_tl
298             { \l__sblidx_range_end_position_int }
299         }
300         {
301             \tl_set:Ne
302             \l__sblidx_compressed_range_end_tl
303             {
304                 \tl_range:Nnn
305                 \l__sblidx_range_end_tl
306                 { \l__sblidx_range_end_position_int }
307                 { \l__sblidx_range_length_int }
308             }
309             \tl_map_break:
310         }
311     }

```

Remove leading zeroes from compressed range.

```

312         \tl_regex_replace_once:Nnn
313         \l__sblidx_compressed_range_end_tl
314         { \A 0+ }
315         { }
316     }
317 }
318 }

```

```

319     }
320     {

```

Check for non-Arabic range.

```

321     \seq_set_regex_extract_once:NNn
322     \l__sblidx_range_seq \c__sblidx_range_regex {#1}
323     \seq_if_empty:NF \l__sblidx_range_seq
324     {
325         \tl_set:Ne \l__sblidx_range_start_tl
326         { \seq_item:Nn \l__sblidx_range_seq { 2 } }
327         \tl_set:Ne \l__sblidx_range_end_tl
328         { \seq_item:Nn \l__sblidx_range_seq { 3 } }
329         \tl_set_eq:NN
330         \l__sblidx_compressed_range_end_tl \l__sblidx_range_end_tl
331     }
332 }
333 }

```

(End of definition for `__sblidx_compress_range:n`.)

```

\__sblidx_compress_note_range:n
\__sblidx_compress_page_range:n
\__sblidx_hyper_compress_page_range:n

```

{*number range*}

Compress a note or page range according to SBL style. Non-Arabic numbers and non-ranges are returned unaltered. `hyperref` is supported for page numbers by `\@@_hyper_compress_page_range:n`.

```

334 \cs_set:Nn \__sblidx_compress_note_range:n
335 { \__sblidx_compress_range:n {#1} }
336 \cs_set:Nn \__sblidx_compress_page_range:n
337 { \__sblidx_compress_range:n {#1} }

338 \cs_new_protected:Nn \__sblidx_hyper_compress_page_range:n
339 {
340     \__sblidx_get_compressed_range_end:n {#1}
341     \tl_if_empty:NTF \l__sblidx_compressed_range_end_tl
342     {
343         \hyperlink { page. #1 } {#1}
344     }
345     {
346         \tl_set:Ne \l__sblidx_compressed_range_tl
347         {
348             \exp_not:N \hyperlink
349             { page. \l__sblidx_range_start_tl }
350             { \l__sblidx_range_start_tl }
351             --
352             \exp_not:N \hyperlink
353             { page. \l__sblidx_range_end_tl }
354             { \l__sblidx_compressed_range_end_tl }
355         }
356         \tl_use:N \l__sblidx_compressed_range_tl
357     }
358 }

```

(End of definition for `__sblidx_compress_note_range:n`, `__sblidx_compress_page_range:n`, and `__sblidx_hyper_compress_page_range:n`.)

`_sblidx_do_page:n` $\{\langle page \rangle\}$
`_sblidx_hyper_do_page:n` Create a hyperlink for $\langle page \rangle$. This does nothing if hyperref is not loaded.

```

359 \cs_new:Nn \_sblidx_do_number:n {#1}
360 \cs_new:Nn \_sblidx_do_note:n {#1}
361 \cs_new:Nn \_sblidx_do_page:n {#1}

362 \cs_new:Nn \_sblidx_hyper_do_page:n
363   {
364     \exp_not:N \hyperlink { page. #1 } {#1}
365   }

```

(End of definition for `_sblidx_do_page:n` and `_sblidx_hyper_do_page:n`.)

`_sblidx_compress_list:nN` $\{\langle sorted\ numbers \rangle\} \langle compressed\ list \rangle$

Format a comma separated list of page or note numbers to use SBL style compressed ranges when there are three or more consecutive numbers and store the result in the $\langle compressed\ list \rangle$ `clist` variable.

