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 bibleref-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

*Email: dcpurton@marshwiggle.net

[1] See *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 LaTeX source showing subject, ancient sources and modern authors indices.

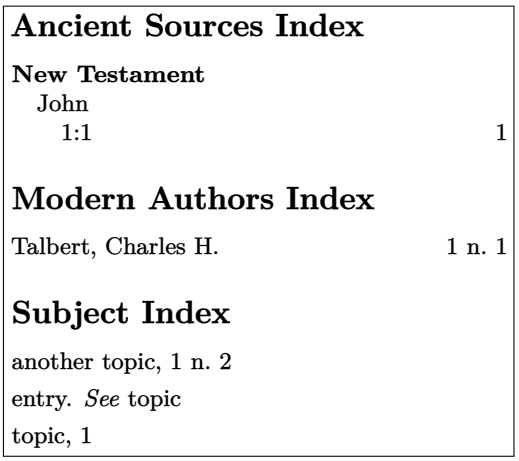

Figure 2: Example output showing subject, ancient sources and modern authors indices.

2 Package Options

`ancient␣sources` `ancient sources = true | false` **Default: false**

Enable an index of ancient sources. Bible books (including deutero-canonical 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

\SBLFootnoteIndex `\SBLFootnotes [⟨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`. In general this macro should not be called directly as it automatically includes the current value of the `footnote` counter.

\SBLPageWithNote `\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.

\SBLPrintIndices `\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.

\SBLProcessIndexPages `\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`.

\SBLSetMaxRomanPage `\SetMaxRomanPage {⟨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.

\see `\see {⟨index entry⟩} {⟨page number⟩}`
\seealso `\seealso {⟨index entry⟩} {⟨page number⟩}`

`\see` and `\seealso` are used for index cross-references. These are slightly redefined from the default as SBL style requires capital letters for *See* and *See also*. They are called in the standard way for the `\index` command:

Page 2:	`\index{at}`	at, 2
Page 2:	`\index{at!bat|see{bat, at}}`	bat. *See* 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.

```
delim_0 "\\SBLProcessIndexPages{"
delim_1 "\\SBLProcessIndexPages{"
delim_2 "\\SBLProcessIndexPages{"
delim_t "}"
```

⟨/indexstyle⟩

4.2 Main Package

⟨*package⟩
⟨@@=sblidx⟩

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesExplPackage{sblidx}{2026-04-04}{1.0}
  {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.

```
\RequirePackage { imakeidx }
\RequirePackage { idxlayout }
\idxlayout
  {
    , columnsep    = 0.5in
    , font         = small
    , hangindent   = 0.25in
    , initsep      = \bigskipamount
    , itemlayout   = relhang
    , subindent    = 0.25in
    , subsubindent = 0.5in
    , totoc
  }
```

4.2.1 Define and Process Package Options

```
\keys_define:nn { sblidx }
  {
    , ancient~sources       .bool_set:N = \l__sblidx_ancient_sources_bool
    , ancient~sources~title .tl_set:N   = \l__sblidx_ancient_sources_title_tl
    , ancient~sources~title .initial:n  = Ancient~Sources~Index
    , modern~authors        .bool_set:N = \l__sblidx_modern_authors_bool
    , modern~authors~title  .tl_set:N   = \l__sblidx_modern_authors_title_tl
    , modern~authors~title  .initial:n  = Modern~Authors~Index
    , subject               .bool_set:N = \l__sblidx_subject_bool
    , subject               .initial:n  = true
    , subject~title         .tl_set:N   = \l__sblidx_subject_title_tl
    , subject~title         .initial:n  = Subject~Index
  }
\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`.

```
\tl_const:Nn \c__sblidx_delim_tl { , \c_space_tl }
\tl_new:N \l__sblidx_delim_first_tl
\tl_const:Nn \c__sblidx_delim_see_tl { . \c_space_tl }
```

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

```
\bool_if:NT \l__sblidx_subject_bool
  {
    \makeindex
      [
        , title   = \l__sblidx_subject_title_tl
        , options = -s ~ sblidx.ist ~ -q
      ]
  }
\bool_if:NT \l__sblidx_ancient_sources_bool
  {
    \RequirePackage { bibleref-parse }
    \RequirePackage { bibleref-sbl }
    \makeindex
      [
        name    = \jobname-ancient-sources ,
        title   = \l__sblidx_ancient_sources_title_tl ,
        options = -s ~ sblidx.ist ~ -q
      ]
    \cs_set_nopar:Npn \biblerefindex
      {
        \index [ \jobname-ancient-sources ]
      }
  }
\bool_if:NT \l__sblidx_modern_authors_bool
  {
    \RequirePackage { biblatex }
    \ExecuteBibliographyOptions { indexing = cite }
    \makeindex
      [
        name    = \jobname-modern-authors ,
        title   = \l__sblidx_modern_authors_title_tl ,
        options = -s ~ sblidx.ist ~ -q
      ]
    \DeclareIndexNameFormat { default }
      {
        \usebibmacro { index:name }
          { \index [ \jobname-modern-authors ] }
          { \namepartfamily }
          { \namepartgiven }
          { \namepartprefix }
          { \namepartsuffix }
      }
  }
```

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.

