glossaries-extra.sty v1.37: documented code

Nicola L.C. Talbot

Dickimaw Books
http://www.dickimaw-books.com/

2018-11-30

Abstract

This is the documented code for the glossaries-extra package. See glossaries-extra-manual.pdf for the user manual.

This package is experimental and not stable. It's provided for testing purposes only. Future versions may not be compatible with this version. Once it has stabilised I'll add it to CTAN, at which point compatibility with the first stable version will be maintained.

Contents

1 Main Package Code (glossaries-extra.sty)

1.1 Package Initialisation and Options

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-extra}[2018/11/30 1.37 (NLCT)]
```

Requires xkeyval to define package options.

```
\RequirePackage{xkeyval}
```

Requires etoolbox package.

```
\RequirePackage{etoolbox}
```

Has glossaries already been loaded?

```
\@ifpackageloaded{glossaries}
{%
```

Already loaded so pass any options to `\setupglossaries`. This means that the options that can only be set when glossaries is loaded can't be used.

```
  \newcommand{\glsxtr@dooption}[1]{\setupglossaries{#1}}%
  \let\@glsxtr@declareoption\@gls@declareoption
}
{%
```

Not already loaded, so pass options to glossaries.

```
  \newcommand{\glsxtr@dooption}[1]{%
    \PassOptionsToPackage{#1}{glossaries}%
  }%
```

Set the defaults.

```
  \PassOptionsToPackage{toc}{glossaries}
  \PassOptionsToPackage{nopostdot}{glossaries}
  \PassOptionsToPackage{noredefwarn}{glossaries}
  \@ifpackageloaded{polyglossia}%
  {}%
  {%
    \@ifpackageloaded{babel}%
    {\PassOptionsToPackage{translate=babel}{glossaries}}%
    {}%
  }%
  \newcommand*{\@glsxtr@declareoption}[2]{%
    \DeclareOptionX{#1}{#2}%
    \DeclareOption{#1}{#2}%
  }
}
```

Declare package options.

sxtrundefaction Determines what to do if an entry hasn't been defined. The two arguments are the error or warning message and the help message if an error should be produced.

```
\newcommand*{\glsxtrundefaction}[2]{%
  \@glsxtrundeftag\PackageError{glossaries-extra}{#1}{#2}%
}
```

arnonexistsordo If user wants undefaction=warn, then glossaries v4.19 is required.

```
\newcommand*{\glsxtr@warnonexistsordo}[1]{}
```

\glsxtrundeftag Text to display when an entry doesn't exist.

```
\newcommand*{\glsxtrundeftag}{??}
\newcommand*{\@glsxtrundeftag}{}
```

This text is switched on at the start of the document to prevent unwanted text inserted into the preamble if any tests are made before the start of the document.

arn@undefaction This is how \glsxtrundefaction should behave if undefaction=warn is set.

```
\newcommand*{\@glsxtr@warn@undefaction}[2]{%
  \@glsxtrundeftag\GlossariesExtraWarning{#1}%
}
```

err@undefaction This is how \glsxtrundefaction should behave if undefaction=error is set.

```
\newcommand*{\@glsxtr@err@undefaction}[2]{%
  \@glsxtrundeftag\PackageError{glossaries-extra}{#1}{#2}%
}
```

rn@onexistsordo This is how \glsxtr@warnonexistsordo should behave if undefaction=warn is set.

```
\newcommand*{\@glsxtr@warn@onexistsordo}[1]{%
  \GlossariesExtraWarning{\string#1\space hasn't been defined, so
  some errors won't be converted to warnings.
  (This most likely means your version of
  glossaries.sty is below version 4.19.)}%
}
```

f@forglsentries

```
\newcommand*{\@glsxtr@redef@forglsentries}{}
```

f@forglsentries

```
\newcommand*{\@glsxtr@do@redef@forglsentries}{%
  \renewcommand*{\forglsentries}[3][\glsdefaulttype]{%
    \edef\@@glo@list{\csname glolist@##1\endcsname}%
    \ifdefstring{\@@glo@list}{,}%
    {%
      \GlossariesExtraWarning{No entries defined in glossary `##1'}%
    }%
    {%
      \@for##2:=\@@glo@list\do
```

```
      {%
        \ifdefempty{##2}{}{##3}%
      }%
    }%
  }%
}%
```

undefaction

```
\define@choicekey{glossaries-extra.sty}{undefaction}%
 [\glsxtr@undefaction@val\glsxtr@undefaction@nr]%
 {warn,error}%
 {%
   \ifcase\glsxtr@undefaction@nr\relax
     \let\glsxtrundefaction\@glsxtr@warn@undefaction
     \let\glsxtr@warnonexistsordo\@glsxtr@warn@onexistsordo
     \let\@glsxtr@redef@forglsentries\@glsxtr@do@redef@forglsentries
   \or
     \let\glsxtrundefaction\@glsxtr@err@undefaction
     \let\glsxtr@warnonexistsordo\@gobble
     \let\@glsxtr@redef@forglsentries\relax
   \fi
 }
```

To assist bib2gls, v1.08 introduces the record option, which will write information to the aux file whenever an entry needs to be indexed.

\@glsxtr@record Does nothing by default.

```
\newcommand*{\@glsxtr@record}[3]{}
```

lsxtr@recordsee Does nothing by default.

```
\newcommand*{\glsxtr@recordsee}[2]{}
```

ultnumberformat

```
\newcommand*{\@glsxtr@defaultnumberformat}{glsnumberformat}%
```

ultNumberFormat

```
\newcommand*{\GlsXtrSetDefaultNumberFormat}[1]{%
  \renewcommand*{\@glsxtr@defaultnumberformat}{#1}%
}%
```

The record option is somewhat problematic. On the first LaTeX run the entries aren't defined. This isn't as straight-forward as commands like \cite since attributes associated with the entry's category may switch off the indexing or the entry's glossary type might require a particular counter. This kind of information can't be determined until the entry has been defined. So there are two different commands here. One that's used if the entry hasn't been defined, which tries to use sensible defaults, and one which is used when the entry has been defined.

cord@wrglossary The record=only option sets \@@do@wrglossary to this command, which means it's done within \glsadd and \@gls@link, and so is only done if the entry exists.

```
\newcommand*{\@glsxtr@do@record@wrglossary}[1]{%
 \begingroup
   \ifKV@glslink@noindex
   \else
     \edef\@gls@label{\glsdetoklabel{#1}}%
     \let\glslabel\@gls@label
     \glswriteentry{#1}%
     {%
       \ifdefempty{\@glsxtr@thevalue}%
       {%
         \ifx\@glsxtr@org@theHvalue\@glsxtr@theHvalue
         \else
           \let\theHglsentrycounter\@glsxtr@theHvalue
         \fi
         \glsxtr@saveentrycounter
         \let\@@do@@wrglossary\@glsxtr@dorecord
       }%
       {%
         \let\theglsentrycounter\@glsxtr@thevalue
         \let\theHglsentrycounter\@glsxtr@theHvalue
         \let\@@do@@wrglossary\@glsxtr@dorecordnodefer
       }%
       \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
         \glsxtr@@do@wrglossary{#1}%
       \else
         \@@glsxtrwrglossmark
```

Increment associated counter.

```
         \glsxtr@inc@wrglossaryctr{#1}%
         \@@do@@wrglossary
       \fi
     }%
   \fi
 \endgroup
}
```

ndex@wrglossary The record=alsoindex option needs to both record and index.

```
\newcommand*{\glsxtr@do@alsoindex@wrglossary}[1]{%
  \glsxtr@@do@wrglossary{#1}%
  \@glsxtr@do@record@wrglossary{#1}%
}
```

@@glsxtr@record The record=only option sets \@glsxtr@record to this. This performs the recording if the entry *doesn't exist* and is done at the start of \@gls@field@link and commands like \@gls@ (before the existence test). This means that it disregards the wrgloss key.

The first argument is the option list (as passed in the first optional argument to commands like \gls). This allows the noindex setting to be picked up. The second argument is the entry's

label. The third argument is the key family (glslink in most cases, glossadd for \glsadd).

```
\newcommand*{\@@glsxtr@record}[3]{%
 \ifglsentryexists{#2}{}%
 {%
   \@@glsxtrwrglossmark
   \begingroup
```

Save the label in case it's needed.

```
     \edef\@gls@label{\glsdetoklabel{#2}}%
     \let\glslabel\@gls@label
     \let\@glsnumberformat\@glsxtr@defaultnumberformat
     \def\@glsxtr@thevalue{}%
     \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
     \let\@glsxtr@org@theHvalue\@glsxtr@theHvalue
```

Entry hasn't been defined, so we'll have to assume it's \glscounter by default.

```
     \let\@gls@counter\glscounter
```

Unless the equations option is on and this is inside a numbered maths environment.

```
     \if@glsxtr@equations
       \@glsxtr@use@equation@counter
     \fi
```

Check for default options (which may switch off indexing).

```
     \@gls@setdefault@glslink@opts
```

Implement any pre-key settings.

```
     \csuse{@glsxtr@#3@prekeys}%
```

Assign keys.

```
     \setkeys{#3}{#1}%
```

Implement any post-key settings. Is the auto-add on?

```
     \glsxtr@do@autoadd{#3}%
```

Check post-key hook.

```
     \csuse{@glsxtr@#3@postkeys}%
```

Increment associated counter.

```
     \glsxtr@inc@wrglossaryctr{#2}%
```

Check if noindex option has been used.

```
     \ifKV@glslink@noindex
     \else
       \glswriteentry{#2}%
       {%
```

Check if thevalue has been set.

```
         \ifdefempty{\@glsxtr@thevalue}%
         {%
```

Key thevalue hasn't been set, but check if theHvalue has been set. (Not particularly likely, but allow for it.)

```
           \ifx\@glsxtr@org@theHvalue\@glsxtr@theHvalue
```

```
            \else
              \let\theHglsentrycounter\@glsxtr@theHvalue
            \fi
```

Save the entry counter.

```
            \glsxtr@saveentrycounter
```

Temporarily redefine `\@@do@@wrglossary` for use with `\glsxtr@@do@wrglossary`.

```
            \let\@@do@@wrglossary\@glsxtr@dorecord
          }%
          {%
```

thevalue has been set, so there's no need to defer writing the location value. (If it's dependent on the page counter, the counter key should be set instead.)

```
            \let\theglsentrycounter\@glsxtr@thevalue
            \let\theHglsentrycounter\@glsxtr@theHvalue
            \let\@@do@@wrglossary\@glsxtr@dorecordnodefer
          }%
          \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
            \glsxtr@@do@wrglossary{#2}%
          \else
```

No need to escape special characters.

```
            \@@do@@wrglossary
          \fi
        }%
      \fi
    \endgroup
  }%
}
```

`glslink@prekeys`

```
\newcommand{\@glsxtr@glslink@prekeys}{\glslinkpresetkeys}
```

`lslink@postkeys`

```
\newcommand{\@glsxtr@glslink@postkeys}{\glslinkpostsetkeys}
```

`lossadd@prekeys`

```
\newcommand{\@glsxtr@glossadd@prekeys}{\glsaddpresetkeys}
```

`ossadd@postkeys`

```
\newcommand{\@glsxtr@glossadd@postkeys}{\glsaddpostsetkeys}
```

`glsxtr@dorecord` If record=alsoindex is used, then `\@glslocref` may have been escaped, but this isn't appropriate here.

```
\newcommand*\@glsxtr@dorecord{%
  \global\let\@glsrecordlocref\theglsentrycounter
  \let\@glsxtr@orgprefix\@glo@counterprefix
  \ifx\theglsentrycounter\theHglsentrycounter
    \def\@glo@counterprefix{}%
```

```
  \else
    \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
      {\theglsentrycounter}{\theHglsentrycounter}%
    }%
    \@do@gls@getcounterprefix
  \fi
```

Don't protect the `\@glsrecordlocref` from premature expansion. If the counter isn't

`page` then it needs expanding. If the location includes `\thepage` then `\protected@write` will automatically deal with it.

```
  \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
    \@glsxtr@do@nameref@record
     {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
     {\@glsrecordlocref}%
  \else
    \protected@write\@auxout{}{\string\glsxtr@record
       {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
       {\@glsrecordlocref}}%
  \fi
  \@glsxtr@counterrecordhook
  \let\@glo@counterprefix\@glsxtr@orgprefix
}
```

`dorecordnodefer` As above, but don't defer expansion of location. This uses `\theglsentrycounter` directly for the location rather than `\@glslocref` since there's no need to guard against premature expansion of the page counter.

```
\newcommand*\@glsxtr@dorecordnodefer{%
  \ifx\theglsentrycounter\theHglsentrycounter
    \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
      \@glsxtr@do@nameref@record
         {\@gls@label}{}{\@gls@counter}{\@glsnumberformat}%
         {\theglsentrycounter}%
    \else
      \protected@write\@auxout{}{\string\glsxtr@record
         {\@gls@label}{}{\@gls@counter}{\@glsnumberformat}%
         {\theglsentrycounter}}%
    \fi
  \else
    \edef\@do@gls@getcounterprefix{\noexpand\@gls@getcounterprefix
      {\theglsentrycounter}{\theHglsentrycounter}%
    }%
    \@do@gls@getcounterprefix
    \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
      \@glsxtr@do@nameref@record
         {\@gls@label}{\@glo@counterprefix}{\@gls@counter}%
         {\@glsnumberformat}{\theglsentrycounter}%
    \else
      \protected@write\@auxout{}{\string\glsxtr@record
        {\@gls@label}{\@glo@counterprefix}{\@gls@counter}{\@glsnumberformat}%
```

```
          {\theglsentrycounter}}%
      \fi
    \fi
    \@glsxtr@counterrecordhook
}
```

xtr@ifnum@mmode

Check if in a numbered maths environment. The amsmath package is automatically loaded by datatool-base, which is required by glossaries, so `\ifst@rred` and `\if@display` should both be defined.

```
\newcommand{\@glsxtr@ifnum@mmode}[2]{%
 \ifmmode
   \ifst@rred
    #2%
   \else
```

Non-amsmath environments and regular inline math mode isn't flagged as starred by amsmath, but we can't use `\mathchoice` in this case as it's not the current style that's relevant. Instead we can use amsmath's `\if@display`. This may not work for environments that aren't provided by amsmath.

```
     \if@display #1\else #2\fi
   \fi
 \else
   #2%
 \fi
}
```

@nameref@record

With record=nameref, the current label information is included in the record, but this may not have been defined, so `\csuse` will prevent an undefined control sequence error and just leave the last two arguments blank if there's no information. In the event that a record is in amsmath's align environment `\@currentHref` will be out. There may be other instances where `\@currentHref` is out, so this also saves `\theHglsentrycounter`, which is useful if it can't be obtained by prefixing `\theglsentrycounter`.

```
\newcommand*{\@glsxtr@do@nameref@record}[5]{%
  \gls@ifnotmeasuring
  {%
    \protected@write\@auxout{}{\string\glsxtr@record@nameref
     {#1}{#2}{#3}{#4}{#5}%
     {\csuse{@currentlabelname}}{\csuse{@currentHref}}%
     {\theHglsentrycounter}}%
  }%
}
```

r@recordcounter

```
\newcommand*{\@@glsxtr@recordcounter}{%
  \@glsxtr@noop@recordcounter
}
```

p@recordcounter

```
\newcommand*{\@glsxtr@noop@recordcounter}[1]{%
  \PackageError{glossaries-extra}{\string\GlsXtrRecordCounter\space
   requires record=only or record=alsoindex package option}{}%
}
```

p@recordcounter

```
\newcommand*{\@glsxtr@op@recordcounter}[1]{%
  \eappto\@glsxtr@counterrecordhook{\noexpand\@glsxtr@docounterrecord{#1}}%
}
```

lsxtr@recordsee Deal with \glssee in record mode. (This doesn't increment the associated counter.)

```
\newcommand*{\@glsxtr@recordsee}[2]{%
 \@@glsxtrwrglossmark
 \def\@gls@xref{#2}%
 \@onelevel@sanitize\@gls@xref
 \protected@write\@auxout{}{\string\glsxtr@recordsee{#1}{\@gls@xref}}%
}
```

srtglossaryunit

```
\newcommand{\printunsrtglossaryunit}{%
  \print@noop@unsrtglossaryunit
}
```

tr@setup@record Initialise.

```
\newcommand*{\glsxtr@setup@record}{\let\@@do@wrglossary\glsxtr@@do@wrglossary}
```

aveentrycounter Only store the entry counter information if the indexing is on.

```
\newcommand*{\glsxtr@indexonly@saveentrycounter}{%
 \ifKV@glslink@noindex
 \else
   \glsxtr@saveentrycounter
 \fi
}
```

addloclistfield

```
\newcommand*{\glsxtr@addloclistfield}{%
 \key@ifundefined{glossentry}{loclist}%
 {%
   \define@key{glossentry}{loclist}{\def\@glo@loclist{##1}}%
   \appto\@gls@keymap{,{loclist}{loclist}}%
   \appto\@newglossaryentryprehook{\def\@glo@loclist{}}%
   \appto\@newglossaryentryposthook{%
     \gls@assign@field{}{\@glo@label}{loclist}{\@glo@loclist}%
   }%
   \glssetnoexpandfield{loclist}%
 }%
 {}%
```

The loclist field is just a comma-separated list. The location field is the formatted list.

```
\key@ifundefined{glossentry}{location}%
{%
  \define@key{glossentry}{location}{\def\@glo@location{##1}}%
  \appto\@gls@keymap{,{location}{location}}%
  \appto\@newglossaryentryprehook{\def\@glo@location{}}%
  \appto\@newglossaryentryposthook{%
    \gls@assign@field{}{\@glo@label}{location}{\@glo@location}%
  }%
  \glssetnoexpandfield{location}%
}%
{}%
```

Add a key to store the group heading.

```
\key@ifundefined{glossentry}{group}%
{%
  \define@key{glossentry}{group}{\def\@glo@group{##1}}%
  \appto\@gls@keymap{,{group}{group}}%
  \appto\@newglossaryentryprehook{\def\@glo@group{}}%
  \appto\@newglossaryentryposthook{%
    \gls@assign@field{}{\@glo@label}{group}{\@glo@group}%
  }%
  \glssetnoexpandfield{group}%
}%
{}%
}
```

`@record@setting` Keep track of the record package option.

```
\newcommand*{\@glsxtr@record@setting}{off}
```

`tting@alsoindex`

```
\newcommand*{\@glsxtr@record@setting@alsoindex}{alsoindex}
```

`rd@setting@only`

```
\newcommand*{\@glsxtr@record@setting@only}{only}
```

`setting@nameref`

```
\newcommand*{\@glsxtr@record@setting@nameref}{nameref}
```

`@if@record@only`

```
\newcommand*{\@glsxtr@if@record@only}[2]{%
 \ifx\@glsxtr@record@setting\@glsxtr@record@setting@only
  #1%
 \else
  \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
    #1%
  \else
    #2%
  \fi
```

```
 \fi
}
```

ord@setting@off

```
\newcommand*{\@glsxtr@record@setting@off}{off}
```

cord@only@setup Initialisation code for record=only and record=nameref

```
\newcommand*{\@glsxtr@record@only@setup}{%
 \def\glsxtr@setup@record{%
   \@glsxtr@autoseeindexfalse
   \let\@do@seeglossary\@glsxtr@recordsee
   \let\@glsxtr@record\@@glsxtr@record
   \let\@@do@wrglossary\@glsxtr@do@record@wrglossary
   \let\@gls@saveentrycounter\relax
   \let\glsxtrundefaction\@glsxtr@warn@undefaction
   \let\glsxtr@warnonexistsordo\@glsxtr@warn@onexistsordo
   \glsxtr@addloclistfield
   \renewcommand*{\@glsxtr@autoindexcrossrefs}{}%
   \let\@@glsxtr@recordcounter\@glsxtr@op@recordcounter
   \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
```

Switch off the index suppression for aliased entries. (bib2gls will deal with them.)

```
   \def\glsxtrsetaliasnoindex{}%
```

\@gls@setupsort@none was only introduced to glossaries v4.30, so it may not be available. If it's defined, use it to remove the unnecessary overhead of escaping and sanitizing the sort value.

```
   \ifdef\@gls@setupsort@none{\@gls@setupsort@none}{}%
```

Warn about using \printglossary:

```
   \def\glsxtrNoGlossaryWarning{\@glsxtr@record@noglossarywarning}%
```

Load glossaries-extra-bib2gls:

```
   \RequirePackage{glossaries-extra-bib2gls}%
 }%
}
```

record Now define the record package option.

```
\define@choicekey{glossaries-extra.sty}{record}
 [\@glsxtr@record@setting\glsxtr@record@nr]%
 {off,only,alsoindex,nameref}%
 [only]%
 {%
   \ifcase\glsxtr@record@nr\relax
```

Don't record.

```
     \def\glsxtr@setup@record{%
       \renewcommand*{\@do@seeglossary}{\@glsxtr@doseeglossary}%
       \renewcommand*{\@glsxtr@record}[3]{}%
       \let\@@do@wrglossary\glsxtr@@do@wrglossary
       \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
```

```
      \let\glsxtrundefaction\@glsxtr@err@undefaction
      \let\glsxtr@warnonexistsordo\@gobble
      \let\@@glsxtr@recordcounter\@glsxtr@noop@recordcounter
      \def\printunsrtglossaryunit{\print@noop@unsrtglossaryunit}%
      \undef\glsxtrsetaliasnoindex
    }%
  \or
```

Only record (don't index).

```
    \@glsxtr@record@only@setup
  \or
```

Record and index. This option doesn't load glossaries-extra-bib2gls as the sorting is performed by `xindy` or `makeindex`.

```
    \def\glsxtr@setup@record{%
      \renewcommand*{\@do@seeglossary}{\@glsxtr@dosee@alsoindex@glossary}%
      \let\@glsxtr@record\@@glsxtr@record
      \let\@@do@wrglossary\glsxtr@do@alsoindex@wrglossary
      \let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
      \let\glsxtrundefaction\@glsxtr@warn@undefaction
      \let\glsxtr@warnonexistsordo\@glsxtr@warn@onexistsordo
      \glsxtr@addloclistfield
      \let\@@glsxtr@recordcounter\@glsxtr@op@recordcounter
      \def\printunsrtglossaryunit{\print@op@unsrtglossaryunit}%
      \undef\glsxtrsetaliasnoindex
    }%
  \or
```

Only record (don't index) but also include nameref information.

```
    \@glsxtr@record@only@setup
    \ifundef\hyperlink
    {\GlossariesExtraWarning{You have requested record=nameref but
     the document doesn't support hyperlinks}}%
    {}%
  \fi
}
```

Version 1.06 changes the docdef option to a choice rather than boolean setting. The available values are: false, true or restricted. The restricted option permits document definitions as long as they occur before the first glossary is displayed.

`\glsxtr@docdefval` The docdef value is stored as an integer: 0 (false), 1 (true) and 2 (restricted).

```
\newcommand*{\@glsxtr@docdefval}{0}
```

Need to provide conditional commands that are backward compatible:

`\if@glsxtrdocdef`

```
\newcommand*{\if@glsxtrdocdef}{\ifnum\@glsxtr@docdefval>0 }
```

`\glsxtrdocdeftrue`

```
\newcommand*{\@glsxtrdocdeftrue}{\def\@glsxtr@docdefval{1}}
```

sxtrdocdeffalse

```
\newcommand*{\@glsxtrdocdeffalse}{\def\@glsxtr@docdefval{0}}
```

docdef By default don't allow entries to be defined in the document to encourage the user to define them in the preamble, but if the user is really determined to define them in the document allow them to request this.

```
\define@choicekey{glossaries-extra.sty}{docdef}
 [\@glsxtr@docdefsetting\@glsxtr@docdefval]%
 {false,true,restricted,atom}[true]%
{%
  \ifnum\@glsxtr@docdefval>1\relax
    \renewcommand*{\@glsdoifexistsorwarn}{\glsdoifexists}%
  \else
    \renewcommand*{\@glsdoifexistsorwarn}{\glsdoifexistsorwarn}%
  \fi
}
```

ocdefrestricted

```
\newcommand*{\if@glsxtrdocdefrestricted}{\ifnum\@glsxtr@docdefval>1 }
```

oifexistsorwarn Need an error to notify user if an undefined entry is being referenced in the glossary for the docdef=restricted option. This is used by `\glossentryname` (but not by `\glossentrydesc` etc as one error per entry is sufficient).

```
\newcommand*{\@glsdoifexistsorwarn}{\glsdoifexistsorwarn}
```

indexcrossrefs Automatically index cross references at the end of the document

```
\define@boolkey{glossaries-extra.sty}[@glsxtr]{indexcrossrefs}[true]{%
 \if@glsxtrindexcrossrefs
 \else
  \renewcommand*{\@glsxtr@autoindexcrossrefs}{}%
 \fi
}
```

Switch off since this can increase the build time.

```
\@glsxtrindexcrossrefsfalse
```

But allow see key to switch it on automatically.

oindexcrossrefs

```
\newcommand*{\@glsxtr@autoindexcrossrefs}{\@glsxtrindexcrossrefstrue}
```

autoseeindex Provide a boolean option to allow the user to prevent the automatic indexing of the cross-referencing keys see, seealso and alias.

```
\define@boolkey{glossaries-extra.sty}[@glsxtr@]{autoseeindex}[true]{%
}
\@glsxtr@autoseeindextrue
```

equations Provide a boolean option to automatically switch to the equation counter when in a numbered maths environment.

```
\define@boolkey{glossaries-extra.sty}[@glsxtr@]{equations}[true]{%
}
\@glsxtr@equationsfalse
```

\glsxtr@float

```
\let\glsxtr@float\@float
```

glsxtr@dblfloat

```
\let\glsxtr@dblfloat\@dblfloat
```

floats Provide a boolean option to automatically switch to the the corresponding counter when in a float.

```
\define@boolkey{glossaries-extra.sty}[@glsxtr@]{floats}[true]{%
  \if@glsxtr@floats
   \renewcommand*{\@float}[1]{\renewcommand{\glscounter}{##1}\glsxtr@float{##1}}%
   \renewcommand*{\@dblfloat}[1]{\renewcommand{\glscounter}{##1}\glsxtr@dblfloat{##1}}%
  \else
   \let\@float\glsxtr@float
   \let\@dblfloat\glsxtr@dblfloat
  \fi
}
\@glsxtr@floatsfalse
```

iesExtraWarning Allow users to suppress warnings.

```
\newcommand*{\GlossariesExtraWarning}[1]{\PackageWarning{glossaries-extra}{#1}}
```

raWarningNoLine Allow users to suppress warnings.

```
\newcommand*{\GlossariesExtraWarningNoLine}[1]{%
 \PackageWarningNoLine{glossaries-extra}{#1}}

\@glsxtr@declareoption{nowarn}{%
  \let\GlossariesExtraWarning\@gobble
  \let\GlossariesExtraWarningNoLine\@gobble
  \glsxtr@dooption{nowarn}%
}
```

xtr@defpostpunc Redefines \glspostdescription. The postdot and nopostdot options will have to redefine this.

```
\newcommand*{\@glsxtr@defpostpunc}{}
```

postdot Shortcut for nopostdot=false

```
\@glsxtr@declareoption{postdot}{%
  \glsxtr@dooption{nopostdot=false}%
  \renewcommand*{\@glsxtr@defpostpunc}{%
    \renewcommand*{\glspostdescription}{%
      \ifglsnopostdot\else.\spacefactor\sfcode`\. \fi}%
```

```
  }%
}
```

nopostdot Needs to redefine \@glsxtr@defpostpunc

```
\define@choicekey{glossaries-extra.sty}{nopostdot}{true,false}[true]{%
  \glsxtr@dooption{nopostdot=#1}%
  \renewcommand*{\@glsxtr@defpostpunc}{%
    \renewcommand*{\glspostdescription}{%
      \ifglsnopostdot\else.\spacefactor\sfcode`\. \fi}%
  }%
}
```

postpunc Set the post-description punctuation. This also sets the \ifglsnopostdot conditional, which now indicates if the post-description punctuation has been suppressed.

```
\define@key{glossaries-extra.sty}{postpunc}{%
  \glsxtr@dooption{nopostdot=false}%
  \ifstrequal{#1}{dot}%
  {%
    \renewcommand*{\@glsxtr@defpostpunc}{%
      \renewcommand*{\glspostdescription}{.\spacefactor\sfcode`\. }%
    }%
  }%
  {%
    \ifstrequal{#1}{comma}%
    {%
      \renewcommand*{\@glsxtr@defpostpunc}{%
        \renewcommand*{\glspostdescription}{,}%
      }%
    }%
    {%
      \ifstrequal{#1}{none}%
      {%
        \glsxtr@dooption{nopostdot=true}%
        \renewcommand*{\@glsxtr@defpostpunc}{%
          \renewcommand*{\glspostdescription}{}%
        }%
      }%
      {%
        \renewcommand*{\@glsxtr@defpostpunc}{%
          \renewcommand*{\glspostdescription}{#1}%
        }%
      }%
    }%
  }%
}
```

glsxtrabbrvtype Glossary type for abbreviations.

```
\newcommand*{\glsxtrabbrvtype}{\glsdefaulttype}
```

bbreviationsdef Set by abbreviations option.

```
\newcommand*{\@glsxtr@abbreviationsdef}{}
```

bbreviationsdef

```
\newcommand*{\@glsxtr@doabbreviationsdef}{%
  \@ifpackageloaded{babel}%
  {\providecommand{\abbreviationsname}{\acronymname}}%
  {\providecommand{\abbreviationsname}{Abbreviations}}%
  \newglossary[glg-abr]{abbreviations}{gls-abr}{glo-abr}{\abbreviationsname}%
  \renewcommand*{\glsxtrabbrvtype}{abbreviations}%
  \newcommand*{\printabbreviations}[1][]{%
    \printglossary[type=\glsxtrabbrvtype,##1]%
  }%
  \disable@keys{glossaries-extra.sty}{abbreviations}%
```

If the acronym option hasn't been used, change `\acronymtype` to `\glsxtrabbrvtype`.

```
  \ifglsacronym
  \else
    \renewcommand*{\acronymtype}{\glsxtrabbrvtype}%
  \fi
}%
```

abbreviations If abbreviations, create a new glossary type for abbreviations.

```
\@glsxtr@declareoption{abbreviations}{%
  \let\@glsxtr@abbreviationsdef\@glsxtr@doabbreviationsdef
}
```

iationShortcuts Enable shortcut commands for the abbreviations. Unlike the analogous command provided by glossaries, this uses `\newcommand` instead of `\let` as a safety feature (except for `\newabbr` which is also provided with `\GlsXtrDefineAcShortcuts`).

```
\newcommand*{\GlsXtrDefineAbbreviationShortcuts}{%
  \newcommand*{\ab}{\cgls}%
  \newcommand*{\abp}{\cglspl}%
  \newcommand*{\as}{\glsxtrshort}%
  \newcommand*{\asp}{\glsxtrshortpl}%
  \newcommand*{\al}{\glsxtrlong}%
  \newcommand*{\alp}{\glsxtrlongpl}%
  \newcommand*{\af}{\glsxtrfull}%
  \newcommand*{\afp}{\glsxtrfullpl}%
  \newcommand*{\Ab}{\cGls}%
  \newcommand*{\Abp}{\cGlspl}%
  \newcommand*{\As}{\Glsxtrshort}%
  \newcommand*{\Asp}{\Glsxtrshortpl}%
  \newcommand*{\Al}{\Glsxtrlong}%
  \newcommand*{\Alp}{\Glsxtrlongpl}%
  \newcommand*{\Af}{\Glsxtrfull}%
  \newcommand*{\Afp}{\Glsxtrfullpl}%
  \newcommand*{\AB}{\cGLS}%
  \newcommand*{\ABP}{\cGLSpl}%
```

```
  \newcommand*{\AS}{\GLSxtrshort}%
  \newcommand*{\ASP}{\GLSxtrshortpl}%
  \newcommand*{\AL}{\GLSxtrlong}%
  \newcommand*{\ALP}{\GLSxtrlongpl}%
  \newcommand*{\AF}{\GLSxtrfull}%
  \newcommand*{\AFP}{\GLSxtrfullpl}%

  \providecommand*{\newabbr}{\newabbreviation}%
```

Disable this command after it's been used.

```
  \let\GlsXtrDefineAbbreviationShortcuts\relax
}
```

fineAcShortcuts Enable shortcut commands for the abbreviations, but uses the analogous commands provided by glossaries.

```
\newcommand*{\GlsXtrDefineAcShortcuts}{%
  \newcommand*{\ac}{\cgls}%
  \newcommand*{\acp}{\cglspl}%
  \newcommand*{\acs}{\glsxtrshort}%
  \newcommand*{\acsp}{\glsxtrshortpl}%
  \newcommand*{\acl}{\glsxtrlong}%
  \newcommand*{\aclp}{\glsxtrlongpl}%
  \newcommand*{\acf}{\glsxtrfull}%
  \newcommand*{\acfp}{\glsxtrfullpl}%
  \newcommand*{\Ac}{\cGls}%
  \newcommand*{\Acp}{\cGlspl}%
  \newcommand*{\Acs}{\Glsxtrshort}%
  \newcommand*{\Acsp}{\Glsxtrshortpl}%
  \newcommand*{\Acl}{\Glsxtrlong}%
  \newcommand*{\Aclp}{\Glsxtrlongpl}%
  \newcommand*{\Acf}{\Glsxtrfull}%
  \newcommand*{\Acfp}{\Glsxtrfullpl}%
  \newcommand*{\AC}{\cGLS}%
  \newcommand*{\ACP}{\cGLSpl}%
  \newcommand*{\ACS}{\GLSxtrshort}%
  \newcommand*{\ACSP}{\GLSxtrshortpl}%
  \newcommand*{\ACL}{\GLSxtrlong}%
  \newcommand*{\ACLP}{\GLSxtrlongpl}%
  \newcommand*{\ACF}{\GLSxtrfull}%
  \newcommand*{\ACFP}{\GLSxtrfullpl}%

  \providecommand*{\newabbr}{\newabbreviation}%
```

Disable this command after it's been used.

```
  \let\GlsXtrDefineAcShortcuts\relax
}
```

eOtherShortcuts Similarly provide shortcut versions for the commands provided by the symbols and numbers options.

```
\newcommand*{\GlsXtrDefineOtherShortcuts}{%
```

```
  \newcommand*{\newentry}{\newglossaryentry}%
  \ifdef\printsymbols
  {%
    \newcommand*{\newsym}{\glsxtrnewsymbol}%
  }{}%
  \ifdef\printnumbers
  {%
    \newcommand*{\newnum}{\glsxtrnewnumber}%
  }{}%
  \let\GlsXtrDefineOtherShortcuts\relax
}
```

Always use the long forms, not the shortcuts, where portability is an issue. (For example, when defining entries in a file that may be input by multiple documents.)

`@setupshortcuts` Command used to set the shortcuts option.

```
\newcommand*{\@glsxtr@setupshortcuts}{}
```

`tr@shortcutsval` Store the value of the shortcuts option. (Needed by bib2gls.)

```
\newcommand*{\@glsxtr@shortcutsval}{\ifglsacrshortcuts acro\else none\fi}%
```

`shortcuts` Provide shortcuts option. Unlike the glossaries version, this is a choice rather than a boolean key but it also provides shortcuts=true and shortcuts=false, which are equivalent to shortcuts=all and shortcuts=none. Multiple use of this option in the *same* option list will override each other. New to v1.17: shortcuts=ac which implements `\GlsXtrDefineAcShortcuts` (not included in shortcuts=all as it conflicts with other shortcuts).