```

366 \clist_new:N \l__sblidx_uncompressed_clist
367 \tl_new:N \l__sblidx_start_tl
368 \tl_new:N \l__sblidx_previous_tl

369 \cs_new_protected:Nn \_sblidx_compress_list:nN
370   {
371     \clist_set:Nn \l__sblidx_uncompressed_clist {#1}
372     \clist_clear:N #2
373     \clist_pop:NNT \l__sblidx_uncompressed_clist \l__sblidx_start_tl
374     {
375       \tl_set_eq:NN \l__sblidx_previous_tl \l__sblidx_start_tl
376       \clist_map_inline:Nn \l__sblidx_uncompressed_clist
377         {
378           \_sblidx_page_compare:eNeTF
379             {##1}
380             =
381             { \_sblidx_page_incr:V \l__sblidx_previous_tl }
382             {
383               \tl_set:Nn \l__sblidx_previous_tl {##1}
384             }
385             {
386               \_sblidx_append_compressed_list:VVN
387                 \l__sblidx_start_tl \l__sblidx_previous_tl #2
388               \tl_set:Nn \l__sblidx_start_tl {##1}
389               \tl_set:Nn \l__sblidx_previous_tl {##1}
390             }
391           }
392       \_sblidx_append_compressed_list:VVN
393         \l__sblidx_start_tl \l__sblidx_previous_tl #2
394     }
395   }

```

`\@@_append_compressed_list:nnN` and `\@@_append_compressed_list:VVN` are helper functions called by `\@@_compress_list:nN` to append the next number or number range to the current list when building a compressed list.

```

396 \cs_new_protected:Nn \_sblidx_append_compressed_list:nnN

```

```

397 {
398   \__sblidx_page_compare:eNeTF {#2} > { \__sblidx_page_incr:n {#1} }
399   {
400     \clist_put_right:Ne #3
401     { \__sblidx_do_compress_range:n { #1 -- #2 } }
402   }
403   {
404     \clist_put_right:Nn #3
405     {
406       \__sblidx_do_number:n {#1}
407     }
408     \tl_if_eq:nnF {#1} {#2}
409     {
410       \clist_put_right:Nn #3
411       {
412         \__sblidx_do_number:n {#2}
413       }
414     }
415   }
416 }

```

```

417 \cs_generate_variant:Nn \__sblidx_append_compressed_list:nnN { VVN }

```

\@@_do_compress_range:n needs to be set to either \@@_compress_note_range:n or \@@_compress_page_range:n before calling to ensure only page numbers are hyperlinked when hyperref is loaded.

```

418 \cs_set:Nn \__sblidx_do_compress_range:n
419   { \__sblidx_compress_range:n {#1} }

```

(End of definition for __sblidx_compress_list:nN.)

__sblidx_compress_note_list:n {<sorted note numbers>}

Format a comma separated list of note numbers to use SBL style compressed ranges when there are three or more consecutive numbers.

```

420 \clist_new:N \l__sblidx_compressed_clist
421 \cs_new_protected:Nn \__sblidx_compress_note_list:n
422   {
423     \cs_set:Nn \__sblidx_do_compress_range:n
424       { \__sblidx_compress_note_range:n {##1} }
425     \cs_set:Nn \__sblidx_do_number:n
426       { \__sblidx_do_note:n {##1} }
427     \__sblidx_compress_list:nN {#1} \l__sblidx_compressed_clist
428     \clist_use:Nnnn
429       \l__sblidx_compressed_clist
430       { ~ and ~ } { , ~ } { , ~ and ~ }
431   }

```

(End of definition for __sblidx_compress_note_list:n.)

__sblidx_compress_page_list:n {<sorted page numbers>} <list of compressed page>

__sblidx_compress_page_list:VN

Format a comma separated list of page numbers to use SBL style compressed ranges when there are three or more consecutive numbers.

```

432 \cs_new_protected:Nn \__sblidx_compress_page_list:nN
433 {
434   \cs_set:Nn \__sblidx_do_compress_range:n
435     { \__sblidx_compress_page_range:n {##1} }
436   \cs_set:Nn \__sblidx_do_number:n
437     { \__sblidx_do_page:n {##1} }
438   \__sblidx_compress_list:nN {#1} \l__sblidx_compressed_clist
439   \tl_set:Ne
440     #2
441     {
442       \clist_use:Nn \l__sblidx_compressed_clist { \c__sblidx_delim_tl }
443     }
444 }
445 \cs_generate_variant:Nn \__sblidx_compress_page_list:nN { VN }

```

(End of definition for `__sblidx_compress_page_list:nN`)

```

\__sblidx_page_compare:nNnTF {<first page number>} <relation> {<second page number>} {<true code>} {<false code>}
\__sblidx_page_compare:eNeTF
\__sblidx_page_compare:VNVT

```

Compare pages taking into account Roman and Arabic page numbering systems. Note that this macro only takes into account a standard book with the front matter numbered with Roman numbers and the main matter numbered with Arabic numbers. The relations supported are =, > and <. T and TF variants are available.