```
\cs_new_protected:Nn \__sblidx_set_up_footnotes:
  {
    \cs_set_eq:NN \@sblidx@index \index
    \bool_if:NT \l__sblidx_ancient_sources_bool
```

```
      {
        \cs_set_nopar:Npn \bvidxpgformat
          { SBLPageWithNote { \thefootnote } }
      }
    \cs_set_nopar:Npn \index { \SBLFootnoteIndex }
  }
```

(*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.

```
\IfDocumentMetadataTF
  {
    \hook_gput_code:nnn { fntext } { sblidx } { \__sblidx_set_up_footnotes: }
  }
  {
    \cs_set_eq:NN \__sblidx_footnote: \footnote
    \RenewDocumentCommand { \footnote } { o+m }
      {
        \group_begin:
          \__sblidx_set_up_footnotes:
          \tl_if_novalue:nTF {#1}
            { \__sblidx_footnote: {#2} }
            { \__sblidx_footnote: [#1] {#2} }
        \group_end:
      }
  }
```

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

```
\cs_set_nopar:Npn \see #1 #2
  {
    \emph { \text_titlecase_first:n { \seename } }
    \c_space_tl #1
  }
\cs_set_nopar:Npn \seealso #1 #2
  {
    \emph { \text_titlecase_first:n { \alsoname } }
    \c_space_tl #1
  }
```

(*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`.

```
\NewDocumentCommand { \SBLFootnoteIndex } { om }
  {
    \tl_if_novalue:nTF {#1}
```

```
      {
        \str_if_in:nnTF {#2} { | }
          { \@sblidx@index {#2} }
          {
            \@sblidx@index
              { #2 | SBLPageWithNote { \thefootnote } }
          }
      }
      {
        \str_if_in:nnTF {#2} { | }
          { \@sblidx@index [#1] {#2} }
          {
            \@sblidx@index [ #1 ]
              { #2 | SBLPageWithNote { \thefootnote } }
          }
      }
  }
```

(*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}`.

```
\cs_new_protected:Npn \SBLPageWithNote #1 #2
  {
    #2
    \c_space_tl
    \int_compare:nNnTF { \clist_count:n {#1} } > { \c_one_int }
      { nn. }
      { n. }
    \nobreakspace
    \__sblidx_compress_note_list:n {#1}
  }
```

(*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.

```
\cs_new_protected:Npn \SBLPrintIndices
  {
    \__sblidx_set_up_hyperref:
    \bool_if:NT \l__sblidx_ancient_sources_bool
      {
        \group_begin:
          \idxlayout
            {
              subindent    = 0pt ,
              subsubindent = 0.25in ,
              hangindent   = 0.25in
```

```
            }
          \cs_set_nopar:Npn \@idxitem
            { \indexspace \bfseries }
          \cs_set_nopar:Npn \subitem
            { \indexspace \normalfont }
          \tl_set:Nn \l__sblidx_delim_first_tl { \quad \hfill }
          \printindex [ \jobname-ancient-sources ]
        \group_end:
      }
    \bool_if:NT \l__sblidx_modern_authors_bool
      {
        \group_begin:
          \tl_set:Nn \l__sblidx_delim_first_tl { \quad \hfill }
          \printindex [ \jobname-modern-authors ]
        \group_end:
      }
    \bool_if:NT \l__sblidx_subject_bool
      {
        \group_begin:
          \tl_set:Nn \l__sblidx_delim_first_tl { , \c_space_tl }
          \printindex
        \group_end:
      }
  }
```

(*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`.

```
\cs_new_protected:Npn \SBLProcessIndexPages #1
  {
    \__sblidx_process_index_pages:n {#1}
  }
```

(*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.

```
\cs_new_protected:Npn \SBLSetMaxRomanPage #1
  {
    \__sblidx_set_max_roman_page:n {#1}
  }
```

(*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.

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

\cs_new_protected:Nn \__sblidx_set_max_roman_page:n
  {
    \int_gset:Nn \g__sblidx_max_roman_page_int {#1}
  }
```

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

```
\hook_gput_code:nnn { cmd / mainmatter / before } { sblidx }
  {
    \int_gset_eq:NN \g__sblidx_max_roman_page_int \c@page
    \legacy_if:nT { @twoside }
      {
        \int_if_odd:nT { \c@page }
          { \int_gincr:N \g__sblidx_max_roman_page_int }
      }
  }
```

(*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.

```
\int_new:N \l__sblidx_range_start_int
\int_new:N \l__sblidx_range_end_int
\int_new:N \l__sblidx_range_length_int
\int_new:N \l__sblidx_range_end_position_int
\seq_new:N \l__sblidx_range_seq
\tl_new:N  \l__sblidx_compressed_range_tl
\tl_new:N  \l__sblidx_compressed_range_end_tl
\tl_new:N  \l__sblidx_range_start_tl
\tl_new:N  \l__sblidx_range_end_tl

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

\cs_new_protected:Nn \__sblidx_compress_range:n
  {
    \__sblidx_get_compressed_range_end:n {#1}
    \tl_if_empty:NTF \l__sblidx_compressed_range_end_tl
      {#1}
      {
        \tl_set:Ne \l__sblidx_compressed_range_tl
          {
```

```
            \l__sblidx_range_start_tl
            --
            \l__sblidx_compressed_range_end_tl
          }
        \tl_use:N \l__sblidx_compressed_range_tl
      }
  }
```