```
\define@choicekey{glossaries-extra.sty}{shortcuts}%
 [\@glsxtr@shortcutsval\@glsxtr@shortcutsnr]%
 {acronyms,acro,abbreviations,abbr,other,all,true,ac,none,false}[true]{%
   \ifcase\@glsxtr@shortcutsnr\relax % acronyms
     \renewcommand*{\@glsxtr@setupshortcuts}{%
       \glsacrshortcutstrue
       \DefineAcronymSynonyms
     }%
   \or % acro
     \renewcommand*{\@glsxtr@setupshortcuts}{%
       \glsacrshortcutstrue
       \DefineAcronymSynonyms
     }%
   \or % abbreviations
     \renewcommand*{\@glsxtr@setupshortcuts}{%
       \GlsXtrDefineAbbreviationShortcuts
     }%
   \or % abbr
     \renewcommand*{\@glsxtr@setupshortcuts}{%
       \GlsXtrDefineAbbreviationShortcuts
     }%
   \or % other
```

```
    \renewcommand*{\@glsxtr@setupshortcuts}{%
      \GlsXtrDefineOtherShortcuts
    }%
  \or % all
    \renewcommand*{\@glsxtr@setupshortcuts}{%
      \glsacrshortcutstrue
      \GlsXtrDefineAcShortcuts
      \GlsXtrDefineAbbreviationShortcuts
      \GlsXtrDefineOtherShortcuts
    }%
  \or % true
    \renewcommand*{\@glsxtr@setupshortcuts}{%
      \glsacrshortcutstrue
      \GlsXtrDefineAcShortcuts
      \GlsXtrDefineAbbreviationShortcuts
      \GlsXtrDefineOtherShortcuts
    }%
  \or % ac
    \renewcommand*{\@glsxtr@setupshortcuts}{%
      \glsacrshortcutstrue
      \GlsXtrDefineAcShortcuts
    }%
```

Leave none and false as last option.

```
  \else % none, false
    \renewcommand*{\@glsxtr@setupshortcuts}{}%
  \fi
}
```

`\@glsxtr@doaccsupp`

```
\newcommand*{\@glsxtr@doaccsupp}{}
```

`accsupp` If accsupp, load glossaries-accsupp package.

```
\@glsxtr@declareoption{accsupp}{%
 \renewcommand*{\@glsxtr@doaccsupp}{\RequirePackage{glossaries-accsupp}}}
```

`GlossaryWarning` Warning text displayed in document if the external glossary file given by the argument is missing.

```
\newcommand{\glsxtrNoGlossaryWarning}[1]{%
  \GlossariesExtraWarning{Glossary `#1' is missing}%
  \@glsxtr@defaultnoglossarywarning{#1}%
}
```

`omissingglstext` If true, suppress the text and warning produced if the external glossary file is missing.

```
\define@choicekey{glossaries-extra.sty}{nomissingglstext}
 [\@glsxtr@nomissingglstextval\@glsxtr@nomissingglstextnr]%
```

```
{true,false}[true]{%
  \ifcase\@glsxtr@nomissingglstextnr\relax % true
    \renewcommand{\glsxtrNoGlossaryWarning}[1]{\null}%
  \else % false
    \renewcommand{\glsxtrNoGlossaryWarning}[1]{%
      \@glsxtr@defaultnoglossarywarning{#1}%
    }%
  \fi
}
```

Provide option to load glossaries-extra-stylemods (Deferred to the end.)

xtr@redefstyles

```
\newcommand*{\@glsxtr@redefstyles}{}
```

stylemods

```
\define@key{glossaries-extra.sty}{stylemods}[default]{%
  \ifstrequal{#1}{default}%
  {%
    \renewcommand*{\@glsxtr@redefstyles}{%
      \RequirePackage{glossaries-extra-stylemods}}%
  }%
  {%
    \ifstrequal{#1}{all}%
    {%
      \renewcommand*{\@glsxtr@redefstyles}{%
        \PassOptionsToPackage{all}{glossaries-extra-stylemods}%
        \RequirePackage{glossaries-extra-stylemods}%
      }%
    }%
    {%
      \renewcommand*{\@glsxtr@redefstyles}{}%
      \@for\@glsxtr@tmp:=#1\do{%
        \IfFileExists{glossary-\@glsxtr@tmp.sty}%
        {%
          \eappto\@glsxtr@redefstyles{%
            \noexpand\RequirePackage{glossary-\@glsxtr@tmp}}%
        }%
        {%
           \PackageError{glossaries-extra}%
             {Glossaries style package `glossary-\@glsxtr@tmp.sty'
              doesn't exist (did you mean to use the `style' key?)}%
             {The list of values (#1) in the `stylemods' key should
              match the glossary-xxx.sty files provided with
              glossaries.sty}%
        }%
      }%
      \appto\@glsxtr@redefstyles{\RequirePackage{glossaries-extra-stylemods}}%
  }
```

```
  }%
}
```

glsxtr@do@style

```
\newcommand*{\@glsxtr@do@style}{}
```

style Since the stylemods option can automatically load extra style packages, deal with the style option after those packages have been loaded.

```
\define@key{glossaries-extra.sty}{style}{%
```

Defer actual style change:

```
 \renewcommand*{\@glsxtr@do@style}{%
```

Set this as the default style:

```
   \setkeys{glossaries.sty}{style={#1}}%
```

Set this style:

```
   \setglossarystyle{#1}%
 }%
}
```

c@wrglossaryctr Increments the associated counter if enabled. Does nothing by default. The optional argument is the entry label in case it's required, but the wrglossary counter is globally used by all entries.

```
\newcommand*{\glsxtr@inc@wrglossaryctr}[1]{}
```

ocationHyperlink `\glsxtrinternallocationhyperlink{⟨counter⟩}{⟨prefix⟩}{⟨location⟩}`

The first two arguments are always control sequences.

```
\newcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
  \glsxtrhyperlink{#1#2#3}{#3}%
}
```

cationhyperlink

```
\newcommand*{\@glsxtr@wrglossary@locationhyperlink}[3]{%
  \pageref{wrglossary.#3}%
}
```

indexcounter Define the wrglossary counter that's incremented every time an entry is indexed, except for cross-references. This is designed for use with bib2gls v1.4+. It can work with the other indexing methods but it will interfere with the number list collation. This option automatically implements counter=wrglossary.

Since glossaries automatically loads amsmath, there may be a problem if the indexing occurs in the equation environment, because only one \label is allowed in each instance of that environment. It's best to change the counter when in maths mode.

```
\@glsxtr@declareoption{indexcounter}{%
```

```
  \glsxtr@dooption{counter=wrglossary}%
  \ifundef\c@wrglossary
  {%
    \newcounter{wrglossary}%
    \renewcommand{\thewrglossary}{\arabic{wrglossary}}%
  }%
  {}%
  \renewcommand*{\glsxtr@inc@wrglossaryctr}[1]{%
```

Only increment if the current counter is wrglossary.

```
    \ifdefstring\@gls@counter{wrglossary}%
    {%
      \refstepcounter{wrglossary}%
      \label{wrglossary.\thewrglossary}%
    }%
    {}%
  }%
  \renewcommand*{\GlsXtrInternalLocationHyperlink}[3]{%
    \ifdefstring\glsentrycounter{wrglossary}%
    {%
      \@glsxtr@wrglossary@locationhyperlink{##1}{##2}{##3}%
    }%
    {\glsxtrhyperlink{##1##2##3}{##3}}%
  }%
}
```

`sxtrwrglossmark` Marks the place where indexing occurs. Does nothing by default.

```
\newcommand*{\@glsxtrwrglossmark}{}
```

`sxtrwrglossmark` Since `\glsadd` can be used in the preamble, this action needs to be disabled until the start of the document.

```
\newcommand*{\@@glsxtrwrglossmark}{}
\AtBeginDocument{\renewcommand*{\@@glsxtrwrglossmark}{\@glsxtrwrglossmark}}
```

`sxtrwrglossmark` Does nothing by default.

```
\newcommand*{\glsxtrwrglossmark}{\ensuremath{\cdot}}
```

`debug` Provide extra debug options.

```
\define@choicekey{glossaries-extra.sty}{debug}
 [\@glsxtr@debugval\@glsxtr@debugnr]%
 {true,false,showtargets,showwrgloss,all}[true]{%
   \ifcase\@glsxtr@debugnr\relax % true
    \glsxtr@dooption{debug=true}%
    \renewcommand*{\@glsxtrwrglossmark}{}%
   \or % false
    \glsxtr@dooption{debug=false}%
    \renewcommand*{\@glsxtrwrglossmark}{}%
   \or % showtargets
    \glsxtr@dooption{debug=showtargets}%
```

```
  \or % showwrgloss
   \glsxtr@dooption{debug=true}%
   \renewcommand*{\@glsxtrwrglossmark}{\glsxtrwrglossmark}%
  \or % all
   \glsxtr@dooption{debug=showtargets}%
   \renewcommand*{\@glsxtrwrglossmark}{\glsxtrwrglossmark}%
  \fi
}
```

Pass all other options to glossaries.

```
\DeclareOptionX*{%
 \expandafter\glsxtr@dooption\expandafter{\CurrentOption}}
```

Process options.

```
\ProcessOptionsX
```

Load glossaries if not already loaded.

```
\RequirePackage{glossaries}
```

Load the glossaries-accsupp package if required.

```
\@glsxtr@doaccsupp
```

Redefine `\glspostdescription` if required.

```
\@glsxtr@defpostpunc
```

`\glsshowtarget` This command was introduced to glossaries v4.32 so it may not be defined. Therefore it's defined here using `\def`.

```
\def\glsshowtarget#1{%
 \glsxtrtitleorpdforheading
 {%
   \ifmmode
     \texttt{\small [#1]}%
   \else
     \ifinner
       \texttt{\small [#1]}%
     \else
       \marginpar{\texttt{\small #1}}%
     \fi
   \fi
 }%
 {[#1]}%
 {\texttt{\small [#1]}}%
}
```

`g@doseeglossary` Save original definition of `\@do@seeglossary`

```
\let\@glsxtr@org@doseeglossary\@do@seeglossary
```

`r@doseeglossary` This doesn't increment the associated counter.

```
\newcommand*{\@glsxtr@doseeglossary}[2]{%
  \glsdoifexists{#1}%
```

```
  {%
    \@@glsxtrwrglossmark
    \@glsxtr@org@doseeglossary{#1}{#2}%
  }%
}
```

oindex@glossary

```
\newcommand*{\@glsxtr@dosee@alsoindex@glossary}[2]{%
  \@glsxtr@recordsee{#1}{#2}%
  \@glsxtr@doseeglossary{#1}{#2}%
}
```

@org@gloautosee Save and restore original definition of \@glo@autosee. (That command may not be defined as it was only introduced to glossaries v4.30, in which case the synonym won't be defined either.)

```
\let\@glsxtr@org@gloautosee\@glo@autosee
```

Check if user tried autoseeindex=false when it can't be supported.

```
\if@glsxtr@autoseeindex
\else
  \ifdef\@glsxtr@org@gloautosee
  {}%
  {\PackageError{glossaries-extra}{`autoseeindex=false' package
   option requires at least v4.30 of glossaries.sty}%
   {You need to update the glossaries.sty package}%
  }
\fi
```

\@glo@autosee If \@glo@autosee has been defined (glossaries v4.30 onwards), redefine it to test the autoseeindex option.

```
\ifdef\@glo@autosee
{%
  \renewcommand*{\@glo@autosee}{%
    \if@glsxtr@autoseeindex\@glsxtr@org@gloautosee\fi}%
}%
{}
```

checkseeallowed Don't prohibit the use of the see key before the indexing files have been opened if the automatic see indexing has been disabled, since it's no longer an issue.

```
\renewcommand*{\gls@checkseeallowed}{%
 \if@glsxtr@autoseeindex\@gls@see@noindex\fi
}
```

Define abbreviations glossaries if required.

```
\@glsxtr@abbreviationsdef
\let\@glsxtr@abbreviationsdef\relax
```

Setup shortcuts if required.

```
\@glsxtr@setupshortcuts
```

Redefine \@glsxtr@redef@forglsentries if required.

```
\@glsxtr@redef@forglsentries
```

ariesextrasetup Allow user to set options after the package has been loaded. First modify \glsxtr@dooption so that it now uses \setupglossaries:

```
\renewcommand{\glsxtr@dooption}[1]{\setupglossaries{#1}}%
```

Now define the user command:

```
\newcommand*{\glossariesextrasetup}[1]{%
  \let\glsxtr@setup@record\relax
  \let\@glsxtr@setupshortcuts\relax
  \let\@glsxtr@redef@forglsentries\relax
  \setkeys{glossaries-extra.sty}{#1}%
  \@glsxtr@abbreviationsdef
  \let\@glsxtr@abbreviationsdef\relax
  \@glsxtr@setupshortcuts
  \glsxtr@setup@record
  \@glsxtr@redef@forglsentries
}
```

@@do@wrglossary Save original definition of \@@do@wrglossary.

```
\let\glsxtr@org@@do@wrglossary\@@do@wrglossary
```

@@do@wrglossary The new version adds code that can show a marker for debugging and increments the associated counter if enabled.

```
\newcommand*{\glsxtr@@do@wrglossary}[1]{%
 \@@glsxtrwrglossmark
 \glsxtr@inc@wrglossaryctr{#1}%
 \glsxtr@org@@do@wrglossary{#1}%
}
```

aveentrycounter Save original definition of \@gls@saveentrycounter.

```
\let\glsxtr@saveentrycounter\@gls@saveentrycounter
```

aveentrycounter Change \@gls@saveentrycounter so that it only stores the entry counter information if the indexing is on.

```
\let\@gls@saveentrycounter\glsxtr@indexonly@saveentrycounter
```

etcounterprefix This command is provided by the base glossaries package, but is redefined here. The standard indexing methods don't directly store the hypertarget but instead need to split it into the counter, prefix and location parts, which can be reconstituted in the location list. Unfortunately, not all targets are in this form, so the links fail. With record=nameref, the complete target name can be saved, so this modification adjusts the warning.

```
\renewcommand*\@gls@getcounterprefix[2]{%
  \protected@edef\@gls@thisloc{#1}\protected@edef\@gls@thisHloc{#2}%
  \ifx\@gls@thisloc\@gls@thisHloc
    \def\@glo@counterprefix{}%
  \else
```

```
    \def\@gls@get@counterprefix##1.#1##2\end@getprefix{%
      \def\@glo@tmp{##2}%
      \ifx\@glo@tmp\@empty
        \def\@glo@counterprefix{}%
      \else
        \def\@glo@counterprefix{##1}%
      \fi
    }%
    \@gls@get@counterprefix#2.#1\end@getprefix
```

Warn if no prefix can be formed, unless record=nameref.

```
    \ifx\@glo@counterprefix\@empty
      \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
      \else
        \GlossariesExtraWarning{Hyper target `#2' can't be formed by
         prefixing^^Jlocation `#1'. You need to modify the
         definition of \string\theH\@gls@counter^^Jotherwise you
         will get the warning: "`name{\@gls@counter.#1}' has been^^J
         referenced but does not exist"%
         \ifx\@glsxtr@record@setting\@glsxtr@record@setting@only
         . You may want to consider using record=nameref instead%
         \fi}%
      \fi
    \fi
  \fi
}
```

Provide script dialect hook (does nothing unless redefined by glossaries-extra-bib2gls).

sxtrdialecthook

```
\newcommand*{\@glsxtrdialecthook}{}
```

Set up record option if required.

```
\glsxtr@setup@record
```

Disable preamble-only options and switch on the undefined tag at the start of the document.

```
\AtBeginDocument{%
  \disable@keys{glossaries-extra.sty}{abbreviations,docdef,record}%
  \def\@glsxtrundeftag{\glsxtrundeftag}%
}
```

1.2 Extra Utilities

nusedOrUndefined

\GlsXtrIfUnusedOrUndefined{⟨*label*⟩}{⟨*true*⟩}{⟨*false*⟩}

Does ⟨*true*⟩ if the entry given by ⟨*label*⟩ is either undefined or hasn't been used (or has had the first use flag reset).

```
\newcommand*{\GlsXtrIfUnusedOrUndefined}[3]{%
  \ifglsentryexists{#1}%
  {\ifbool{glo@\glsdetoklabel{#1}@flag}{#3}{#2}}%
  {#2}%
}
```

rifemptyglossary

\glsxtrifemptyglossary{⟨*type*⟩}{⟨*true*⟩}{⟨*false*⟩}

Provide command to determine if any entries have been added to the glossary (where the glossary label is provided in the first argument). The entries are stored in the comma-separated list \glolist@⟨*type*⟩. If this hasn't been defined, the glosary doesn't exist. If it has been defined and is simply a comma, the glossary exists and is empty. (It's initialised to a comma.)

```
\newcommand{\glsxtrifemptyglossary}[3]{%
  \ifcsdef{glolist@#1}%
  {%
    \ifcsstring{glolist@#1}{,}{#2}{#3}%
  }%
  {%
    \glsxtrundefaction{Glossary type `#1' doesn't exist}{}%
    #2%
  }%
}
```

xtrifkeydefined

Tests if the key given in the first argument has been defined.

```
\newcommand*{\glsxtrifkeydefined}[3]{%
  \key@ifundefined{glossentry}{#1}{#3}{#2}%
}
```

ovidestoragekey

Like \glsaddstoragekey but does nothing if the key has already been defined.

```
\newcommand*{\glsxtrprovidestoragekey}{%
  \@ifstar\@sglsxtr@provide@storagekey\@glsxtr@provide@storagekey
}
```

vide@storagekey

Unstarred version.

```
\newcommand*{\@glsxtr@provide@storagekey}[3]{%
  \key@ifundefined{glossentry}{#1}%
  {%
    \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
    \appto\@gls@keymap{,{#1}{#1}}%
    \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
    \appto\@newglossaryentryposthook{%
      \letcs{\@glo@tmp}{@glo@#1}%
```

```
      \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
    }%
```

Allow the user to omit the user level command if they only intended fetching the value with `\glsxtrusefield`

```
    \ifblank{#3}
    {}%
    {%
       \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
    }%
  }%
  {%
```

Provide the no-link command if not already defined.

```
    \ifblank{#3}
    {}%
    {%
      \providecommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
    }%
  }%
}
```

`vide@storagekey` Starred version.

```
\newcommand*{\s@glsxtr@provide@storagekey}[1]{%
  \key@ifundefined{glossentry}{#1}%
  {%
    \expandafter\newcommand\expandafter*\expandafter
     {\csname gls@assign@#1@field\endcsname}[2]{%
       \@@gls@expand@field{##1}{#1}{##2}%
     }%
  }%
  {}%
  \@glsxtr@provide@addstoragekey{#1}%
}
```

The name of a text-block control sequence can be stored in a field (given by `\GlsXtrFmtField`). This command can then be used with `\glsxtrfmt[`⟨*options*⟩`]{`⟨*label*⟩`}{`⟨*text*⟩`}` which effectively does `\glslink[`⟨*options*⟩`]{`⟨*label*⟩`}{`⟨*cs*⟩`}{`⟨*text*⟩`}}` If the field hasn't been set for that entry just ⟨*text*⟩ is done.

`\GlsXtrFmtField`

```
\newcommand{\GlsXtrFmtField}{useri}
```

`tDefaultOptions`

```
\newcommand{\GlsXtrFmtDefaultOptions}{noindex}
```

`\glsxtrfmt` The post-link hook isn't done. This now has a starred form that checks for a final optional argument.

```
\newrobustcmd*{\glsxtrfmt}{\@ifstar\s@glsxtrfmt\@glsxtrfmt}
```

\@glsxtrfmt Unstarred form.

```
\newcommand*{\@glsxtrfmt}[3][]{\@@glsxtrfmt{#1}{#2}{#3}{}}
```

\s@glsxtrfmt Starred form.

```
\newcommand*{\s@glsxtrfmt}[3][]{%
 \new@ifnextchar[{\s@@glsxtrfmt{#1}{#2}{#3}}%
  {\@@glsxtrfmt{#1}{#2}{#3}{}}%
}
```

\s@@glsxtrfmt Pick up final optional argument.

```
\def\s@@glsxtrfmt#1#2#3[#4]{\@@glsxtrfmt{#1}{#2}{#3}{#4}}
```

\@@glsxtrfmt Actual inner working.

```
\newcommand*{\@@glsxtrfmt}[4]{%
```

Since there's no post-link hook to worry about, grouping can be added to provide some protection against nesting (but in general nested link text should be avoided).

```
 \begingroup
   \def\glslabel{#2}%
   \glsdoifexistsordo{#2}%
   {%
     \ifglshasfield{\GlsXtrFmtField}{#2}%
     {%
       \let\do@gls@link@checkfirsthyper\relax
       \expandafter\@gls@link\expandafter[\GlsXtrFmtDefaultOptions,#1]{#2}%
         {\glsxtrfmtdisplay{\glscurrentfieldvalue}{#3}{#4}}%
     }%
     {\glsxtrfmtdisplay{@firstofone}{#3}{#4}}%
   }%
   {%
```

Has the default `noindex` been counteracted? If so, this needs `\glsadd` in case `bib2gls` needs to pick up the record.

```
     \begingroup
       \@gls@setdefault@glslink@opts
       \setkeys{glslink}{\GlsXtrFmtDefaultOptions,#1}%
       \ifKV@glslink@noindex\else\glsadd{#2}\fi
     \endgroup
     \glsxtrfmtdisplay{@firstofone}{#3}{#4}%
   }%
 \endgroup
}
```

lsxtrfmtdisplay The command used internally by `\glsxtrfmt` to do the actual formatting. The first argument is the control sequence name, the second is the control sequence's argument, the third is the inserted material (if starred form used).

```
\newcommand{\glsxtrfmtdisplay}[3]{\csuse{#1}{#2}#3}
```

\glsxtrentryfmt No link or indexing.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsxtrentryfmt}[2]{%
    \texorpdfstring{\@glsxtrentryfmt{#1}{#2}}{#2}%
  }
}
{
  \newcommand*{\glsxtrentryfmt}{\@glsxtrentryfmt}
}
```

\@glsxtrentryfmt

```
\newrobustcmd*{\@glsxtrentryfmt}[2]{%
 \glsdoifexistsordo{#1}%
 {%
   \ifglshasfield{\GlsXtrFmtField}{#1}%
   {%
     \csuse{\glscurrentfieldvalue}{#2}%
   }%
   {#2}%
 }%
 {#2}%
}
```

\xtrfieldlistadd If a field stores an etoolbox internal list (e.g. loclist) then this macro provides a convenient way of adding to the list via etoolbox's \listcsadd. The first argument is the entry's label, the second is the field label and the third is the element to add to the list.

```
\newcommand*{\glsxtrfieldlistadd}[3]{%
  \listcsadd{glo@\glsdetoklabel{#1}@#2}{#3}%
}
```

\trfieldlistgadd Similarly but uses \listcsgadd.

```
\newcommand*{\glsxtrfieldlistgadd}[3]{%
  \listcsgadd{glo@\glsdetoklabel{#1}@#2}{#3}%
}
```

\trfieldlisteadd Similarly but uses \listcseadd.

```
\newcommand*{\glsxtrfieldlisteadd}[3]{%
  \listcseadd{glo@\glsdetoklabel{#1}@#2}{#3}%
}
```

\trfieldlistxadd Similarly but uses \listcsxadd.

```
\newcommand*{\glsxtrfieldlistxadd}[3]{%
  \listcsxadd{glo@\glsdetoklabel{#1}@#2}{#3}%
}
```

Now provide commands to iterate over these lists.

fielddolistloop

```
\newcommand*{\glsxtrfielddolistloop}[2]{%
  \dolistcsloop{glo@\glsdetoklabel{#1}@#2}%
}
```

ieldforlistloop

```
\newcommand*{\glsxtrfieldforlistloop}[3]{%
  \forlistcsloop{#3}{glo@\glsdetoklabel{#1}@#2}%
}
```

List element tests:

trfieldifinlist First argument label, second argument field, third argument item, fourth true part and fifth false part.

```
\newcommand*{\glsxtrfieldifinlist}[5]{%
  \ifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
}
```

rfieldxifinlist Expands item.

```
\newcommand*{\glsxtrfieldxifinlist}[5]{%
  \xifinlistcs{#3}{glo@\glsdetoklabel{#1}@#2}{#4}{#5}%
}
```

lsxtrforcsvfield `\glsxtrforcsvfield{⟨label⟩}{⟨field⟩}{⟨cs handler⟩}`

```
\newcommand*{\glsxtrforcsvfield}[3]{%
 \@glsxtrifhasfield{#2}{#1}%
 {%
   \let\glsxtrendfor\@endfortrue
   \@for\@glsxtr@label:=\glscurrentfieldvalue\do
   {\expandafter#3\expandafter{\@glsxtr@label}}}%
 {}%
}
```

lsxtrifhasfield A simpler alternative to `\ifglshasfield` that doesn't complain if the entry or the field doesn't exist. (No mapping is used.) Grouping is added to the unstarred version allow for nested use.

```
\newrobustcmd{\glsxtrifhasfield}{%
  \@ifstar{\s@glsxtrifhasfield}{\@glsxtrifhasfield}%
}
```

lsxtrifhasfield Unstarred version adds grouping.

```
\newcommand{\@glsxtrifhasfield}[4]{%
  {\s@glsxtrifhasfield{#1}{#2}{#3}{#4}}%
}
```

lsxtrifhasfield Starred version omits grouping.

```
\newcommand{\s@glsxtrifhasfield}[4]{%
  \letcs{\glscurrentfieldvalue}{glo@\glsdetoklabel{#2}@#1}%
  \ifundef\glscurrentfieldvalue
  {#4}%
  {%
   \ifdefempty\glscurrentfieldvalue{#4}{#3}%
  }%
}
```

rIfFieldNonZero Designed for numeric fields.

```
\newcommand{\GlsXtrIfFieldNonZero}[4]{%
  \GlsXtrIfFieldCmpNum{#1}{#2}{=}{0}{#4}{#3}%
}
```

sXtrIfFieldEqNum `\GlsXtrIfFieldEqNum{⟨field⟩}{⟨label⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩}`

Designed for numeric fields.

```
\newcommand{\GlsXtrIfFieldEqNum}[5]{%
  \GlsXtrIfFieldCmpNum{#1}{#2}{=}{#3}{#4}{#5}%
}
```

XtrIfFieldCmpNum `\GlsXtrIfFieldCmpNum{⟨field⟩}{⟨label⟩}{⟨comparison⟩}{⟨value⟩}{⟨true⟩}{⟨false⟩}`

Designed for numeric fields.

```
\newcommand{\GlsXtrIfFieldCmpNum}[6]{%
  {%
    \letcs{\glscurrentfieldvalue}{glo@\glsdetoklabel{#2}@#1}%
    \ifundef\glscurrentfieldvalue
    {\def\glscurrentfieldvalue{0}}%
    {%
     \ifdefempty\glscurrentfieldvalue
     {\def\glscurrentfieldvalue{0}}%
     {}%
    }%
    \ifnum\glscurrentfieldvalue#3#4\relax #5\else #6\fi
  }%
}
```

sXtrIfFieldUndef `\GlsXtrIfFieldUndef{⟨field⟩}{⟨label⟩}{⟨true⟩}{⟨false⟩}`

Just uses \ifcsundef.

```
\newcommand{\GlsXtrIfFieldUndef}[2]{%
 \ifcsundef{glo@\glsdetoklabel{#2}@#1}%
}
```

\glsxtrusefield Provide a user-level alternative to \@gls@entry@field. The first argument is the entry label. The second argument is the field label.

```
\newcommand*{\glsxtrusefield}[2]{%
  \@gls@entry@field{#1}{#2}%
}
```

\Glsxtrusefield Provide a user-level alternative to \@Gls@entry@field.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsxtrusefield}[2]{%
    \texorpdfstring
     {\@Gls@entry@field{#1}{#2}}
     {\@gls@entry@field{#1}{#2}}%
  }
}
{
  \newcommand*{\Glsxtrusefield}[2]{%
    \@Gls@entry@field{#1}{#2}%
  }
}
```

\GLSxtrusefield As above but convert to all caps.

```
\ifdef\texorpdfstring
{
  \newcommand*{\GLSxtrusefield}[2]{%
    \texorpdfstring
     {\glsdoifexists{#1}{\mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}}}%
     {\@gls@entry@field{#1}{#2}}%
  }
}
{
  \newcommand*{\GLSxtrusefield}[2]{%
    \glsdoifexists{#1}{\mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}}%
  }
}
```

\glsxtrdeffield Just use \csdef to provide a field value for the given entry.

```
\newcommand*{\glsxtrdeffield}[2]{\csdef{glo@\glsdetoklabel{#1}@#2}}
```

glsxtredeffield Just use \csedef to provide a field value for the given entry.

```
\newcommand*{\glsxtredeffield}[2]{\protected@csedef{glo@\glsdetoklabel{#1}@#2}}
```

etfieldifexists

```
\newcommand*{\glsxtrsetfieldifexists}[3]{\glsdoifexists{#1}{#3}}
```

\GlsXtrSetField Allow the user to set a field. First argument entry label, second argument field label, third argument value.

```
\newrobustcmd*{\GlsXtrSetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\csdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

\GlsXtrLetField Uses \cslet instead. Third argument should be a macro.

```
\newrobustcmd*{\GlstrLetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\cslet{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

sGlsXtrLetField Uses \csletcs instead. Third argument should be a control sequence name.

```
\newrobustcmd*{\csGlsXtrLetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\csletcs{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

LetFieldToField Sets the field for one entry to the field for another entry. Third argument should be the other entry and the fourth argument that other field label.

```
\newrobustcmd*{\GlsXtrLetFieldToField}[4]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\csletcs{glo@\glsdetoklabel{#1}@#2}{glo@\glsdetoklabel{#3}@#4}}%
}
```

gGlsXtrSetField Allow the user to set a field. First argument entry label, second argument field label, third argument value.

```
\newrobustcmd*{\gGlsXtrSetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\csgdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

xGlsXtrSetField

```
\newrobustcmd*{\xGlsXtrSetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\protected@csxdef{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

eGlsXtrSetField

```
\newrobustcmd*{\eGlsXtrSetField}[3]{%
  \glsxtrsetfieldifexists{#1}{#2}%
  {\protected@csedef{glo@\glsdetoklabel{#1}@#2}{#3}}%
}
```

XtrIfFieldEqStr

```
\newrobustcmd*{\GlsXtrIfFieldEqStr}[5]{%
```

```
  \glsxtrifhasfield{#1}{#2}%
  {%
    \ifdefstring{\glscurrentfieldvalue}{#3}{#4}{#5}%
  }%
  {#5}%
}
```

rIfFieldEqXpStr Like the above but first expands the string.

```
\newrobustcmd*{\GlsXtrIfFieldEqXpStr}[5]{%
  \glsxtrifhasfield{#1}{#2}%
  {%
    \protected@edef\@gls@tmp{#3}%
    \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
  }%
  {#5}%
}
```

fXpFieldEqXpStr Like the above but also expands the field value.

```
\newrobustcmd*{\GlsXtrIfXpFieldEqXpStr}[5]{%
  \glsxtrifhasfield{#1}{#2}%
  {%
    \protected@edef\@gls@tmp{\glscurrentfieldvalue}%
    \let\glscurrentfieldvalue\@gls@tmp
    \protected@edef\@gls@tmp{#3}%
    \ifdefequal{\glscurrentfieldvalue}{\@gls@tmp}{#4}{#5}%
  }%
  {#5}%
}
```

lsXtrForeignText `\GlsXtrForeignText{⟨entry label⟩}{⟨text⟩}`

If a field is used to store a language tag (such as `en-GB` or `de-CH-1996`) then this command uses tracklang's interface to encapsulate ⟨*text*⟩. The field identifying the locale is given by `\GlsXtrForeignTextField`.

```
\ifdef\foreignlanguage
{
  \ifdef\GetTrackedDialectFromLanguageTag
  {
    \newcommand{\GlsXtrForeignText}[2]{%
```

In case this is used inside the argument of `\glsxtrifhasfield`, save and restore `\glscurrentfieldvalue`.

```
      \let\@glsxtr@org@currentfieldvalue\glscurrentfieldvalue
      \glsxtrifhasfield{\GlsXtrForeignTextField}{#1}%
      {%
        \expandafter\GetTrackedDialectFromLanguageTag\expandafter
          {\glscurrentfieldvalue}{\@glsxtr@dialect}%
```

```
      \let\@glsxtr@locale\glscurrentfieldvalue
      \let\glscurrentfieldvalue\@glsxtr@org@currentfieldvalue
      \ifdefempty\@glsxtr@dialect
      {%
```

An exact match hasn't been found. A partial match can only be obtained with at least tracklang v1.3.6.

```
        \ifundef\TrackedDialectClosestSubMatch
        {%
          \GlossariesExtraWarning{Can't obtain dialect label
            (tracklang v1.3.6+ required)}%
        }%
        {\let\@glsxtr@dialect\TrackedDialectClosestSubMatch}%
      }%
      {}%
      \ifdefempty\@glsxtr@dialect
      {%
```

No tracked dialect found for the root language.

```
      }%
      {%
```

Check if there's a caption hook for the given dialect label.

```
        \ifcsundef{captions\@glsxtr@dialect}{}%
        {%
```

Dialect label not recognised. Check if there's a known mapping.

```
          \IfTrackedDialectHasMapping{\@glsxtr@dialect}%
          {%
            \edef\@glsxtr@dialect{%
              \GetTrackedDialectToMapping{\@glsxtr@dialect}}%
```

Does a caption hook exist for this?

```
            \ifcsundef{captions\@glsxtr@dialect}{}%
            {%
```

No mapping. Try root language label instead.

```
              \ifcsundef{captions\@tracklang@lang}{}%
              {%
                \let\@glsxtr@dialect\@tracklang@lang
              }%
            }%
          }%
          {%
```

No mapping. Try root language label instead.

```
            \ifcsundef{captions\@tracklang@lang}{}%
            {%
              \let\@glsxtr@dialect\@tracklang@lang
            }%
          }%
        }%
```

```
        }%
        \ifdefempty\@glsxtr@dialect
        {%
          \GlsXtrUnknownDialectWarning{\@glsxtr@locale}{\@tracklang@lang}%
          #2%
        }%
        {\foreignlanguage{\@glsxtr@dialect}{#2}}%
      }%
      {#2}% key not set
    }
  }
  {
    \newcommand{\GlsXtrForeignText}[2]{%
      \GlossariesExtraWarning{Can't encapsulate foreign text:
        tracklang v1.3.6+ required}%
      #2%
    }
  }
}
{
```

\foreignlanguage isn't defined so just do ⟨*text*⟩.

```
  \newcommand{\GlsXtrForeignText}[2]{#2}
}
```

oreignTextField This is the user2 field by default but may be redefined as required.

```
\newcommand*{\GlsXtrForeignTextField}{userii}
```

nDialectWarning

```
\newcommand*{\GlsXtrUnknownDialectWarning}[2]{%
  \GlossariesExtraWarning{Can't determine valid dialect label
   for locale '#1' (root language: #2)}%
}
```

\glsxtrpageref Like \glsrefentry but references the page number instead (if entry counting is on). The base glossaries package only introduced \GlsEntryCounterLabelPrefix in version 4.38, so it may not be defined.

```
\ifdef\GlsEntryCounterLabelPrefix
{%
  \newcommand*{\glsxtrpageref}[1]{%
    \ifglsentrycounter
      \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
    \else
      \ifglssubentrycounter
        \pageref{\GlsEntryCounterLabelPrefix\glsdetoklabel{#1}}%
      \else
        \gls{#1}%
      \fi
    \fi
```

```
  }
}%
{%
  \newcommand*{\glsxtrpageref}[1]{%
    \ifglsentrycounter
      \pageref{glsentry-\glsdetoklabel{#1}}%
    \else
      \ifglssubentrycounter
        \pageref{glsentry-\glsdetoklabel{#1}}%
      \else
        \gls{#1}%
      \fi
    \fi
  }
}%
```

lossarypreamble

```
\newcommand{\apptoglossarypreamble}[2][\glsdefaulttype]{%
  \ifcsdef{glolist@#1}%
  {%
   \ifcsundef{@glossarypreamble@#1}%
   {\csdef{@glossarypreamble@#1}{}}%
   {}%
   \csappto{@glossarypreamble@#1}{#2}%
  }%
  {%
    \GlossariesExtraWarning{Glossary `#1' is not defined}%
  }%
}
```

lossarypreamble

```
\newcommand{\preglossarypreamble}[2][\glsdefaulttype]{%
  \ifcsdef{glolist@#1}%
  {%
   \ifcsundef{@glossarypreamble@#1}%
   {\csdef{@glossarypreamble@#1}{}}%
   {}%
   \cspreto{@glossarypreamble@#1}{#2}%
  }%
  {%
    \GlossariesExtraWarning{Glossary `#1' is not defined}%
  }%
}
```

1.3 Modifications to Commands Provided by glossaries

Some of the commands provided by glossaries are modified to take into account new options or to change default behaviour.