```

446 \int_new:N \l__sblidx_first_int
447 \int_new:N \l__sblidx_second_int
448 \prg_new_protected_conditional:Npnn \__sblidx_page_compare:nNn #1 #2 #3
449   { T, TF }
450   {

```

Offset Arabic page numbers by `\g__max_roman_page_int` so they are always greater than any Roman page number.

```

451   \regex_if_match:nnTF { \A \d+ \Z } {#1}
452   {
453     \int_set:Nn \l__sblidx_first_int
454       { #1 + \g__sblidx_max_roman_page_int }
455   }
456   {
457     \int_set:Nn \l__sblidx_first_int
458       { \exp_args:Ne \int_from_roman:n {#1} }
459   }
460   \regex_if_match:nnTF { \A \d+ \Z } {#3}
461   {
462     \int_set:Nn \l__sblidx_second_int
463       { #3 + \g__sblidx_max_roman_page_int }
464   }
465   {
466     \int_set:Nn \l__sblidx_second_int
467       { \exp_args:Ne \int_from_roman:n {#3} }
468   }

```

Now we just need a simple integer comparison.

```

469   \if_int_compare:w \l__sblidx_first_int #2 \l__sblidx_second_int
470     \prg_return_true:

```



```

471 \else:
472 \prg_return_false:
473 \fi:
474 }

475 \cs_generate_variant:Nn \__sblidx_page_compare:nNnT { VNVNT }
476 \cs_generate_variant:Nn \__sblidx_page_compare:nNnTF { eNeTF }

(End of definition for \__sblidx_page_compare:nNnTF and \__sblidx_page_compare:VNVNT.)

```

`__sblidx_page_incr:n` $\{\langle\text{page number}\rangle\}$

Increment a page number taking into account whether it is Roman or Arabic, leaving the result on the input stream. Trying to increment something other than a Roman or positive Arabic number results in an empty token list. This function is fully expandable, but as a result the test for an Arabic page number checks if the first character is a digit and otherwise assumes the input is roman.

```

477 \cs_new:Nn \__sblidx_page_incr:n
478 {
479 \exp_args:Ne \__sblidx_page_incr:nn { \str_head:n {#1} } {#1}
480 }

```

`\@@_page_incr:nn` is a helper function that allows `\@@_page_incr:n` to be fully expandable.

```

481 \cs_new:Nn \__sblidx_page_incr:nn
482 {
483 \str_case:nnTF {#1}
484 {
485 { 0 } { } { 1 } { } { 2 } { } { 3 } { } { 4 } { }
486 { 5 } { } { 6 } { } { 7 } { } { 8 } { } { 9 } { }
487 }
488 {
489 \int_eval:n { #2 + 1 }
490 }
491 {
492 \int_compare:nNnTF
493 { \int_from_roman:n {#2} } = { \g__sblidx_max_roman_page_int }
494 { 1 }
495 { \int_to_roman:n { \int_from_roman:n {#2} + 1 } }
496 }
497 }

```

```

498 \cs_generate_variant:Nn \__sblidx_page_incr:n { V }

```

(End of definition for `__sblidx_page_incr:n`.)

4.2.4 Index Processing Code

The following code processes page numbers and notes for an index entry and produces an index entry in SBL style.

`__sblidx_set_up_hyperref:` The `hyperref` package changes the format of the index, so needs special handling at a number of points.

```

499 \bool_new:N \l__sblidx_hyperref_bool
500 \regex_new:N \l__sblidx_page_regex

```