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

```
\cs_new_protected:Nn \__sblidx_get_compressed_range_end:n
  {
    \tl_clear:N \l__sblidx_compressed_range_end_tl
```

Only Arabic number ranges are compressed

```
    \seq_set_regex_extract_once:NNn
      \l__sblidx_range_seq \c__sblidx_number_range_regex {#1}
    \int_compare:nNnTF
      { \seq_count:N \l__sblidx_range_seq } = { 3 }
      {
        \int_set:Nn \l__sblidx_range_start_int
          { \seq_item:Nn \l__sblidx_range_seq { 2 } }
        \int_set:Nn \l__sblidx_range_end_int
          { \seq_item:Nn \l__sblidx_range_seq { 3 } }
```

First number must be greater than 100.

```
        \int_compare:nNnT
          { \l__sblidx_range_start_int } > { 100 }
          {
```

First number must not be divisible by 100.

```
            \int_compare:nNnT
              {
                \int_mod:nn
                  { \l__sblidx_range_start_int }
                  { 100 }
              }
              >
              { \c_zero_int }
              {
```

Pages must be the same number of digits.

```
                \int_set:Nn \l__sblidx_range_length_int
                  {
                    \tl_count:e { \int_use:N \l__sblidx_range_start_int }
                  }
                \int_compare:nNnT
                  { \l__sblidx_range_length_int }
                  =
                  {
                    \tl_count:e { \int_use:N \l__sblidx_range_end_int }
                  }
                  {
```

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.

```
                \tl_set:No \l__sblidx_range_start_tl
                  { \int_use:N \l__sblidx_range_start_int }
                \tl_set:No \l__sblidx_range_end_tl
                  { \int_use:N \l__sblidx_range_end_int }
                \int_zero:N \l__sblidx_range_end_position_int
                \tl_map_inline:Nn \l__sblidx_range_start_tl
                  {
                    \int_incr:N \l__sblidx_range_end_position_int
                    \int_compare:nNnT
                      { \l__sblidx_range_end_position_int + 1 }
                      =
                      { \l__sblidx_range_length_int }
                      {
                        \tl_set:Ne
                          \l__sblidx_compressed_range_end_tl
                          {
                            \tl_range:Nnn
                              \l__sblidx_range_end_tl
                              { \l__sblidx_range_end_position_int }
                              { \l__sblidx_range_length_int }
                          }
                        \tl_map_break:
                      }
                    \tl_if_eq:neF
                      {##1}
                      {
                        \tl_item:Nn
                          \l__sblidx_range_end_tl
                          { \l__sblidx_range_end_position_int }
                      }
                      {
                        \tl_set:Ne
                          \l__sblidx_compressed_range_end_tl
                          {
                            \tl_range:Nnn
                              \l__sblidx_range_end_tl
                              { \l__sblidx_range_end_position_int }
                              { \l__sblidx_range_length_int }
                          }
                        \tl_map_break:
                      }
                  }
```

Remove leading zeroes from compressed range.

```
                \tl_regex_replace_once:Nnn
                  \l__sblidx_compressed_range_end_tl
                  { \A 0+ }
                  { }
              }
          }
      }
```

```
      }
      {
```

Check for non-Arabic range.

```
        \seq_set_regex_extract_once:NNn
          \l__sblidx_range_seq \c__sblidx_range_regex {#1}
        \seq_if_empty:NF \l__sblidx_range_seq
          {
            \tl_set:Ne \l__sblidx_range_start_tl
              { \seq_item:Nn \l__sblidx_range_seq { 2 } }
            \tl_set:Ne \l__sblidx_range_end_tl
              { \seq_item:Nn \l__sblidx_range_seq { 3 } }
            \tl_set_eq:NN
              \l__sblidx_compressed_range_end_tl \l__sblidx_range_end_tl
          }
      }
  }
```

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

`\__sblidx_compress_note_range:n`
`\__sblidx_compress_page_range:n`
`\__sblidx_hyper_compress_page_rage: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`.

```
\cs_set:Nn \__sblidx_compress_note_range:n
  { \__sblidx_compress_range:n {#1} }
\cs_set:Nn \__sblidx_compress_page_range:n
  { \__sblidx_compress_range:n {#1} }

\cs_new_protected:Nn \__sblidx_hyper_compress_page_range:n
  {
    \__sblidx_get_compressed_range_end:n {#1}
    \tl_if_empty:NTF \l__sblidx_compressed_range_end_tl
      {
        \hyperlink { page. #1 } {#1}
      }
      {
        \tl_set:Ne \l__sblidx_compressed_range_tl
          {
            \exp_not:N \hyperlink
              { page. \l__sblidx_range_start_tl }
              { \l__sblidx_range_start_tl }
            --
            \exp_not:N \hyperlink
              { page. \l__sblidx_range_end_tl }
              { \l__sblidx_compressed_range_end_tl }
          }
        \tl_use:N \l__sblidx_compressed_range_tl
      }
  }
```

(*End of definition for* `\__sblidx_compress_note_range:n`, `\__sblidx_compress_page_range:n`, *and* `\__sblidx_hyper_compress_page_rage:n`.)