\ifglsused

\ifglsused{⟨*label*⟩}{⟨*true part*⟩}{⟨*false part*⟩}

In the event that undefined entries should trigger a warning rather than an error, \ifglsused needs to be modified to check for existence. If the boolean variable is undefined, then its state is indeterminate and is neither true nor false, so neither ⟨*true part*⟩ nor ⟨*false*⟩ part will be performed if ⟨*label*⟩ is undefined.

```
\renewcommand*{\ifglsused}[3]{%
  \glsdoifexists{#1}{\ifbool{glo@\glsdetoklabel{#1}@flag}{#2}{#3}}%
}
```

Provide a starred version of \longnewglossaryentry that doesn't automatically insert \leavevmode\unskip\nopostdesc at the end of the description. The unstarred version is modified to use \glsxtrpostlongdescription instead.

ewglossaryentry

```
\renewcommand*{\longnewglossaryentry}{%
 \@ifstar\@glsxtr@s@longnewglossaryentry\@glsxtr@longnewglossaryentry
}
```

ewglossaryentry Starred version.

```
\newcommand{\@glsxtr@s@longnewglossaryentry}[3]{%
  \glsdoifnoexists{#1}%
  {%
     \bgroup
       \let\@org@newglossaryentryprehook\@newglossaryentryprehook
       \long\def\@newglossaryentryprehook{%
         \long\def\@glo@desc{#3}%
         \@org@newglossaryentryprehook
       }%
       \renewcommand*{\gls@assign@desc}[1]{%
          \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
          \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
        }
       \gls@defglossaryentry{#1}{#2}%
     \egroup
  }%
}
```

ewglossaryentry Unstarred version.

```
\newcommand{\@glsxtr@longnewglossaryentry}[3]{%
  \glsdoifnoexists{#1}%
  {%
     \bgroup
       \let\@org@newglossaryentryprehook\@newglossaryentryprehook
       \long\def\@newglossaryentryprehook{%
         \long\def\@glo@desc{#3\glsxtrpostlongdescription}%
```

```
          \@org@newglossaryentryprehook
        }%
        \renewcommand*{\gls@assign@desc}[1]{%
           \global\cslet{glo@\glsdetoklabel{#1}@desc}{\@glo@desc}%
```

The following is different from the base glossaries.sty:

```
           \global\cslet{glo@\glsdetoklabel{#1}@descplural}{\@glo@descplural}%
         }
        \gls@defglossaryentry{#1}{#2}%
      \egroup
  }%
}
```

longdescription Hook at the end of the description when using the unstarred \longnewglossaryentry.

```
\newcommand*{\glsxtrpostlongdescription}{\leavevmode\unskip\nopostdesc}
```

Provide a starred version of \newignoredglossary that doesn't add the glossary to the nohyperlist list.

ignoredglossary Redefine to check for star.

```
\renewcommand{\newignoredglossary}{%
 \@ifstar\glsxtr@s@newignoredglossary\glsxtr@org@newignoredglossary
}
```

ignoredglossary The original definition is patched to check for existence.

```
\newcommand*{\glsxtr@org@newignoredglossary}[1]{%
  \ifcsdef{glolist@#1}
  {%
    \glsxtrundefaction{Glossary type `#1' already exists}{}%
  }%
  {%
    \ifdefempty\@ignored@glossaries
    {%
      \edef\@ignored@glossaries{#1}%
    }%
    {%
      \eappto\@ignored@glossaries{,#1}%
    }%
    \csgdef{glolist@#1}{,}%
    \ifcsundef{gls@#1@entryfmt}%
    {%
      \defglsentryfmt[#1]{\glsentryfmt}%
    }%
    {}%
    \ifdefempty\@gls@nohyperlist
    {%
       \renewcommand*{\@gls@nohyperlist}{#1}%
    }%
    {%
```

```
      \eappto\@gls@nohyperlist{,#1}%
    }%
  }%
}
```

ignoredglossary Starred form.

```
\newcommand*{\glsxtr@s@newignoredglossary}[1]{%
  \ifcsdef{glolist@#1}
  {%
    \glsxtrundefaction{Glossary type `#1' already exists}{}%
  }%
  {%
    \ifdefempty\@ignored@glossaries
    {%
      \edef\@ignored@glossaries{#1}%
    }%
    {%
      \eappto\@ignored@glossaries{,#1}%
    }%
    \csgdef{glolist@#1}{,}%
    \ifcsundef{gls@#1@entryfmt}%
    {%
      \defglsentryfmt[#1]{\glsentryfmt}%
    }%
    {}%
  }%
}
```

\glssettoctitle Ignored glossaries don't have an associated title, so modify \glssettoctitle to check for it to prevent an undefined command written to the toc file.

```
\glsifusetranslator
{%
  \renewcommand*{\glssettoctitle}[1]{%
    \ifcsdef{gls@tr@set@#1@toctitle}%
    {%
      \csuse{gls@tr@set@#1@toctitle}%
    }%
    {%
      \ifcsdef{@glotype@#1@title}%
      {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
      {\def\glossarytoctitle{\glossarytitle}}%
    }%
  }%
}
{
  \renewcommand*{\glssettoctitle}[1]{%
    \ifcsdef{@glotype@#1@title}%
    {\def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}%
    {\def\glossarytoctitle{\glossarytitle}}%
```

```
  }
}
```

ignoredglossary As above but won't do anything if the glossary already exists.

```
\newcommand{\provideignoredglossary}{%
 \@ifstar\glsxtr@s@provideignoredglossary\glsxtr@provideignoredglossary
}
```

ignoredglossary Unstarred version.

```
\newcommand*{\glsxtr@provideignoredglossary}[1]{%
  \ifcsdef{glolist@#1}
  {}%
  {%
    \ifdefempty\@ignored@glossaries
    {%
      \edef\@ignored@glossaries{#1}%
    }%
    {%
      \eappto\@ignored@glossaries{,#1}%
    }%
    \csgdef{glolist@#1}{,}%
    \ifcsundef{gls@#1@entryfmt}%
    {%
      \defglsentryfmt[#1]{\glsentryfmt}%
    }%
    {}%
    \ifdefempty\@gls@nohyperlist
    {%
       \renewcommand*{\@gls@nohyperlist}{#1}%
    }%
    {%
       \eappto\@gls@nohyperlist{,#1}%
    }%
  }%
}
```

ignoredglossary Starred form.

```
\newcommand*{\glsxtr@s@provideignoredglossary}[1]{%
  \ifcsdef{glolist@#1}
  {}%
  {%
    \ifdefempty\@ignored@glossaries
    {%
      \edef\@ignored@glossaries{#1}%
    }%
    {%
      \eappto\@ignored@glossaries{,#1}%
    }%
    \csgdef{glolist@#1}{,}%
```

```
    \ifcsundef{gls@#1@entryfmt}%
    {%
      \defglsentryfmt[#1]{\glsentryfmt}%
    }%
    {}%
  }%
}
```

rcopytoglossary Adds an entry label to another glossary list. First argument is entry label. Second argument is glossary label.

```
\newcommand*{\glsxtrcopytoglossary}[2]{%
  \glsdoifexists{#1}%
  {%
    \ifcsdef{glolist@#2}
    {%
      \cseappto{glolist@#2}{#1,}%
    }%
    {%
      \glsxtrundefaction{Glossary type `#2' doesn't exist}{}%
    }%
  }%
}
```

1.3.1 Existence Checks

\glsdoifexists Modify \glsdoifexists to take account of the undefaction setting.

```
\renewcommand{\glsdoifexists}[2]{%
  \ifglsentryexists{#1}{#2}%
  {%
```

Define \glslabel in case it's needed after this command (for example in the post-link hook).

```
    \edef\glslabel{\glsdetoklabel{#1}}%
    \glsxtrundefaction{Glossary entry `\glslabel'
    has not been defined}{You need to define a glossary entry before
    you can reference it.}%
  }%
}
```

glsdoifnoexists Modify \glsdoifnoexists to take account of the undefaction setting.

```
\renewcommand{\glsdoifnoexists}[2]{%
   \ifglsentryexists{#1}{%
    \glsxtrundefaction{Glossary entry `\glsdetoklabel{#1}'
    has already been defined}{}}{#2}%
}
```

sdoifexistsordo Modify \glsdoifexistsordo to take account of the undefaction setting. This command was introduced in glossaries version 4.19, so check if it has been defined first.

```
\ifdef\glsdoifexistsordo
{%
  \renewcommand{\glsdoifexistsordo}[3]{%
    \ifglsentryexists{#1}{#2}%
    {%
      \glsxtrundefaction{Glossary entry `\glsdetoklabel{#1}'
      has not been defined}{You need to define a glossary entry
      before you can use it.}%
      #3%
    }%
  }%
}
{%
  \glsxtr@warnonexistsordo\glsdoifexistsordo
  \newcommand{\glsdoifexistsordo}[3]{%
    \ifglsentryexists{#1}{#2}%
    {%
      \glsxtrundefaction{Glossary entry `\glsdetoklabel{#1}'
      has not been defined}{You need to define a glossary entry
      before you can use it.}%
      #3%
    }%
  }%
}
```

arynoexistsordo Similarly for \doifglossarynoexistsordo.

```
\ifdef\doifglossarynoexistsordo
{%
  \renewcommand{\doifglossarynoexistsordo}[3]{%
    \ifglossaryexists{#1}%
    {%
      \glsxtrundefaction{Glossary type `#1' already exists}{}%
      #3%
    }%
    {#2}%
  }%
}
{%
  \glsxtr@warnonexistsordo\doifglossarynoexistsordo
  \newcommand{\doifglossarynoexistsordo}[3]{%
    \ifglossaryexists{#1}%
    {%
      \glsxtrundefaction{Glossary type `#1' already exists}{}%
      #3%
    }%
    {#2}%
  }%
}

```

There are now three types of cross-references: the see key (as original), the alias key (from glossaries-extra v1.12) and the seealso key (from glossaries-extra v1.16). The original see key needs to have a corresponding field (which it doesn't with the base glossaries package).

ryentryposthook Hook into end of \newglossaryentry to add "see" value as a field.

```
\appto\@newglossaryentryposthook{%
  \ifdefvoid\@glo@see
   {\csxdef{glo@\@glo@label @see}{}}%
   {%
     \csxdef{glo@\@glo@label @see}{\@glo@see}%
     \if@glsxtr@autoseeindex
       \@glsxtr@autoindexcrossrefs
     \fi
   }%
}
\appto\@gls@keymap{,{see}{see}}
```

\glsxtrusesee Apply \glsseeformat to the see key if not empty.

```
\newcommand*{\glsxtrusesee}[1]{%
  \glsdoifexists{#1}%
  {%
    \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
    \ifdefempty\@glo@see
    {}%
    {%
      \expandafter\glsxtr@usesee\@glo@see\@end@glsxtr@usesee
    }%
  }%
}
```

\glsxtr@usesee

```
\newcommand*{\glsxtr@usesee}[1][\seename]{%
  \@glsxtr@usesee[#1]%
}
```

\@glsxtr@usesee

```
\def\@glsxtr@usesee[#1]#2\@end@glsxtr@usesee{%
  \glsxtruseseeformat{#1}{#2}%
}
```

xtruseseeformat The format used by \glsxtrusesee. The first argument is the tag (such as \seename). The second argument is the comma-separated list of cross-referenced labels.

```
\newcommand*{\glsxtruseseeformat}[2]{%
  \glsseeformat[#1]{#2}{}%
}
```

lsseeitemformat glossaries originally defined \glsseeitemformat to use \glsentryname but in v3.0 this was switched to use \glsentrytext due to problems occurring with the name field being sanitized. Since this is no longer a problem, glossaries-extra restores the original definition as it

makes more sense to use the name in the cross-reference list. This still uses \glsaccesstext for abbreviations.

```
\renewcommand*{\glsseeitemformat}[1]{%
 \ifglshasshort{#1}{\glsaccesstext{#1}}{\glsaccessname{#1}}%
}
```

\glsxtrhiername `\glsxtrhiername{⟨label⟩}`

Displays the hierarchical name for the given entry. The cross-reference format \glsseeitemformat may be redefined to use this command to show the hierarchy, if required.

```
\newcommand*{\glsxtrhiername}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {\glsxtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep}%
    {}%
    \ifglshasshort{#1}{\glsaccessshort{#1}}{\glsaccessname{#1}}%
  }%
}
```

\Glsxtrhiername `\Glsxtrhiername{⟨label⟩}`

As above but displays the top-level name with an initial capital.

```
\newcommand*{\Glsxtrhiername}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {%
      \Glsxtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep
      \ifglshasshort{#1}{\glsaccessshort{#1}}{\glsaccessname{#1}}%
    }%
    {\ifglshasshort{#1}{\Glsaccessshort{#1}}{\Glsaccessname{#1}}}%
  }%
}
```

\GlsXtrhiername `\GlsXtrhiername{⟨label⟩}`

As above but converts the first letter of each name to a capital.

```
\newcommand*{\GlsXtrhiername}[1]{%
  \glsdoifexists{#1}%
```

```
  {%
    \glsxtrifhasfield{parent}{#1}%
    {\GlsXtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep}%
    {}%
    \ifglshasshort{#1}{\Glsaccessshort{#1}}{\Glsaccessname{#1}}%
  }%
}
```

\GLSxtrhiername `\GLSxtrhiername{⟨label⟩}`

As above but displays the top-level name in all-caps.

```
\newcommand*{\GLSxtrhiername}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {%
      \GLSxtrhiername{\glscurrentfieldvalue}\glsxtrhiernamesep
      \ifglshasshort{#1}{\glsaccessshort{#1}}{\glsaccessname{#1}}%
    }%
    {\ifglshasshort{#1}{\GLSaccessshort{#1}}{\GLSaccessname{#1}}}%
  }%
}
```

\GLSXTRhiername `\GLSXTRhiername{⟨label⟩}`

As above but displays all names in all-caps.

```
\newcommand*{\GLSXTRhiername}[1]{%
  \glsdoifexists{#1}%
  {%
    \glsxtrifhasfield{parent}{#1}%
    {\GLSXTRhiername{\glscurrentfieldvalue}\glsxtrhiernamesep}%
    {}
    \ifglshasshort{#1}{\GLSaccessshort{#1}}{\GLSaccessname{#1}}%
  }%
}
```

sxtrhiernamesep Separator used in \glsxtrhiername and variants.

```
\newcommand*{\glsxtrhiernamesep}{\,{\small$\triangleright$}\,}
```

lsxtruseseealso Apply \glsseeformat to the seealso key if not empty. There's no optional tag to worry about here.

```
\newcommand*{\glsxtruseseealso}[1]{%
  \glsdoifexists{#1}%
```

```
  {%
    \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%
    \ifdefempty\@glo@see
    {}%
    {%
      \expandafter\glsxtruseseealsoformat\expandafter{\@glo@see}%
    }%
  }%
}
```

seseealsoformat The format used by \glsxtruseseealso. The argument is the comma-separated list of cross-referenced labels.

```
\newcommand*{\glsxtruseseealsoformat}[1]{%
  \glsseeformat[\seealsoname]{#1}{}%
}
```

\glsxtrseelist Fully expands argument before passing to \glsseelist. (The argument to \glsseelist must be a comma-separated list of entry labels.)

```
\newrobustcmd{\glsxtrseelist}[1]{%
  \edef\@glo@tmp{\noexpand\glsseelist{#1}}\@glo@tmp
}
```

\seealsoname In case this command hasn't been defined. (Should be provided by language packages.)

```
\providecommand{\seealsoname}{see also}
```

xtrindexseealso If \@xdycrossrefhook is defined, provide a seealso crossref class. Otherwise this just does \glssee with \seealsoname as the tag. The hook is only defined if both xindy and glossaries v4.30+ are being used.

```
\ifdef\@xdycrossrefhook
{
```

Add the cross-reference class definition to the hook.

```
  \appto\@xdycrossrefhook{%
    \write\glswrite{(define-crossref-class \string"seealso\string"
      :unverified )}%
    \write\glswrite{(markup-crossref-list
       :class \string"seealso\string"^^J\space\space\space
       :open \string"\string\glsxtruseseealsoformat\glsopenbrace\string"
       :close \string"\glsclosebrace\string")}%
  }
```

Append to class list.

```
  \appto\@xdylocationclassorder{\space\string"seealso\string"}
```

This essentially works like \@do@seeglossary but uses the seealso class. This doesn't increment the associated counter.

```
  \newrobustcmd*{\glsxtrindexseealso}[2]{%
    \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
      \@glsxtr@recordsee{#1}{#2}%
```

```
    \fi
    \glsdoifexists{#1}%
    {%
      \@@glsxtrwrglossmark
      \def\@gls@xref{#2}%
      \@onelevel@sanitize\@gls@xref
      \@gls@checkmkidxchars\@gls@xref
      \gls@glossary{\csname glo@#1@type\endcsname}{%
        (indexentry
          :tkey (\csname glo@#1@index\endcsname)
          :xref (\string"\@gls@xref\string")
          :attr \string"seealso\string"
        )
      }%
    }%
  }
}
{
```

xindy not in use or glossaries version too old to support this.

```
  \newrobustcmd*{\glsxtrindexseealso}{\glssee[\seealsoname]}
}
```

The alias key should be set to the label of the synonymous entry. The seealso key essentially behaves like `see=[\seealsoname]{`⟨*xr-list*⟩`}`. Neither of these new keys has the optional tag part allowed with see.

If `\gls@set@xr@key` has been defined (glossaries v4.30), use that, otherwise just use `\glsaddstoragekey`.

```
\ifdef\gls@set@xr@key
{
```

We have at least glossaries v4.30. This means the new keys can be governed by the same settings as the see key.

```
  \define@key{glossentry}{alias}{%
    \gls@set@xr@key{alias}{\@glo@alias}{#1}%
  }
  \define@key{glossentry}{seealso}{%
    \gls@set@xr@key{seealso}{\@glo@seealso}{#1}%
  }
```

Add to the key mappings.

```
  \appto\@gls@keymap{,{alias}{alias},{seealso}{seealso}}
```

Set the default value.

```
  \appto\@newglossaryentryprehook{\def\@glo@alias{}\def\@glo@seealso{}}%
```

Assign the field values.

```
  \appto\@newglossaryentryposthook{%
    \ifdefvoid\@glo@seealso
     {\csxdef{glo@\@glo@label @seealso}{}}%
```

```
    {%
      \csxdef{glo@\@glo@label @seealso}{\@glo@seealso}%
      \if@glsxtr@autoseeindex
        \@glsxtr@autoindexcrossrefs
      \fi
    }%
```

The alias field doesn't trigger the automatic cross-reference indexing performed at the end of the document.

```
  \ifdefvoid\@glo@alias
   {\csxdef{glo@\@glo@label @alias}{}}%
   {%
     \csxdef{glo@\@glo@label @alias}{\@glo@alias}%
   }%
}
```

Provide user-level commands to access the values.

\glsxtralias

```
\newcommand*{\glsxtralias}[1]{\@gls@entry@field{#1}{alias}}
```

trseealsolabels

```
\newcommand*{\glsxtrseealsolabels}[1]{\@gls@entry@field{#1}{seealso}}
```

Add to the \@glo@autosee hook.

```
\appto\@glo@autoseehook{%
  \ifdefvoid\@glo@alias
  {%
    \ifdefvoid\@glo@seealso
    {}%
    {%
      \edef\@do@glssee{\noexpand\glsxtrindexseealso
        {\@glo@label}{\@glo@seealso}}%
      \@do@glssee
    }%
  }%
  {%
```

Add cross-reference if see key hasn't been used.

```
    \ifdefvoid\@glo@see
    {%
      \edef\@do@glssee{\noexpand\glssee{\@glo@label}{\@glo@alias}}%
      \@do@glssee
    }%
    {}%
  }%
}%
}
{
```

We have an older version of glossaries, so just use \glsaddstoragekey.

\glsxtralias

```
1648 \glsaddstoragekey*{alias}{}{\glsxtralias}
```

trseealsolabels

```
1649 \glsaddstoragekey*{seealso}{}{\glsxtrseealsolabels}
```

If \gls@set@xr@key isn't defined, then \@glo@autosee won't be either, so use the post entry definition hook.

ryentryposthook Append to the hook to check for the alias and seealso keys.

```
1650 \appto\@newglossaryentryposthook{%
1651   \ifcsvoid{glo@\@glo@label @alias}%
1652   {%
1653     \ifcsvoid{glo@\@glo@label @seealso}%
1654     {}%
1655     {%
1656       \edef\@do@glssee{\noexpand\glsxtrindexseealso
1657         {\@glo@label}{\csuse{glo@\@glo@label @seealso}}}%
1658       \@do@glssee
1659     }%
1660   }%
1661   {%
```

Add cross-reference if see key hasn't been used.

```
1662     \ifdefvoid\@glo@see
1663     {%
1664       \edef\@do@glssee{\noexpand\glssee
1665         {\@glo@label}{\csuse{glo@\@glo@label @alias}}}%
1666       \@do@glssee
1667     }%
1668     {}%
1669   }%
1670 }
```

```
1671 }
```

Add all unused cross-references at the end of the document.

```
1672 \AtEndDocument{\if@glsxtrindexcrossrefs\glsxtraddallcrossrefs\fi}
```

addallcrossrefs Iterate through all used entries and if they have a cross-reference, make sure the cross-reference has been added.

```
1673 \newcommand*{\glsxtraddallcrossrefs}{%
1674   \forallglossaries{\@glo@type}%
1675   {%
1676     \forglsentries[\@glo@type]{\@glo@label}%
1677     {%
1678       \ifglsused{\@glo@label}%
1679       {\expandafter\@glsxtr@addunusedxrefs\expandafter{\@glo@label}}{}%
1680     }%
```

```
  }%
}
```

@addunusedxrefs If the given entry has a see or seealso field add all unused cross-references. (The alias field isn't checked.)

```
\newcommand*{\@glsxtr@addunusedxrefs}[1]{%
  \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@see}%
  \ifdefvoid\@glo@see
  {}%
  {%
    \expandafter\glsxtr@addunused\@glo@see\@end@glsxtr@addunused
  }%
  \letcs{\@glo@see}{glo@\glsdetoklabel{#1}@seealso}%
  \ifdefvoid\@glo@see
  {}%
  {%
    \expandafter\glsxtr@addunused\@glo@see\@end@glsxtr@addunused
  }%
}
```

lsxtr@addunused Adds all the entries if they haven't been used.

```
\newcommand*{\glsxtr@addunused}[1][]{%
  \@glsxtr@addunused
}
```

lsxtr@addunused Adds all the entries if they haven't been used.

```
\def\@glsxtr@addunused#1\@end@glsxtr@addunused{%
 \@for\@glsxtr@label:=#1\do
 {%
   \ifglsused{\@glsxtr@label}{}%
   {%
     \glsadd[format=glsxtrunusedformat]{\@glsxtr@label}%
     \glsunset{\@glsxtr@label}%
     \expandafter\@glsxtr@addunusedxrefs\expandafter{\@glsxtr@label}%
   }%
 }%
}
```

xtrunusedformat

```
\newcommand*{\glsxtrunusedformat}[1]{\unskip}
```

1.3.2 Document Definitions

ls@begindocdefs This command was only introduced to glossaries v4.37, so it may not be defined. If it has been defined, redefine it to check \@glsxtr@docdefval so that it only inputs the .glsdefs file if docdef=true.

```
\ifdef\gls@begindocdefs
{%
```

```
\renewcommand*{\gls@begindocdefs}{%
  \ifnum\@glsxtr@docdefval=1\relax
    \@gls@enablesavenonumberlist
    \edef\@gls@restoreat{%
      \noexpand\catcode`\noexpand\@=\number\catcode`\@\relax}%
    \makeatletter
    \InputIfFileExists{\jobname.glsdefs}{}{}%
    \@gls@restoreat
    \undef\@gls@restoreat
    \gls@defdocnewglossaryentry
  \else
    \ifnum\@glsxtr@docdefval=3\relax
```

The docdef=atom package option has been set. Create the .glsdefs file for the autocomplete support but don't read it.

```
      \@gls@enablesavenonumberlist
      \let\gls@checkseeallowed\relax
      \let\newglossaryentry\new@atom@glossaryentry
      \global\newwrite\@gls@deffile
      \immediate\openout\@gls@deffile=\jobname.glsdefs
```

Write all currently defined entries.

```
      \forallglsentries{\@glsentry}{\@gls@writedef{\@glsentry}}%
    \fi
  \fi
}
}
{%
  \ifnum\@glsxtr@docdefval=3\relax
    \PackageError{glossaries-extra}{Package option
    `docdef=\@glsxtr@docdefsetting' requires at least version 4.37
    of the base glossaries.sty package}{}
  \fi
}
```

m@glossaryentry

```
\newrobustcmd{\new@atom@glossaryentry}[2]{%
  \gls@defglossaryentry{#1}{#2}%
  \@gls@writedef{#1}%
}
```

noidxglossaries

Modify \makenoidxglossaries so that it automatically sets docdef=false (unless the restricted setting is on) and disables the docdef key. This command isn't allowed with the record option.

```
\let\glsxtr@orgmakenoidxglossaries\makenoidxglossaries
\renewcommand{\makenoidxglossaries}{%
  \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
  {%
    \glsxtr@orgmakenoidxglossaries
```

Add marker to \@do@seeglossary but don't increment associated counter.

```
    \renewcommand{\@do@seeglossary}[2]{%
      \@@glsxtrwrglossmark
      \edef\@gls@label{\glsdetoklabel{##1}}%
      \protected@write\@auxout{}{%
        \string\@gls@reference
          {\csname glo@\@gls@label @type\endcsname}%
          {\@gls@label}%
          {%
            \string\glsseeformat##2{}%
          }%
      }%
    }%
```

Check for docdefs=restricted:

```
    \if@glsxtrdocdefrestricted
```

If restricted document definitions allowed, adjust \@gls@reference so that it doesn't test for existence.

```
      \renewcommand*{\@gls@reference}[3]{%
        \ifcsundef{@glsref@##1}{\csgdef{@glsref@##1}{}}{}%
        \ifinlistcs{##2}{@glsref@##1}%
        {}%
        {\listcsgadd{@glsref@##1}{##2}}%
        \ifcsundef{glo@\glsdetoklabel{##2}@loclist}%
        {\csgdef{glo@\glsdetoklabel{##2}@loclist}{}}%
        {}%
        \listcsgadd{glo@\glsdetoklabel{##2}@loclist}{##3}%
      }%
    \else
```

Disable document definitions.

```
      \@glsxtrdocdeffalse
    \fi
    \disable@keys{glossaries-extra.sty}{docdef}%
  }%
  {%
    \PackageError{glossaries-extra}{\string\makenoidxglossaries\space
     not permitted\MessageBreak
     with record=\@glsxtr@record@setting\space package option}%
    {You may only use \string\makenoidxglossaries\ space with the
     record=off option}%
  }%
}
```

ewglossaryentry Modify \gls@defdocnewglossaryentry so that it checks the docdef value.

```
\renewcommand*{\gls@defdocnewglossaryentry}{%
  \ifcase\@glsxtr@docdefval
```

docdef=false:

```
    \renewcommand*{\newglossaryentry}[2]{%
      \PackageError{glossaries-extra}{Glossary entries must
       be \MessageBreak defined in the preamble with \MessageBreak
       package option 'docdef=false'\MessageBreak(consider using
       'docdef=restricted')}{Move your glossary definitions to
       the preamble. You can also put them in a \MessageBreak separate file
       and load them with \string\loadglsentries.}%
    }%
  \or
```

(docdef=true case.) Since the see value is now saved in a field, it can be used by entries that have been defined in the document.

```
    \let\gls@checkseeallowed\relax
    \let\newglossaryentry\new@glossaryentry
  \else
```

Restricted mode just needs to allow the see value.

```
    \let\gls@checkseeallowed\relax
  \fi
}%
```

Permit a special form of document definition, but only allow it if the glossaries come at the end of the document. These commands behave a little like a combination of `\newterm` and `\gls`. This must be explicitly enabled with the following.

rEnableOnTheFly

```
\newcommand*{\GlsXtrEnableOnTheFly}{%
  \@ifstar\@sGlsXtrEnableOnTheFly\@GlsXtrEnableOnTheFly
}
```

rEnableOnTheFly The starred version attempts to allow UTF8 characters in the label, but this may break! (Formatting commands mustn't be used in the label, but the label may be a command whose replacement text is the actual label. This doesn't take into account a command that's defined in terms of another command that may eventually expand to the label text.)

```
\newcommand*{\@sGlsXtrEnableOnTheFly}{%
  \renewcommand*{\glsdetoklabel}[1]{%
    \expandafter\@glsxtr@ifcsstart\string##1 \@glsxtr@end@
    {%
      \expandafter\detokenize\expandafter{##1}%
    }%
    {\detokenize{##1}}%
  }%
  \@GlsXtrEnableOnTheFly
}
\def\@glsxtr@ifcsstart#1#2\@glsxtr@end@#3#4{%
  \expandafter\if\glsbackslash#1%
    #3%
  \else
    #4%
```

```
  \fi
}
```

sxtrstarflywarn

```
\newcommand*{\glsxtrstarflywarn}{%
  \GlossariesExtraWarning{Experimental starred version of
  \string\GlsXtrEnableOnTheFly\space in use (please ensure you have
  read the warnings in the glossaries-extra user manual)}%
}
```

rEnableOnTheFly

```
\newcommand*{\@GlsXtrEnableOnTheFly}{%
```

Don't redefine `\glsdetoklabel` if LuaTeX or XeTeX is being used, since it's mainly to allow accented characters in the label.

These definitions are all assigned the category given by:

\glsxtrcat

```
  \newcommand*{\glsxtrcat}{general}
```

\glsxtr

```
  \newcommand*{\glsxtr}[1][]{%
   \def\glsxtr@keylist{##1}%
   \@glsxtr
  }
```

\@glsxtr

```
  \newcommand*{\@glsxtr}[2][]{%
   \ifglsentryexists{##2}%
   {%
     \ifblank{##1}{}{\GlsXtrWarning{##1}{##2}}%
   }%
   {%
     \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
       description={\nopostdesc},##1}%
   }%
   \expandafter\gls\expandafter[\glsxtr@keylist]{##2}%
  }
```

\Glsxtr

```
  \newcommand*{\Glsxtr}[1][]{%
   \def\glsxtr@keylist{##1}%
   \@Glsxtr
  }
```

\@Glsxtr

```
  \newcommand*{\@Glsxtr}[2][]{%
   \ifglsentryexists{##2}%
```

```
  {%
    \ifblank{##1}{}{\GlsXtrWarning{##1}{##2}}%
  }%
  {%
    \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
      description={\nopostdesc},##1}%
  }%
  \expandafter\Gls\expandafter[\glsxtr@keylist]{##2}%
}
```

\glsxtrpl

```
\newcommand*{\glsxtrpl}[1][]{%
 \def\glsxtr@keylist{##1}%
 \@glsxtrpl
}
```

\@glsxtrpl

```
\newcommand*{\@glsxtrpl}[2][]{%
 \ifglsentryexists{##2}%
 {%
   \ifblank{##1}{}{\GlsXtrWarning{##1}{##2}}%
 }%
 {%
   \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
     description={\nopostdesc},##1}%
 }%
 \expandafter\glspl\expandafter[\glsxtr@keylist]{##2}%
}
```

\Glsxtrpl

```
\newcommand*{\Glsxtrpl}[1][]{%
 \def\glsxtr@keylist{##1}%
 \@Glsxtrpl
}
```

\@Glsxtrpl

```
\newcommand*{\@Glsxtrpl}[2][]{%
 \ifglsentryexists{##2}
 {%
   \ifblank{##1}{}{\GlsXtrWarning{##1}{##2}}%
 }%
 {%
   \gls@defglossaryentry{##2}{name={##2},category=\glsxtrcat,
     description={\nopostdesc},##1}%
 }%
 \expandafter\Glspl\expandafter[\glsxtr@keylist]{##2}%
}
```

\GlsXtrWarning

```
\newcommand*{\GlsXtrWarning}[2]{%
  \def\@glsxtr@optlist{##1}%
  \@onelevel@sanitize\@glsxtr@optlist
  \GlossariesExtraWarning{The options `\@glsxtr@optlist' have
  been ignored for entry `##2' as it has already been defined}%
}
```

Disable commands after the glossary:

```
\renewcommand\@printglossary[2]{%
  \def\@glsxtr@printglossopts{##1}%
  \@glsxtr@orgprintglossary{##1}{##2}%
  \def\@glsxtr{\@glsxtr@disabledflycommand\glsxtr}%
  \def\@glsxtrpl{\@glsxtr@disabledflycommand\glsxtrpl}%
  \def\@Glsxtr{\@glsxtr@disabledflycommand\Glsxtr}%
  \def\@Glsxtrpl{\@glsxtr@disabledflycommand\Glsxtrpl}%
}
```

abledflycommand

```
\newcommand*{\@glsxtr@disabledflycommand}[1]{%
  \PackageError{glossaries-extra}%
  {\string##1\space can't be used after any of the \MessageBreak
   glossaries have been displayed}%
  {The on-the-fly commands enabled by
   \string\GlsXtrEnableOnTheFly\space may only be used \MessageBreak
   before the glossaries. If you want to use any entries \MessageBreak
   after any of the glossaries, you must use the standard \MessageBreak
   method of first defining the entry and then using the \MessageBreak
   entry with commands like \string\gls}%
   \@@glsxtr@disabledflycommand
}%
\newcommand*{\@@glsxtr@disabledflycommand}[2][]{##2}
```

End of \GlsXtrEnableOnTheFly. Disable since it can only be used once.

```
  \let\GlsXtrEnableOnTheFly\relax
}
\@onlypreamble\GlsXtrEnableOnTheFly
```

1.3.3 Existing Glossary Style Modifications

Modify \setglossarystyle to keep track of the current style. This allows the \glossaries-extra-stylemods package to reset the current style after the required modifications have been made.

r@current@style Initialise the current style to the default style.

```
\newcommand*{\@glsxtr@current@style}{\@glossary@default@style}
```

Modify \setglossarystyle to set \@glsxtr@current@style.

etglossarystyle

```
\renewcommand*{\setglossarystyle}[1]{%
```

```
  \ifcsundef{@glsstyle@#1}%
  {%
    \PackageError{glossaries-extra}{Glossary style `#1' undefined}{}%
  }%
  {%
    \csname @glsstyle@#1\endcsname
```

Only set the current style if it exists.

```
    \protected@edef\@glsxtr@current@style{#1}%
  }%
  \ifx\@glossary@default@style\relax
    \protected@edef\@glossary@default@style{#1}%
  \fi
}
```

In case we have an old version of glossaries:

```
\ifdef\@glossary@default@style
{}
{%
  \let\@glossary@default@style\relax
}
```

listdottedwidth

If `\glslistdottedwidth` has been defined and is currently equal to `.5\hsize` then make the modification suggested in bug report #92

```
\ifdef\glslistdottedwidth
{%
  \ifdim\glslistdottedwidth=.5\hsize
    \setlength{\glslistdottedwidth}{-\dimexpr\maxdimen-1sp\relax}
    \AtBeginDocument{%
      \ifdim\glslistdottedwidth=-\dimexpr\maxdimen-1sp\relax
       \setlength{\glslistdottedwidth}{.5\columnwidth}%
      \fi
    }%
  \fi
}
{}%
```

Similarly for `\glsdescwidth`:

\glsdescwidth

```
\ifdef\glsdescwidth
{%
  \ifdim\glsdescwidth=.6\hsize
    \setlength{\glsdescwidth}{-\dimexpr\maxdimen-1sp\relax}
    \AtBeginDocument{%
      \ifdim\glsdescwidth=-\dimexpr\maxdimen-1sp\relax
       \setlength{\glsdescwidth}{.6\columnwidth}%
      \fi
    }%
  \fi
```

```
}
{}%
```

and for \glspagelistwidth:

lspagelistwidth

```
\ifdef\glspagelistwidth
{%
  \ifdim\glspagelistwidth=.1\hsize
    \setlength{\glspagelistwidth}{-\dimexpr\maxdimen-1sp\relax}
    \AtBeginDocument{%
      \ifdim\glspagelistwidth=-\dimexpr\maxdimen-1sp\relax
       \setlength{\glspagelistwidth}{.1\columnwidth}%
      \fi
    }%
  \fi
}
{}%
```

aryentrynumbers Has the nonumberlist option been used?

```
\def\org@glossaryentrynumbers#1{#1\gls@save@numberlist{#1}}%
\ifx\org@glossaryentrynumbers\glossaryentrynumbers
  \glsnonumberlistfalse
  \renewcommand*{\glossaryentrynumbers}[1]{%
    \ifglsentryexists{\glscurrententrylabel}%
    {%
      \@glsxtrpreloctag
      \GlsXtrFormatLocationList{#1}%
      \@glsxtrpostloctag
      \gls@save@numberlist{#1}%
    }{}%
  }%
\else
  \glsnonumberlisttrue
  \renewcommand*{\glossaryentrynumbers}[1]{%
    \ifglsentryexists{\glscurrententrylabel}%
    {%
      \gls@save@numberlist{#1}%
    }{}%
  }%
\fi
```

matLocationList Provide an easy interface to change the format of the location list without removing the save number list stuff.

```
\newcommand*{\GlsXtrFormatLocationList}[1]{#1}
```

Sometimes users want to prefix the location list with “page”/“pages”. The simplest way to determine if the location list consists of a single location is to check for instances of \delimN

or \delimR, but this isn't so easy to do as they might be embedded inside the argument of formatting commands. With a bit of trickery we can find out by adjusting \delimN and \delimR to set a flag and then save information to the auxiliary file for the next run.

ePreLocationTag

```
\newcommand*{\GlsXtrEnablePreLocationTag}[2]{%
  \let\@glsxtrpreloctag\@@glsxtrpreloctag
  \let\@glsxtrpostloctag\@@glsxtrpostloctag
  \renewcommand*{\@glsxtr@pagetag}{#1}%
  \renewcommand*{\@glsxtr@pagestag}{#2}%
  \renewcommand*{\@glsxtr@saveprelocatag}[2]{%
    \csgdef{@glsxtr@preloctag@##1}{##2}%
  }%
  \renewcommand*{\@glsxtr@doloctag}{%
    \ifcsundef{@glsxtr@preloctag@\glscurrententrylabel}%
    {%
      \GlossariesWarning{Missing pre-location tag for `\glscurrententrylabel'.
        Rerun required}%
    }%
    {%
      \csuse{@glsxtr@preloctag@\glscurrententrylabel}%
    }%
  }%
}
\@onlypreamble\GlsXtrEnablePreLocationTag
```

glsxtrpreloctag

```
\newcommand*{\@@glsxtrpreloctag}{%
  \let\@glsxtr@org@delimN\delimN
  \let\@glsxtr@org@delimR\delimR
  \let\@glsxtr@org@glsignore\glsignore
```

\gdef is required as the delimiters may occur inside a scope.

```
  \gdef\@glsxtr@thisloctag{\@glsxtr@pagetag}%
  \renewcommand*{\delimN}{%
    \gdef\@glsxtr@thisloctag{\@glsxtr@pagestag}%
    \@glsxtr@org@delimN}%
  \renewcommand*{\delimR}{%
    \gdef\@glsxtr@thisloctag{\@glsxtr@pagestag}%
    \@glsxtr@org@delimR}%
  \renewcommand*{\glsignore}[1]{%
    \gdef\@glsxtr@thisloctag{\relax}%
    \@glsxtr@org@glsignore{##1}}%
  \@glsxtr@doloctag
}
```

glsxtrpreloctag

```
\newcommand*{\@glsxtrpreloctag}{}
```

@glsxtr@pagetag

```
\newcommand*{\@glsxtr@pagetag}{}%
```

glsxtr@pagestag

```
\newcommand*{\@glsxtr@pagestag}{}%
```

lsxtrpostloctag

```
\newcommand*{\@@glsxtrpostloctag}{%
   \let\delimN\@glsxtr@org@delimN
   \let\delimR\@glsxtr@org@delimR
   \let\glsignore\@glsxtr@org@glsignore
   \protected@write\@auxout{}%
    {\string\@glsxtr@savepreloctag{\glscurrententrylabel}{\@glsxtr@thisloctag}}%
}
```

lsxtrpostloctag

```
\newcommand*{\@glsxtrpostloctag}{}
```

lsxtr@preloctag

```
\newcommand*{\@glsxtr@savepreloctag}[2]{}
\protected@write\@auxout{}{%
  \string\providecommand\string\@glsxtr@savepreloctag[2]{}}
```

glsxtr@doloctag

```
\newcommand*{\@glsxtr@doloctag}{}
```

ss@nonumberlist Modify the nonumberlist key to use `\GlsXtrFormatLocationList` (and also save the number list):

```
\renewcommand*{\KV@printgloss@nonumberlist}[1]{%
 \XKV@plfalse
 \XKV@sttrue
 \XKV@checkchoice[\XKV@resa]{#1}{true,false}%
 {%
   \csname glsnonumberlist\XKV@resa\endcsname
   \ifglsnonumberlist
     \def\glossaryentrynumbers##1{\gls@save@numberlist{##1}}%
   \else
     \def\glossaryentrynumbers##1{%
       \@glsxtrpreloctag
       \GlsXtrFormatLocationList{##1}%
       \@glsxtrpostloctag
       \gls@save@numberlist{##1}}%
   \fi
 }%
}
```

1.3.4 Entry Formatting, Hyperlinks and Indexing

\glsentryfmt Change default entry format. Use the generic format for regular terms (that is, entries that have a category with the regular attribute set) or non-regular terms without a short value and use the abbreviation format for non-regular terms that have a short value. If further attributes need to be checked, then \glsentryfmt will need redefining as appropriate (or use \defglsentryfmt). The abbreviation format is set here for entries that have a short form, even if they are regular entries to ensure the abbreviation fonts are correct.

```
\renewcommand*{\glsentryfmt}{%
  \ifglshasshort{\glslabel}{\glssetabbrvfmt{\glscategory{\glslabel}}}{}%
  \glsifregular{\glslabel}%
  {\glsxtrregularfont{\glsgenentryfmt}}%
  {%
    \ifglshasshort{\glslabel}%
    {\glsxtrabbreviationfont{\glsxtrgenabbrvfmt}}%
    {\glsxtrregularfont{\glsgenentryfmt}}%
  }%
}
```

sxtrregularfont Font used for regular entries.

```
\newcommand*{\glsxtrregularfont}[1]{#1}
```

bbreviationfont Font used for abbreviation entries.

```
\newcommand*{\glsxtrabbreviationfont}[1]{#1}
```

Commands like \glsifplural are only used by the \gls-like commands in the glossaries package, but it might be useful for the postlink hook to know if the user has used, say, \glsfirst or \glsplural. This can provide better consistency with the formatting of the \gls-like commands, even though they don't use \glsentryfmt.

@gls@field@link Redefine \@gls@field@link so that commands like \glsfirst can setup \glsxtrifwasfirstuse etc to allow the postlink hook to work better. This now has an optional argument that sets up the defaults.

```
\renewcommand{\@gls@field@link}[4][]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#2}{#3}{glslink}%
  \glsdoifexists{#3}%
  {%
```

Save and restore the hyper setting (\@gls@link also does this, but that's too late if the optional argument of \@gls@field@link modifies it).

```
    \let\glsxtrorg@ifKV@glslink@hyper\ifKV@glslink@hyper
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \def\glscustomtext{#4}%
    \@glsxtr@field@linkdefs
    #1%
```

```
    \@gls@link[#2]{#3}{#4}%
    \let\ifKV@glslink@hyper\glsxtrorg@ifKV@glslink@hyper
  }%
  \glspostlinkhook
}
```

The commands \gls, \Gls etc don't use \@gls@field@link, so they need modifying as well to use \@glsxtr@record.

\@gls@ Save the original definition and redefine.

```
\let\@glsxtr@org@gls@\@gls@
\def\@gls@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@gls@{#1}{#2}%
}%
```

\@glspl@ Save the original definition and redefine.

```
\let\@glsxtr@org@glspl@\@glspl@
\def\@glspl@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@glspl@{#1}{#2}%
}%
```

\@Gls@ Save the original definition and redefine.

```
\let\@glsxtr@org@Gls@\@Gls@
\def\@Gls@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@Gls@{#1}{#2}%
}%
```

\@Glspl@ Save the original definition and redefine.

```
\let\@glsxtr@org@Glspl@\@Glspl@
\def\@Glspl@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@Glspl@{#1}{#2}%
}%
```

\@GLS@ Save the original definition and redefine.

```
\let\@glsxtr@org@GLS@\@GLS@
\def\@GLS@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@GLS@{#1}{#2}%
}%
```

\@GLSpl@ Save the original definition and redefine.

```
\let\@glsxtr@org@GLSpl@\@GLSpl@
\def\@GLSpl@#1#2{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \@glsxtr@org@GLSpl@{#1}{#2}%
}%
```

\@glsdisp This is redefined to allow the recording on the first run. Can't save and restore \@glsdisp since it has an optional argument.

```
\renewcommand*{\@glsdisp}[3][]{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}{%
    \let\do@gls@link@checkfirsthyper\@gls@link@checkfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \def\glscustomtext{#3}%
    \def\glsinsert{}%
    \def\@glo@text{\csname gls@\glstype @entryfmt\endcsname}%
    \@gls@link[#1]{#2}{\@glo@text}%
    \ifKV@glslink@local
      \glslocalunset{#2}%
    \else
      \glsunset{#2}%
    \fi
  }%
  \glspostlinkhook
}
```

\@gls@@link@ Redefine to include \@glsxtr@record

```
\renewcommand*{\@gls@@link}[3][]{%
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexistsordo{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\relax
```

Post-link hook commands need initialising.

```
    \def\glscustomtext{#3}%
    \@glsxtr@field@linkdefs
    \@gls@link[#1]{#2}{#3}%
  }%
  {%
    \glstextformat{#3}%
  }%
  \glspostlinkhook
}
```

sxtrinitwrgloss Set the default if the wrgloss is omitted.

```
\newcommand*{\glsxtrinitwrgloss}{%
 \glsifattribute{\glslabel}{wrgloss}{after}%
 {%
   \glsxtrinitwrglossbeforefalse
 }%
 {%
   \glsxtrinitwrglossbeforetrue
 }%
}
```

trwrglossbefore Conditional to determine if the indexing should be done before the link text.

```
\newif\ifglsxtrinitwrglossbefore
\glsxtrinitwrglossbeforetrue
```

Define a wrgloss key to determine whether to write the glossary information before or after the link text.

```
\define@choicekey{glslink}{wrgloss}%
[\@glsxtr@wrglossval\@glsxtr@wrglossnr]%
{before,after}%
{%
  \ifcase\@glsxtr@wrglossnr\relax
    \glsxtrinitwrglossbeforetrue
  \or
    \glsxtrinitwrglossbeforefalse
  \fi
}

\define@key{glslink}{thevalue}{\def\@glsxtr@thevalue{#1}}

\define@key{glslink}{theHvalue}{\def\@glsxtr@theHvalue{#1}}
```

tr@hyperoutside Define a hyperoutside key to determine whether \hyperlink should be outside \glstextformat.

```
\define@boolkey{glslink}[glsxtr@]{hyperoutside}[true]{}
\glsxtr@hyperoutsidetrue
```

ocal@textformat Provide a key to locally change the text format.

```
\define@key{glslink}{textformat}{%
  \ifcsdef{#1}
  {%
    \letcs{\@glsxtr@local@textformat}{#1}%
  }%
  {%
    \PackageError{glossaries-extra}{Unknown control sequence name `#1'}{}%
  }%
}

\define@key{glslink}{prefix}{\def\glolinkprefix{#1}}
```

nithyperoutside Set the default if the hyperoutside is omitted.

```
\newcommand*{\glsxtrinithyperoutside}{%
 \glsifattribute{\glslabel}{hyperoutside}{false}%
 {%
   \glsxtr@hyperoutsidefalse
 }%
 {%
   \glsxtr@hyperoutsidetrue
 }%
}
```

r@inc@linkcount Does nothing by default.

```
\newcommand*{\glsxtr@inc@linkcount}{}
```

slinkpresetkeys User hook performed immediately before options are set. Does nothing by default.

```
\newcommand*{\glslinkpresetkeys}{}
```

sXtrExpandedFmt Helper command that (protected) fully expands second argument and then applies it to the first, which must be a command that takes a single argument.

```
\newrobustcmd*{\GlsXtrExpandedFmt}[2]{%
  \protected@edef\@glsxtr@tmp{#2}%
  \expandafter#1\expandafter{\@glsxtr@tmp}%
}
```

tion@counter@or If in a numbered equation, change the counter to equation. This can be overridden by explicitly setting the counter in the optional argument of commands like \gls and \glslink.

```
\newcommand*{\@glsxtr@use@equation@counter}{%
  \@glsxtr@ifnum@mmode{\def\@gls@counter{equation}}{}%
}
```

sxtr@do@autoadd If \GlsXtrAutoAddOnFormat is used, this will automatically use \glsadd. It's therefore only used with \@gls@link not with \glsadd otherwise it could trigger an infinite loop. The argument indicates the key family (glslink or glossadd).

```
\newcommand*{\glsxtr@do@autoadd}[1]{}
```

rAutoAddOnFormat \GlsXtrAutoAddOnFormat[⟨*label*⟩]{⟨*format list*⟩}{⟨*glsadd options*⟩}

If an entry is indexed with the format set to one identified in the comma-separated list, then automatically index it using \glsadd with the given options, which may override the current options. Scoping is needed to prevent leakage.

```
\newcommand*{\GlsXtrAutoAddOnFormat}[3][\glslabel]{%
  \renewcommand*{\glsxtr@do@autoadd}[1]{%
    \begingroup
      \protected@edef\@glsxtr@do@autoadd{%
         \noexpand\ifstrequal{##1}{glslink}%
         {%
           \noexpand\DTLifinlist{\@glsnumberformat}{#2}{\noexpand\glsadd[format={\@glsnumberfor
         }%
         {}%
      }%
      \@glsxtr@do@autoadd
    \endgroup
  }%
}
```

\@gls@link Redefine to allow the indexing to be placed after the link text. By default this is done before the link text to prevent problems that can occur from the whatsit, but there may be times when the user would like the indexing done afterwards even though it causes a whatsit.

```
\def\@gls@link[#1]#2#3{%
  \leavevmode
  \edef\glslabel{\glsdetoklabel{#2}}%
  \def\@gls@link@opts{#1}%
  \let\@gls@link@label\glslabel
  \let\@glsnumberformat\@glsxtr@defaultnumberformat
  \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
  \edef\glstype{\csname glo@\glslabel @type\endcsname}%
  \let\org@ifKV@glslink@hyper\ifKV@glslink@hyper
```

Save current value of \glolinkprefix:

```
  \let\@glsxtr@org@glolinkprefix\glolinkprefix
```

Initialise \@glsxtr@local@textformat

```
  \let\@glsxtr@local@textformat\relax
```

Initialise thevalue and theHvalue (v1.19).

```
  \def\@glsxtr@thevalue{}%
  \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

Initialise when indexing should occur (new to v1.14).

```
  \glsxtrinitwrgloss
```

Initialise whether \hyperlink should be outside \glstextformat (new to v1.21).

```
  \glsxtrinithyperoutside
```

Note that the default link options may override \glsxtrinitwrgloss.

```
  \@gls@setdefault@glslink@opts
```

Increment link counter if enabled (new to v1.26).

```
   \glsxtr@inc@linkcount
```

Check if the equations option has been set (new to v1.37).

```
   \if@glsxtr@equations
     \@glsxtr@use@equation@counter
   \fi
```

As the original definition.

```
  \do@glsdisablehyperinlist
  \do@gls@link@checkfirsthyper
```

User hook before options are set (new to v1.26):

```
  \glslinkpresetkeys
```

Set options.

```
  \setkeys{glslink}{#1}%
```

Perform auto add if set (new to v1.37)

```
  \glsxtr@do@autoadd{glslink}%
```

User hook after options are set:

```
\glslinkpostsetkeys
```

Check thevalue and theHvalue before saving (v1.19).

```
\ifdefempty{\@glsxtr@thevalue}%
{%
  \@gls@saveentrycounter
}%
{%
  \let\theglsentrycounter\@glsxtr@thevalue
  \def\theHglsentrycounter{\@glsxtr@theHvalue}%
}%
\@gls@setsort{\glslabel}%
```

Check if the textformat key has been used.

```
\ifx\@glsxtr@local@textformat\relax
```

Check textformat attribute (new to v1.21).

```
   \glshasattribute{\glslabel}{textformat}%
   {%
     \edef\@glsxtr@attrval{\glsgetattribute{\glslabel}{textformat}}%
     \ifcsdef{\@glsxtr@attrval}%
     {%
       \letcs{\@glsxtr@textformat}{\@glsxtr@attrval}%
     }%
     {%
       \GlossariesExtraWarning{Unknown control sequence name
       `\@glsxtr@attrval' supplied in textformat attribute
       for entry `\glslabel'. Reverting to default \string\glstextformat}%
       \let\@glsxtr@textformat\glstextformat
     }%
   }%
   {%
     \let\@glsxtr@textformat\glstextformat
   }%
\else
   \let\@glsxtr@textformat\@glsxtr@local@textformat
\fi
```

Do write if it should occur before the link text:

```
\ifglsxtrinitwrglossbefore
  \@do@wrglossary{#2}%
\fi
```

Do the link text:

```
\ifKV@glslink@hyper
  \ifglsxtr@hyperoutside
    \@glslink{\glolinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
  \else
    \@glsxtr@textformat{\@glslink{\glolinkprefix\glslabel}{#3}}%
  \fi
```

```
  \else
    \ifglsxtr@hyperoutside
      \glsdonohyperlink{\glolinkprefix\glslabel}{\@glsxtr@textformat{#3}}%
    \else
      \@glsxtr@textformat{\glsdonohyperlink{\glolinkprefix\glslabel}{#3}}%
    \fi
  \fi
```

Do write if it should occur after the link text:

```
  \ifglsxtrinitwrglossbefore
  \else
    \@do@wrglossary{#2}%
  \fi
```

Restore original value of `\glolinkprefix`:

```
  \let\glolinkprefix\@glsxtr@org@glolinkprefix
```

As the original definition:

```
  \let\ifKV@glslink@hyper\org@ifKV@glslink@hyper
}
```

```
\define@key{glossadd}{thevalue}{\def\@glsxtr@thevalue{#1}}
```

```
\define@key{glossadd}{theHvalue}{\def\@glsxtr@theHvalue{#1}}
```

`lsaddpresetkeys`

```
\newcommand*{\glsaddpresetkeys}{}
```

`saddpostsetkeys`

```
\newcommand*{\glsaddpostsetkeys}{}
```

`\glsadd` Redefine to include `\@glsxtr@record` and suppress in headings

```
\renewrobustcmd*{\glsadd}[2][]{%
  \glsxtrifinmark
  {}%
  {%
    \@gls@adjustmode
    \begingroup
      \@glsxtr@record{#1}{#2}{glossadd}%
      \glsdoifexists{#2}%
      {%
        \let\@glsnumberformat\@glsxtr@defaultnumberformat
        \edef\@gls@counter{\csname glo@\glsdetoklabel{#2}@counter\endcsname}%
        \def\@glsxtr@thevalue{}%
        \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
```

Implement any default settings (before options are set)

```
        \glsaddpresetkeys
        \setkeys{glossadd}{#1}%
```

Implement any default settings (after options are set)

```
        \glsaddpostsetkeys
        \ifdefempty{\@glsxtr@thevalue}%
        {%
          \@gls@saveentrycounter
        }%
        {%
          \let\theglsentrycounter\@glsxtr@thevalue
          \def\theHglsentrycounter{\@glsxtr@theHvalue}%
        }%
```

Define sort key if necessary (in case of sort=use):

```
        \@gls@setsort{#2}%
```

Ensure that indexing occurs (since that's the point of \glsadd). If indexing has been switched off by default, don't want the setting to affect \glsadd. The ignored format \glsignore can be used for selection without location, but the indexing still needs to be performed.

```
        \KV@glslink@noindexfalse
        \@@do@wrglossary{#2}%
      }%
    \endgroup
  }%
}
```

\glsaddeach Performs \glsadd for each entry listed in the mandatory argument.

```
\newrobustcmd{\glsaddeach}[2][]{%
  \@for\@gls@thislabel:=#2\do{\glsadd[#1]{\@gls@thislabel}}%
}
```

@field@linkdefs Default settings for \@gls@field@link

```
\newcommand*{\@glsxtr@field@linkdefs}{%
  \let\glsxtrifwasfirstuse\@secondoftwo
  \let\glsifplural\@secondoftwo
  \let\glscapscase\@firstofthree
  \let\glsinsert\@empty
}
```

Redefine the field link commands that need to modify the above. Also add accessibility support and set the abbreviation styles if required.

assignfieldfont

```
\newcommand*{\glsxtrassignfieldfont}[1]{%
  \ifglsentryexists{#1}%
  {%
    \ifglshasshort{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glsifregular{#1}%
      {\let\@gls@field@font\glsxtrregularfont}%
```

```
      {\let\@gls@field@font\@firstofone}%
    }%
    {%
      \glsifnotregular{#1}%
      {\let\@gls@field@font\@firstofone}%
      {\let\@gls@field@font\glsxtrregularfont}%
    }%
  }%
  {%
    \let\@gls@field@font\@gobble
  }%
}
```

`\@glstext@` The abbreviation format may also need setting.

```
\def\@glstext@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesstext{#2}#3}}%
}
```

`\@GLStext@` All uppercase version of `\glstext`. The abbreviation format may also need setting.

```
\def\@GLStext@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdofthree]{#1}{#2}%
    {\@gls@field@font{\GLSaccesstext{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`\@Glstext@` First letter uppercase version. The abbreviation format may also need setting.

```
\def\@Glstext@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@secondofthree]{#1}{#2}%
    {\@gls@field@font{\Glsaccesstext{#2}#3}}%
}
```

Version 1.07 ensures that `\glsfirst` etc honours the nohyperfirst attribute. Allow a convenient way for the user to revert to ignoring this attribute for these commands.

`ecknohyperfirst`

```
\newcommand*{\glsxtrchecknohyperfirst}[1]{%
  \glsifattribute{#1}{nohyperfirst}{true}{\KV@glslink@hyperfalse}{}%
}
```

`\@glsfirst@` No case changing version. The abbreviation format may also need setting.

```
\def\@glsfirst@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\glsfirst` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \glsxtrchecknohyperfirst{#2}%
```

```
  ]{#1}{#2}%
  {\@gls@field@font{\glsaccessfirst{#2}#3}}%
}
```

`\@Glsfirst@` First letter uppercase version. The abbreviation format may also need setting.

```
\def\@Glsfirst@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\Glsfirst` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glscapscase\@secondofthree
   \glsxtrchecknohyperfirst{#2}%
  ]%
   {#1}{#2}{\@gls@field@font{\Glsaccessfirst{#2}#3}}%
}
```

`\@GLSfirst@` All uppercase version. The abbreviation format may also need setting.

```
\def\@GLSfirst@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\GLSfirst` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glscapscase\@thirdofthree
   \glsxtrchecknohyperfirst{#2}%
  ]%
   {#1}{#2}{\@gls@field@font{\GLSaccessfirst{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`\@glsplural@` No case changing version. The abbreviation format may also need setting.

```
\def\@glsplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glsifplural\@firstoftwo]{#1}{#2}%
    {\@gls@field@font{\glsaccessplural{#2}#3}}%
}
```

`\@Glsplural@` First letter uppercase version. The abbreviation format may also need setting.

```
\def\@Glsplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glsifplural\@firstoftwo
   \let\glscapscase\@secondofthree
  ]%
    {#1}{#2}{\@gls@field@font{\Glsaccessplural{#2}#3}}%
}
```

`\@GLSplural@` All uppercase version. The abbreviation format may also need setting.

```
\def\@GLSplural@#1#2[#3]{%
```

```
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glsifplural\@firstoftwo
   \let\glscapscase\@thirdofthree
  ]%
    {#1}{#2}{\@gls@field@font{\GLSaccessplural{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`glsfirstplural@` No case changing version. The abbreviation format may also need setting.

```
\def\@glsfirstplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\glsfirstplural` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glsifplural\@firstoftwo
   \glsxtrchecknohyperfirst{#2}%
  ]%
   {#1}{#2}{\@gls@field@font{\glsaccessfirstplural{#2}#3}}%
}
```

`Glsfirstplural@` First letter uppercase version. The abbreviation format may also need setting.

```
\def\@Glsfirstplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\glsfirstplural` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glsifplural\@firstoftwo
   \let\glscapscase\@secondofthree
   \glsxtrchecknohyperfirst{#2}%
  ]%
   {#1}{#2}{\@gls@field@font{\Glsaccessfirstplural{#2}#3}}%
}
```

`GLSfirstplural@` All uppercase version. The abbreviation format may also need setting.

```
\def\@GLSfirstplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

Ensure that `\glsfirstplural` honours the nohyperfirst attribute.

```
  \@gls@field@link
  [\let\glsxtrifwasfirstuse\@firstoftwo
   \let\glsifplural\@firstoftwo
   \let\glscapscase\@thirdofthree
   \glsxtrchecknohyperfirst{#2}%
  ]%
   {#1}{#2}%
   {\@gls@field@font{\GLSaccessfirstplural{#2}\mfirstucMakeUppercase{#3}}}%
}
```

\@glsname@ Redefine to use accessibility support. The abbreviation format may also need setting.

```
\def\@glsname@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessname{#2}#3}}%
}
```

\@Glsname@ First letter uppercase version. The abbreviation format may also need setting.

```
\def\@Glsname@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]{#1}{#2}%
  {\@gls@field@font{\Glsaccessname{#2}#3}}%
}
```

\@GLSname@ All uppercase version. The abbreviation format may also need setting.

```
\def\@GLSname@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}%
    {\@gls@field@font{\GLSaccessname{#2}\mfirstucMakeUppercase{#3}}}%
}
```

\@glsdesc@

```
\def\@glsdesc@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccessdesc{#2}#3}}%
}
```

\@Glsdesc@ First letter uppercase version.

```
\def\@Glsdesc@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]{#1}{#2}%
  {\@gls@field@font{\Glsaccessdesc{#2}#3}}%
}
```

\@GLSdesc@ All uppercase version.

```
\def\@GLSdesc@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\GLSaccessdesc{#2}\mfirstucMakeUppercase{#3}}}%
}
```

@glsdescplural@ No case-changing version.

```
\def\@glsdescplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo
```

```
    \let\glsifplural\@firstoftwo
  ]{#1}{#2}{\@gls@field@font{\glsaccessdescplural{#2}#3}}%
}
```

`@Glsdescplural@` First letter uppercase version.

```
\def\@Glsdescplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo
   \let\glsifplural\@firstoftwo
  ]{#1}{#2}{\@gls@field@font{\Glsaccessdescplural{#2}#3}}%
}
```

`@GLSdescplural@` All uppercase version.

```
\def\@GLSdesc@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@thirdoftwo
   \let\glsifplural\@firstoftwo
  ]%
    {#1}{#2}%
    {\@gls@field@font{\GLSaccessdescplural{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`\@glssymbol@`

```
\def\@glssymbol@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link{#1}{#2}{\@gls@field@font{\glsaccesssymbol{#2}#3}}%
}
```

`\@Glssymbol@` First letter uppercase version.

```
\def\@Glssymbol@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsaccesssymbol{#2}#3}}%
}
```

`\@GLSsymbol@` All uppercase version.

```
\def\@GLSsymbol@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\GLSaccesssymbol{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`lssymbolplural@` No case-changing version.

```
\def\@glssymbolplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
```

```
  \@gls@field@link
  [\let\glscapscase\@secondoftwo
   \let\glsifplural\@firstoftwo
  ]{#1}{#2}{\@gls@field@font{\glsaccesssymbolplural{#2}#3}}%
}
```

`lssymbolplural@` First letter uppercase version.

```
\def\@Glssymbolplural@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo
   \let\glsifplural\@firstoftwo
  ]{#1}{#2}{\@gls@field@font{\Glsaccesssymbolplural{#2}#3}}%
}
```

`LSsymbolplural@` All uppercase version.

```
\def\@GLSsymbol@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@thirdoftwo
   \let\glsifplural\@firstoftwo
  ]%
    {#1}{#2}%
    {\@gls@field@font{\GLSaccesssymbolplural{#2}\mfirstucMakeUppercase{#3}}}%
}
```

`\@Glsuseri@` First letter uppercase version.

```
\def\@Glsuseri@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]{#1}{#2}%
  {\@gls@field@font{\Glsentryuseri{#2}#3}}%
}
```

`\@GLSuseri@` All uppercase version.

```
\def\@GLSuseri@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseri{#2}#3}}}%
}
```

`\@Glsuserii@` First letter uppercase version.

```
\def\@Glsuserii@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsentryuserii{#2}#3}}%
}
```

\@GLSuserii@ All uppercase version.

```
\def\@GLSuserii@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuserii{#2}#3}}}%
}
```

\@Glsuseriii@ First letter uppercase version.

```
\def\@Glsuseriii@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsentryuseriii{#2}#3}}%
}
```

\@GLSuseriii@ All uppercase version.

```
\def\@GLSuseriii@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseriii{#2}#3}}}%
}
```

\@Glsuseriv@ First letter uppercase version.

```
\def\@Glsuseriv@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsentryuseriv{#2}#3}}%
}
```

\@GLSuseriv@ All uppercase version.

```
\def\@GLSuseriv@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}%
    {\@gls@field@font{\mfirstucMakeUppercase{\glsentryuseriv{#2}#3}}}%
}
```

\@Glsuserv@ First letter uppercase version.

```
\def\@Glsuserv@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsentryuserv{#2}#3}}%
}
```

\@GLSuserv@ All uppercase version.

```
\def\@GLSuserv@#1#2[#3]{%
```

```
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuserv{#2}#3}}}%
}
```

`\@Glsuservi@` First letter uppercase version.

```
\def\@Glsuservi@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link
  [\let\glscapscase\@secondoftwo]%
   {#1}{#2}{\@gls@field@font{\Glsentryuservi{#2}#3}}%
}
```

`\@GLSuservi@` All uppercase version.

```
\def\@GLSuservi@#1#2[#3]{%
  \glsxtrassignfieldfont{#2}%
  \@gls@field@link[\let\glscapscase\@thirdoftwo]%
    {#1}{#2}{\@gls@field@font{\mfirstucMakeUppercase{\glsentryuservi{#2}#3}}}%
}
```

Commands like `\acrshort` already set `\glsifplural`, but they don't set `\glsxtrifwasfirstuse` so they need adjusting.

`\@acrshort` No case change.

```
\def\@acrshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsaccessshort{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@Acrshort` First letter uppercase.

```
\def\@Acrshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
```

```
    \def\glscustomtext{%
      \acronymfont{\Glsaccessshort{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@ACRshort` All uppercase.

```
\def\@ACRshort#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\acronymfont{\glsaccessshort{#2}}#3}%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@acrshortpl` No case change.

```
\def\@acrshortpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsaccessshortpl{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@Acrshortpl` First letter uppercase.

```
\def\@Acrshortpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
```

```
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\Glsaccessshortpl{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@ACRshortpl` All uppercase.

```
\def\@ACRshortpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\acronymfont{\glsaccessshortpl{#2}}#3}%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@acrlong` No case change.

```
\def\@acrlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsaccesslong{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@Acrlong` First letter uppercase.

```
\def\@Acrlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
```

```
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\Glsaccesslong{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@ACRlong` All uppercase.

```
\def\@ACRlong#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\acronymfont{\glsaccesslong{#2}}#3}%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@acrlongpl` No case change.

```
\def\@acrlongpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\glsaccesslongpl{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@Acrlongpl` First letter uppercase.

```
\def\@Acrlongpl#1#2[#3]{%
  \glsdoifexists{#2}%
```

```
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \acronymfont{\Glsaccesslongpl{#2}}#3%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\@ACRlongpl` All uppercase.

```
\def\@ACRlongpl#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\acronymfont{\glsaccesslongpl{#2}}#3}%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

Modify `\@glsaddkey` so additional keys provided by the user can be treated in a similar way.

`\@glsaddkey`

```
\renewcommand*{\@glsaddkey}[7]{%
  \key@ifundefined{glossentry}{#1}%
  {%
    \define@key{glossentry}{#1}{\csdef{@glo@#1}{##1}}%
    \appto\@gls@keymap{,{#1}{#1}}%
    \appto\@newglossaryentryprehook{\csdef{@glo@#1}{#2}}%
    \appto\@newglossaryentryposthook{%
      \letcs{\@glo@tmp}{@glo@#1}%
      \gls@assign@field{#2}{\@glo@label}{#1}{\@glo@tmp}%
    }%
    \newcommand*{#3}[1]{\@gls@entry@field{##1}{#1}}%
    \newcommand*{#4}[1]{\@Gls@entry@field{##1}{#1}}%
```

Now for the commands with links. First the version with no case change (same as before):

```
    \ifcsdef{@gls@user@#1@}%
```

```
    {%
       \PackageError{glossaries}%
       {Can't define `\string#5' as helper command
        `\expandafter\string\csname @gls@user@#1@\endcsname' already
        exists}%
       {}%
    }%
    {%
      \expandafter\newcommand\expandafter*\expandafter
        {\csname @gls@user@#1\endcsname}[2][]{%
          \new@ifnextchar[%
            {\csuse{@gls@user@#1@}{##1}{##2}}%
            {\csuse{@gls@user@#1@}{##1}{##2}[]}}%
      \csdef{@gls@user@#1@}##1##2[##3]{%
        \@gls@field@link{##1}{##2}{#3{##2}##3}%
      }%
      \newrobustcmd*{#5}{%
        \expandafter\@gls@hyp@opt\csname @gls@user@#1\endcsname}%
    }%
```

Next the version with the first letter converted to upper case (modified):

```
    \ifcsdef{@Gls@user@#1@}%
    {%
       \PackageError{glossaries}%
       {Can't define `\string#6' as helper command
        `\expandafter\string\csname @Gls@user@#1@\endcsname' already
         exists}%
       {}%
    }%
    {%
      \expandafter\newcommand\expandafter*\expandafter
        {\csname @Gls@user@#1\endcsname}[2][]{%
          \new@ifnextchar[%
            {\csuse{@Gls@user@#1@}{##1}{##2}}%
            {\csuse{@Gls@user@#1@}{##1}{##2}[]}}%
      \csdef{@Gls@user@#1@}##1##2[##3]{%
        \@gls@field@link[\let\glscapscase\@secondofthree]%
          {##1}{##2}{#4{##2}##3}%
      }%
      \newrobustcmd*{#6}{%
        \expandafter\@gls@hyp@opt\csname @Gls@user@#1\endcsname}%
    }%
```

Finally the all caps version (modified):

```
    \ifcsdef{@GLS@user@#1@}%
    {%
       \PackageError{glossaries}%
       {Can't define `\string#7' as helper command
        `\expandafter\string\csname @GLS@user@#1@\endcsname' already
         exists}%
```

```
      {}%
    }%
    {%
      \expandafter\newcommand\expandafter*\expandafter
        {\csname @GLS@user@#1\endcsname}[2][]{%
          \new@ifnextchar[%
            {\csuse{@GLS@user@#1@}{##1}{##2}}%
            {\csuse{@GLS@user@#1@}{##1}{##2}[]}}%
      \csdef{@GLS@user@#1@}##1##2[##3]{%
        \@gls@field@link[\let\glscapscase\@thirdofthree]%
           {##1}{##2}{\mfirstucMakeUppercase{#3{##2}##3}}%
      }%
      \newrobustcmd*{#7}{%
        \expandafter\@gls@hyp@opt\csname @GLS@user@#1\endcsname}%
    }%
  }%
  {%
    \PackageError{glossaries-extra}{Key '#1' already exists}{}%
  }%
}
```

checkfirsthyper Old versions of glossaries don't define this, so provide it just in case it hasn't been defined.