```

501 \regex_new:N \l__sblidx_page_encap_regex
502 \regex_new:N \l__sblidx_see_also_regex
503 \regex_new:N \l__sblidx_page_with_note_regex
504 \cs_new_protected:Nn \__sblidx_set_up_hyperref:
505 {
506   \cs_if_exist:NTF \hyperindexformat
507   {
508     \bool_set_true:N \l__sblidx_hyperref_bool
509     \cs_set:Nn \__sblidx_compress_page_range:n
510     { \__sblidx_hyper_compress_page_range:n {##1} }
511     \cs_set:Nn \__sblidx_do_page:n
512     { \__sblidx_hyper_do_page:n {##1} }
513     \regex_set:Nn \l__sblidx_page_regex
514     { \A \c{ hyperpage } \cB{ ( .*? ) \cE\} \Z }
515     \regex_set:Nn \l__sblidx_page_encap_regex
516     {
517       \A \c{ hyperindexformat } \cB{ ( \c{ .* } .* ) \cE\}
518       \cB{ ( .*? ) \cE\} \Z
519     }
520     \regex_set:Nn \l__sblidx_see_also_regex
521     {
522       \A \c{ hyperindexformat }
523       \cB{ ( [ \c{ see } \c{ seealso } ] .* ) \cE\}
524       \cB{ .*? \cE\} \Z
525     }
526     \regex_set:Nn \l__sblidx_page_with_note_regex
527     {
528       \A \c{ hyperindexformat }
529       \cB{ \c{ SBLPageWithNote } \cB{ ( .*? ) \cE\} \cE\}
530       \cB{ ( .*? ) \cE\} \Z
531     }
532   }
533 {
534   \cs_set:Nn \__sblidx_compress_page_range:n
535   { \__sblidx_compress_range:n {##1} }
536   \cs_set:Nn \__sblidx_do_page:n {##1}
537   \regex_set:Nn \l__sblidx_page_encap_regex
538   { \A ( \c{ .* } .* ) \cB{ ( .*? ) \cE\} \Z }
539   \regex_set:Nn \l__sblidx_see_also_regex
540   { \A ( [ \c{ see } \c{ seealso } ] .* ) \cB{ .*? \cE\} \Z }
541   \regex_set:Nn \l__sblidx_page_with_note_regex
542   {
543     \A \c{ SBLPageWithNote }
544     \cB{ ( .*? ) \cE\} \cB{ ( .*? ) \cE\} \Z
545   }
546 }
547 }

```

(End of definition for __sblidx_set_up_hyperref:.)

\l__sblidx_page_data_seq Create a data structure for storing and manipulating page data. These can then added to the \l_@@_page_data_seq sequence. All terms should always be specified and their values should be braced. E.g., page={1},note={},encap={}

```

548 \seq_new:N      \l__sblidx_page_data_seq
549 \tl_new:N      \l__sblidx_page_tl
550 \tl_new:N      \l__sblidx_note_tl
551 \cs_new_nopar:Nn \__sblidx_encap: { \exp_not:n {} }

552 \keys_define:nn { sblidx / pagedata }
553 {
554   , page .tl_set:N = \l__sblidx_page_tl
555   , note .tl_set:N = \l__sblidx_note_tl
556   , encap .code:n = \cs_set_nopar:Nn \__sblidx_encap:
557                     { \exp_not:n {#1} }
558 }

```

(End of definition for `\l__sblidx_page_data_seq`.)

`__sblidx_sort_page_data:N` *<page data sequence>*

Sort the specified page data sequence by page number. Encapsulated page numbers sort ahead of plain page numbers which sort ahead of page numbers with notes.

```

559 \bool_new:N \l__sblidx_sort_returned_bool
560 \cs_set_nopar:Nn \__sblidx_empty_encap: { \exp_not:n {} }

561 \cs_new_protected:Nn \__sblidx_sort_page_data:N
562 {
563   \seq_sort:Nn #1
564   {
565     \bool_set_false:N \l__sblidx_sort_returned_bool
566     \keys_set:nn { sblidx / pagedata } {##1}
567     \tl_set_eq:NN \l_tmpa_tl \l__sblidx_page_tl
568     \tl_set_eq:NN \l_tmpb_tl \l__sblidx_note_tl
569     \cs_set_eq:NN \__sblidx_tmpa: \__sblidx_encap:
570     \keys_set:nn { sblidx / pagedata } {##2}

```

Page is primary sort key.

```

571     \__sblidx_page_compare:VNVT \l_tmpa_tl > \l__sblidx_page_tl
572     {
573       \bool_set_true:N \l__sblidx_sort_returned_bool
574       \sort_return_swapped:
575     }
576     \__sblidx_page_compare:VNVT \l_tmpa_tl < \l__sblidx_page_tl
577     {
578       \bool_set_true:N \l__sblidx_sort_returned_bool
579       \sort_return_same:
580     }
581     \tl_if_eq:NNT \l_tmpa_tl \l__sblidx_page_tl
582     {

```

If the page is the same, then calculate a sorting weight for each item.

```

583       \__sblidx_page_data_weight:NNN
584       \l_tmpb_tl \__sblidx_tmpa: \l_tmpa_int
585       \__sblidx_page_data_weight:NNN
586       \l__sblidx_note_tl \__sblidx_encap: \l_tmpb_int
587       \int_compare:nNnT { \l_tmpa_int } < { \l_tmpb_int }
588       {
589         \bool_set_true:N \l__sblidx_sort_returned_bool

```

```

590         \sort_return_same:
591     }
592     \int_compare:nNtT { \l_tmpa_int } > { \l_tmpb_int }
593     {
594         \bool_set_true:N \l__sblidx_sort_returned_bool
595         \sort_return_swapped:
596     }

```

If the sorting weight indicates both items have a note then sort by note.