__sblidx_do_page:n {⟨*page*⟩}
__sblidx_hyper_do_page:n Create a hyperlink for ⟨***page***⟩. This does nothing if hyperref is not loaded.

```
\cs_new:Nn \__sblidx_do_number:n {#1}
\cs_new:Nn \__sblidx_do_note:n {#1}
\cs_new:Nn \__sblidx_do_page:n {#1}

\cs_new:Nn \__sblidx_hyper_do_page:n
  {
    \exp_not:N \hyperlink { page. #1 } {#1}
  }
```

(*End of definition for* `\__sblidx_do_page:n` *and* `\__sblidx_hyper_do_page:n`.)

__sblidx_compress_list:nN {⟨*sorted numbers*⟩} ⟨***compressed list***⟩

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 ⟨***compressed list***⟩ `clist` variable.

```
\clist_new:N \l__sblidx_uncompressed_clist
\tl_new:N    \l__sblidx_start_tl
\tl_new:N    \l__sblidx_previous_tl

\cs_new_protected:Nn \__sblidx_compress_list:nN
  {
    \clist_set:Nn \l__sblidx_uncompressed_clist {#1}
    \clist_clear:N #2
    \clist_pop:NNT \l__sblidx_uncompressed_clist \l__sblidx_start_tl
      {
        \tl_set_eq:NN \l__sblidx_previous_tl \l__sblidx_start_tl
        \clist_map_inline:Nn \l__sblidx_uncompressed_clist
          {
            \__sblidx_page_compare:eNeTF
              {##1}
              =
              { \__sblidx_page_incr:V \l__sblidx_previous_tl }
              {
                \tl_set:Nn \l__sblidx_previous_tl {##1}
              }
              {
                \__sblidx_append_compressed_list:VVN
                  \l__sblidx_start_tl \l__sblidx_previous_tl #2
                \tl_set:Nn \l__sblidx_start_tl {##1}
                \tl_set:Nn \l__sblidx_previous_tl {##1}
              }
          }
        \__sblidx_append_compressed_list:VVN
          \l__sblidx_start_tl \l__sblidx_previous_tl #2
      }
  }
```

`\@@_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.

```
\cs_new_protected:Nn \__sblidx_append_compressed_list:nnN
```

```
  {
    \__sblidx_page_compare:eNeTF {#2} > { \__sblidx_page_incr:n {#1} }
      {
        \clist_put_right:Ne #3
          { \__sblidx_do_compress_range:n { #1 -- #2 } }
      }
      {
        \clist_put_right:Nn #3
          {
            \__sblidx_do_number:n {#1}
          }
        \tl_if_eq:nnF {#1} {#2}
          {
            \clist_put_right:Nn #3
              {
                \__sblidx_do_number:n {#2}
              }
          }
      }
  }
\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.

```
\cs_set:Nn \__sblidx_do_compress_range:n
  { \__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.

```
\clist_new:N \l__sblidx_compressed_clist
\cs_new_protected:Nn \__sblidx_compress_note_list:n
  {
    \cs_set:Nn \__sblidx_do_compress_range:n
      { \__sblidx_compress_note_range:n {##1} }
    \cs_set:Nn \__sblidx_do_number:n
      { \__sblidx_do_note:n {##1} }
    \__sblidx_compress_list:nN {#1} \l__sblidx_compressed_clist
    \clist_use:Nnnn
      \l__sblidx_compressed_clist
      { ~ and ~ } { , ~ } { , ~ and ~ }
  }
```

(*End of definition for* `\__sblidx_compress_note_list:n`.)

`\__sblidx_compress_page_list:nN` `\__sblidx_compress_page_list:VN` `{`⟨*sorted page numbers*⟩`}` ⟨***list of compressed page***⟩

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

```
\cs_new_protected:Nn \__sblidx_compress_page_list:nN
  {
    \cs_set:Nn \__sblidx_do_compress_range:n
      { \__sblidx_compress_page_range:n {##1} }
    \cs_set:Nn \__sblidx_do_number:n
      { \__sblidx_do_page:n {##1} }
    \__sblidx_compress_list:nN {#1} \l__sblidx_compressed_clist
    \tl_set:Ne
      #2
      {
        \clist_use:Nn \l__sblidx_compressed_clist { \c__sblidx_delim_tl }
      }
  }
\cs_generate_variant:Nn \__sblidx_compress_page_list:nN { VN }
```

(*End of definition for* `\__sblidx_compress_page_list:nN`.)