```
\providecommand*{\@gls@link@nocheckfirsthyper}{}
```

checkfirsthyper Modify check to determine if the hyperlink should be automatically suppressed, but save the original in case the acronyms are restored.

```
\let\@glsxtr@org@checkfirsthyper\@gls@link@checkfirsthyper
\renewcommand*{\@gls@link@checkfirsthyper}{%
```

\ifglsused isn't useful in the post link hook as it's already been unset by then, so define a command that can be used in the post link hook. Since \@gls@link@checkfirsthyper is only used by commands like \gls but not by other commands, this seems the best place to put it to automatically set the value for the commands that change the first use flag. The other commands should set \glsxtrifwasfirstuse to \@secondoftwo (which is done in \@glsxtr@field@linkdefs).

```
  \ifglsused{\glslabel}%
   {\let\glsxtrifwasfirstuse\@secondoftwo}
   {\let\glsxtrifwasfirstuse\@firstoftwo}%
```

Store the category label for convenience.

```
  \edef\glscategorylabel{\glscategory{\glslabel}}%
  \ifglsused{\glslabel}%
  {%
    \glsifcategoryattribute{\glscategorylabel}{nohypernext}{true}%
      {\KV@glslink@hyperfalse}{}%
  }%
  {%
    \glsifcategoryattribute{\glscategorylabel}{nohyperfirst}{true}%
      {\KV@glslink@hyperfalse}{}%
```

```
  }%
  \glslinkcheckfirsthyperhook
}
```

ablehyperinlist This command was introduced in glossaries v4.19. If it hasn't been defined, we're using an earlier version, in which case the nohyper attribute can't be implemented.

```
\ifdef\do@glsdisablehyperinlist
{%
  \let\@glsxtr@do@glsdisablehyperinlist\do@glsdisablehyperinlist
  \renewcommand*{\do@glsdisablehyperinlist}{%
    \@glsxtr@do@glsdisablehyperinlist
    \glsifattribute{\glslabel}{nohyper}{true}{\KV@glslink@hyperfalse}{}%
  }
}
{}
```

Define a noindex key to prevent writing information to the external file.

```
\define@boolkey{glslink}{noindex}[true]{}
\KV@glslink@noindexfalse
```

If `\@gls@setdefault@glslink@opts` has been defined (glossaries v4.20) use it to set the default keys in `\@glslink`.

lt@glslink@opts

```
\ifdef\@gls@setdefault@glslink@opts
{
  \renewcommand*{\@gls@setdefault@glslink@opts}{%
    \KV@glslink@noindexfalse
    \@glsxtrsetaliasnoindex
  }
}
{
```

Not defined so prepend it to `\do@glsdisablehyperinlist` to achieve the same effect.

```
  \newcommand*{\@gls@setdefault@glslink@opts}{%
    \KV@glslink@noindexfalse
    \@glsxtrsetaliasnoindex
  }
  \preto\do@glsdisablehyperinlist{\@gls@setdefault@glslink@opts}
}
```

setaliasnoindex Allow user to hook into the alias noindex setting. Default behaviour switches off indexing for aliases. If the record option is on, this will have been defined to do nothing. (bib2gls will deal with records for aliased entries.)

```
\providecommand*{\glsxtrsetaliasnoindex}{%
 \KV@glslink@noindextrue
}
```

setaliasnoindex

```
\newcommand*{\@glsxtrsetaliasnoindex}{%
 \glsxtrifhasfield{alias}{\glslabel}%
 {%
   \let\glsxtrindexaliased\@glsxtrindexaliased
   \glsxtrsetaliasnoindex
   \let\glsxtrindexaliased\@no@glsxtrindexaliased
 }%
 {}%
}
```

xtrindexaliased

```
\newcommand{\@glsxtrindexaliased}{%
 \ifKV@glslink@noindex
 \else
   \begingroup
   \let\@glsnumberformat\@glsxtr@defaultnumberformat
   \edef\@gls@counter{\csname glo@\glsdetoklabel{\glslabel}@counter\endcsname}%
   \glsxtr@saveentrycounter
   \@@do@wrglossary{\glsxtralias{\glslabel}}%
   \endgroup
 \fi
}
```

xtrindexaliased

```
\newcommand{\@no@glsxtrindexaliased}{%
  \PackageError{glossaries-extra}{\string\glsxtrindexaliased\space
  not permitted outside definition of \string\glsxtrsetaliasnoindex}%
  {}%
}
```

xtrindexaliased Provide a command to redirect alias indexing, but only allow it to be used within \glsxtrsetaliasnoindex.

```
\let\glsxtrindexaliased\@no@glsxtrindexaliased
```

tDefaultGlsOpts Set the default options for \glslink etc.

```
\newcommand*{\GlsXtrSetDefaultGlsOpts}[1]{%
  \renewcommand*{\@gls@setdefault@glslink@opts}{%
    \setkeys{glslink}{#1}%
    \@glsxtrsetaliasnoindex
  }%
}
```

lsxtrifindexing Provide user level command to access it in \glswriteentry.

```
\newcommand*{\glsxtrifindexing}[2]{%
  \ifKV@glslink@noindex #2\else #1\fi
}
```

\glswriteentry Redefine to test for indexonlyfirst category attribute.

```
\renewcommand*{\glswriteentry}[2]{%
```

```
  \glsxtrifindexing
  {%
   \ifglsindexonlyfirst
     \ifglsused{#1}
     {\glsxtrdoautoindexname{#1}{dualindex}}%
     {#2}%
   \else
     \glsifattribute{#1}{indexonlyfirst}{true}%
     {\ifglsused{#1}
      {\glsxtrdoautoindexname{#1}{dualindex}}%
      {#2}}%
     {#2}%
   \fi
  }%
  {}%
}
```

@do@@wrglossary Hook into glossary indexing command so that it can also use \index at the same time if required and add user hook.

```
\appto\@@do@@wrglossary{\@glsxtr@do@@wrindex
  \glsxtrdowrglossaryhook{\@gls@label}%
}
```

(The label can be obtained from \@gls@label at this point.)

Similarly for the "noidx" version:

s@noidxglossary

```
\appto\gls@noidxglossary{\@glsxtr@do@@wrindex
  \glsxtrdowrglossaryhook{\@gls@label}%
}
```

xtr@do@@wrindex

```
\newcommand*{\@glsxtr@do@@wrindex}{%
  \glsxtrdoautoindexname{\@gls@label}{dualindex}%
}
```

owrglossaryhook Allow user to hook into indexing code. (Always used by \glsadd. Used by \gls when indexing, which may or may not occur depending on the indexing settings.)

```
\newcommand*{\glsxtrdowrglossaryhook}[1]{}
```

gls@alt@hyp@opt Commands like \gls have a star or plus version. Provide a third symbol that the user can adapt for convenience.

```
\newcommand*{\@gls@alt@hyp@opt}[1]{%
 \let\glslinkvar\@firstofthree
 \let\@gls@hyp@opt@cs#1\relax
 \@ifstar{\s@gls@hyp@opt}%
 {\@ifnextchar+%
   {\@firstoftwo{\p@gls@hyp@opt}}%
```

```
    {%
      \expandafter\@ifnextchar\@gls@alt@hyp@opt@char
      {\@firstoftwo{\@alt@gls@hyp@opt}}%
      {#1}%
    }%
  }%
}
```

`alt@gls@hyp@opt` User version

```
\newcommand*{\@alt@gls@hyp@opt}[1][]{%
 \let\glslinkvar\@firstofthree
 \expandafter\@gls@hyp@opt@cs\expandafter[\@gls@alt@hyp@opt@keys,#1]}
```

`lt@hyp@opt@char` Contains the character used as the command modifier.

```
\newcommand*{\@gls@alt@hyp@opt@char}{}
```

`lt@hyp@opt@keys` Contains the option list used as the command modifier.

```
\newcommand*{\@gls@alt@hyp@opt@keys}{}
```

`rSetAltModifier`

```
\newcommand*{\GlsXtrSetAltModifier}[2]{%
  \let\@gls@hyp@opt\@gls@alt@hyp@opt
  \def\@gls@alt@hyp@opt@char{#1}%
  \def\@gls@alt@hyp@opt@keys{#2}%
  \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
  {}%
  {%
```

Let bib2gls know the modifier.

```
    \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@altmodifier}[1]{}}%
    \protected@write\@auxout{}{\string\@glsxtr@altmodifier{#1}}%
  }%
}
```

`org@dohyperlink`

```
\let\glsxtr@org@dohyperlink\glsdohyperlink
```

`glsnavhyperlink` Now that `\glsdohyperlink` (used by `\@glslink`) references `\glslabel` it's necessary to patch `\glsnavhyperlink` to avoid using it (since `\glslabel` won't be defined). This means temporarily redefining `\glsdohyperlink` to its original definition.

This command is provided by glossary-hypernav so it may not exist.

```
\ifdef\glsnavhyperlink
{
  \renewcommand*{\glsnavhyperlink}[3][\@glo@type]{%
    \edef\gls@grplabel{#2}\protected@edef\@gls@grptitle{#3}%
```

Scope:

```
    {%
      \let\glsdohyperlink\glsxtr@org@dohyperlink
```

```
      \@glslink{\glsnavhyperlinkname{#1}{#2}}{#3}%
    }%
  }%
}
{}
```

\glsdohyperlink Unpleasant complications can occur if the text or first key etc contains \gls, particularly if there are hyperlinks. To get around this problem, patch \glsdohyperlink so that it temporarily makes \gls behave like \glstext[⟨*hyper=false,noindex*⟩]. (This will be overridden if the user explicitly cancels either of those options in the optional argument of \gls or using the plus version.) This also patches the short form commands like \acrshort and \glsxtrshort to use \glsentryshort and, similarly, the long form commands like \acrlong and \glsxtrlong to use \glsentrylong. Added attribute check.

```
\renewcommand*{\glsdohyperlink}[2]{%
 \glshasattribute{\glslabel}{targeturl}%
 {%
   \glshasattribute{\glslabel}{targetname}%
   {%
     \glshasattribute{\glslabel}{targetcategory}%
     {%
       \hyperref{\glsgetattribute{\glslabel}{targeturl}}%
         {\glsgetattribute{\glslabel}{targetcategory}}%
         {\glsgetattribute{\glslabel}{targetname}}%
         {{\glsxtrprotectlinks#2}}%
     }%
     {%
       \hyperref{\glsgetattribute{\glslabel}{targeturl}}%
         {}%
         {\glsgetattribute{\glslabel}{targetname}}%
         {{\glsxtrprotectlinks#2}}%
     }%
   }%
   {%
     \href{\glsgetattribute{\glslabel}{targeturl}}%
       {{\glsxtrprotectlinks#2}}%
   }%
 }%
 {%
```

Check for alias.

```
   \glsfieldfetch{\glslabel}{alias}{\gloaliaslabel}%
   \ifdefvoid\gloaliaslabel
   {%
     \glsxtrhyperlink{#1}{{\glsxtrprotectlinks#2}}%
   }%
   {%
```

Redirect link to the alias target.

```
     \glsxtrhyperlink
```

```
      {\glolinkprefix\glsdetoklabel{\gloaliaslabel}}%
      {{\glsxtrprotectlinks#2}}%
    }%
  }%
}
```

glsxtrhyperlink Allows integration with the base glossaries package's debug=showtargets option.

```
\ifdef\@glsshowtarget
{
  \newcommand{\glsxtrhyperlink}[2]{%
    \@glsshowtarget{#1}%
    \hyperlink{#1}{#2}%
  }%
}
{
  \newcommand{\glsxtrhyperlink}[2]{\hyperlink{#1}{#2}}%
}
```

glsdisablehyper Redefine to set \glslabel (to allow it to be picked up by \glsdohyperlink). Also made it robust and added grouping to localise the definition of \glslabel. The original internal command @glo@label could probably be simply replaced with \glslabel, but it's retained in case its removal causes unexpected problems.

```
\renewrobustcmd*{\glshyperlink}[2][\glsentrytext{\@glo@label}]{%
 \glsdoifexists{#2}%
 {%
   \def\@glo@label{#2}%
   {\edef\glslabel{#2}%
   \@glslink{\glolinkprefix\glslabel}{#1}}%
 }%
}
```

glsdisablehyper Redefine in case we have an old version of glossaries. This now uses \def rather than \let to allow for redefinitions of \glsdonohyperlink.

```
\renewcommand{\glsdisablehyper}{%
  \KV@glslink@hyperfalse
  \def\@glslink{\glsdonohyperlink}%
  \let\@glstarget\@secondoftwo
}
```

\glsenablehyper This now uses \def rather than \let to allow for redefinitions of \glsdohypertarget and \glsdohyperlink.

```
\renewcommand{\glsenablehyper}{%
 \KV@glslink@hypertrue
 \def\@glslink{\glsdohyperlink}%
 \def\@glstarget{\glsdohypertarget}%
}
```

lsdonohyperlink This command was only introduced in glossaries v4.20, so it may not be defined (therefore use \def). For older glossaries versions, this won't be used if hyperref hasn't been loaded,

which means the indexing will still take place. The generated text is scoped (the link text in \hyperlink is also scoped, so it's consistent).

```
\def\glsdonohyperlink#1#2{{\glsxtrprotectlinks #2}}
```

\@glslink Reset \@glslink with patched versions:

```
\ifcsundef{hyperlink}%
{%
  \def\@glslink{\glsdonohyperlink}
}%
{%
  \def\@glslink{\glsdohyperlink}
}
```

xtrprotectlinks Make \gls (and variants) behave like the corresponding \glstext (and variants) with hyperlinking and indexing off.

```
\newcommand*{\glsxtrprotectlinks}{%
  \KV@glslink@hyperfalse
  \KV@glslink@noindextrue
  \let\@gls@\@glsxtr@p@text@
  \let\@Gls@\@Glsxtr@p@text@
  \let\@GLS@\@GLSxtr@p@text@
  \let\@glspl@\@glsxtr@p@plural@
  \let\@Glspl@\@Glsxtr@p@plural@
  \let\@GLSpl@\@GLSxtr@p@plural@
  \let\@glsxtrshort\@glsxtr@p@short@
  \let\@Glsxtrshort\@Glsxtr@p@short@
  \let\@GLSxtrshort\@GLSxtr@p@short@
  \let\@glsxtrlong\@glsxtr@p@long@
  \let\@Glsxtrlong\@Glsxtr@p@long@
  \let\@GLSxtrlong\@GLSxtr@p@long@
  \let\@glsxtrshortpl\@glsxtr@p@shortpl@
  \let\@Glsxtrshortpl\@Glsxtr@p@shortpl@
  \let\@GLSxtrshortpl\@GLSxtr@p@shortpl@
  \let\@glsxtrlongpl\@glsxtr@p@longpl@
  \let\@Glsxtrlongpl\@Glsxtr@p@longpl@
  \let\@GLSxtrlongpl\@GLSxtr@p@longpl@
  \let\@acrshort\@glsxtr@p@acrshort@
  \let\@Acrshort\@Glsxtr@p@acrshort@
  \let\@ACRshort\@GLSxtr@p@acrshort@
  \let\@acrshortpl\@glsxtr@p@acrshortpl@
  \let\@Acrshortpl\@Glsxtr@p@acrshortpl@
  \let\@ACRshortpl\@GLSxtr@p@acrshortpl@
  \let\@acrlong\@glsxtr@p@acrlong@
  \let\@Acrlong\@Glsxtr@p@acrlong@
  \let\@ACRlong\@GLSxtr@p@acrlong@
  \let\@acrlongpl\@glsxtr@p@acrlongpl@
  \let\@Acrlongpl\@Glsxtr@p@acrlongpl@
  \let\@ACRlongpl\@GLSxtr@p@acrlongpl@
}
```

These protected versions need grouping to prevent the label from getting confused.

@glsxtr@p@text@

```
\def\@glsxtr@p@text@#1#2[#3]{{\@glstext@{#1}{#2}[#3]}}
```

@Glsxtr@p@text@

```
\def\@Glsxtr@p@text@#1#2[#3]{{\@Glstext@{#1}{#2}[#3]}}
```

@GLSxtr@p@text@

```
\def\@GLSxtr@p@text@#1#2[#3]{{\@GLStext@{#1}{#2}[#3]}}
```

lsxtr@p@plural@

```
\def\@glsxtr@p@plural@#1#2[#3]{{\@glsplural@{#1}{#2}[#3]}}
```

lsxtr@p@plural@

```
\def\@Glsxtr@p@plural@#1#2[#3]{{\@Glsplural@{#1}{#2}[#3]}}
```

LSxtr@p@plural@

```
\def\@GLSxtr@p@plural@#1#2[#3]{{\@GLSplural@{#1}{#2}[#3]}}
```

glsxtr@p@short@

```
\def\@glsxtr@p@short@#1#2[#3]{%
 {%
  \glssetabbrvfmt{\glscategory{#2}}%
  \glsabbrvfont{\glsentryshort{#2}}#3%
 }%
}
```

Glsxtr@p@short@

```
\def\@Glsxtr@p@short@#1#2[#3]{%
 {%
   \glssetabbrvfmt{\glscategory{#2}}%
   \glsabbrvfont{\Glsentryshort{#2}}#3%
 }%
}
```

GLSxtr@p@short@

```
\def\@GLSxtr@p@short@#1#2[#3]{%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshort{#2}}#3}%
  }%
}
```

sxtr@p@shortpl@

```
\def\@glsxtr@p@shortpl@#1#2[#3]{%
 {%
   \glssetabbrvfmt{\glscategory{#2}}%
```

```
    \glsabbrvfont{\glsentryshortpl{#2}}#3%
  }%
}

sxtr@p@shortpl@
\def\@Glsxtr@p@shortpl@#1#2[#3]{%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \glsabbrvfont{\Glsentryshortpl{#2}}#3%
  }%
}

Sxtr@p@shortpl@
\def\@GLSxtr@p@shortpl@#1#2[#3]{%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \mfirstucMakeUppercase{\glsabbrvfont{\glsentryshortpl{#2}}#3}%
  }%
}

@glsxtr@p@long@
\def\@glsxtr@p@long@#1#2[#3]{{\glsentrylong{#2}#3}}

@Glsxtr@p@long@
\def\@Glsxtr@p@long@#1#2[#3]{{\Glsentrylong{#2}#3}}

@GLSxtr@p@long@
\def\@GLSxtr@p@long@#1#2[#3]{%
  {\mfirstucMakeUppercase{\glslongfont{\glsentrylong{#2}}#3}}}

lsxtr@p@longpl@
\def\@glsxtr@p@longpl@#1#2[#3]{{\glsentrylongpl{#2}#3}}

lsxtr@p@longpl@
\def\@Glsxtr@p@longpl@#1#2[#3]{{\glslongfont{\Glsentrylongpl{#2}}#3}}

LSxtr@p@longpl@
\def\@GLSxtr@p@longpl@#1#2[#3]{%
  {\mfirstucMakeUppercase{\glslongfont{\glsentrylongpl{#2}}#3}}}

xtr@p@acrshort@
\def\@glsxtr@p@acrshort@#1#2[#3]{{\acronymfont{\glsentryshort{#2}}#3}}

xtr@p@acrshort@
\def\@Glsxtr@p@acrshort@#1#2[#3]{{\acronymfont{\Glsentryshort{#2}}#3}}

xtr@p@acrshort@
\def\@GLSxtr@p@acrshort@#1#2[#3]{%
  {\mfirstucMakeUppercase{\acronymfont{\glsentryshort{#2}}#3}}}
```

```
r@p@acrshortpl@
                3188 \def\@glsxtr@p@acrshortpl@#1#2[#3]{{\acronymfont{\glsentryshortpl{#2}}#3}}
r@p@acrshortpl@
                3189 \def\@Glsxtr@p@acrshortpl@#1#2[#3]{{\acronymfont{\Glsentryshortpl{#2}}#3}}
r@p@acrshortpl@
                3190 \def\@GLSxtr@p@acrshortpl@#1#2[#3]{%
                3191   {\mfirstucMakeUppercase{\acronymfont{\glsentryshortpl{#2}}#3}}}
sxtr@p@acrlong@
                3192 \def\@glsxtr@p@acrlong@#1#2[#3]{{\glsentrylong{#2}#3}}
sxtr@p@acrlong@
                3193 \def\@Glsxtr@p@acrlong@#1#2[#3]{{\Glsentrylong{#2}#3}}
Sxtr@p@acrlong@
                3194 \def\@GLSxtr@p@acrlong@#1#2[#3]{%
                3195  {\mfirstucMakeUppercase{\glsentrylong{#2}#3}}}
tr@p@acrlongpl@
                3196 \def\@glsxtr@p@acrlongpl@#1#2[#3]{{\glsentrylongpl{#2}#3}}
tr@p@acrlongpl@
                3197 \def\@Glsxtr@p@acrlongpl@#1#2[#3]{{\Glsentrylongpl{#2}#3}}
tr@p@acrlongpl@
                3198 \def\@GLSxtr@p@acrlongpl@#1#2[#3]{%
                3199  {\mfirstucMakeUppercase{\glsentrylongpl{#2}#3}}}
```

Commands to minimise conflict.

```
  \@glsxtrp@opt
                3200 \newcommand*{\@glsxtrp@opt}{hyper=false,noindex}
```

\glsxtrsetpopts Used in glossary to switch hyperlinks on for the \glsxtrp type of commands.

```
                3201 \newcommand*{\glsxtrsetpopts}[1]{%
                3202   \renewcommand*{\@glsxtrp@opt}{#1}%
                3203 }
```

lossxtrsetpopts Used in glossary to switch hyperlinks on for the \glsxtrp type of commands.

```
                3204 \newcommand*{\glossxtrsetpopts}{%
                3205   \glsxtrsetpopts{noindex}%
                3206 }
     \@@glsxtrp
                3207 \newrobustcmd*{\@@glsxtrp}[2]{%
```

Add scope.

```
  {%
    \let\glspostlinkhook\relax
    \csname#1\expandafter\endcsname\expandafter[\@glsxtrp@opt]{#2}[]%
  }%
}
```

\@glsxtrp

```
\newrobustcmd*{\@glsxtrp}[2]{%
  \ifcsdef{gls#1}%
  {%
    \@@glsxtrp{gls#1}{#2}%
  }%
  {%
    \ifcsdef{glsxtr#1}%
    {%
      \@@glsxtrp{glsxtr#1}{#2}%
    }%
    {%
      \PackageError{glossaries-extra}{`#1' not recognised by
        \string\glsxtrp}{}%
    }%
  }%
}
```

\@Glsxtrp

```
\newrobustcmd*{\@Glsxtrp}[2]{%
  \ifcsdef{Gls#1}%
  {%
    \@@glsxtrp{Gls#1}{#2}%
  }%
  {%
    \ifcsdef{Glsxtr#1}%
    {%
      \@@glsxtrp{Glsxtr#1}{#2}%
    }%
    {%
      \PackageError{glossaries-extra}{`#1' not recognised by
        \string\Glsxtrp}{}%
    }%
  }%
}
```

\@GLSxtrp

```
\newrobustcmd*{\@GLSxtrp}[2]{%
  \ifcsdef{GLS#1}%
  {%
    \@@glsxtrp{GLS#1}{#2}%
  }%
```

```
  {%
    \ifcsdef{GLSxtr#1}%
    {%
      \@@glsxtrp{GLSxtr#1}{#2}%
    }%
    {%
      \PackageError{glossaries-extra}{`#1' not recognised by
        \string\GLSxtrp}{}%
    }%
  }%
}
```

\glsxtr@entry@p

```
\newrobustcmd*{\glsxtr@headentry@p}[2]{%
 \glsifattribute{#1}{headuc}{true}%
 {%
   \mfirstucMakeUppercase{\@gls@entry@field{#1}{#2}}%
 }%
 {%
   \@gls@entry@field{#1}{#2}%
 }%
}
```

\glsxtrp Not robust as it needs to expand somewhat.

```
\ifdef\texorpdfstring
{
  \newcommand{\glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\texorpdfstring
      {%
        \protect\glsxtrifinmark
        {%
          \ifcsdef{glsxtrhead#1}%
          {%
            {\protect\csuse{glsxtrhead#1}{#2}}%
          }%
          {%
            \glsxtr@headentry@p{#2}{#1}%
          }%
        }%
        {%
          \@glsxtrp{#1}{#2}%
        }%
      }%
      {%
        \protect\@gls@entry@field{#2}{#1}%
      }%
    }%
```

```
  }
}
{
  \newcommand{\glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\glsxtrifinmark
      {%
        \ifcsdef{glsxtrhead#1}%
        {%
          {\protect\csuse{glsxtrhead#1}}%
        }%
        {%
          \glsxtr@headentry@p{#2}{#1}%
        }%
      }%
      {%
        \@glsxtrp{#1}{#2}%
      }%
    }%
  }
}
```

Provide short synonyms for the most common option.

\glsps

```
\newcommand*{\glsps}{\glsxtrp{short}}
```

\glspt

```
\newcommand*{\glspt}{\glsxtrp{text}}
```

\Glsxtrp As above but use first letter upper case (but not for the bookmarks, which can't process \uppercase).

```
\ifdef\texorpdfstring
{
  \newcommand{\Glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\texorpdfstring
      {%
        \protect\glsxtrifinmark
        {%
          \ifcsdef{Glsxtrhead#1}%
          {%
            {\protect\csuse{Glsxtrhead#1}{#2}}%
          }%
          {%
            \protect\@Gls@entry@field{#2}{#1}%
          }%
```

```
        }%
        {%
          \@Glsxtrp{#1}{#2}%
        }%
      }%
      {%
        \protect\@gls@entry@field{#2}{#1}%
      }%
    }%
  }
}
{
  \newcommand{\Glsxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\glsxtrifinmark
      {%
        \ifcsdef{Glsxtrhead#1}%
        {%
          {\protect\csuse{Glsxtrhead#1}}%
        }%
        {%
          \protect\@Gls@entry@field{#2}{#1}%
        }%
      }%
      {%
        \@Glsxtrp{#1}{#2}%
      }%
    }%
  }
}
```

\GLSxtrp As above but all upper case (but not for the bookmarks, which can't process \uppercase).

```
\ifdef\texorpdfstring
{
  \newcommand{\GLSxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\texorpdfstring
      {%
        \protect\glsxtrifinmark
        {%
          \ifcsdef{GLSxtr#1}%
          {%
            {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
          }%
          {%
            \protect\mfirstucMakeUppercase
            {%
```

```
              \protect\@gls@entry@field{#2}{#1}%
            }%
          }%
        }%
        {%
          \@GLSxtrp{#1}{#2}%
        }%
      }%
      {%
        \protect\@gls@entry@field{#2}{#1}%
      }%
    }%
  }
}
{
  \newcommand{\GLSxtrp}[2]{%
    \protect\NoCaseChange
    {%
      \protect\glsxtrifinmark
      {%
        \ifcsdef{GLSxtr#1}%
        {%
          {\protect\GLSxtrshort[noindex,hyper=false]{#1}[]}%
        }%
        {%
          \protect\mfirstucMakeUppercase
          {%
            \protect\@gls@entry@field{#2}{#1}%
          }%
        }%
      }%
      {%
        \@GLSxtrp{#1}{#2}%
      }%
    }%
  }
}
```

1.3.5 Entry Counting

The (use) entry counting mechanism from glossaries is adjusted here to work with category attributes. Provide a convenient command to enable entry counting, set the entrycount attribute for given categories and redefine `\gls` etc to use `\cgls` instead. This form of entry counting is provided to adjust the formatting if the number of times an entry has been used (through commands that unset the first use flag) doesn't exceeding the specified threshold. For link counting, see Section 1.4.

First adjust definitions of the unset and reset commands to provide a hook, but changing the flag can cause problems in certain situations, so to allow the normal unsetting to be

temporarily disabled, \@glsunset is let to \@glsxtr@unset, which performs the actual unsetting through \@@glsunset and then does the hook. This means that the unsetting (and the hook) can switched off by redefining \@glsunset and then switched back on again by changing the definition back to \@glsxtr@unset.

\@glsxtr@unset Global unset.

```
\newcommand*{\@glsxtr@unset}[1]{%
  \@@glsunset{#1}%
  \glsxtrpostunset{#1}%
}%
```

\@glsunset Global unset.

```
\let\@glsunset\@glsxtr@unset
```

glsxtrpostunset

```
\newcommand*{\glsxtrpostunset}[1]{}
```

Provide a command to store a list of labels that will need unsetting.

tUnsetBuffering

```
\newcommand*{\GlsXtrStartUnsetBuffering}{%
  \@ifstar\s@GlsXtrStartUnsetBuffering\@GlsXtrStartUnsetBuffering
}
```

tUnsetBuffering Unstarred version doesn't check for duplicates.

```
\newcommand*{\@GlsXtrStartUnsetBuffering}{%
  \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
  \def\@glsxtr@unset@buffer{}%
  \let\@glsunset\@glsxtrbuffer@unset
}
```

tUnsetBuffering Starred version checks for duplicates.

```
\newcommand*{\s@GlsXtrStartUnsetBuffering}{%
  \let\@glsxtr@org@unset@buffer\@glsxtr@unset@buffer
  \def\@glsxtr@unset@buffer{}%
  \let\@glsunset\@glsxtrbuffer@nodup@unset
}
```

xtrbuffer@unset This must use a global change since \gls may have to be placed inside \mbox (for example, with soul commands).

```
\newcommand*{\@glsxtrbuffer@unset}[1]{%
  \listxadd\@glsxtr@unset@buffer{#1}%
}
```

fer@nodup@unset Alternative version that avoids duplicates. One level of expansion is performed on the argument in case it's a control sequence containing the label. (Not using \xifinlist as the added complexity might cause problems that the buffering is trying to overcome.)

```
\newcommand*{\@glsxtrbuffer@nodup@unset}[1]{%
  \expandafter\ifinlist\expandafter{#1}{\@glsxtr@unset@buffer}{}%
  {\listxadd\@glsxtr@unset@buffer{#1}}%
}
```

pUnsetBuffering

```
\newcommand*{\GlsXtrStopUnsetBuffering}{%
  \@ifstar\s@GlsXtrStopUnsetBuffering\@GlsXtrStopUnsetBuffering
}
```

pUnsetBuffering Unstarred form (global unset).

```
\newcommand*{\@GlsXtrStopUnsetBuffering}{%
  \let\@glsunset\@glsxtr@unset
  \forlistloop\@glsunset\@glsxtr@unset@buffer
  \let\@glsxtr@unset@buffer\@glsxtr@org@unset@buffer
}
```

pUnsetBuffering Starred form (local unset).

```
\newcommand*{\s@GlsXtrStopUnsetBuffering}{%
  \forlistloop\@glslocalunset\@glsxtr@unset@buffer
  \let\@glsunset\@glsxtr@unset
}
```

setBufferedList Iterate over labels stored in the current buffer. The argument is the handler macro.

```
\newcommand*{\GlsXtrForUnsetBufferedList}[1]{%
  \forlistloop#1\@glsxtr@unset@buffer
}
```

\@glslocalunset Local unset.

```
\renewcommand*{\@glslocalunset}[1]{%
  \@@glslocalunset{#1}%
  \glsxtrpostlocalunset{#1}%
}%
```

rpostlocalunset

```
\newcommand*{\glsxtrpostlocalunset}[1]{}
```

\@glsreset Global reset.

```
\renewcommand*{\@glsreset}[1]{%
  \@@glsreset{#1}%
  \glsxtrpostreset{#1}%
}%
```

glsxtrpostreset

```
\newcommand*{\glsxtrpostreset}[1]{}
```

\@glslocalreset Local reset.

```
\renewcommand*{\@glslocalreset}[1]{%
  \@@glslocalreset{#1}%
  \glsxtrpostlocalreset{#1}%
}%
```

rpostlocalreset

```
\newcommand*{\glsxtrpostlocalreset}[1]{}
```

slocalreseteach Locally reset a list of entries.

```
\newcommand*{\glslocalreseteach}[1]{%
  \gls@ifnotmeasuring
  {%
    \@for\@gls@thislabel:=#1\do{%
      \glsdoifexists{\@gls@thislabel}%
      {%
        \@glslocalreset{\@gls@thislabel}%
      }%
    }%
  }%
}
```

slocalunseteach Locally unset a list of entries.

```
\newcommand*{\glslocalunseteach}[1]{%
  \gls@ifnotmeasuring
  {%
    \@for\@gls@thislabel:=#1\do{%
      \glsdoifexists{\@gls@thislabel}%
      {%
        \@glslocalunset{\@gls@thislabel}%
      }%
    }%
  }%
}
```

leEntryCounting The first argument is the list of categories and the second argument is the value of the entrycount attribute.

```
\newcommand*{\GlsXtrEnableEntryCounting}[2]{%
```

Enable entry counting:

```
  \glsenableentrycount
```

Redefine \gls etc:

```
  \renewcommand*{\gls}{\cgls}%
  \renewcommand*{\Gls}{\cGls}%
  \renewcommand*{\glspl}{\cglspl}%
  \renewcommand*{\Glspl}{\cGlspl}%
  \renewcommand*{\GLS}{\cGLS}%
  \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
  \@glsxtr@setentrycountunsetattr{#1}{#2}%
```

In case this command is used again:

```
  \let\GlsXtrEnableEntryCounting\@glsxtr@setentrycountunsetattr
  \renewcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryUnitCounting\space
    can't be used with \string\GlsXtrEnableEntryCounting}%
   {Use one or other but not both commands}}%
}
```

ycountunsetattr

```
\newcommand*{\@glsxtr@setentrycountunsetattr}[2]{%
 \@for\@glsxtr@cat:=#1\do
 {%
   \ifdefempty{\@glsxtr@cat}{}%
   {%
     \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
   }%
 }%
}
```

Redefine the entry counting commands to take into account the entrycount attribute.

nableentrycount

```
\renewcommand*{\glsenableentrycount}{%
```

Enable new fields:

```
  \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defcounters}%
```

Just in case the user has switched on the docdef option.

```
  \renewcommand*{\gls@defdocnewglossaryentry}{%
    \renewcommand*\newglossaryentry[2]{%
      \PackageError{glossaries}{\string\newglossaryentry\space
      may only be used in the preamble when entry counting has
      been activated}{If you use \string\glsenableentrycount\space
      you must place all entry definitions in the preamble not in
      the document environment}%
    }%
  }%
```

New commands to access new fields:

```
  \newcommand*{\glsentrycurrcount}[1]{%
   \ifcsundef{glo@\glsdetoklabel{##1}@currcount}%
   {0}{\@gls@entry@field{##1}{currcount}}%
  }%
  \newcommand*{\glsentryprevcount}[1]{%
   \ifcsundef{glo@\glsdetoklabel{##1}@prevcount}%
   {0}{\@gls@entry@field{##1}{prevcount}}%
  }%
```

Adjust post unset and reset:

```
\let\@glsxtr@entrycount@org@unset\glsxtrpostunset
\renewcommand*{\glsxtrpostunset}[1]{%
  \@glsxtr@entrycount@org@unset{##1}%
  \@gls@increment@currcount{##1}%
}%
\let\@glsxtr@entrycount@org@localunset\glsxtrpostlocalunset
\renewcommand*{\glsxtrpostlocalunset}[1]{%
  \@glsxtr@entrycount@org@localunset{##1}%
  \@gls@local@increment@currcount{##1}%
}%
\let\@glsxtr@entrycount@org@reset\glsxtrpostreset
\renewcommand*{\glsxtrpostreset}[1]{%
  \@glsxtr@entrycount@org@reset{##1}%
  \csgdef{glo@\glsdetoklabel{##1}@currcount}{0}%
}%
\let\@glsxtr@entrycount@org@localreset\glsxtrpostlocalreset
\renewcommand*{\glsxtrpostlocalreset}[1]{%
  \@glsxtr@entrycount@org@localreset{##1}%
  \csdef{glo@\glsdetoklabel{##1}@currcount}{0}%
}%
```

Modifications to take into account the attributes that govern whether the entry should be unset.