```

597     \int_compare:nNtT { \l_tmpa_int } = { \l_tmpb_int }
598     {
599         \bool_set_true:N \l__sblidx_sort_returned_bool
600         \int_compare:nNtTF { \l_tmpa_int } = { 2 }
601         {
602             \tl_if_empty:NTF \l__sblidx_note_tl
603             { \int_zero:N \l_tmpa_int }
604             { \int_set:Nn \l_tmpa_int { \l__sblidx_note_tl } }
605             \tl_if_empty:NTF \l_tmpb_tl
606             { \int_zero:N \l_tmpb_int }
607             { \int_set:Nn \l_tmpb_int { \l__sblidx_note_tl } }
608             \int_compare:nNtTF { \l_tmpa_int } < { \l_tmpb_int }
609             { \sort_return_swapped: }
610             { \sort_return_same: }
611         }
612         { \sort_return_same: }
613     }
614 }

```

Make sure we return for a cases where \l_@@_page_tl is a roman numeral range.

```

615     \bool_if:NF \l__sblidx_sort_returned_bool
616     { \sort_return_same: }
617 }
618 }

```

(End of definition for __sblidx_sort_page_data:N.)

_sblidx_page_data_weight:NNN <note> <encap> <weight>

Return a sorting weight depending on the presence and value of <note> and <encap>. Set the <weight> integer variable to 0 if <note> is an empty token list and <encap> is \@@_empty_encap:, 1 if there is a <note> and <encap> is \@@_empty_encap: and 2 if <note> is an empty token list and <encap> is not \@@_empty_encap:.

```

619 \cs_new_protected:Nn \__sblidx_page_data_weight:NNN
620 {
621     \cs_if_eq:NNTF #2 \__sblidx_empty_encap:
622     {
623         \tl_if_empty:NTF #1
624         {
625             \int_set:Nn #3 { \c_one_int }
626         }
627         {
628             \int_set:Nn #3 { 2 }
629         }
630     }

```

```

631     { \int_zero:N #3 }
632   }

```

(End of definition for `_sblidx_page_data_weight:NNN`.)

```
\_sblidx_page_data_to_tl:N <page data sequence> <token list>
```

Convert a page data sequence to a token list applying compression and encapsulation. The page data sequence must already be sorted and consolidated.

```

633 \bool_new:N \l__sblidx_item_added_bool
634 \tl_new:N \l__sblidx_compressed_page_tl

635 \cs_new_protected:Nn \_sblidx_page_data_to_tl:N
636 {
637   \tl_clear:N \l_tmpa_tl
638   \seq_map_inline:Nn #1
639   {
640     \bool_set_false:N \l__sblidx_item_added_bool
641     \keys_set:nn { sblidx / pagedata } {##1}
642     \_sblidx_compress_page_list:VN
643     \l__sblidx_page_tl
644     \l__sblidx_compressed_page_tl
645     \cs_if_eq:NNF \_sblidx_encap: \_sblidx_empty_encap:
646     {

```

Test if the encap is `\see` or `\seealso` and if it is insert a period instead of a comma.

```

647       \tl_set:Ne \l_tmpb_tl { \_sblidx_encap: }
648       \tl_if_in:NnTF \l_tmpb_tl { \see }
649       {
650         \tl_put_right:NV \l_tmpa_tl \c__sblidx_delim_see_tl
651       }
652       {
653         \tl_if_in:NnTF \l_tmpb_tl { \seealso }
654         {
655           \tl_put_right:NV \l_tmpa_tl \c__sblidx_delim_see_tl
656         }
657         {
658           \tl_if_empty:NTF \l_tmpa_tl
659           { \tl_set:NV \l_tmpa_tl \l__sblidx_delim_first_tl }
660           { \tl_put_right:NV \l_tmpa_tl \c__sblidx_delim_tl }
661         }
662       }
663       \tl_put_right:Ne \l_tmpa_tl
664       {
665         \_sblidx_encap: { \l__sblidx_compressed_page_tl }
666       }
667       \bool_set_true:N \l__sblidx_item_added_bool
668     }
669   \bool_if:NF \l__sblidx_item_added_bool
670   {
671     \tl_if_empty:NF \l__sblidx_note_tl
672     {
673       \tl_if_empty:NTF \l_tmpa_tl
674       { \tl_set:NV \l_tmpa_tl \l__sblidx_delim_first_tl }
675       { \tl_put_right:NV \l_tmpa_tl \c__sblidx_delim_tl }

```