`\__sblidx_page_compare:nNnTF`
`\__sblidx_page_compare:eNeTF`
`\__sblidx_page_compare:VNVT`

`{`⟨*first page number*⟩`}` ⟨`relation`⟩ `{`⟨*second page number*⟩`}` `{`⟨*true code*⟩`}` `{`⟨*false code*⟩`}`

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.

```
\int_new:N \l__sblidx_first_int
\int_new:N \l__sblidx_second_int
\prg_new_protected_conditional:Npnn \__sblidx_page_compare:nNn #1 #2 #3
  { T, TF }
  {
```

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

```
    \regex_if_match:nnTF { \A \d+ \Z } {#1}
      {
        \int_set:Nn \l__sblidx_first_int
          { #1 + \g__sblidx_max_roman_page_int }
      }
      {
        \int_set:Nn \l__sblidx_first_int
          { \exp_args:Ne \int_from_roman:n {#1} }
      }
    \regex_if_match:nnTF { \A \d+ \Z } {#3}
      {
        \int_set:Nn \l__sblidx_second_int
          { #3 + \g__sblidx_max_roman_page_int }
      }
      {
        \int_set:Nn \l__sblidx_second_int
          { \exp_args:Ne \int_from_roman:n {#3} }
      }
```

Now we just need a simple integer comparison.

```
    \if_int_compare:w \l__sblidx_first_int #2 \l__sblidx_second_int
      \prg_return_true:
```

```
    \else:
      \prg_return_false:
    \fi:
  }
\cs_generate_variant:Nn \__sblidx_page_compare:nNnT  { VNVT }
\cs_generate_variant:Nn \__sblidx_page_compare:nNnTF { eNeTF }
```

(*End of definition for* `\__sblidx_page_compare:nNnTF` *and* `\__sblidx_page_compare:VNVT`.)

`\__sblidx_page_incr:n` `{`⟨*page number*⟩`}`

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.

```
\cs_new:Nn \__sblidx_page_incr:n
  {
    \exp_args:Ne \__sblidx_page_incr:nn { \str_head:n {#1} } {#1}
  }
```

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

```
\cs_new:Nn \__sblidx_page_incr:nn
  {
    \str_case:nnTF {#1}
      {
        { 0 } { } { 1 } { } { 2 } { } { 3 } { } { 4 } { }
        { 5 } { } { 6 } { } { 7 } { } { 8 } { } { 9 } { }
      }
      {
        \int_eval:n { #2 + 1 }
      }
      {
        \int_compare:nNnTF
          { \int_from_roman:n {#2} } = { \g__sblidx_max_roman_page_int }
          { 1 }
          { \int_to_roman:n { \int_from_roman:n {#2} + 1 } }
      }
  }
\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.

```
\bool_new:N  \l__sblidx_hyperref_bool
\regex_new:N \l__sblidx_page_regex
```

```
\regex_new:N \l__sblidx_page_encap_regex
\regex_new:N \l__sblidx_see_also_regex
\regex_new:N \l__sblidx_page_with_note_regex
\cs_new_protected:Nn \__sblidx_set_up_hyperref:
  {
    \cs_if_exist:NTF \hyperxindexformat
      {
        \bool_set_true:N \l__sblidx_hyperref_bool
        \cs_set:Nn \__sblidx_compress_page_range:n
          { \__sblidx_hyper_compress_page_range:n {##1} }
        \cs_set:Nn \__sblidx_do_page:n
          { \__sblidx_hyper_do_page:n {##1} }
        \regex_set:Nn \l__sblidx_page_regex
          { \A \c{ hyperpage } \cB\{ ( .*? ) \cE\} \Z }
        \regex_set:Nn \l__sblidx_page_encap_regex
          {
            \A \c{ hyperxindexformat } \cB\{ ( \c{ .* } .* ) \cE\}
              \cB\{ ( .*? ) \cE\} \Z
          }
        \regex_set:Nn \l__sblidx_see_also_regex
          {
            \A \c{ hyperxindexformat }
              \cB\{ ( [ \c{ see } \c{ seealso } ] .* ) \cE\}
              \cB\{ .*? \cE\} \Z
          }
        \regex_set:Nn \l__sblidx_page_with_note_regex
          {
            \A \c{ hyperxindexformat }
              \cB\{ \c{ SBLPageWithNote } \cB\{ ( .*? ) \cE\} \cE\}
              \cB\{ ( .*? ) \cE\} \Z
          }
      }
      {
        \cs_set:Nn \__sblidx_compress_page_range:n
          { \__sblidx_compress_range:n {##1} }
        \cs_set:Nn \__sblidx_do_page:n {##1}
        \regex_set:Nn \l__sblidx_page_encap_regex
          { \A ( \c{ .* } .* ) \cB\{ ( .*? ) \cE\} \Z }
        \regex_set:Nn \l__sblidx_see_also_regex
          { \A ( [ \c{ see } \c{ seealso } ] .* ) \cB\{ .*? \cE\} \Z }
        \regex_set:Nn \l__sblidx_page_with_note_regex
          {
            \A \c{ SBLPageWithNote }
              \cB\{ ( .*? ) \cE\} \cB\{ ( .*? ) \cE\} \Z
          }
      }
  }
```

(*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={}`.