```
\let\@cgls@\@@cgls@
\let\@cglspl@\@@cglspl@

\let\@cGls@\@@cGls@
\let\@cGlspl@\@@cGlspl@
\let\@cGLS@\@@cGLS@
\let\@cGLSpl@\@@cGLSpl@
```

The rest is as the original definition.

```
\AtEndDocument{\@gls@write@entrycounts}%
\renewcommand*{\@gls@entry@count}[2]{%
  \csgdef{glo@\glsdetoklabel{##1}@prevcount}{##2}%
}%
\let\glsenableentrycount\relax
\renewcommand*{\glsenableentryunitcount}{%
  \PackageError{glossaries-extra}{\string\glsenableentryunitcount\space
   can't be used with \string\glsenableentrycount}%
  {Use one or other but not both commands}%
}%
}
```

ite@entrycounts Modify this command so that it only writes the information for entries with the entrycount attribute and issue warning if no entries have this attribute set.

```
\renewcommand*{\@gls@write@entrycounts}{%
\immediate\write\@auxout
  {\string\providecommand*{\string\@gls@entry@count}[2]{}}%
```

```
  \count@=0\relax
  \forallglsentries{\@glsentry}{%
    \glshasattribute{\@glsentry}{entrycount}%
    {%
      \ifglsused{\@glsentry}%
      {%
        \immediate\write\@auxout
         {\string\@gls@entry@count{\@glsentry}{\glsentrycurrcount{\@glsentry}}}%
      }%
      {}%
      \advance\count@ by \@ne
    }%
    {}%
  }%
  \ifnum\count@=0
    \GlossariesExtraWarningNoLine{Entry counting has been enabled
     \MessageBreak with \string\glsenableentrycount\space but the
     \MessageBreak attribute 'entrycount' hasn't
     \MessageBreak been assigned to any of the defined
     \MessageBreak entries}%
  \fi
}
```

trifcounttrigger `\glsxtrifcounttrigger{⟨label⟩}{⟨trigger format⟩}{⟨normal⟩}`

```
\newcommand*{\glsxtrifcounttrigger}[3]{%
 \glshasattribute{#1}{entrycount}%
 {%
   \ifnum\glsentryprevcount{#1}>\glsgetattribute{#1}{entrycount}\relax
    #3%
   \else
    #2%
   \fi
 }%
 {#3}%
}
```

Actual internal definitions of `\cgls` used when entry counting is enabled.

`\@@cgls@`

```
\def\@@cgls@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cglsformat{#2}{#3}%
    \glsunset{#2}%
  }%
```

```
  {%
    \@gls@{#1}{#2}[#3]%
  }%
}%
```

\@@cglspl@

```
\def\@@cglspl@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cglsplformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
    \@glspl@{#1}{#2}[#3]%
  }%
}%
```

\@@cGls@

```
\def\@@cGls@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cGlsformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
    \@Gls@{#1}{#2}[#3]%
  }%
}%
```

\@@cGlspl@

```
\def\@@cGlspl@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cGlsplformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
    \@Glspl@{#1}{#2}[#3]%
  }%
}%
```

\@@cGLS@

```
\def\@@cGLS@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cGLSformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
```

```
    \@GLS@{#1}{#2}[#3]%
  }%
}%
```

\@@cGLSpl@

```
\def\@@cGLSpl@#1#2[#3]{%
  \glsxtrifcounttrigger{#2}%
  {%
    \cGLSplformat{#2}{#3}%
    \glsunset{#2}%
  }%
  {%
    \@GLSpl@{#1}{#2}[#3]%
  }%
}%
```

Remove default warnings from \cgls etc so that it can be used interchangeable with \gls etc.

\@cgls@

```
\def\@cgls@#1#2[#3]{\@gls@{#1}{#2}[#3]}
```

\@cGls@

```
\def\@cGls@#1#2[#3]{\@Gls@{#1}{#2}[#3]}
```

\@cglspl@

```
\def\@cglspl@#1#2[#3]{\@glspl@{#1}{#2}[#3]}
```

\@cGlspl@

```
\def\@cGlspl@#1#2[#3]{\@Glspl@{#1}{#2}[#3]}
```

Add all upper case versions not provided by glossaries.

\cGLS

```
\newrobustcmd*{\cGLS}{\@gls@hyp@opt\@cGLS}
```

\@cGLS Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@cGLS}[2][]{%
  \new@ifnextchar[{\@cGLS@{#1}{#2}}{\@cGLS@{#1}{#2}[]}%
}
```

\@cGLS@

```
\def\@cGLS@#1#2[#3]{\@GLS@{#1}{#2}[#3]}
```

\cGLSformat Format used by \cGLS if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
\newcommand*{\cGLSformat}[2]{%
 \expandafter\mfirstucMakeUppercase\expandafter{\cglsformat{#1}{#2}}%
}
```

\cGLSpl

```
\newrobustcmd*{\cGLSpl}{\@gls@hyp@opt\@cGLSpl}
```

\@cGLSpl Defined the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\@cGLSpl}[2][]{%
  \new@ifnextchar[{\@cGLSpl@{#1}{#2}}{\@cGLSpl@{#1}{#2}[]}%
}
```

\@cGLSpl@

```
\def\@cGLSpl@#1#2[#3]{\@GLSpl@{#1}{#2}[#3]}
```

\cGLSplformat Format used by \cGLSpl if entry only used once on previous run. The first argument is the label, the second argument is the insert text.

```
\newcommand*{\cGLSplformat}[2]{%
 \expandafter\mfirstucMakeUppercase\expandafter{\cglsplformat{#1}{#2}}%
}
```

Modify the trigger formats to check for the regular attribute.

\cglsformat

```
\renewcommand*{\cglsformat}[2]{%
  \glsifregular{#1}
  {\glsentryfirst{#1}}%
  {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}#2%
}
```

\cGlsformat

```
\renewcommand*{\cGlsformat}[2]{%
  \glsifregular{#1}
  {\Glsentryfirst{#1}}%
  {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}#2%
}
```

\cglsplformat

```
\renewcommand*{\cglsplformat}[2]{%
  \glsifregular{#1}
  {\glsentryfirstplural{#1}}%
  {\ifglshaslong{#1}{\glsentrylongpl{#1}}{\glsentryfirstplural{#1}}}#2%
}
```

\cGlsplformat

```
\renewcommand*{\cGlsplformat}[2]{%
  \glsifregular{#1}
  {\Glsentryfirstplural{#1}}%
  {\ifglshaslong{#1}{\Glsentrylongpl{#1}}{\Glsentryfirstplural{#1}}}#2%
}
```

New code similar to above for unit counting.

defunitcounters

```
\newcommand*{\@@newglossaryentry@defunitcounters}{%
  \edef\@glo@countunit{\csuse{@glsxtr@categoryattr@@\@glo@category @unitcount}}%
  \ifdefvoid\@glo@countunit
  {}%
  {%
    \@glsxtr@ifunitcounter{\@glo@countunit}%
    {}%
    {\expandafter\@glsxtr@addunitcounter\expandafter{\@glo@countunit}}%
  }%
}
```

r@unitcountlist List to keep track of which counters are being used by the entry unit count facility.

```
\newcommand*{\@glsxtr@unitcountlist}{}
```

@addunitcounter

```
\newcommand*{\@glsxtr@addunitcounter}[1]{%
 \listadd{\@glsxtr@unitcountlist}{#1}%
 \ifcsundef{glsxtr@theunit@#1}
 {%
   \ifcsdef{theH#1}%
   {\csdef{glsxtr@theunit@#1}{\csuse{theH#1}}}%
   {\csdef{glsxtr@theunit@#1}{\csuse{the#1}}}%
 }%
 {}%
}
```

r@ifunitcounter

```
\newcommand*{\@glsxtr@ifunitcounter}[3]{%
  \xifinlist{#1}{\@glsxtr@unitcountlist}{#2}{#3}%
}
```

urrentunitcount

```
\newcommand*\@glsxtr@currentunitcount[1]{%
 glo@\glsdetoklabel{#1}@currunit@\glsgetattribute{#1}{unitcount}.%
 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
}
```

eviousunitcount

```
\newcommand*\@glsxtr@previousunitcount[1]{%
 glo@\glsdetoklabel{#1}@prevunit@\glsgetattribute{#1}{unitcount}.%
 \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
}
```

t@currunitcount

```
\newcommand*{\@gls@increment@currunitcount}[1]{%
  \glshasattribute{#1}{unitcount}%
  {%
```

```
    \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
    \ifcsundef{\@glsxtr@csname}%
    {%
      \csgdef{\@glsxtr@csname}{1}%
      \listcsxadd
       {glo@\glsdetoklabel{#1}@unitlist}%
       {\glsgetattribute{#1}{unitcount}.%
        \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
       }%
    }%
    {%
      \csxdef{\@glsxtr@csname}%
      {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
    }%
  }%
  {}%
}

t@currunitcount
\newcommand*{\@gls@local@increment@currunitcount}[1]{%
  \glshasattribute{#1}{unitcount}%
  {%
    \edef\@glsxtr@csname{\@glsxtr@currentunitcount{#1}}%
    \ifcsundef{\@glsxtr@csname}%
    {%
      \csdef{\@glsxtr@csname}{1}%
      \listcseadd
       {glo@\glsdetoklabel{#1}@unitlist}%
       {\glsgetattribute{#1}{unitcount}.%
        \csuse{glsxtr@theunit@\glsgetattribute{#1}{unitcount}}%
       }%
    }%
    {%
      \csedef{\@glsxtr@csname}%
      {\number\numexpr\csname\@glsxtr@csname\endcsname+1}%
    }%
  }%
  {}%
}

r@currunitcount
\newcommand*{\@glsxtr@currunitcount}[2]{%
 \ifcsundef
 {glo@\glsdetoklabel{#1}@currunit@#2}%
 {0}%
 {\csuse{glo@\glsdetoklabel{#1}@currunit@#2}}%
}%

r@prevunitcount
```

```
\newcommand*{\@glsxtr@prevunitcount}[2]{%
 \ifcsundef
 {glo@\glsdetoklabel{#1}@prevunit@#2}%
 {0}%
 {\csuse{glo@\glsdetoklabel{#1}@prevunit@#2}}%
}%
```

eentryunitcount

```
\newcommand*{\glsenableentryunitcount}{%
```

Enable new fields:

```
  \appto\@newglossaryentry@defcounters{\@@newglossaryentry@defunitcounters}%
```

Just in case the user has switched on the docdef option.

```
  \renewcommand*{\gls@defdocnewglossaryentry}{%
    \renewcommand*\newglossaryentry[2]{%
      \PackageError{glossaries}{\string\newglossaryentry\space
      may only be used in the preamble when entry counting has
      been activated}{If you use \string\glsenableentryunitcount\space
      you must place all entry definitions in the preamble not in
      the document environment}%
    }%
  }%
```

New commands to access new fields:

```
  \newcommand*{\glsentrycurrcount}[1]{%
    \@glsxtr@currunitcount{##1}{\glsgetattribute{##1}{unitcount}.%
     \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
  }%
  \newcommand*{\glsentryprevcount}[1]{%
    \@glsxtr@prevunitcount{##1}{\glsgetattribute{##1}{unitcount}.%
     \csuse{glsxtr@theunit@\glsgetattribute{##1}{unitcount}}}%
  }%
```

Access total count:

```
  \newcommand*{\glsentryprevtotalcount}[1]{%
    \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
    {0}%
    {%
      \number\csuse{glo@\glsdetoklabel{##1}@prevunittotal}
    }%
  }%
```

Access max value:

```
  \newcommand*{\glsentryprevmaxcount}[1]{%
    \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
    {0}%
    {%
      \number\csuse{glo@\glsdetoklabel{##1}@prevunitmax}
    }%
  }%
```

Adjust post unset and reset:

```
\let\@glsxtr@entryunitcount@org@unset\glsxtrpostunset
\renewcommand*{\glsxtrpostunset}[1]{%
  \@glsxtr@entryunitcount@org@unset{##1}%
  \@gls@increment@currunitcount{##1}%
}%
\let\@glsxtr@entryunitcount@org@localunset\glsxtrpostlocalunset
\renewcommand*{\glsxtrpostlocalunset}[1]{%
  \@glsxtr@entryunitcount@org@localunset{##1}%
  \@gls@local@increment@currunitcount{##1}%
}%
\let\@glsxtr@entryunitcount@org@reset\glsxtrpostreset
\renewcommand*{\glsxtrpostreset}[1]{%
  \glshasattribute{##1}{unitcount}%
  {%
    \edef\@glsxtr@csname{\@glsxtr@currentunitcount{##1}}%
    \ifcsundef{\@glsxtr@csname}%
    {}%
    {\csgdef{\@glsxtr@csname}{0}}%
  }%
  {}%
}%
\let\@glsxtr@entryunitcount@org@localreset\glsxtrpostlocalreset
\renewcommand*{\glsxtrpostlocalreset}[1]{%
  \@glsxtr@entryunitcount@org@localreset{##1}%
  \glshasattribute{##1}{unitcount}%
  {%
    \edef\@glsxtr@csname{\@glsxtr@currentunitcount{##1}}%
    \ifcsundef{\@glsxtr@csname}%
    {}%
    {\csdef{\@glsxtr@csname}{0}}%
  }%
  {}%
}%
```

Modifications to take into account the attributes that govern whether the entry should be unset.

```
\let\@cgls@\@@cgls@
\let\@cglspl@\@@cglspl@

\let\@cGls@\@@cGls@
\let\@cGlspl@\@@cGlspl@
\let\@cGLS@\@@cGLS@
\let\@cGLSpl@\@@cGLSpl@
```

Write information to the aux file.

```
\AtEndDocument{\@gls@write@entryunitcounts}%
\renewcommand*{\@gls@entry@unitcount}[3]{%
  \csgdef{glo@\glsdetoklabel{##1}@prevunit@##3}{##2}%
  \ifcsundef{glo@\glsdetoklabel{##1}@prevunittotal}%
```

```
    {\csgdef{glo@\glsdetoklabel{##1}@prevunittotal}{##2}}%
    {%
      \csxdef{glo@\glsdetoklabel{##1}@prevunittotal}{
        \number\numexpr\csuse{glo@\glsdetoklabel{##1}@prevunittotal}+##2}%
    }%
    \ifcsundef{glo@\glsdetoklabel{##1}@prevunitmax}%
    {\csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}}%
    {%
      \ifnum\csuse{glo@\glsdetoklabel{##1}@prevunitmax}<##2
       \csgdef{glo@\glsdetoklabel{##1}@prevunitmax}{##2}%
      \fi
    }%
  }%
  \let\glsenableentryunitcount\relax
  \renewcommand*{\glsenableentrycount}{%
    \PackageError{glossaries-extra}{\string\glsenableentrycount\space
     can't be used with \string\glsenableentryunitcount}%
    {Use one or other but not both commands}%
  }%
}
\@onlypreamble\glsenableentryunitcount
```

entry@unitcount

```
\newcommand*{\@gls@entry@unitcount}[3]{}
```

ryunitcounts@do

```
\newcommand*{\@gls@write@entryunitcounts@do}[1]{%
  \immediate\write\@auxout
   {\string\@gls@entry@unitcount
     {\@glsentry}%
     {\@glsxtr@currunitcount{\@glsentry}{#1}%
     }%
     {#1}}%
}
```

entryunitcounts

```
\newcommand*{\@gls@write@entryunitcounts}{%
  \immediate\write\@auxout
    {\string\providecommand*{\string\@gls@entry@unitcount}[3]{}}%
  \count@=0\relax
  \forallglsentries{\@glsentry}{%
    \glshasattribute{\@glsentry}{unitcount}%
    {%
      \ifglsused{\@glsentry}%
      {%
        \forlistcsloop
          {\@gls@write@entryunitcounts@do}%
          {glo@\glsdetoklabel{\@glsentry}@unitlist}%
      }%
```

```
      {}%
      \advance\count@ by \@ne
    }%
    {}%
  }%
  \ifnum\count@=0
    \GlossariesExtraWarningNoLine{Entry counting has been enabled
     \MessageBreak with \string\glsenableentryunitcount\space but the
     \MessageBreak attribute 'unitcount' hasn't
     \MessageBreak been assigned to any of the defined
     \MessageBreak entries}%
  \fi
}
```

tryUnitCounting The first argument is the list of categories, the second argument is the value of the entrycount attribute and the third is the counter name.

```
\newcommand*{\GlsXtrEnableEntryUnitCounting}[3]{%
```

Enable entry counting:

```
  \glsenableentryunitcount
```

Redefine \gls etc:

```
  \renewcommand*{\gls}{\cgls}%
  \renewcommand*{\Gls}{\cGls}%
  \renewcommand*{\glspl}{\cglspl}%
  \renewcommand*{\Glspl}{\cGlspl}%
  \renewcommand*{\GLS}{\cGLS}%
  \renewcommand*{\GLSpl}{\cGLSpl}%
```

Set the entrycount attribute:

```
  \@glsxtr@setentryunitcountunsetattr{#1}{#2}{#3}%
```

In case this command is used again:

```
  \let\GlsXtrEnableEntryUnitCounting\@glsxtr@setentryunitcountunsetattr
  \renewcommand*{\GlsXtrEnableEntryCounting}[2]{%
   \PackageError{glossaries-extra}{\string\GlsXtrEnableEntryCounting\space
    can't be used with \string\GlsXtrEnableEntryUnitCounting}%
   {Use one or other but not both commands}}%
}
```

tcountunsetattr

```
\newcommand*{\@glsxtr@setentryunitcountunsetattr}[3]{%
 \@for\@glsxtr@cat:=#1\do
 {%
   \ifdefempty{\@glsxtr@cat}{}%
   {%
     \glssetcategoryattribute{\@glsxtr@cat}{entrycount}{#2}%
     \glssetcategoryattribute{\@glsxtr@cat}{unitcount}{#3}%
   }%
 }%
}
```

1.3.6 Acronym Modifications

It's more consistent to use the abbreviation code for acronyms, but make some adjustments to allow for continued use of the glossaries package's custom acronym format. (For example, user may already have defined some acronym styles with `\newacronymstyle` which they would like to continue to use.) The original glossaries acronym code can be restored with `\RestoreAcronyms`, but adjust `\SetGenericNewAcronym` so that `\newacronym` adds the category.

nericNewAcronym

```
\renewcommand*{\SetGenericNewAcronym}{%
  \let\@Gls@entryname\@Gls@acrentryname
  \renewcommand{\newacronym}[4][]{%
    \ifdefempty{\@glsacronymlists}%
    {%
      \def\@glo@type{\acronymtype}%
      \setkeys{glossentry}{##1}%
      \DeclareAcronymList{\@glo@type}%
    }%
    {}%
    \glskeylisttok{##1}%
    \glslabeltok{##2}%
    \glsshorttok{##3}%
    \glslongtok{##4}%
    \newacronymhook
    \protected@edef\@do@newglossaryentry{%
      \noexpand\newglossaryentry{\the\glslabeltok}%
      {%
        type=\acronymtype,%
        name={\expandonce{\acronymentry{##2}}},%
        sort={\acronymsort{\the\glsshorttok}{\the\glslongtok}},%
        text={\the\glsshorttok},%
        short={\the\glsshorttok},%
        shortplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
        long={\the\glslongtok},%
        longplural={\the\glslongtok\noexpand\acrpluralsuffix},%
        category=acronym,
        \GenericAcronymFields,%
        \the\glskeylisttok
      }%
    }%
    \@do@newglossaryentry
  }%
  \renewcommand*{\acrfullfmt}[3]{%
    \glslink[##1]{##2}{\genacrfullformat{##2}{##3}}}%
  \renewcommand*{\Acrfullfmt}[3]{%
    \glslink[##1]{##2}{\Genacrfullformat{##2}{##3}}}%
  \renewcommand*{\ACRfullfmt}[3]{%
    \glslink[##1]{##2}{%
```

```
      \mfirstucMakeUppercase{\genacrfullformat{##2}{##3}}}}%
  \renewcommand*{\acrfullplfmt}[3]{%
    \glslink[##1]{##2}{\genplacrfullformat{##2}{##3}}}%
  \renewcommand*{\Acrfullplfmt}[3]{%
    \glslink[##1]{##2}{\Genplacrfullformat{##2}{##3}}}%
  \renewcommand*{\ACRfullplfmt}[3]{%
    \glslink[##1]{##2}{%
      \mfirstucMakeUppercase{\genplacrfullformat{##2}{##3}}}}%
  \renewcommand*{\glsentryfull}[1]{\genacrfullformat{##1}{}}%
  \renewcommand*{\Glsentryfull}[1]{\Genacrfullformat{##1}{}}%
  \renewcommand*{\glsentryfullpl}[1]{\genplacrfullformat{##1}{}}%
  \renewcommand*{\Glsentryfullpl}[1]{\Genplacrfullformat{##1}{}}%
}
```

This will cause a problem for glossaries that contain a mixture of acronyms and abbreviations, so redefine `\newacronym` to use the new abbreviation interface.

First save the original definitions:

```
\let\@glsxtr@org@setacronymstyle\setacronymstyle
\let\@glsxtr@org@newacronymstyle\newacronymstyle
```

`msAbbreviations` Make acronyms use the same interface as abbreviations. Note that `\newacronymstyle` has a different implementation to `\newabbrevationstyle` so disable `\newacronymstyle` and `\setacronymstyle`.

```
\newcommand*{\MakeAcronymsAbbreviations}{%
  \renewcommand*{\newacronym}[4][]{%
    \glsxtr@newabbreviation{type=\acronymtype,category=acronym,##1}{##2}{##3}{##4}%
  }%
  \renewcommand*{\firstacronymfont}[1]{\glsfirstabbrvfont{##1}}%
  \renewcommand*{\acronymfont}[1]{\glsabbrvfont{##1}}%
  \renewcommand*{\setacronymstyle}[1]{%
    \PackageError{glossaries-extra}{\string\setacronymstyle{##1}
    unavailable.
    Use \string\setabbreviationstyle\space instead.
    The original acronym interface can be restored with
    \string\RestoreAcronyms}{}%
  }%
  \renewcommand*{\newacronymstyle}[1]{%
    \GlossariesExtraWarning{New acronym style `##1' won't be
    available unless you restore the original acronym interface with
    \string\RestoreAcronyms}%
    \@glsxtr@org@newacronymstyle{##1}%
  }%
}
```

Switch acronyms to abbreviations:

```
\MakeAcronymsAbbreviations
```

`RestoreAcronyms` Restore acronyms to glossaries interface.

```
\newcommand*{\RestoreAcronyms}{%
  \SetGenericNewAcronym
  \renewcommand{\firstacronymfont}[1]{\acronymfont{##1}}%
  \renewcommand{\acronymfont}[1]{##1}%
  \let\setacronymstyle\@glsxtr@org@setacronymstyle
  \let\newacronymstyle\@glsxtr@org@newacronymstyle
```

Need to restore the original definition of `\@gls@link@checkfirsthyper` but `\glsxtrifwasfirstuse` still needs setting for the benefit of the post-link hook.

```
  \renewcommand*\@gls@link@checkfirsthyper{%
    \ifglsused{\glslabel}%
    {\let\glsxtrifwasfirstuse\@secondoftwo}
    {\let\glsxtrifwasfirstuse\@firstoftwo}%
    \@glsxtr@org@checkfirsthyper
  }
  \glssetcategoryattribute{acronym}{regular}{false}%
  \setacronymstyle{long-short}%
}
```

`\glsacspace` Allow the user to customise the maximum value.

```
\renewcommand*{\glsacspace}[1]{%
  \settowidth{\dimen@}{(\firstacronymfont{\glsentryshort{#1}})}%
  \ifdim\dimen@<\glsacspacemax~\else\space\fi
}
```

`\glsacspacemax` Value used in the above.

```
\newcommand*{\glsacspacemax}{3em}
```

1.3.7 Indexing and Displaying Glossaries

From time-to-time users ask if they can have one glossary sorted normally and another sorted by definition or usage. With the base glossaries package this can only be achieved with the "noidx" commands (Option 1). This is an attempt to mix and match.

First we need a list of the glossaries that require makeindex/xindy.

`r@reg@glosslist`

```
\newcommand*{\@glsxtr@reg@glosslist}{}
```

Save the original definition of `\makeglossaries`:

```
\let\@glsxtr@org@makeglossaries\makeglossaries
```

Redefine `\makeglossaries` to take an optional argument. This should be empty for the usual behaviour (all glossaries need processing with an indexing application) or a comma-separated list of glossary labels indicating those glossaries that should be processed with an indexing application. The optional argument version shouldn't be used with record.

`\makeglossaries`

```
\renewcommand*{\makeglossaries}[1][]{%
 \@glsxtr@if@record@only
```

```
{%
 \PackageError{glossaries-extra}{\string\makeglossaries\space
  not permitted\MessageBreak with record=\@glsxtr@record@setting\space
  package option}%
 {You may only use \string\makeglossaries\space with
  record=off or record=alsoindex options}%
}%
{%
  \ifblank{#1}%
  {\@glsxtr@org@makeglossaries}%
  {%
    \ifx\@glsxtr@record@setting\@glsxtr@record@setting@alsoindex
      \PackageError{glossaries-extra}{\string\makeglossaries[#1]\space
      not permitted\MessageBreak with record=alsoindex package option}%
      {You may only use the hybrid \string\makeglossaries[...]\space with
       record=off option}%
    \else
      \edef\@glsxtr@reg@glosslist{#1}%
      \ifundef{\glswrite}{\newwrite\glswrite}{}%
      \protected@write\@auxout{}{\string\providecommand
        \string\@glsorder[1]{}}
      \protected@write\@auxout{}{\string\providecommand
        \string\@istfilename[1]{}}
      \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
      \protected@write\@auxout{}{\string\@glsorder{\glsorder}}
      \protected@write\@auxout{}{\string\glsxtr@makeglossaries{#1}}
      \write\@auxout{\string\providecommand\string\@gls@reference[3]{}}%
```

Iterate through each supplied glossary type and activate it.

```
      \@for\@glo@type:=#1\do{%
       \ifdefempty{\@glo@type}{}{\@makeglossary{\@glo@type}}%
      }%
```

New glossaries must be created before `\makeglossaries`:

```
      \renewcommand*\newglossary[4][]{%
      \PackageError{glossaries}{New glossaries
      must be created before \string\makeglossaries}{You need
      to move \string\makeglossaries\space after all your
      \string\newglossary\space commands}}%
```

Any subsequence instances of this command should have no effect

```
       \let\@makeglossary\relax
       \let\makeglossary\relax
       \renewcommand\makeglossaries[1][]{}%
```

Disable all commands that have no effect after `\makeglossaries`

```
       \@disable@onlypremakeg
```

Allow see key:

```
       \let\gls@checkseeallowed\relax
```

Adjust \@do@seeglossary. This needs to check for the entry's existence but don't increment associated counter.

```
\renewcommand*{\@do@seeglossary}[2]{%
  \glsdoifexists{##1}%
  {%
    \edef\@gls@label{\glsdetoklabel{##1}}%
    \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
    \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
    {\@glsxtr@org@doseeglossary{##1}{##2}}%
    {%
      \@@glsxtrwrglossmark
      \protected@write\@auxout{}{%
        \string\@gls@reference
          {\gls@type}{\@gls@label}{\string\glsseeformat##2{}}%
      }%
    }%
  }%
}%
```

Adjust \@@do@@wrglossary

```
\let\@glsxtr@@do@@wrglossary\@@do@@wrglossary
\def\@@do@@wrglossary{%
  \edef\@gls@type{\csname glo@\@gls@label @type\endcsname}%
  \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
  {\@glsxtr@@do@@wrglossary}%
  {\gls@noidxglossary}%
}%
```

Suppress warning about no \makeglossaries

```
\let\warn@nomakeglossaries\relax
\def\warn@noprintglossary{%
  \GlossariesWarningNoLine{No \string\printglossary\space
    or \string\printglossaries\space
    found.^^J(Remove \string\makeglossaries\space if you don't want
    any glossaries.)^^JThis document will not have a glossary}%
}%
```

Only warn for glossaries not listed.

```
\renewcommand{\@gls@noref@warn}[1]{%
  \edef\@gls@type{##1}%
  \expandafter\DTLifinlist\expandafter{\@gls@type}{\@glsxtr@reg@glosslist}%
  {%
    \GlossariesExtraWarning{Can't use
       \string\printnoidxglossary[type={\@gls@type}]
       when `\@gls@type' is listed in the optional argument of
       \string\makeglossaries}%
  }%
  {%
    \GlossariesWarning{Empty glossary for
    \string\printnoidxglossary[type={##1}].
```

```
4135            Rerun may be required (or you may have forgotten to use
4136            commands like \string\gls)}%
4137          }%
4138        }%
```

Adjust display number list to check for type:

```
4139        \renewcommand*{\glsdisplaynumberlist}[1]{%
4140          \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4141          {\@glsxtr@idx@displaynumberlist{##1}}%
4142          {\@glsxtr@noidx@displaynumberlist{##1}}%
4143        }%
```

Adjust entry list:

```
4144        \renewcommand*{\glsentrynumberlist}[1]{%
4145          \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4146          {\@glsxtr@idx@entrynumberlist{##1}}%
4147          {\@glsxtr@noidx@entrynumberlist{##1}}%
4148        }%
```

Adjust number list loop

```
4149        \renewcommand*{\glsnumberlistloop}[2]{%
4150          \expandafter\DTLifinlist\expandafter{##1}{\@glsxtr@reg@glosslist}%
4151          {%
4152             \PackageError{glossaries-extra}{\string\glsnumberlistloop\space
4153              not available for glossary `##1'}{}%
4154          }%
4155          {\@glsxtr@noidx@numberlistloop{##1}{##2}}%
4156        }%
```

Only sanitize sort for normal indexing glossaries.

```
4157        \renewcommand*{\glsprestandardsort}[3]{%
4158          \expandafter\DTLifinlist\expandafter{##2}{\@glsxtr@reg@glosslist}%
4159          {%
4160            \glsdosanitizesort
4161          }%
4162          {%
4163            \ifglssanitizesort
4164             \@gls@noidx@sanitizesort
4165            \else
4166             \@gls@noidx@nosanitizesort
4167            \fi
4168          }%
4169        }%
```

Unlike `\makenoidxglossaries` we can't automatically set sanitizesort=false. All entries must be defined in the preamble.

```
4170        \renewcommand*\new@glossaryentry[2]{%
4171          \PackageError{glossaries-extra}{Glossary entries must be defined
4172           in the preamble\MessageBreak when you use the optional argument
4173           of \string\makeglossaries}{Either move your definitions to the
4174           preamble or don't use the optional argument of
```

```
            \string\makeglossaries}%
        }%
```

Only activate sort key for glossaries that aren't listed in #1 (glossary label is stored in \@glo@type but this defaults to \glsdefaulttype so some expansion is required).

```
        \let\@glo@assign@sortkey\@glsxtr@mixed@assign@sortkey
        \renewcommand*{\@printgloss@setsort}{%
```

Need to extract just the type value.

```
          \expandafter\@glsxtr@gettype\expandafter,\@glsxtr@printglossopts,%
            type=\glsdefaulttype,\@end@glsxtr@gettype
          \def\@glo@sorttype{\@glo@default@sorttype}%
        }%
```

Check automake setting:

```
        \ifglsautomake
          \renewcommand*{\@gls@doautomake}{%
            \@for\@gls@type:=\@glsxtr@reg@glosslist\do{%
              \ifdefempty{\@gls@type}{}{\@gls@automake{\@gls@type}}%
            }%
          }%
        \fi
```

Check the sort setting (glossaries v4.30 onwards):

```
        \ifdef\@glo@check@sortallowed{\@glo@check@sortallowed\makeglossaries}{}%
      \fi
    }%
  }%
}
```

The optional argument version of \makeglossaries needs an adjustment to \@printglossary to allow \@glo@assign@sortkey to pick up the glossary type.

rgprintglossary This no longer simply saves \@printglossary with \let but is actually defined to check for the existence of the title, since ignored glossaries don't have a title assigned. (bib2gls writes \provideignoredglossary to the glstex file for some settings, so the glossary might not have been defined.) (This command is also used for on-the-fly setting.)

```
\newcommand{\@glsxtr@orgprintglossary}[2]{%
  \def\@glo@type{\glsdefaulttype}%
```

Add check here.

```
  \def\glossarytitle{%
     \ifcsdef{@glotype@\@glo@type @title}%
     {\csuse{@glotype@\@glo@type @title}}%
     {\glossaryname}}%
  \def\glossarytoctitle{\glossarytitle}%
  \let\org@glossarytitle\glossarytitle
  \def\@glossarystyle{%
    \ifx\@glossary@default@style\relax
      \GlossariesWarning{No default glossary style provided \MessageBreak
        for the glossary `\@glo@type'. \MessageBreak
```

```
        Using deprecated fallback. \MessageBreak
        To fix this set the style with \MessageBreak
        \string\setglossarystyle\space or use the \MessageBreak
        style key=value option}%
    \fi
  }%
  \def\gls@dotoctitle{\glssettoctitle{\@glo@type}}%
  \let\@org@glossaryentrynumbers\glossaryentrynumbers
  \bgroup
    \@printgloss@setsort
    \setkeys{printgloss}{#1}%
    \ifx\glossarytitle\org@glossarytitle
    \else
      \cslet{@glotype@\@glo@type @title}{\glossarytitle}%
    \fi
    \let\currentglossary\@glo@type
    \let\org@glossaryentrynumbers\glossaryentrynumbers
    \let\glsnonextpages\@glsnonextpages
    \let\glsnextpages\@glsnextpages

    \glsxtractivatenopost
    \gls@dotoctitle
    \@glossarystyle
    \let\gls@org@glossaryentryfield\glossentry
    \let\gls@org@glossarysubentryfield\subglossentry
    \renewcommand{\glossentry}[1]{%
      \xdef\glscurrententrylabel{\glsdetoklabel{##1}}%
      \gls@org@glossaryentryfield{##1}%
    }%
    \renewcommand{\subglossentry}[2]{%
      \xdef\glscurrententrylabel{\glsdetoklabel{##2}}%
      \gls@org@glossarysubentryfield{##1}{##2}%
    }%
    \@gls@preglossaryhook
    #2%
  \egroup
  \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
  \global\let\warn@noprintglossary\relax
}
```

ractivatenopost Change \nopostdesc and \glsxtrnopostpunc to behave as they do in the glossary.

```
\newcommand*{\glsxtractivatenopost}{%
  \let\nopostdesc\@nopostdesc
  \let\glsxtrnopostpunc\@glsxtr@nopostpunc
}
```

lsxtrnopostpunc

```
\newrobustcmd*{\glsxtrnopostpunc}{}
```

sxtr@nopostpunc Provide a command that works like \nopostdesc but only switches of the punctuation without suppressing the post-description hook.

```
\newcommand{\@glsxtr@nopostpunc}{%
 \let\@@glsxtr@org@postdescription\glspostdescription
 \ifglsnopostdot
   \renewcommand{\glspostdescription}{%
     \glsnopostdottrue
     \let\glspostdescription\@@glsxtr@org@postdescription
     \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
     \glsxtrpostdescription
     \@glsxtr@nopostpunc@postdesc}%
 \else
   \renewcommand{\glspostdescription}{%
     \let\glspostdescription\@@glsxtr@org@postdescription
     \let\glsxtrrestorepostpunc\@glsxtr@restore@postpunc
     \glsxtrpostdescription
     \@glsxtr@nopostpunc@postdesc}%
 \fi
 \glsnopostdotfalse
}
```

stpunc@postdesc

```
\newcommand*{\@glsxtr@nopostpunc@postdesc}{}
```

estore@postpunc

```
\newcommand*{\@glsxtr@restore@postpunc}{%
 \def\@glsxtr@nopostpunc@postdesc{%
   \@glsxtr@org@postdescription
   \let\@glsxtr@nopostpunc@postdesc\@empty
   \let\glsxtrrestorepostpunc\@empty
 }%
}
```

restorepostpunc Does nothing outside of glossary.

```
\newcommand*{\glsxtrrestorepostpunc}{}
```

\@printglossary Redefine.

```
\renewcommand{\@printglossary}[2]{%
  \def\@glsxtr@printglossopts{#1}%
  \@glsxtr@orgprintglossary{#1}{#2}%
}
```