```

676         \tl_put_right:Ne \l_tmpa_tl
677         {
678             \SBLPageWithNote
679             { \l__sblidx_note_tl }
680             { \l__sblidx_compressed_page_tl }
681         }
682         \bool_set_true:N \l__sblidx_item_added_bool
683     }
684 }
685 \bool_if:NF \l__sblidx_item_added_bool
686 {
687     \tl_if_empty:NTF \l_tmpa_tl
688     { \tl_set:NV \l_tmpa_tl \l__sblidx_delim_first_tl }
689     { \tl_put_right:NV \l_tmpa_tl \c__sblidx_delim_tl }
690     \tl_put_right:Ne \l_tmpa_tl
691     { \l__sblidx_compressed_page_tl }
692 }
693 }

```

Put formatted index pages on to the input stream.

```

694     \tl_use:N \l_tmpa_tl
695 }

```

(End of definition for `__sblidx_page_data_to_tl:N`.)

`\g__sblidx_status_code_int` The functions below set a status code so that the result of the most recent function can be tested.

```

696 \bool_new:N \l__sblidx_stepping_page_bool
697 \int_new:N \g__sblidx_status_code_int
698 \tl_new:N \l__sblidx_end_tl

```

(End of definition for `\g__sblidx_status_code_int`.)

`__sblidx_range_to_clist:nN` `{<range>}` `<list>`

`__sblidx_range_to_clist:eN` Convert a range of the form `<integer>--<integer>` to a comma separated list of integers, leaving the result in `<list>` clist variable. If the form is not found the input is added unaltered to `<list>`. A status code of 0 indicates that a range was found and successfully added to `<list>`.

```

699 \cs_new_protected:Nn \__sblidx_range_to_clist:nN
700 {
701     \seq_set_regex_extract_once:NNn
702     \l_tmpb_seq \c__sblidx_range_regex {#1}
703     \seq_if_empty:NTF \l_tmpb_seq
704     {
705         \clist_set:Nn #2 {#1}
706         \int_gset:Nn \g__sblidx_status_code_int { \c_one_int }
707     }
708     {
709         \clist_clear:N #2
710         \bool_set_true:N \l__sblidx_stepping_page_bool
711         \tl_set:Ne \l__sblidx_page_tl { \seq_item:Nn \l_tmpb_seq { 2 } }
712         \tl_set:Ne \l__sblidx_end_tl { \seq_item:Nn \l_tmpb_seq { 3 } }
713         \bool_do_while:nn { \l__sblidx_stepping_page_bool }

```

```

714     {
715         \clist_put_right:NV #2 \l__sblidx_page_tl
716         \tl_set:Ne \l__sblidx_page_tl
717             { \__sblidx_page_incr:V \l__sblidx_page_tl }
718         \__sblidx_page_compare:VNVT \l__sblidx_page_tl = \l__sblidx_end_tl
719         {
720             \clist_put_right:NV #2 \l__sblidx_page_tl
721             \bool_set_false:N \l__sblidx_stepping_page_bool
722         }
723     }
724     \int_gzero:N \g__sblidx_status_code_int
725 }
726 }

```

```

727 \cs_generate_variant:Nn \__sblidx_range_to_clist:nN { eN }

```

(End of definition for `__sblidx_range_to_clist:nN`.)

`__sblidx_check_page:n` {<*index item*>}

Assume an *<index item>* is a plain page or range and append it the page to the *<page data>* sequence. A status code of 0 indicates a page or range was successfully appended to *<page data>*.

```

728 \cs_new_protected:Nn \__sblidx_check_page:n
729 {
730     \tl_clear:N \l_tmpa_tl
731     \bool_if:NTF \l__sblidx_hyperref_bool
732     {
733         \seq_set_regex_extract_once:NNn
734             \l_tmpa_seq \l__sblidx_page_regex {#1}
735         \seq_if_empty:NF \l_tmpa_seq
736         {
737             \tl_set:Ne \l_tmpa_tl { \seq_item:Nn \l_tmpa_seq { 2 } }
738         }
739     }
740     {
741         \tl_set:Nn \l_tmpa_tl {#1}
742     }
743     \tl_if_empty:NTF \l_tmpa_tl
744     {
745         \int_gset:Nn \g__sblidx_status_code_int { \c_one_int }
746     }
747     {
748         \clist_set:NV \l_tmpa_clist \l_tmpa_tl
749         \clist_map_inline:Nn \l_tmpa_clist
750         {
751             \__sblidx_range_to_clist:nN {##1} \l_tmpb_clist
752             \clist_map_inline:Nn \l_tmpb_clist
753             {
754                 \seq_put_right:Ne \l__sblidx_page_data_seq
755                 {
756                     page = {###1} ,
757                     note = { } ,
758                     encap = { }
759                 }

```

```

760     }
761   }
762   \int_gzero:N \g__sblidx_status_code_int
763 }
764 }

```

(End of definition for `__sblidx_check_page:n`.)