```
\seq_new:N       \l__sblidx_page_data_seq
\tl_new:N        \l__sblidx_page_tl
\tl_new:N        \l__sblidx_note_tl
\cs_new_nopar:Nn \__sblidx_encap: { \exp_not:n {} }

\keys_define:nn { sblidx / pagedata }
  {
    , page  .tl_set:N = \l__sblidx_page_tl
    , note  .tl_set:N = \l__sblidx_note_tl
    , encap .code:n   = \cs_set_nopar:Nn \__sblidx_encap:
                          { \exp_not:n {#1} }
  }
```

(*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.

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

\cs_new_protected:Nn \__sblidx_sort_page_data:N
  {
    \seq_sort:Nn #1
      {
        \bool_set_false:N \l__sblidx_sort_returned_bool
        \keys_set:nn { sblidx / pagedata } {##1}
        \tl_set_eq:NN \l_tmpa_tl \l__sblidx_page_tl
        \tl_set_eq:NN \l_tmpb_tl \l__sblidx_note_tl
        \cs_set_eq:NN \__sblidx_tmpa: \__sblidx_encap:
        \keys_set:nn { sblidx / pagedata } {##2}
```

Page is primary sort key.

```
        \__sblidx_page_compare:VNVT \l_tmpa_tl > \l__sblidx_page_tl
          {
            \bool_set_true:N \l__sblidx_sort_returned_bool
            \sort_return_swapped:
          }
        \__sblidx_page_compare:VNVT \l_tmpa_tl < \l__sblidx_page_tl
          {
            \bool_set_true:N \l__sblidx_sort_returned_bool
            \sort_return_same:
          }
        \tl_if_eq:NNT \l_tmpa_tl \l__sblidx_page_tl
          {
```

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

```
            \__sblidx_page_data_weight:NNN
              \l_tmpb_tl \__sblidx_tmpa: \l_tmpa_int
            \__sblidx_page_data_weight:NNN
              \l__sblidx_note_tl \__sblidx_encap: \l_tmpb_int
            \int_compare:nNnT { \l_tmpa_int } < { \l_tmpb_int }
              {
                \bool_set_true:N \l__sblidx_sort_returned_bool
```

```
                \sort_return_same:
              }
            \int_compare:nNnT { \l_tmpa_int } > { \l_tmpb_int }
              {
                \bool_set_true:N \l__sblidx_sort_returned_bool
                \sort_return_swapped:
              }
```

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

```
            \int_compare:nNnT { \l_tmpa_int } = { \l_tmpb_int }
              {
                \bool_set_true:N \l__sblidx_sort_returned_bool
                \int_compare:nNnTF { \l_tmpa_int } = { 2 }
                  {
                    \tl_if_empty:NTF \l__sblidx_note_tl
                      { \int_zero:N \l_tmpa_int }
                      { \int_set:Nn \l_tmpa_int { \l__sblidx_note_tl } }
                    \tl_if_empty:NTF \l_tmpb_tl
                      { \int_zero:N \l_tmpb_int }
                      { \int_set:Nn \l_tmpb_int { \l_tmpb_tl } }
                    \int_compare:nNnTF { \l_tmpa_int } < { \l_tmpb_int }
                      { \sort_return_swapped: }
                      { \sort_return_same: }
                  }
                  { \sort_return_same: }
              }
          }
```

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

```
        \bool_if:NF \l__sblidx_sort_returned_bool
          { \sort_return_same: }
      }
  }
```

(*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:`.

```
\cs_new_protected:Nn \__sblidx_page_data_weight:NNN
  {
    \cs_if_eq:NNTF #2 \__sblidx_empty_encap:
      {
        \tl_if_empty:NTF #1
          {
            \int_set:Nn #3 { \c_one_int }
          }
          {
            \int_set:Nn #3 { 2 }
          }
      }
```

```
      { \int_zero:N #3 }
  }
```

(*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.

```
\bool_new:N \l__sblidx_item_added_bool
\tl_new:N   \l__sblidx_compressed_page_tl

\cs_new_protected:Nn \__sblidx_page_data_to_tl:N
  {
    \tl_clear:N \l_tmpa_tl
    \seq_map_inline:Nn #1
      {
        \bool_set_false:N \l__sblidx_item_added_bool
        \keys_set:nn { sblidx / pagedata } {##1}
        \__sblidx_compress_page_list:VN
          \l__sblidx_page_tl
          \l__sblidx_compressed_page_tl
        \cs_if_eq:NNF \__sblidx_encap: \__sblidx_empty_encap:
          {
```

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

```
            \tl_set:Ne \l_tmpb_tl { \__sblidx_encap: }
            \tl_if_in:NnTF \l_tmpb_tl { \see }
              {
                \tl_put_right:NV \l_tmpa_tl \c__sblidx_delim_see_tl
              }
              {
                \tl_if_in:NnTF \l_tmpb_tl { \seealso }
                  {
                    \tl_put_right:NV \l_tmpa_tl \c__sblidx_delim_see_tl
                  }
                  {
                    \tl_if_empty:NTF \l_tmpa_tl
                      { \tl_set:NV \l_tmpa_tl \l__sblidx_delim_first_tl }
                      { \tl_put_right:NV \l_tmpa_tl \c__sblidx_delim_tl }
                  }
              }
            \tl_put_right:Ne \l_tmpa_tl
              {
                \__sblidx_encap: { \l__sblidx_compressed_page_tl }
              }
            \bool_set_true:N \l__sblidx_item_added_bool
          }
        \bool_if:NF \l__sblidx_item_added_bool
          {
            \tl_if_empty:NF \l__sblidx_note_tl
              {
                \tl_if_empty:NTF \l_tmpa_tl
                  { \tl_set:NV \l_tmpa_tl \l__sblidx_delim_first_tl }
                  { \tl_put_right:NV \l_tmpa_tl \c__sblidx_delim_tl }
```

```
                \tl_put_right:Ne \l_tmpa_tl
                  {
                    \SBLPageWithNote
                      { \l__sblidx_note_tl }
                      { \l__sblidx_compressed_page_tl }
                  }
                \bool_set_true:N \l__sblidx_item_added_bool
              }
          }
        \bool_if:NF \l__sblidx_item_added_bool
          {
            \tl_if_empty:NTF \l_tmpa_tl
              { \tl_set:NV \l_tmpa_tl \l__sblidx_delim_first_tl }
              { \tl_put_right:NV \l_tmpa_tl \c__sblidx_delim_tl }
            \tl_put_right:Ne \l_tmpa_tl
              { \l__sblidx_compressed_page_tl }
          }
      }
```