Add a key that switches off the entry targets:

```
\define@choicekey{printgloss}{target}
[\@glsxtr@printglossval\@glsxtr@printglossnr]%
{true,false}[true]%
{%
  \ifcase\@glsxtr@printglossnr
```

```
    \def\@glstarget{\glsdohypertarget}%
  \else
    \let\@glstarget\@secondoftwo
  \fi
}
```

hypernameprefix

```
\newcommand{\@glsxtrhypernameprefix}{}
```

New to v1.20:

```
\define@key{printgloss}{targetnameprefix}{%
  \renewcommand{\@glsxtrhypernameprefix}{#1}%
}
```

```
\define@key{printgloss}{prefix}{%
  \renewcommand{\glolinkprefix}{#1}%
}
```

lsdohypertarget Redefine to insert \@glsxtrhypernameprefix before the target name.

```
\let\@glsxtr@org@glsdohypertarget\glsdohypertarget
\renewcommand{\glsdohypertarget}[2]{%
  \@glsxtr@org@glsdohypertarget{\@glsxtrhypernameprefix#1}{#2}%
}
```

Update \@glstarget to use \def instead being assigned with \let so that it can pick up the new definition and allow any further redefinitions:

```
\ifx\@glstarget\@glsxtr@org@glsdohypertarget
 \def\@glstarget{\glsdohypertarget}%
\fi
%\end{macro}
```

@makeglossaries For the benefit of makeglossaries

```
\newcommand*{\glsxtr@makeglossaries}[1]{}
```

@glsxtr@gettype Get just the type.

```
\def\@glsxtr@gettype#1,type=#2,#3\@end@glsxtr@gettype{%
  \def\@glo@type{#2}%
}
```

@assign@sortkey Assign the sort key.

```
\newcommand\@glsxtr@mixed@assign@sortkey[1]{%
  \edef\@glo@type{\@glo@type}%
  \expandafter\DTLifinlist\expandafter{\@glo@type}{\@glsxtr@reg@glosslist}%
  {%
    \@glo@no@assign@sortkey{#1}%
  }%
  {%
    \@@glo@assign@sortkey{#1}%
  }%
}%
```

Display number list for the regular version:

splaynumberlist

```
\let\@glsxtr@idx@displaynumberlist\glsdisplaynumberlist
```

Display number list for the “noidx” version:

splaynumberlist

```
\newcommand*{\@glsxtr@noidx@displaynumberlist}[1]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \ifdef\@gls@loclist
  {%
    \def\@gls@noidxloclist@sep{%
      \def\@gls@noidxloclist@sep{%
        \def\@gls@noidxloclist@sep{%
          \glsnumlistsep
        }%
        \def\@gls@noidxloclist@finalsep{\glsnumlistlastsep}%
      }%
    }%
    \def\@gls@noidxloclist@finalsep{}%
    \def\@gls@noidxloclist@prev{}%
    \forlistloop{\glsnoidxdisplayloclisthandler}{\@gls@loclist}%
    \@gls@noidxloclist@finalsep
    \@gls@noidxloclist@prev
  }%
  {%

    \glsxtrundeftag
    \glsdoifexists{#1}%
    {%
      \GlossariesWarning{Missing location list for ‘#1’. Either
        a rerun is required or you haven’t referenced the entry.}%
    }%
  }%
}%

```

And for the number list loop:

@numberlistloop

```
\newcommand*{\@glsxtr@noidx@numberlistloop}[3]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \let\@gls@org@glsnoidxdisplayloc\glsnoidxdisplayloc
  \let\@gls@org@glsseeformat\glsseeformat
  \let\glsnoidxdisplayloc#2\relax
  \let\glsseeformat#3\relax
  \ifdef\@gls@loclist
  {%
    \forlistloop{\glsnoidxnumberlistloophandler}{\@gls@loclist}%
  }%
```

```
  {%

    \glsxtrundeftag
    \glsdoifexists{#1}%
    {%
      \GlossariesWarning{Missing location list for ‘##1’. Either
        a rerun is required or you haven’t referenced the entry.}%
    }%
  }%
  \let\glsnoidxdisplayloc\@gls@org@glsnoidxdisplayloc
  \let\glsseeformat\@gls@org@glsseeformat
}%
```

Same for entry number list.

entrynumberlist

```
\newcommand*{\@glsxtr@noidx@entrynumberlist}[1]{%
  \letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
  \ifdef\@gls@loclist
  {%
    \glsnoidxloclist{\@gls@loclist}%
  }%
  {%

    \glsxtrundeftag
    \glsdoifexists{#1}%
    {%
      \GlossariesWarning{Missing location list for ‘#1’. Either
        a rerun is required or you haven’t referenced the entry.}%
    }%
  }%
}%
```

entrynumberlist

```
\newcommand*{\@glsxtr@idx@entrynumberlist}[1]{\glsentrynumberlist{#1}}
```

x@getgrouptitle Patch.

```
\renewcommand*{\@gls@noidx@getgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{#1}%
  \ifdefvoid\@glsxtr@titlelabel
  {}%
  {%
    \protected@edef\@glsxtr@titlelabel{\csuse{glsxtr@grouptitle@#1}}%
  }%
  \ifdefvoid{\@glsxtr@titlelabel}%
  {%
    \DTLifint{#1}%
    {%
      \ifnum#1<256\relax
        \edef#2{\char#1\relax}%
```

```
      \else
        \edef#2{#1}%
      \fi
    }%
    {%
      \ifcsundef{#1groupname}%
      {\def#2{#1}}%
      {\letcs#2{#1groupname}}%
    }%
  }%
  {%
    \let#2\@glsxtr@titlelabel
  }%
}
```

g@getgrouptitle Save original definition of `\@gls@getgrouptitle`

```
\let\glsxtr@org@getgrouptitle\@gls@getgrouptitle
```

trgetgrouptitle Provide a user-level command to fetch the group title. The first argument is the group label. The second argument is a control sequence in which to store the title.

```
\newrobustcmd{\glsxtrgetgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \ifcsdef{\@glsxtr@titlelabel}
  {\letcs{#2}{\@glsxtr@titlelabel}}%
  {\glsxtr@org@getgrouptitle{#1}{#2}}%
}
\let\@gls@getgrouptitle\glsxtrgetgrouptitle
```

trsetgrouptitle Sets the title for the given group label.

```
\newcommand{\glsxtrsetgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \protected@csxdef{\@glsxtr@titlelabel}{#2}%
}
```

alsetgrouptitle As above put only locally defines the title.

```
\newcommand{\glsxtrlocalsetgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \protected@csedef{\@glsxtr@titlelabel}{#2}%
}
```

\glsnavigation Redefine to use new user-level command.

```
\renewcommand*{\glsnavigation}{%
  \def\@gls@between{}%
  \ifcsundef{@gls@hypergrouplist@\@glo@type}%
  {%
```

```
    \def\@gls@list{}%
  }%
  {%
    \expandafter\let\expandafter\@gls@list
      \csname @gls@hypergrouplist@\@glo@type\endcsname
  }%
  \@for\@gls@tmp:=\@gls@list\do{%
    \@gls@between
    \glsxtrgetgrouptitle{\@gls@tmp}{\@gls@grptitle}%
    \glsnavhyperlink{\@gls@tmp}{\@gls@grptitle}%
    \let\@gls@between\glshypernavsep
  }%
}
```

`@noidx@glossary`

```
\renewcommand*{\@print@noidx@glossary}{%
  \ifcsdef{@glsref@\@glo@type}%
  {%
    \ifcsdef{@glo@sortmacro@\@glo@sorttype}%
    {%
      \csuse{@glo@sortmacro@\@glo@sorttype}{\@glo@type}%
    }%
    {%
      \PackageError{glossaries}{Unknown sort handler `\@glo@sorttype'}{}%
    }%
    \glossarysection[\glossarytoctitle]{\glossarytitle}%
    \glossarypreamble
```

Moved this command definition outside of environment in case of scoping issues (e.g. in tabular-like styles).

```
    \def\@gls@currentlettergroup{}%
    \begin{theglossary}%
    \glossaryheader
    \glsresetentrylist
    \forlistcsloop{\@gls@noidx@do}{@glsref@\@glo@type}%
    \end{theglossary}%
    \glossarypostamble
  }%
  {%
```

Add section header if there are actually entries defined in this glossary as the document is likely pending a re-run.

```
    \glsxtrifemptyglossary{\@glo@type}%
    {}%
    {\glossarysection[\glossarytoctitle]{\glossarytitle}}%
    \@gls@noref@warn{\@glo@type}%
  }%
}
```

`noidxdisplayloc` Patch to check for range formations.

```
\renewcommand*{\glsnoidxdisplayloc}[4]{%
  \setentrycounter[#1]{#2}%
  \@glsxtr@display@loc#3\empty\end@glsxtr@display@loc{#4}%
}
```

xtr@display@loc Patch to check for range formations.

```
\def\@glsxtr@display@loc#1#2\end@glsxtr@display@loc#3{%
  \ifx#1(\relax
    \glsxtrdisplaystartloc{#2}{#3}%
  \else
    \ifx#1)\relax
      \glsxtrdisplayendloc{#2}{#3}%
    \else
      \glsxtrdisplaysingleloc{#1#2}{#3}%
    \fi
  \fi
}
```

isplaysingleloc Single location.

```
\newcommand*{\glsxtrdisplaysingleloc}[2]{%
  \csuse{#1}{#2}%
}
```

By default the range identifiers are simply ignored. A custom list loop handler can be defined by the user to test for ranges by checking the definition of `\glsxtrlocrangefmt`.

displaystartloc Start of a location range.

```
\newcommand*{\glsxtrdisplaystartloc}[2]{%
  \edef\glsxtrlocrangefmt{#1}%
  \ifx\glsxtrlocrangefmt\empty
    \def\glsxtrlocrangefmt{glsnumberformat}%
  \fi
  \expandafter\glsxtrdisplaysingleloc
    \expandafter{\glsxtrlocrangefmt}{#2}%
}
```

trdisplayendloc End of a location range.

```
\newcommand*{\glsxtrdisplayendloc}[2]{%
  \edef\@glsxtr@tmp{#1}%
  \ifdefempty{\@glsxtr@tmp}{\def\@glsxtr@tmp{glsnumberformat}}{}%
  \ifx\glsxtrlocrangefmt\@glsxtr@tmp
  \else
    \GlossariesExtraWarning{Mismatched end location range
      (start=\glsxtrlocrangefmt, end=\@glsxtr@tmp)}%
  \fi
  \expandafter\glsxtrdisplayendlochook\expandafter{\@glsxtr@tmp}{#2}%
  \expandafter\glsxtrdisplaysingleloc
    \expandafter{\glsxtrlocrangefmt}{#2}%
  \def\glsxtrlocrangefmt{}%
}
```

splayendlochook Allow the user to hook into the end of range command.

```
\newcommand*{\glsxtrdisplayendlochook}[2]{}
```

sxtrlocrangefmt Current range format. Empty if not in a range.

```
\newcommand*{\glsxtrlocrangefmt}{}
```

setentrycounter Adjust \setentrycounter to save the original prefix.

```
\renewcommand*{\setentrycounter}[2][]{%
  \def\glsxtrcounterprefix{#1}%
  \ifx\glsxtrcounterprefix\@empty
    \def\@glo@counterprefix{.}%
  \else
    \def\@glo@counterprefix{.#1.}%
  \fi
  \def\glsentrycounter{#2}%
}
```

ls@removespaces Redefine to allow adjustments to location hyperlink.

```
\def\@gls@removespaces#1 #2\@nil{%
 \toks@=\expandafter{\the\toks@#1}%
 \ifx\\#2\\%
   \edef\x{\the\toks@}%
   \ifx\x\empty
   \else
```

Expand location (just in case \toks@ is needed for something else).

```
     \expandafter\glsxtrlocationhyperlink\expandafter
      \glsentrycounter\expandafter\@glo@counterprefix\expandafter{\the\toks@}%
   \fi
 \else
   \@gls@ReturnAfterFi{%
     \@gls@removespaces#2\@nil
   }%
 \fi
}
```

ocationhyperlink \glsxtrlocationhyperlink{⟨*counter*⟩}{⟨*prefix*⟩}{⟨*location*⟩}

```
\newcommand*{\glsxtrlocationhyperlink}[3]{%
  \ifdefvoid\glsxtrsupplocationurl
  {%
    \GlsXtrInternalLocationHyperlink{#1}{#2}{#3}%
  }%
  {%
    \hyperref{\glsxtrsupplocationurl}{}{#1#2#3}{#3}%
  }%
}
```

supphypernumber

```
\newcommand*{\glsxtrsupphypernumber}[1]{%
 {%
   \glshasattribute{\glscurrententrylabel}{externallocation}%
   {%
     \def\glsxtrsupplocationurl{%
       \glsgetattribute{\glscurrententrylabel}{externallocation}}%
   }%
   {%
     \def\glsxtrsupplocationurl{}%
   }%
   \glshypernumber{#1}%
 }%
}
```

Give a bit of assistance to new users who are confused and don't know how to read transcript messages.

@print@glossary

```
\renewcommand{\@print@glossary}{%
  \makeatletter
  \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
  \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
  {}%
  {\glsxtrNoGlossaryWarning{\@glo@type}}%
  \ifglsxindy
    \ifcsundef{@xdy@\@glo@type @language}%
    {%
      \edef\@do@auxoutstuff{%
        \noexpand\AtEndDocument{%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\providecommand\string\@xdylanguage[2]{}}%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\@xdylanguage{\@glo@type}{\@xdy@main@language}}%
        }%
      }%
    }%
    {%
      \edef\@do@auxoutstuff{%
        \noexpand\AtEndDocument{%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\providecommand\string\@xdylanguage[2]{}}%
          \noexpand\immediate\noexpand\write\@auxout{%
            \string\@xdylanguage{\@glo@type}{\csname @xdy@\@glo@type
              @language\endcsname}}%
        }%
      }%
    }%
    \@do@auxoutstuff
```

```
    \edef\@do@auxoutstuff{%
      \noexpand\AtEndDocument{%
         \noexpand\immediate\noexpand\write\@auxout{%
          \string\providecommand\string\@gls@codepage[2]{}}%
         \noexpand\immediate\noexpand\write\@auxout{%
          \string\@gls@codepage{\@glo@type}{\gls@codepage}}%
      }%
    }%
    \@do@auxoutstuff
  \fi
  \renewcommand*{\@warn@nomakeglossaries}{%
    \GlossariesWarningNoLine{\string\makeglossaries\space
    hasn't been used,^^Jthe glossaries will not be updated}%
  }%
}
```

Setup the warning text to display if the external file for the given glossary is missing.

oGlsWarningHead Header message.

```
\newcommand{\GlsXtrNoGlsWarningHead}[2]{%
 This document is incomplete. The external file associated with
 the glossary '#1' (which should be called \texttt{#2})
 hasn't been created.%
}
```

rningEmptyStart No entries have been added to the glossary.

```
\newcommand{\GlsXtrNoGlsWarningEmptyStart}{%
  This has probably happened because there are no entries defined
  in this glossary.%
}
```

arningEmptyMain The default "main" glossary is empty.

```
\newcommand{\GlsXtrNoGlsWarningEmptyMain}{%
 If you don't want this glossary,
 add \texttt{nomain} to your package option list when you load
 \texttt{glossaries-extra.sty}. For example:%
}
```

ingEmptyNotMain A glossary that isn't the default "main" glossary is empty.

```
\newcommand{\GlsXtrNoGlsWarningEmptyNotMain}[1]{%
 Did you forget to use \texttt{type=#1} when you defined your
 entries? If you tried to load entries into this glossary with
 \texttt{\string\loadglsentries} did you remember to use
 \texttt{[#1]} as the optional argument? If you did, check that
 the definitions in the file you loaded all had the type set
 to \texttt{\string\glsdefaulttype}.%
}
```

arningCheckFile Advisory message to check the file contents.

```
\newcommand{\GlsXtrNoGlsWarningCheckFile}[1]{%
  Check the contents of the file \texttt{#1}. If
  it's empty, that means you haven't indexed any of your entries in this
  glossary (using commands like \texttt{\string\gls} or
  \texttt{\string\glsadd}) so this list can't be generated.
  If the file isn't empty, the document build process hasn't been
  completed.%
}
```

WarningAutoMake Message when automake option has been used.

```
\newcommand{\GlsXtrNoGlsWarningAutoMake}[1]{%
  You may need to rerun \LaTeX. If you already have, it may be that
  \TeX's shell escape doesn't allow you to run
  \ifglsxindy xindy\else makeindex\fi. Check the
  transcript file \texttt{\jobname.log}. If the shell escape is
  disabled, try one of the following:

  \begin{itemize}
    \item Run the external (Lua) application:

       \texttt{makeglossaries-lite.lua \string"\jobname\string"}

    \item Run the external (Perl) application:

       \texttt{makeglossaries \string"\jobname\string"}
  \end{itemize}

  Then rerun \LaTeX\ on this document.
  \GlossariesExtraWarning{Rerun required to build the
  glossary `#1' or check TeX's shell escape allows
  you to run \ifglsxindy xindy\else makeindex\fi}%
}
```

WarningMisMatch Mismatching \makenoidxglossaries.

```
\newcommand{\GlsXtrNoGlsWarningMisMatch}{%
  You need to either replace \texttt{\string\makenoidxglossaries}
  with \texttt{\string\makeglossaries} or replace
  \texttt{\string\printglossary} (or \texttt{\string\printglossaries}) with
  \texttt{\string\printnoidxglossary}
  (or \texttt{\string\printnoidxglossaries}) and then rebuild
  this document.%
}
```

arningBuildInfo Build advice.

```
\newcommand{\GlsXtrNoGlsWarningBuildInfo}{%
  Try one of the following:
  \begin{itemize}
    \item Add \texttt{automake} to your package option list when you load
```

```
          \texttt{glossaries-extra.sty}. For example:

          \texttt{\string\usepackage[automake]%
              \glsopenbrace glossaries-extra\glsclosebrace}

    \item Run the external (Lua) application:

          \texttt{makeglossaries-lite.lua \string"\jobname\string"}

    \item Run the external (Perl) application:

          \texttt{makeglossaries \string"\jobname\string"}
  \end{itemize}

  Then rerun \LaTeX\ on this document.%
}
```

trRecordWarning Paragraph for record=only.

```
\newcommand{\GlsXtrRecordWarning}[1]{%
 \texttt{\string\printglossary} doesn't work
 with the \texttt{record=only} package option
 use\par\texttt{\string\printunsrtglossary[type=#1]}\par
 instead (or change the package option).%
}
```

oGlsWarningTail Final paragraph.

```
\newcommand{\GlsXtrNoGlsWarningTail}{%
 This message will be removed once the problem has been fixed.%
}
```

GlsWarningNoOut No out file created. Build advice.

```
\newcommand{\GlsXtrNoGlsWarningNoOut}[1]{%
  The file \texttt{#1} doesn't exist. This most likely means you haven't used
  \texttt{\string\makeglossaries} or you have used
  \texttt{\string\nofiles}. If this is just a draft version of the
  document, you can suppress this message using the
  \texttt{nomissingglstext} package option.%
}
```

glossarywarning

```
\newcommand*{\@glsxtr@defaultnoglossarywarning}[1]{%
 \glossarysection[\glossarytoctitle]{\glossarytitle}
 \GlsXtrNoGlsWarningHead{#1}{\jobname.\csname @glotype@\@glo@type @in\endcsname}
 \par
 \glsxtrifemptyglossary{#1}%
 {%
    \GlsXtrNoGlsWarningEmptyStart\space
    \ifthenelse{\equal{#1}{main}}{\GlsXtrNoGlsWarningEmptyMain\par
    \medskip
```

```
    \noindent\texttt{\string\usepackage[nomain\ifglsacronym ,acronym\fi]%
        \glsopenbrace glossaries-extra\glsclosebrace}
    \medskip
    }%
    {\GlsXtrNoGlsWarningEmptyNotMain{#1}}%
 }%
 {%
   \IfFileExists{\jobname.\csname @glotype@\@glo@type @out\endcsname}
   {%
     \GlsXtrNoGlsWarningCheckFile
       {\jobname.\csname @glotype@\@glo@type @out\endcsname}

     \ifglsautomake

      \GlsXtrNoGlsWarningAutoMake{#1}

     \else

        \ifthenelse{\equal{#1}{main}}%
        {%
          \GlsXtrNoGlsWarningEmptyMain\par
          \medskip
          \noindent\texttt{\string\usepackage[nomain]%
            \glsopenbrace glossaries-extra\glsclosebrace}
          \medskip
        }%
        {}%

        \ifdefequal\makeglossaries\@no@makeglossaries
        {%
          \GlsXtrNoGlsWarningMisMatch
        }%
        {%
          \GlsXtrNoGlsWarningBuildInfo
        }%
     \fi
   }%
   {%
     \GlsXtrNoGlsWarningNoOut
       {\jobname.\csname @glotype@\@glo@type @out\endcsname}%
   }%
 }%
 \par
 \GlsXtrNoGlsWarningTail
}
```

glossarywarning Warn about using \printglossary with record

```
\newcommand*{\@glsxtr@record@noglossarywarning}[1]{%
  \GlossariesExtraWarning{\string\printglossary\space doesn't work\MessageBreak
```

```
  with record=only package option\MessageBreak(use
  \string\printunsrtglossary[type=#1])\MessageBreak
  instead (or change the package option)}%
 \glossarysection[\glossarytoctitle]{\glossarytitle}
 \GlsXtrRecordWarning{#1}
 \GlsXtrNoGlsWarningTail
}
```

Provide some commands to accompany the record option for use with bib2gls.

`xtrresourcefile` Since it's dangerous for an external application to create a file with a .tex extension, as from v1.11 this enforces a .glstex extension to avoid conflict.

```
\newcommand*{\glsxtrresourcefile}[2][]{%
```

The record option can't be set after this command.

```
  \disable@keys{glossaries-extra.sty}{record}%
  \glsxtr@writefields
  \protected@write\@auxout{\glsxtrresourceinit}{\string\glsxtr@resource{#1}{#2}}%
  \let\@glsxtr@org@see@noindex\@gls@see@noindex
  \let\@gls@see@noindex\relax
  \IfFileExists{#2.glstex}%
  {%
```

Can't scope `\@input` so save and restore the category code of `@` to allow for internal commands in the location list.

```
    \edef\@bibgls@restoreat{\noexpand\catcode\noexpand`\noexpand\@=\number\catcode`\@}%
    \makeatletter
    \@input{#2.glstex}%
    \@bibgls@restoreat
```

If the record=nameref option has been set, check if this is supported by the installed version of `bib2gls`.

```
    \@glsxtr@check@bibgls@nameref
  }%
  {%
    \GlossariesExtraWarning{No file `#2.glstex'}%
  }%
  \let\@gls@see@noindex\@glsxtr@org@see@noindex
}
\@onlypreamble\glsxtrresourcefile
```

`@bibgls@nameref` This will only warn after `bib2gls` has created the .glstex file, but there's way to check before.

```
\newcommand{\@glsxtr@check@bibgls@nameref}{%
  \ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
    \ifdef\bibglshrefchar
    {}%
    {%
      \GlossariesExtraWarning{record=nameref requires at least
      version 1.8 of bib2gls}%
    }%
```

```
  \fi
  \let\@glsxtr@check@bibgls@nameref\relax
}
```

xtrresourceinit Code used during the protected write operation.

```
\newcommand*{\glsxtrresourceinit}{}
```

trresourcecount

```
\newcount\glsxtrresourcecount
```

trLoadResources Short cut that uses \glsxtrresourcefile with \jobname as the mandatory argument.

```
\newcommand*{\GlsXtrLoadResources}[1][]{%
  \ifnum\glsxtrresourcecount=0\relax
    \glsxtrresourcefile[#1]{\jobname}%
  \else
    \glsxtrresourcefile[#1]{\jobname-\the\glsxtrresourcecount}%
  \fi
  \advance\glsxtrresourcecount by 1\relax
}
```

glsxtr@resource

```
\newcommand*{\glsxtr@resource}[2]{}
```

\glsxtr@fields

```
\newcommand*{\glsxtr@fields}[1]{}
```

xtr@texencoding

```
\newcommand*{\glsxtr@texencoding}[1]{}
```

\glsxtr@langtag

```
\newcommand*{\glsxtr@langtag}[1]{}
```

@pluralsuffixes

```
\newcommand*{\glsxtr@pluralsuffixes}[4]{}
```

tr@shortcutsval

```
\newcommand*{\glsxtr@shortcutsval}[1]{}
```

sxtr@linkprefix

```
\newcommand*{\glsxtr@linkprefix}[1]{}
```

xtr@writefields This information only needs to be written once, so disable it after it's been used.

```
\newcommand*{\glsxtr@writefields}{%
```

```
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@fields}[1]{}}%
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@resource}[2]{}}%
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@pluralsuffixes}[4]{}}%
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@shortcutsval}[1]{}}%
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@linkprefix}[1]{}}%
\protected@write\@auxout{}{\string\glsxtr@fields{\@gls@keymap}}%
\protected@write\@auxout{}%
 {\string\providecommand*{\string\glsxtr@record}[5]{}}%
\ifx\@glsxtr@record@setting\@glsxtr@record@setting@nameref
  \protected@write\@auxout{}%
   {\string\providecommand*{\string\glsxtr@record@nameref}[8]{}}%
\fi
```

If any languages have been loaded, the language tag will be available in `\CurrentTrackedLanguageTag` (provided by tracklang). For multilingual documents, the required locale will have to be indicated in the `sort` key when using `\glsxtrresourcefile`.

```
\ifdef\CurrentTrackedLanguageTag
{%
   \protected@write\@auxout{}{%
     \string\glsxtr@langtag{\CurrentTrackedLanguageTag}}%
}%
{}%
\protected@write\@auxout{}{\string\glsxtr@pluralsuffixes
  {\glspluralsuffix}{\abbrvpluralsuffix}{\acrpluralsuffix}%
  {\glsxtrabbrvpluralsuffix}}%
\ifdef\inputencodingname
{%
   \protected@write\@auxout{}{\string\glsxtr@texencoding{\inputencodingname}}%
}%
{%
```

If fontspec has been loaded, assume UTF-8. (The encoding can be changed with `\XeTeXinputencoding`, but I can't work out how to determine the current encoding.)

```
   \@ifpackageloaded{fontspec}%
   {\protected@write\@auxout{}{\string\glsxtr@texencoding{utf8}}}%
   {}%
}%
\protected@write\@auxout{}{\string\glsxtr@shortcutsval{\@glsxtr@shortcutsval}}%
```

Prefix deferred until the beginning of the document in case it's redefined later in the preamble. This is picked up by bib2gls when the external option is used.

```
\AtBeginDocument
  {\protected@write\@auxout{}{\string\glsxtr@linkprefix{\glolinkprefix}}}%
\let\glsxtr@writefields\relax
```

If the automake option is on, try running bib2gls if the aux file exists. The double-quotes around \jobname have been removed (v1.19) since \jobname will include double-quotes if the file name has spaces.

```
  \ifglsautomake
    \IfFileExists{\jobname.aux}%
    {\immediate\write18{bib2gls \jobname}}{}%
```

If \makeglossaries is also used, allow makeindex/xindy to also be run, otherwise disable the error message about requiring \makeglossaries with automake=true.

```
    \ifx\@gls@doautomake\@gls@doautomake@err
       \let\@gls@doautomake\relax
    \fi
  \fi
}
```

do@automake@err

```
\newcommand*{\@gls@doautomake@err}{%
  \PackageError{glossaries}{You must use
  \string\makeglossaries\space with automake=true}
  {%
     Either remove the automake=true setting or
     add \string\makeglossaries\space to your document preamble.%
  }%
}
```

Allow locations specific to a particular counter to be recorded.

\glsxtr@record

```
\newcommand*{\glsxtr@record}[5]{}
```

@record@nameref Used with record=nameref to include current label information.

```
\newcommand*{\glsxtr@record@nameref}[8]{}
```

r@counterrecord Aux file command.

```
\newcommand*{\glsxtr@counterrecord}[3]{%
  \glsxtrfieldlistgadd{#1}{record.#2}{#3}%
}
```

unterrecordhook Hook used by \@glsxtr@dorecord.

```
\newcommand*{\@glsxtr@counterrecordhook}{}
```

trRecordCounter Activate recording for a particular counter (identified in the argument).

```
\newcommand*{\GlsXtrRecordCounter}[1]{%
  \@@glsxtr@recordcounter{#1}%
}
\@onlypreamble\GlsXtrRecordCounter
```

docounterrecord

```
\newcommand*{\@glsxtr@docounterrecord}[1]{%
  \protected@write\@auxout{}{\string\glsxtr@counterrecord
    {\@gls@label}{#1}{\csuse{the#1}}}%
}
```

lsxtrglossentry Users may prefer to have entries displayed throughout the document rather than gathered together in a list. This command emulates the way \glossentry behaves (without the style formatting commands like \item). This needs to define \currentglossary to the current glossary type (normally set at the start of \@printglossary) and needs to define \glscurrententrylabel to the entry's label (normally set before \glossentry and \subglossentry). This needs some protection in case it's used in a section heading.

```
\newcommand*{\glsxtrglossentry}[1]{%
  \glsxtrtitleorpdforheading
  {\@glsxtrglossentry{#1}}%
  {\glsentryname{#1}}%
  {\glsxtrheadname{#1}}%
}
```

lsxtrglossentry Another test is needed in case \@glsxtrglossentry has been written to the table of contents.

```
\newrobustcmd*{\@glsxtrglossentry}[1]{%
  \glsxtrtitleorpdforheading
  {%
    \glsdoifexists{#1}%
    {%
      \begingroup
        \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
        \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
        \ifglshasparent{#1}%
        {\GlsXtrStandaloneSubEntryItem{#1}}%
        {\glsentryitem{#1}}%
        \GlsXtrStandaloneEntryName{#1}%
      \endgroup
    }%
  }%
  {\glsentryname{#1}}%
  {\glsxtrheadname{#1}}%
}
```

daloneEntryName

```
\newcommand*{\GlsXtrStandaloneEntryName}[1]{%
  \glstarget{#1}{\glossentryname{#1}}%
}
```

oneGlossaryType To make it easier to adjust the definition of \currentglossary within \glsxtrglossentry, this expands to the default definition. (If redefined, it must fully expand to the appropriate label.)

```
\newcommand{\GlsXtrStandaloneGlossaryType}{\glsentrytype{\glscurrententrylabel}}
```

oneSubEntryItem Used for sub-entries in standalone format. The argument is the entry's label.

```
\newcommand*{\GlsXtrStandaloneSubEntryItem}[1]{%
  \GlsXtrIfFieldEqNum{level}{#1}{1}{\glssubentryitem{#1}}{}%
}
```

glossentryother As \glsxtrglossentry but uses a different field. First argument is code to use in the header. The second argument is the entry's label. The third argument is the internal field label. This needs to be expandable in case it occurs in a sectioning command so it can't have an optional argument.

```
\newcommand*{\glsxtrglossentryother}[3]{%
  \ifstrempty{#1}%
  {%
    \ifcsdef{glsxtrhead#3}%
    {%
      \glsxtrtitleorpdforheading
      {\@glsxtrglossentryother{#2}{#3}{#1}}%
      {\@gls@entry@field{#2}{#3}}%
      {\csuse{glsxtrhead#3}{#2}}%
    }%
    {%
      \glsxtrtitleorpdforheading
      {\@glsxtrglossentryother{#2}{#3}{#1}}%
      {\@gls@entry@field{#2}{#3}}%
      {\@gls@entry@field{\NoCaseChange{#2}}{#3}}%
    }%
  }%
  {%
    \glsxtrtitleorpdforheading
    {\@glsxtrglossentryother{#2}{#3}{#1}}%
    {\@gls@entry@field{#2}{#3}}%
    {#1}%
  }%
}
```

glossentryother As \@glsxtrglossentry but uses a different field.

```
\newrobustcmd*{\@glsxtrglossentryother}[3]{%
  \glsxtrtitleorpdforheading
  {%
    \glsdoifexists{#1}%
    {%
      \begingroup
        \edef\glscurrententrylabel{\glsdetoklabel{#1}}%
        \edef\currentglossary{\GlsXtrStandaloneGlossaryType}%
        \ifglshasparent{#1}%
        {\GlsXtrStandaloneSubEntryItem{#1}}%
        {\glsentryitem{#1}}%
        \GlsXtrStandaloneEntryOther{#1}%
```

```
      \endgroup
    }%
  }%
  {\@gls@entry@field{#1}{#2}}%
  {#3}%
}
```

aloneEntryOther

```
\newcommand*{\GlsXtrStandaloneEntryOther}[2]{%
  \glstarget{#1}{\glossentrynameother{#1}{#2}}%
}
```

ntunsrtglossary Similar to \printnoidxglossary but it displays all entries defined for the given glossary without sorting.

```
\newcommand*{\printunsrtglossary}{%
  \@ifstar\s@printunsrtglossary\@printunsrtglossary
}
```

ntunsrtglossary Unstarred version.

```
\newcommand*{\@printunsrtglossary}[1][]{%
  \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
}
```

ntunsrtglossary Starred version.

```
\newcommand*{\s@printunsrtglossary}[2][]{%
  \begingroup
    #2%
    \@printglossary{type=\glsdefaulttype,#1}{\@print@unsrt@glossary}%
  \endgroup
}
```

unsrtglossaries Similar to \printnoidxglossaries but it displays all entries defined for the given glossary without sorting.

```
\newcommand*{\printunsrtglossaries}{%
  \forallglossaries{\@@glo@type}{\printunsrtglossary[type=\@@glo@type]}%
}
```

@unsrt@glossary

```
\newcommand*{\@print@unsrt@glossary}{%
  \glossarysection[\glossarytoctitle]{\glossarytitle}%
  \glossarypreamble
```

check for empty list

```
  \glsxtrifemptyglossary{\@glo@type}%
  {%
    \GlossariesExtraWarning{No entries defined in glossary `\@glo@type'}%
  }%
  {%
```

```
    \key@ifundefined{glossentry}{group}%
    {\let\@gls@getgrouptitle\@gls@noidx@getgrouptitle}%
    {\let\@gls@getgrouptitle\@glsxtr@unsrt@getgrouptitle}%
    \def\@gls@currentlettergroup{}%
```

A loop within the tabular-like styles can cause problems, so move the loop outside.

```
    \def\@glsxtr@doglossary{%
      \begin{theglossary}%
      \glossaryheader
      \glsresetentrylist
    }%
    \expandafter\@for\expandafter\glscurrententrylabel\expandafter
      :\expandafter=\csname glolist@\@glo@type\endcsname\do{%
      \ifdefempty{\glscurrententrylabel}
      {}%
      {%
```

Provide a hook (for example to measure width).

```
        \let\glsxtr@process\@firstofone
        \let\printunsrtglossaryskipentry
            \@glsxtr@printunsrtglossaryskipentry
        \printunsrtglossaryentryprocesshook{\glscurrententrylabel}%
```