```
\__sblidx_check_page_encap:n {<index item>}
```

Check an *<index item>* if it is encapsulated. If it is then append the page and encap function to the *<page data>* sequence. A status code of 0 indicates an encapsulated page was found and successfully appended to *<page data>*.

```

765 \cs_new_protected:Nn \__sblidx_check_page_encap:n
766 {
767   \seq_set_regex_extract_once:NNn
768     \l_tmpa_seq \l__sblidx_page_encap_regex {#1}
769   \seq_if_empty:NTF \l_tmpa_seq
770   {
771     \int_gset:Nn \g__sblidx_status_code_int { \c_one_int }
772   }
773   {
774     \__sblidx_range_to_clist:eN
775     { \seq_item:Nn \l_tmpa_seq { 3 } }
776     \l_tmpa_clist
777     \clist_map_inline:Nn \l_tmpa_clist
778     {
779       \seq_put_right:Ne \l__sblidx_page_data_seq
780       {
781         page = {##1} ,
782         note = { } ,
783         encap = { \seq_item:Nn \l_tmpa_seq { 2 } }
784       }
785     }
786     \int_gzero:N \g__sblidx_status_code_int
787   }
788 }

```

(End of definition for `__sblidx_check_page_encap:n`.)

```
\__sblidx_check_see_also:n {<index item>}
```

Check an *<index item>* if it use `\see` or `\seealso`. If it does then append to the *<page data>* sequence with a large page number to ensure correct sorting. A status code of 0 indicates `\see` or `\seealso` was found and successfully appended to *<page data>*.

```

789 \int_new:N \l__sblidx_see_also_int
790 \int_set:Nn \l__sblidx_see_also_int { 100000 }
791 \cs_new_protected:Nn \__sblidx_check_see_also:n
792 {
793   \seq_set_regex_extract_once:NNn
794     \l_tmpa_seq \l__sblidx_see_also_regex {#1}
795   \seq_if_empty:NTF \l_tmpa_seq
796   {
797     \int_gset:Nn \g__sblidx_status_code_int { \c_one_int }

```



```

798     }
799     {
800     \int_incr:N \l__sblidx_see_also_int
801     \seq_put_right:Ne \l__sblidx_page_data_seq
802     {
803     page = { \int_use:N \l__sblidx_see_also_int } ,
804     note = { } ,
805     encap = { \seq_item:Nn \l_tmpa_seq { 2 } }
806     }
807     \int_gzero:N \g__sblidx_status_code_int
808     }
809     }

```

(End of definition for __sblidx_check_see_also:n.)

_sblidx_check_page_with_note:n {<index item>}

Check an <index item> if it is in the form \SBLPageWithNote{<note>}{<page>}. If it is then append the page and note to the <page data> sequence. A status code of 0 indicates a page with a note was found and successfully appended to <page data>.

```

810 \cs_new_protected:Nn \__sblidx_check_page_with_note:n
811 {
812   \seq_set_regex_extract_once:NNn
813   \l_tmpa_seq \l__sblidx_page_with_note_regex {#1}
814   \seq_if_empty:NTF \l_tmpa_seq
815   {
816     \int_gset:Nn \g__sblidx_status_code_int { \c_one_int }
817   }
818   {
819     \seq_put_right:Ne \l__sblidx_page_data_seq
820     {
821     page = { \seq_item:Nn \l_tmpa_seq { 3 } } ,
822     note = { \seq_item:Nn \l_tmpa_seq { 2 } } ,
823     encap = { }
824     }
825     \int_gzero:N \g__sblidx_status_code_int
826   }
827 }

```

(End of definition for __sblidx_check_page_with_note:n.)

_sblidx_consolidate_page_data:N <page data>

Loop through a <page data> sequence combining consecutive pages into single terms, notes into single pages and removing unneeded pages.