Put formatted index pages on to the input stream.

```
    \tl_use:N \l_tmpa_tl
  }
```

(*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.

```
\bool_new:N \l__sblidx_stepping_page_bool
\int_new:N  \g__sblidx_status_code_int
\tl_new:N   \l__sblidx_end_tl
```

(*End of definition for* `\g__sblidx_status_code_int`.)

`\__sblidx_range_to_clist:nN` `\__sblidx_range_to_clist:eN`

`{`⟨*range*⟩`}` ⟨`list`⟩

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`⟩.

```
\cs_new_protected:Nn \__sblidx_range_to_clist:nN
  {
    \seq_set_regex_extract_once:NNn
      \l_tmpb_seq \c__sblidx_range_regex {#1}
    \seq_if_empty:NTF \l_tmpb_seq
      {
        \clist_set:Nn #2 {#1}
        \int_gset:Nn \g__sblidx_status_code_int { \c_one_int }
      }
      {
        \clist_clear:N #2
        \bool_set_true:N \l__sblidx_stepping_page_bool
        \tl_set:Ne \l__sblidx_page_tl { \seq_item:Nn \l_tmpb_seq { 2 } }
        \tl_set:Ne \l__sblidx_end_tl { \seq_item:Nn \l_tmpb_seq { 3 } }
        \bool_do_while:nn { \l__sblidx_stepping_page_bool }
```

```
          {
            \clist_put_right:NV #2 \l__sblidx_page_tl
            \tl_set:Ne \l__sblidx_page_tl
              { \__sblidx_page_incr:V \l__sblidx_page_tl }
            \__sblidx_page_compare:VNVT \l__sblidx_page_tl = \l__sblidx_end_tl
              {
                \clist_put_right:NV #2 \l__sblidx_page_tl
                \bool_set_false:N \l__sblidx_stepping_page_bool
              }
          }
        \int_gzero:N \g__sblidx_status_code_int
      }
  }
\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***⟩.

```
\cs_new_protected:Nn \__sblidx_check_page:n
  {
    \tl_clear:N \l_tmpa_tl
    \bool_if:NTF \l__sblidx_hyperref_bool
      {
        \seq_set_regex_extract_once:NNn
          \l_tmpa_seq \l__sblidx_page_regex {#1}
        \seq_if_empty:NF \l_tmpa_seq
          {
            \tl_set:Ne \l_tmpa_tl { \seq_item:Nn \l_tmpa_seq { 2 } }
          }
      }
      {
        \tl_set:Nn \l_tmpa_tl {#1}
      }
    \tl_if_empty:NTF \l_tmpa_tl
      {
        \int_gset:Nn \g__sblidx_status_code_int { \c_one_int }
      }
      {
        \clist_set:NV \l_tmpa_clist \l_tmpa_tl
        \clist_map_inline:Nn \l_tmpa_clist
          {
            \__sblidx_range_to_clist:nN {##1} \l_tmpb_clist
            \clist_map_inline:Nn \l_tmpb_clist
              {
                \seq_put_right:Ne \l__sblidx_page_data_seq
                  {
                    page  = {####1} ,
                    note  = { } ,
                    encap = { }
                  }
```

```
            }
          }
        \int_gzero:N \g__sblidx_status_code_int
      }
  }
```

(*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***⟩.

```
\cs_new_protected:Nn \__sblidx_check_page_encap:n
  {
    \seq_set_regex_extract_once:NNn
      \l_tmpa_seq \l__sblidx_page_encap_regex {#1}
    \seq_if_empty:NTF \l_tmpa_seq
      {
        \int_gset:Nn \g__sblidx_status_code_int { \c_one_int }
      }
      {
        \__sblidx_range_to_clist:eN
          { \seq_item:Nn \l_tmpa_seq { 3 } }
          \l_tmpa_clist
        \clist_map_inline:Nn \l_tmpa_clist
          {
            \seq_put_right:Ne \l__sblidx_page_data_seq
              {
                page  = {##1} ,
                note  = { } ,
                encap = { \seq_item:Nn \l_tmpa_seq { 2 } }
              }
          }
        \int_gzero:N \g__sblidx_status_code_int
      }
  }
```

(*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***⟩.

```
\int_new:N \l__sblidx_see_also_int
\int_set:Nn \l__sblidx_see_also_int { 100000 }

\cs_new_protected:Nn \__sblidx_check_see_also:n
  {
    \seq_set_regex_extract_once:NNn
      \l_tmpa_seq \l__sblidx_see_also_regex {#1}
    \seq_if_empty:NTF \l_tmpa_seq
      {
        \int_gset:Nn \g__sblidx_status_code_int { \c_one_int }
```

```
      }
      {
        \int_incr:N \l__sblidx_see_also_int
        \seq_put_right:Ne \l__sblidx_page_data_seq
          {
            page  = { \int_use:N \l__sblidx_see_also_int } ,
            note  = { } ,
            encap = { \seq_item:Nn \l_tmpa_seq { 2 } }
          }
        \int_gzero:N \g__sblidx_status_code_int
      }
  }
```

(*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***⟩.

```
\cs_new_protected:Nn \__sblidx_check_page_with_note:n
  {
    \seq_set_regex_extract_once:NNn
      \l_tmpa_seq \l__sblidx_page_with_note_regex {#1}
    \seq_if_empty:NTF \l_tmpa_seq
      {
        \int_gset:Nn \g__sblidx_status_code_int { \c_one_int }
      }
      {
        \seq_put_right:Ne \l__sblidx_page_data_seq
          {
            page  = { \seq_item:Nn \l_tmpa_seq { 3 } } ,
            note  = { \seq_item:Nn \l_tmpa_seq { 2 } } ,
            encap = { }
          }
        \int_gzero:N \g__sblidx_status_code_int
      }
  }
```

(*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.

```
\bool_new:N  \l__sblidx_put_page_data_bool
\clist_new:N \l__sblidx_notes_clist
\clist_new:N \l__sblidx_pages_clist

\cs_new_protected:Nn \__sblidx_consolidate_page_data:N
  {
    \keys_set:ne { sblidx / pagedata }
      {
        \seq_item:Nn #1 { \c_one_int }
      }
```

```
    \clist_set:NV \l__sblidx_pages_clist \l__sblidx_page_tl
    \clist_set:NV \l__sblidx_notes_clist \l__sblidx_note_tl
    \cs_set_eq:NN \__sblidx_encap_previous: \__sblidx_encap:
    \seq_clear:N \l_tmpa_seq
    \int_step_inline:nnn { 2 } { \seq_count:N #1 }
      {
        \keys_set:ne { sblidx / pagedata }
          { \seq_item:Nn #1 {##1} }
        \tl_if_eq:eeTF
          { \l__sblidx_page_tl }
          { \clist_item:Nn \l__sblidx_pages_clist { -1 } }
          {
            \clist_if_empty:NF \l__sblidx_notes_clist
              {
                \clist_put_right:NV
                  \l__sblidx_notes_clist \l__sblidx_note_tl
              }
          }
          {
            \bool_set_false:N \l__sblidx_put_page_data_bool
```

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

```
            \cs_if_eq:NNF \__sblidx_encap: \__sblidx_encap_previous:
              { \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.

```
            \clist_if_empty:NF \l__sblidx_notes_clist
              { \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.

```
            \tl_if_empty:NF \l__sblidx_note_tl
              { \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.

```
            \bool_if:NTF \l__sblidx_put_page_data_bool
              {
                \seq_put_right:Ne \l_tmpa_seq
                  {
                    page  = { \clist_use:N \l__sblidx_pages_clist } ,
                    note  = { \clist_use:N \l__sblidx_notes_clist } ,
                    encap = { \__sblidx_encap_previous: }
                  }
                \clist_set:NV \l__sblidx_pages_clist \l__sblidx_page_tl
                \clist_set:NV \l__sblidx_notes_clist \l__sblidx_note_tl
                \cs_set_eq:NN \__sblidx_encap_previous: \__sblidx_encap:
              }
              {
                \clist_put_right:NV
                  \l__sblidx_pages_clist \l__sblidx_page_tl
              }
          }
      }
    \seq_put_right:Ne \l_tmpa_seq
      {
```

```
        page  = { \clist_use:N \l__sblidx_pages_clist } ,
        note  = { \clist_use:N \l__sblidx_notes_clist } ,
        encap = { \__sblidx_encap_previous: }
      }
    \seq_set_eq:NN #1 \l_tmpa_seq
  }
```

(*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.

```
\cs_new_protected:Nn \__sblidx_process_index_pages:n
  {
    \seq_clear:N \l__sblidx_page_data_seq
    \int_set:Nn \l__sblidx_see_also_int { 100000 }
    \clist_map_inline:nn { #1 }
      {
        \__sblidx_check_page_with_note:n {##1}
        \int_compare:nNnF { \g__sblidx_status_code_int } = { \c_zero_int }
          { \__sblidx_check_see_also:n {##1} }
        \int_compare:nNnF { \g__sblidx_status_code_int } = { \c_zero_int }
          { \__sblidx_check_page_encap:n {##1} }
        \int_compare:nNnF { \g__sblidx_status_code_int } = { \c_zero_int }
          { \__sblidx_check_page:n {##1} }
      }
    \__sblidx_sort_page_data:N \l__sblidx_page_data_seq
    \__sblidx_consolidate_page_data:N \l__sblidx_page_data_seq
    \__sblidx_page_data_to_tl:N \l__sblidx_page_data_seq
  }
```

(*End of definition for* `\__sblidx_process_index_pages:n`.)

```
\endinput
```

⟨/package⟩