```

828 \bool_new:N \l__sblidx_put_page_data_bool
829 \clist_new:N \l__sblidx_notes_clist
830 \clist_new:N \l__sblidx_pages_clist
831 \cs_new_protected:Nn \__sblidx_consolidate_page_data:N
832 {
833   \keys_set:ne { sblidx / pagedata }
834   {
835     \seq_item:Nn #1 { \c_one_int }
836   }

```

```

837 \clist_set:NV \l__sblidx_pages_clist \l__sblidx_page_tl
838 \clist_set:NV \l__sblidx_notes_clist \l__sblidx_note_tl
839 \cs_set_eq:NN \__sblidx_encap_previous: \__sblidx_encap:
840 \seq_clear:N \l_tmpa_seq
841 \int_step_inline:nnn { 2 } { \seq_count:N #1 }
842 {
843   \keys_set:ne { sblidx / pagedata }
844   { \seq_item:Nn #1 {##1} }
845   \tl_if_eq:eeTF
846   { \l__sblidx_page_tl }
847   { \clist_item:Nn \l__sblidx_pages_clist { -1 } }
848   {
849     \clist_if_empty:NF \l__sblidx_notes_clist
850     {
851       \clist_put_right:NV
852       \l__sblidx_notes_clist \l__sblidx_note_tl
853     }
854   }
855   {
856     \bool_set_false:N \l__sblidx_put_page_data_bool

```

If the encap changes then add current data to page data sequence.

```

857 \cs_if_eq:NMF \__sblidx_encap: \__sblidx_encap_previous:
858 { \bool_set_true:N \l__sblidx_put_page_data_bool }

```

If there are notes in the list then add current data to page data sequence.

```

859 \clist_if_empty:NF \l__sblidx_notes_clist
860 { \bool_set_true:N \l__sblidx_put_page_data_bool }

```

If the current item has a note then add current data to page data sequence.

```

861 \tl_if_empty:NF \l__sblidx_note_tl
862 { \bool_set_true:N \l__sblidx_put_page_data_bool }

```

Either add data to the page data sequence or append the current page to a page list.

```

863 \bool_if:NTF \l__sblidx_put_page_data_bool
864 {
865   \seq_put_right:Ne \l_tmpa_seq
866   {
867     page = { \clist_use:N \l__sblidx_pages_clist } ,
868     note = { \clist_use:N \l__sblidx_notes_clist } ,
869     encap = { \__sblidx_encap_previous: }
870   }
871   \clist_set:NV \l__sblidx_pages_clist \l__sblidx_page_tl
872   \clist_set:NV \l__sblidx_notes_clist \l__sblidx_note_tl
873   \cs_set_eq:NN \__sblidx_encap_previous: \__sblidx_encap:
874 }
875 {
876   \clist_put_right:NV
877   \l__sblidx_pages_clist \l__sblidx_page_tl
878 }
879 }
880 }
881 \seq_put_right:Ne \l_tmpa_seq
882 {

```

```

883     page = { \clist_use:N \l__sblidx_pages_clist } ,
884     note = { \clist_use:N \l__sblidx_notes_clist } ,
885     encap = { \__sblidx_encap_previous: }
886   }
887   \seq_set_eq:NN #1 \l_tmpa_seq
888 }

```

(End of definition for __sblidx_consolidate_page_data:N.)

__sblidx_process_index_pages:n *{⟨index pages⟩}*

Process a list of pages produced by `makeindex`, compressing page ranges and note ranges as needed according to SBL style.

```

889 \cs_new_protected:Nn \__sblidx_process_index_pages:n
890 {
891   \seq_clear:N \l__sblidx_page_data_seq
892   \int_set:Nn \l__sblidx_see_also_int { 100000 }
893   \clist_map_inline:nn { #1 }
894   {
895     \__sblidx_check_page_with_note:n {##1}
896     \int_compare:nNnF { \g__sblidx_status_code_int } = { \c_zero_int }
897     { \__sblidx_check_see_also:n {##1} }
898     \int_compare:nNnF { \g__sblidx_status_code_int } = { \c_zero_int }
899     { \__sblidx_check_page_encap:n {##1} }
900     \int_compare:nNnF { \g__sblidx_status_code_int } = { \c_zero_int }
901     { \__sblidx_check_page:n {##1} }
902   }
903   \__sblidx_sort_page_data:N \l__sblidx_page_data_seq
904   \__sblidx_consolidate_page_data:N \l__sblidx_page_data_seq
905   \__sblidx_page_data_to_tl:N \l__sblidx_page_data_seq
906 }

```

(End of definition for __sblidx_process_index_pages:n.)

```

907 \endinput
908 </package>

```