Don't check group for child entries.

```
        \glsxtr@process
        {%
          \ifglshasparent{\glscurrententrylabel}{}%
          {%
            \@glsxtr@checkgroup\glscurrententrylabel
            \expandafter\appto\expandafter\@glsxtr@doglossary\expandafter
              {\@glsxtr@groupheading}%
          }%
          \eappto\@glsxtr@doglossary{%
            \noexpand\@printunsrt@glossary@handler{\glscurrententrylabel}}%
        }%
      }%
    }%
    \appto\@glsxtr@doglossary{\end{theglossary}}%
    \printunsrtglossarypredoglossary
    \@glsxtr@doglossary
  }%
  \glossarypostamble
}
```

ntryprocesshook

```
\newcommand*{\printunsrtglossaryentryprocesshook}[1]{}
```

ossaryskipentry

```
\newcommand*{\printunsrtglossaryskipentry}{%
  \PackageError{glossaries-extra}{\string\printunsrtglossaryskipentry\space
```

```
can only be used within \string\printunsrtglossaryentryprocesshook}{}%
}

ntryprocesshook

\newcommand*{\@glsxtr@printunsrtglossaryskipentry}{%
  \let\glsxtr@process\@gobble
}

rypredoglossary

\newcommand*{\printunsrtglossarypredoglossary}{}

lossary@handler

\newcommand{\@printunsrt@glossary@handler}[1]{%
  \xdef\glscurrententrylabel{#1}%
  \printunsrtglossaryhandler\glscurrententrylabel
}

glossaryhandler

\newcommand{\printunsrtglossaryhandler}[1]{%
  \glsxtrunsrtdo{#1}%
}
```

xtriflabelinlist

\glsxtriflabelinlist{⟨*label*⟩}{⟨*list*⟩}{⟨*true*⟩}{⟨*false*⟩}

Might be useful for the handler to check if an entry label or category label is contained in a list, so provide a user-level version of \@gls@ifinlist which ensures the label and list are fully expanded.

```
\newrobustcmd*{\glsxtriflabelinlist}[4]{%
 \protected@edef\@glsxtr@doiflabelinlist{\noexpand\@gls@ifinlist{#1}{#2}}%
 \@glsxtr@doiflabelinlist{#3}{#4}%
}
```

srtglossaryunit

```
\newcommand{\print@op@unsrtglossaryunit}[2][]{%
  \s@printunsrtglossary[type=\glsdefaulttype,#1]{%
    \printunsrtglossaryunitsetup{#2}%
  }%
}

ossaryunitsetup

\newcommand*{\printunsrtglossaryunitsetup}[1]{%
  \renewcommand{\printunsrtglossaryhandler}[1]{%
    \glsxtrfieldxifinlist{##1}{record.#1}{\csuse{the#1}}
    {\glsxtrunsrtdo{##1}}%
    {}%
  }%
```

Only the target names should have the prefixes adjusted as \gls etc need the original \glolinkprefix. The \@gobble part discards \glolinkprefix.

```
  \ifcsundef{theH#1}%
  {%
    \renewcommand*{\@glsxtrhypernameprefix}{record.#1.\csuse{the#1}.\@gobble}%
  }%
  {%
    \renewcommand*{\@glsxtrhypernameprefix}{record.#1.\csuse{theH#1}.\@gobble}%
  }%
  \renewcommand*{\glossarysection}[2][]{}%
  \appto\glossarypostamble{\glspar\medskip\glspar}%
}
```

srtglossaryunit

```
\newcommand{\print@noop@unsrtglossaryunit}[2][]{%
  \PackageError{glossaries-extra}{\string\printunsrtglossaryunit\space
   requires the record=only or record=alsoindex package option}{}%
}
```

t@getgrouptitle

```
\newrobustcmd*{\@glsxtr@unsrt@getgrouptitle}[2]{%
  \protected@edef\@glsxtr@titlelabel{glsxtr@grouptitle@#1}%
  \@onelevel@sanitize\@glsxtr@titlelabel
  \ifcsdef{\@glsxtr@titlelabel}
  {\letcs{#2}{\@glsxtr@titlelabel}}%
  {\def#2{#1}}%
}
```

\glsxtrunsrtdo Provide a user-level call to \@glsxtr@noidx@do to make it easier to define a new handler.

```
\newcommand{\glsxtrunsrtdo}{\@glsxtr@noidx@do}
```

lsxtrgroupfield bib2gls provides a supplementary field labelled secondarygroup for secondary glossaries, so provide a way of switching to that field. (The group key still needs checking. There's no associated key with the internal field).

```
\newcommand*{\glsxtrgroupfield}{group}
```

The tabular-like glossary styles cause quite a problem with the iterative approach. In particular for the group skip. To compensate for this, the groups are now determined while \@glsxtr@doglossary is being constructed rather than in the handler.

sxtr@checkgroup The argument is the entry's label. (This block of code was formerly in \@glsxtr@noidx@do.) Now that this is no longer within a tabular environment, the global definitions aren't needed. The result is now stored in \@glsxtr@groupheading, which will be empty if no heading is required.

```
\newcommand*{\@glsxtr@checkgroup}[1]{%
  \def\@glsxtr@groupheading{}%
  \key@ifundefined{glossentry}{group}%
  {%
```

```
    \letcs{\@gls@sort}{glo@\glsdetoklabel{#1}@sort}%
    \expandafter\glo@grabfirst\@gls@sort{}{}\@nil
  }%
  {%

    \protected@edef\@glo@thislettergrp{%
        \csuse{glo@\glsdetoklabel{#1}@\glsxtrgroupfield}}%
  }%
  \ifdefequal{\@glo@thislettergrp}{\@gls@currentlettergroup}%
  {}%
  {%
    \ifdefempty{\@gls@currentlettergroup}{}%
    {\def\@glsxtr@groupheading{\glsgroupskip}}%
    \eappto\@glsxtr@groupheading{%
      \noexpand\glsgroupheading{\expandonce\@glo@thislettergrp}%
    }%
  }%
  \let\@gls@currentlettergroup\@glo@thislettergrp
}
```

`trLocationField` Stores the internal name of the location field.

```
\newcommand*{\GlsXtrLocationField}{location}
```

`glsxtr@noidx@do` Minor modification of `\@gls@noidx@do` to check for location field if present, but also need to check for the group field.

```
\newcommand{\@glsxtr@noidx@do}[1]{%
  \ifglsentryexists{#1}%
  {%
    \global\letcs{\@gls@loclist}{glo@\glsdetoklabel{#1}@loclist}%
    \global\letcs{\@gls@location}{glo@\glsdetoklabel{#1}@\GlsXtrLocationField}%
    \ifglshasparent{#1}%
    {%
      \gls@level=\csuse{glo@\glsdetoklabel{#1}@level}\relax
      \ifdefvoid{\@gls@location}%
      {%
        \ifdefvoid{\@gls@loclist}%
        {%
          \subglossentry{\gls@level}{#1}{}%
        }%
        {%
          \subglossentry{\gls@level}{#1}%
          {%
            \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
          }%
        }%
      }%
      {%
        \subglossentry{\gls@level}{#1}{\glossaryentrynumbers{\@gls@location}}%
      }%
```

```
    }%
    {%

      \ifdefvoid{\@gls@location}%
      {%
        \ifdefvoid{\@gls@loclist}
        {%
          \glossentry{#1}{}%
        }%
        {%
          \glossentry{#1}%
          {%
            \glossaryentrynumbers{\glsnoidxloclist{\@gls@loclist}}%
          }%
        }%
      }%
      {%
        \glossentry{#1}%
        {%
          \glossaryentrynumbers{\@gls@location}%
        }%
      }%
    }%
  }%
  {}%
}
```

Provide a way to conveniently define commands that behaves like `\gls` with a label prefix.

It's possible that the user might want minor variations with the same prefix but different default options, so use a counter to provide unique inner commands.

`\glsxtrnewgls`

```
\newcount\@glsxtrnewgls@inner
```

(The default options supplied in ⟨*options*⟩ below could possibly be used to form the inner control sequence name to help make it unique, but it might feasibly contain thevalue where the value might contain commands.)

`r@providenewgls`

```
\newcommand*{\@glsxtr@providenewgls}{%
  \protected@write\@auxout{}{\string\providecommand{\string\@glsxtr@newglslike}[2]{}}%
  \let\@glsxtr@providenewgls\relax
}
```

`identifyglslike` Identify the command given in the second argument for the benefit of bib2gls.

```
\newcommand{\glsxtridentifyglslike}[2]{%
 \ifdefequal\@glsxtr@record@setting\@glsxtr@record@setting@off
 {}%
 {%
```

```
    \@glsxtr@providenewgls
    \protected@write\@auxout{}{\string\@glsxtr@newglslike{#1}{\string#2}}%
  }%
}
```

\@glsxtrnewgls `\glsxtrnewgls[⟨options⟩]{⟨prefix⟩}{⟨cs⟩}{⟨inner cs name⟩}`

```
\newcommand*{\@glsxtrnewgls}[4]{%
  \ifdef{#3}%
  {%
    \PackageError{glossaries-extra}{Command \string#3\space already
defined}{}%
  }%
  {%
```

Write information to the aux file for bib2gls.

```
    \glsxtridentifyglslike{#2}{#3}%
    \ifcsdef{@#4like@#2}%
    {%
      \advance\@glsxtrnewgls@inner by \@ne
      \def\@glsxtrnewgls@innercsname{@#4like\number\@glsxtrnewgls@inner @#2}%
    }%
    {\def\@glsxtrnewgls@innercsname{@#4like@#2}}%
    \expandafter\newrobustcmd\expandafter*\expandafter
     #3\expandafter{\expandafter\@gls@hyp@opt\csname\@glsxtrnewgls@innercsname\endcsname}%
    \ifstrempty{#1}%
    {%
      \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
        \new@ifnextchar[%
         {\csname @#4@\endcsname{##1}{#2##2}}%
         {\csname @#4@\endcsname{##1}{#2##2}[]}%
      }%
    }%
    {%
      \expandafter\newcommand\expandafter*\csname\@glsxtrnewgls@innercsname\endcsname[2][]{%
        \new@ifnextchar[%
         {\csname @#4@\endcsname{#1,##1}{#2##2}}%
         {\csname @#4@\endcsname{#1,##1}{#2##2}[]}%
      }%
    }%
  }%
}
```

\glsxtrnewgls `\glsxtrnewgls[⟨options⟩]{⟨prefix⟩}{⟨cs⟩}`

The first argument prepends to the options and the second argument is the prefix.

```
\newrobustcmd*{\glsxtrnewgls}[3][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{gls}%
}
```

lsxtrnewglslike Provide a way to conveniently define commands that behave like \gls, \glspl, \Gls and \Glspl with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
\newrobustcmd*{\glsxtrnewglslike}[6][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{gls}%
  \@glsxtrnewgls{#1}{#2}{#4}{glspl}%
  \@glsxtrnewgls{#1}{#2}{#5}{Gls}%
  \@glsxtrnewgls{#1}{#2}{#6}{Glspl}%
}
```

lsxtrnewGLSlike Provide a way to conveniently define commands that behave like \GLS, \GLSpl with a label prefix. The first argument prepends to the options and the second argument is the prefix.

```
\newrobustcmd*{\glsxtrnewGLSlike}[4][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{GLS}%
  \@glsxtrnewgls{#1}{#2}{#4}{GLSpl}%
}
```

\glsxtrnewrgls As \glsxtrnewgls but for \rgls.

```
\newrobustcmd*{\glsxtrnewrgls}[3][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
}
```

sxtrnewrglslike As \glsxtrnewglslike but for \rgls etc.

```
\newrobustcmd*{\glsxtrnewrglslike}[6][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{rgls}%
  \@glsxtrnewgls{#1}{#2}{#4}{rglspl}%
  \@glsxtrnewgls{#1}{#2}{#5}{rGls}%
  \@glsxtrnewgls{#1}{#2}{#6}{rGlspl}%
}
```

sxtrnewrGLSlike As \glsxtrnewGLSlike but for \rGLS etc.

```
\newrobustcmd*{\glsxtrnewrGLSlike}[4][]{%
  \@glsxtrnewgls{#1}{#2}{#3}{rGLS}%
  \@glsxtrnewgls{#1}{#2}{#4}{rGLSpl}%
}
```

Provide easy access to record count fields.

otalRecordCount Access total record count. This is designed to be expandable. The argument is the label.

```
\newcommand*{\GlsXtrTotalRecordCount}[1]{%
 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount}%
 {\csname glo@\glsdetoklabel{#1}@recordcount\endcsname}%
 {0}%
}
```

sXtrRecordCount Access record count for a particular counter. The first argument is the label. The second argument is the counter name.

```
\newcommand*{\GlsXtrRecordCount}[2]{%
 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2}%
 {\csname glo@\glsdetoklabel{#1}@recordcount.#2\endcsname}%
 {0}%
}
```

tionRecordCount Access record count for a particular counter and location. The first argument is the label. The second argument is the counter name. The third argument is the location. This command shouldn't be used if the location doesn't fully expand unless `\glsxtrdetoklocation` can be set to something sensible.

```
\newcommand*{\GlsXtrLocationRecordCount}[3]{%
 \ifcsdef{glo@\glsdetoklabel{#1}@recordcount.#2.\glsxtrdetoklocation{#3}}%
 {\csname glo@\glsdetoklabel{#1}@recordcount.#2.\glsxtrdetoklocation{#3}\endcsname}%
 {0}%
}
```

trdetoklocation

```
\newcommand*{\glsxtrdetoklocation}[1]{#1}
```

ablerecordcount

```
\newcommand*{\glsxtrenablerecordcount}{%
  \renewcommand*{\gls}{\rgls}%
  \renewcommand*{\Gls}{\rGls}%
  \renewcommand*{\glspl}{\rglspl}%
  \renewcommand*{\Glspl}{\rGlspl}%
  \renewcommand*{\GLS}{\rGLS}%
  \renewcommand*{\GLSpl}{\rGLSpl}%
}
```

ordtriggervalue The value used by the record trigger test. The argument is the entry's label.

```
\newcommand*{\glsxtrrecordtriggervalue}[1]{%
 \GlsXtrTotalRecordCount{#1}%
}
```

dCountAttribute

```
\newcommand*{\GlsXtrSetRecordCountAttribute}[2]{%
 \@for\@glsxtr@cat:=#1\do
 {%
   \ifdefempty{\@glsxtr@cat}{}%
   {%
     \glssetcategoryattribute{\@glsxtr@cat}{recordcount}{#2}%
   }%
 }%
}
```

rifrecordtrigger | \glsxtrifrecordtrigger{⟨*label*⟩}{⟨*trigger format*⟩}{⟨*normal*⟩}

```
\newcommand*{\glsxtrifrecordtrigger}[3]{%
 \glshasattribute{#1}{recordcount}%
 {%
   \ifnum\glsxtrrecordtriggervalue{#1}>\glsgetattribute{#1}{recordcount}\relax
    #3%
   \else
    #2%
   \fi
 }%
 {#3}%
}
```

strigger@record Still need a record to ensure that bib2gls selects the entry.

```
\newcommand*{\@glsxtr@rglstrigger@record}[3]{%
  \edef\glslabel{\glsdetoklabel{#2}}%
  \let\@gls@link@label\glslabel
  \def\@glsxtr@thevalue{}%
  \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
  \def\@glsnumberformat{glstriggerrecordformat}%
  \edef\@gls@counter{\csname glo@\glslabel @counter\endcsname}%
  \edef\glstype{\csname glo@\glslabel @type\endcsname}%
  \def\@glsxtr@thevalue{}%
  \def\@glsxtr@theHvalue{\@glsxtr@thevalue}%
  \glsxtrinitwrgloss
  \glslinkpresetkeys
  \setkeys{glslink}{#1}%
  \glslinkpostsetkeys
  \ifdefempty{\@glsxtr@thevalue}%
  {%
    \@gls@saveentrycounter
  }%
  {%
    \let\theglsentrycounter\@glsxtr@thevalue
    \def\theHglsentrycounter{\@glsxtr@theHvalue}%
  }%
  \ifglsxtrinitwrglossbefore
    \@do@wrglossary{#2}%
  \fi
  #3%
  \ifglsxtrinitwrglossbefore
  \else
    \@do@wrglossary{#2}%
  \fi
  \ifKV@glslink@local
    \glslocalunset{#2}%
```

```
  \else
    \glsunset{#2}%
  \fi
}
```

gerrecordformat Typically won't be used as it should be recognised as a special type of ignored location by bib2gls.

```
\newcommand*{\glstriggerrecordformat}[1]{}
```

\rgls

```
\newrobustcmd*{\rgls}{\@gls@hyp@opt\@rgls}
```

\@rgls

```
\newcommand*{\@rgls}[2][]{%
  \new@ifnextchar[{\@rgls@{#1}{#2}}{\@rgls@{#1}{#2}[]}%
}
```

\@rgls@

```
\def\@rgls@#1#2[#3]{%
  \glsxtrifrecordtrigger{#2}%
  {%
    \@glsxtr@rglstrigger@record{#1}{#2}{\rglsformat{#2}{#3}}%
  }%
  {%
    \@gls@{#1}{#2}[#3]%
  }%
}%
```

\rglspl

```
\newrobustcmd*{\rglspl}{\@gls@hyp@opt\@rglspl}
```

\@rglspl

```
\newcommand*{\@rglspl}[2][]{%
  \new@ifnextchar[{\@rglspl@{#1}{#2}}{\@rglspl@{#1}{#2}[]}%
}
```

\@rglspl@

```
\def\@rglspl@#1#2[#3]{%
  \glsxtrifrecordtrigger{#2}%
  {%
    \@glsxtr@rglstrigger@record{#1}{#2}{\rglsplformat{#2}{#3}}%
  }%
  {%
    \@glspl@{#1}{#2}[#3]%
  }%
}%
```

\rGls

```
\newrobustcmd*{\rGls}{\@gls@hyp@opt\@rGls}
```

```
\@rGls
5330 \newcommand*{\@rGls}[2][]{%
5331   \new@ifnextchar[{\@rGls@{#1}{#2}}{\@rGls@{#1}{#2}[]}%
5332 }

\@rGls@
5333 \def\@rGls@#1#2[#3]{%
5334   \glsxtrifrecordtrigger{#2}%
5335   {%
5336     \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsformat{#2}{#3}}%
5337   }%
5338   {%
5339     \@Gls@{#1}{#2}[#3]%
5340   }%
5341 }%

\rGlspl
5342 \newrobustcmd*{\rGlspl}{\@gls@hyp@opt\@rGlspl}

\@rGlspl
5343 \newcommand*{\@rGlspl}[2][]{%
5344   \new@ifnextchar[{\@rGlspl@{#1}{#2}}{\@rGlspl@{#1}{#2}[]}%
5345 }

\@rGlspl@
5346 \def\@rGlspl@#1#2[#3]{%
5347   \glsxtrifrecordtrigger{#2}%
5348   {%
5349     \@glsxtr@rglstrigger@record{#1}{#2}{\rGlsplformat{#2}{#3}}%
5350   }%
5351   {%
5352     \@Glspl@{#1}{#2}[#3]%
5353   }%
5354 }%

\rGLS
5355 \newrobustcmd*{\rGLS}{\@gls@hyp@opt\@rGLS}

\@rGLS
5356 \newcommand*{\@rGLS}[2][]{%
5357   \new@ifnextchar[{\@rGLS@{#1}{#2}}{\@rGLS@{#1}{#2}[]}%
5358 }

\@rGLS@
5359 \def\@rGLS@#1#2[#3]{%
5360   \glsxtrifrecordtrigger{#2}%
5361   {%
5362     \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSformat{#2}{#3}}%
```

```
  }%
  {%
    \@GLS@{#1}{#2}[#3]%
  }%
}%
```

\rGLSpl

```
\newrobustcmd*{\rGLSpl}{\@gls@hyp@opt\@rGLSpl}
```

\@rGLSpl

```
\newcommand*{\@rGLSpl}[2][]{%
  \new@ifnextchar[{\@rGLSpl@{#1}{#2}}{\@rGLSpl@{#1}{#2}[]}%
}
```

\@rGLSpl@

```
\def\@rGLSpl@#1#2[#3]{%
  \glsxtrifrecordtrigger{#2}%
  {%
    \@glsxtr@rglstrigger@record{#1}{#2}{\rGLSplformat{#2}{#3}}%
  }%
  {%
    \@GLSpl@{#1}{#2}[#3]%
  }%
}%
```

\rglsformat

```
\newcommand*{\rglsformat}[2]{%
  \glsifregular{#1}
  {\glsentryfirst{#1}}%
  {\ifglshaslong{#1}{\glsentrylong{#1}}{\glsentryfirst{#1}}}#2%
}
```

\rglsplformat

```
\newcommand*{\rglsplformat}[2]{%
  \glsifregular{#1}
  {\glsentryfirstplural{#1}}%
  {\ifglshaslong{#1}{\glsentrylongplural{#1}}{\glsentryfirstplural{#1}}}#2%
}
```

\rGlsformat

```
\newcommand*{\rGlsformat}[2]{%
  \glsifregular{#1}
  {\Glsentryfirst{#1}}%
  {\ifglshaslong{#1}{\Glsentrylong{#1}}{\Glsentryfirst{#1}}}#2%
}
```

\rGlsplformat

```
\newcommand*{\rGlsplformat}[2]{%
```

```
  \glsifregular{#1}
  {\Glsentryfirstplural{#1}}%
  {\ifglshaslong{#1}{\Glsentrylongplural{#1}}{\Glsentryfirstplural{#1}}}#2%
}
```

\rGLSformat

```
\newcommand*{\rGLSformat}[2]{%
 \expandafter\mfirstucMakeUppercase\expandafter{\rglsformat{#1}{#2}}%
}
```

\rGLSplformat

```
\newcommand*{\rGLSplformat}[2]{%
 \expandafter\mfirstucMakeUppercase\expandafter{\rglsplformat{#1}{#2}}%
}
```

1.4 Link Counting

This is different to the entry counting provided by the base package (which counts the number of times the first use flag is unset). Instead, this method hooks into \@gls@link (through \glsxtr@inc@linkcount) to increment an associated counter. To preserve resources, the counter is only defined if it needs to be incremented. This method is independent of the presence of hyperlinks. (The "link" part of the name refers to \@gls@link not \hyperlink.)

o@inc@linkcount This performs the actual incrementing and counter definition. The counter is given by \c@glsxtr@linkcount@⟨*label*⟩ where ⟨*label*⟩ is the entry's label. Since this is performed within \@gls@link the label can be accessed with \glslabel.

```
\newcommand{\@glsxtr@do@inc@linkcount}{%
```

Does this entry have the linkcount attribute set?

```
 \glsifattribute{\glslabel}{linkcount}{true}%
 {%
```

Does the counter exist?

```
   \ifcsdef{c@glsxtr@linkcount@\glslabel}{}%
   {%
```

Counter doesn't exist, so define it.

```
     \newcounter{glsxtr@linkcount@\glslabel}%
```

If linkcountmaster is set, add to counter reset.

```
     \glshasattribute{\glslabel}{linkcountmaster}%
     {%
```

Need to ensure values are fully expanded.

```
       \begingroup
        \edef\x{\endgroup\noexpand\@addtoreset{glsxtr@linkcount@\glslabel}%
         {\glsgetattribute{\glslabel}{linkcountmaster}}}%
       \x
     }%
```

```
      {}%
    }%
```

Increment counter:

```
  \glsxtrinclinkcounter{glsxtr@linkcount@\glslabel}%
 }%
 {}%
}
```

rinclinkcounter May be redefined to use `\refstepcounter` if required.

```
\newcommand*{\glsxtrinclinkcounter}[1]{\stepcounter{#1}}
```

inkCounterValue Expands to the associated link counter register or 0 if not defined.

```
\newcommand*{\GlsXtrLinkCounterValue}[1]{%
 \ifcsundef{c@glsxtr@linkcount@#1}{0}{\csname c@glsxtr@linkcount@#1\endcsname}%
}
```

rTheLinkCounter Expands to the display value of the associated link counter or 0 if not defined.

```
\newcommand*{\GlsXtrTheLinkCounter}[1]{%
 \ifcsundef{theglsxtr@linkcount@#1}{0}%
 {\csname theglsxtr@linkcount@#1\endcsname}%
}
```

fLinkCounterDef Tests if the counter has been defined

```
\newcommand*{\GlsXtrIfLinkCounterDef}[3]{%
 \ifcsundef{theglsxtr@linkcount@#1}{#3}{#2}%
}
```

LinkCounterName Expands to the associated link counter name. (No check for existence.)

```
\newcommand*{\GlsXtrLinkCounterName}[1]{glsxtr@linkcount@#1}
```

ableLinkCounting `\GlsXtrEnableLinkCounting[⟨master counter⟩]{⟨categories⟩}`

Enable link counting for the given categories.

```
\newcommand*{\GlsXtrEnableLinkCounting}[2][]{%
 \let\glsxtr@inc@linkcount\@glsxtr@do@inc@linkcount
 \@for\@glsxtr@label:=#2\do
 {%
  \glssetcategoryattribute{\@glsxtr@label}{linkcount}{true}%
  \ifstrempty{#1}{}%
  {%
    \ifcsundef{c@#1}%
    {\@nocounterr{#1}}%
    {\glssetcategoryattribute{\@glsxtr@label}{linkcountmaster}{#1}}%
  }%
 }%
}
\@onlypreamble\GlsXtrEnableLinkCounting
```

1.5 Integration with glossaries-accsupp

Provide better integration with the glossaries-accsupp package. (Must be loaded before the main code of glossaries-extra either explicitly or through the accsupp package option.)

These commands have their definitions set according to whether or not glossaries-extra has been loaded.

```
\@ifpackageloaded{glossaries-accsupp}
{
```

Define (or redefine) commands to use the accessibility information.

\glsaccessname Display the name value (no link and no check for existence).

```
  \newcommand*{\glsaccessname}[1]{%
    \glsnameaccessdisplay
    {%
      \glsentryname{#1}%
    }%
    {#1}%
  }
```

\Glsaccessname Display the name value (no link and no check for existence) with the first letter converted to upper case.

```
  \newcommand*{\Glsaccessname}[1]{%
    \glsnameaccessdisplay
    {%
      \Glsentryname{#1}%
    }%
    {#1}%
  }
```

\GLSaccessname Display the name value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccessname}[1]{%
    \glsnameaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentryname{#1}}%
    }%
    {#1}%
  }
```

\glsaccesstext Display the text value (no link and no check for existence).

```
  \newcommand*{\glsaccesstext}[1]{%
    \glstextaccessdisplay
    {%
      \glsentrytext{#1}%
    }%
    {#1}%
  }
```

\Glsaccesstext Display the text value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccesstext}[1]{%
  \glstextaccessdisplay
  {%
    \Glsentrytext{#1}%
  }%
  {#1}%
}
```

\GLSaccesstext Display the text value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccesstext}[1]{%
  \glstextaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentrytext{#1}}%
  }%
  {#1}%
}
```

glsaccessplural Display the plural value (no link and no check for existence).

```
\newcommand*{\glsaccessplural}[1]{%
  \glspluralaccessdisplay
  {%
    \glsentryplural{#1}%
  }%
  {#1}%
}
```

Glsaccessplural Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessplural}[1]{%
  \glspluralaccessdisplay
  {%
    \Glsentryplural{#1}%
  }%
  {#1}%
}
```

GLSaccessplural Display the plural value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessplural}[1]{%
  \glspluralaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentryplural{#1}}%
  }%
  {#1}%
}
```

\glsaccessfirst Display the first value (no link and no check for existence).

```
\newcommand*{\glsaccessfirst}[1]{%
  \glsfirstaccessdisplay
  {%
    \glsentryfirst{#1}%
  }%
  {#1}%
}
```

\Glsaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessfirst}[1]{%
  \glsfirstaccessdisplay
  {%
    \Glsentryfirst{#1}%
  }%
  {#1}%
}
```

\GLSaccessfirst Display the first value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessfirst}[1]{%
  \glsfirstaccessdisplay
  {%
    \mfirstucMakeUppercase{\glsentryfirst{#1}}%
  }%
  {#1}%
}
```

cessfirstplural Display the firstplural value (no link and no check for existence).

```
\newcommand*{\glsaccessfirstplural}[1]{%
  \glsfirstpluralaccessdisplay
  {%
    \glsentryfirstplural{#1}%
  }%
  {#1}%
}
```

cessfirstplural Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessfirstplural}[1]{%
  \glsfirstpluralaccessdisplay
  {%
    \Glsentryfirstplural{#1}%
  }%
  {#1}%
}
```

cessfirstplural Display the firstplural value (no link and no check for existence) converted to upper case.

```
\newcommand*{\GLSaccessfirstplural}[1]{%
```

```
    \glsfirstpluralaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentryfirstplural{#1}}%
    }%
    {#1}%
  }
```

`glsaccesssymbol` Display the symbol value (no link and no check for existence).

```
  \newcommand*{\glsaccesssymbol}[1]{%
    \glssymbolaccessdisplay
    {%
      \glsentrysymbol{#1}%
    }%
    {#1}%
  }
```

`Glsaccesssymbol` Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```
  \newcommand*{\Glsaccesssymbol}[1]{%
    \glssymbolaccessdisplay
    {%
      \Glsentrysymbol{#1}%
    }%
    {#1}%
  }
```

`GLSaccesssymbol` Display the symbol value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccesssymbol}[1]{%
    \glssymbolaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentrysymbol{#1}}%
    }%
    {#1}%
  }
```

`esssymbolplural` Display the symbolplural value (no link and no check for existence).

```
  \newcommand*{\glsaccesssymbolplural}[1]{%
    \glssymbolpluralaccessdisplay
    {%
      \glsentrysymbolplural{#1}%
    }%
    {#1}%
  }
```

`esssymbolplural` Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```
  \newcommand*{\Glsaccesssymbolplural}[1]{%
    \glssymbolpluralaccessdisplay
```

```
    {%
      \Glsentrysymbolplural{#1}%
    }%
    {#1}%
  }
```

esssymbolplural Display the symbolplural value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccesssymbolplural}[1]{%
    \glssymbolpluralaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentrysymbolplural{#1}}%
    }%
    {#1}%
  }
```

\glsaccessdesc Display the desc value (no link and no check for existence).

```
  \newcommand*{\glsaccessdesc}[1]{%
    \glsdescriptionaccessdisplay
    {%
      \glsentrydesc{#1}%
    }%
    {#1}%
  }
```

\Glsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```
  \newcommand*{\Glsaccessdesc}[1]{%
    \glsdescriptionaccessdisplay
    {%
      \Glsentrydesc{#1}%
    }%
    {#1}%
  }
```

\GLSaccessdesc Display the desc value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccessdesc}[1]{%
    \glsdescriptionaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentrydesc{#1}}%
    }%
    {#1}%
  }
```

ccessdescplural Display the descplural value (no link and no check for existence).

```
  \newcommand*{\glsaccessdescplural}[1]{%
    \glsdescriptionpluralaccessdisplay
    {%
      \glsentrydescplural{#1}%
```

```
    }%
    {#1}%
  }
```

ccessdescplural Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```
  \newcommand*{\Glsaccessdescplural}[1]{%
    \glsdescriptionpluralaccessdisplay
    {%
      \Glsentrydescplural{#1}%
    }%
    {#1}%
  }
```

ccessdescplural Display the descplural value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccessdescplural}[1]{%
    \glsdescriptionpluralaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentrydescplural{#1}}%
    }%
    {#1}%
  }
```

\glsaccessshort Display the short form (no link and no check for existence).

```
  \newcommand*{\glsaccessshort}[1]{%
    \glsshortaccessdisplay
    {%
      \glsentryshort{#1}%
    }%
    {#1}%
  }
```

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).

```
  \newcommand*{\Glsaccessshort}[1]{%
    \glsshortaccessdisplay
    {%
      \Glsentryshort{#1}%
    }%
    {#1}%
  }
```

\GLSaccessshort Display the short value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccessshort}[1]{%
    \glsshortaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentryshort{#1}}%
    }%
```

```
    {#1}%
  }
```

`lsaccessshortpl` Display the short plural form (no link and no check for existence).

```
  \newcommand*{\glsaccessshortpl}[1]{%
    \glsshortpluralaccessdisplay
    {%
      \glsentryshortpl{#1}%
    }%
    {#1}%
  }
```

`lsaccessshortpl` Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```
  \newcommand*{\Glsaccessshortpl}[1]{%
    \glsshortpluralaccessdisplay
    {%
      \Glsentryshortpl{#1}%
    }%
    {#1}%
  }
```

`LSaccessshortpl` Display the shortplural value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccessshortpl}[1]{%
    \glsshortpluralaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentryshortpl{#1}}%
    }%
    {#1}%
  }
```

`\glsaccesslong` Display the long form (no link and no check for existence).

```
  \newcommand*{\glsaccesslong}[1]{%
    \glslongaccessdisplay{\glsentrylong{#1}}{#1}%
  }
```

`\Glsaccesslong` Display the long form (no link and no check for existence).

```

  \newcommand*{\Glsaccesslong}[1]{%
    \glslongaccessdisplay{\Glsentrylong{#1}}{#1}%
  }
```

`\GLSaccesslong` Display the long value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccesslong}[1]{%
    \glslongaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentrylong{#1}}%
    }%
```

```
    {#1}%
  }
```

`glsaccesslongpl` Display the long plural form (no link and no check for existence).

```
  \newcommand*{\glsaccesslongpl}[1]{%
    \glslongpluralaccessdisplay{\glsentrylongpl{#1}}{#1}%
  }
```

`Glsaccesslongpl` Display the long plural form (no link and no check for existence).

```

  \newcommand*{\Glsaccesslongpl}[1]{%
    \glslongpluralaccessdisplay{\Glsentrylongpl{#1}}{#1}%
  }
```

`GLSaccesslongpl` Display the longplural value (no link and no check for existence) converted to upper case.

```
  \newcommand*{\GLSaccesslongpl}[1]{%
    \glslongpluralaccessdisplay
    {%
      \mfirstucMakeUppercase{\glsentrylongpl{#1}}%
    }%
    {#1}%
  }
```

Keys for accessibility support.

```
  \define@key{glsxtrabbrv}{access}{%
    \def\@gls@nameaccess{#1}%
  }
  \define@key{glsxtrabbrv}{textaccess}{%
    \def\@gls@textaccess{#1}%
  }
  \define@key{glsxtrabbrv}{firstaccess}{%
    \def\@gls@firstaccess{#1}%
  }
  \define@key{glsxtrabbrv}{shortaccess}{%
    \def\@gls@shortaccess{#1}%
  }
  \define@key{glsxtrabbrv}{shortpluralaccess}{%
    \def\@gls@shortaccesspl{#1}%
  }
```

`@initaccesskeys`

```
  \newcommand*{\@gls@initaccesskeys}{%
    \def\@gls@nameaccess{}%
    \def\@gls@textaccess{}%
    \def\@gls@firstaccess{}%
    \def\@gls@shortaccess{}%
    \def\@gls@shortaccesspl{}%
  }
```

essattribute@set `\gls@ifaccessattribute@set{⟨attribute⟩}{⟨true⟩}{⟨false⟩}`

```
\newcommand*{\@gls@ifaccessattribute@set}[3]{%
  \glsifcategoryattribute{\glscategorylabel}{access#1}{true}%
  {#2}%
  {%
    \glsifcategoryattribute{\glscategorylabel}{access#1}{false}%
    {#3}%
    {%
      \glsifcategoryattribute{\glscategorylabel}{#1}{true}%
      {#2}%
      {#3}%
    }%
  }%
}
```

lt@short@access Assign the default value of the shortaccess key. The argument is the short value passed to `\newabbreviation`.

```
\newcommand{\@gls@setup@default@short@access}[1]{%
```

Check if the accessinsertdots attribute has been set but only if shortaccess hasn't been set.

```
    \ifdefempty\@gls@shortaccess
    {%
      \glsifcategoryattribute{\glscategorylabel}{accessinsertdots}{true}%
      {%
        \@glsxtr@insertdots\@gls@shortaccess{#1}%
        \eappto\ExtraCustomAbbreviationFields{%
          shortaccess={\expandonce\@gls@shortaccess},}%
      }%
      {}%
    }%
    {}%
```

If the shortaccess field has been set but shortaccessplural hasn't been set, assign plural form.

```
    \ifdefempty\@gls@shortaccess
    {}%
    {%
      \ifdefempty\@gls@shortaccesspl
      {%
        \@gls@ifaccessattribute@set{aposplural}%
        {%
          \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
            \@gls@shortaccess'\abbrvpluralsuffix}%
        }%
        {%
          \@gls@ifaccessattribute@set{noshortplural}%
          {%
```

```
            \let\@gls@shortaccesspl\@gls@shortaccess
          }%
          {%
            \expandafter\def\expandafter\@gls@shortaccesspl\expandafter{%
              \@gls@shortaccess\abbrvpluralsuffix}%
          }%
        }%
        \eappto\ExtraCustomAbbreviationFields{%
          shortpluralaccess={\expandonce\@gls@shortaccesspl},}%
      }%
      {}%
    }%
```

If access key hasn't been set, check if the nameshortaccess attribute has been set.

```
    \ifdefempty\@gls@nameaccess
    {%
      \glsifcategoryattribute{\glscategorylabel}{nameshortaccess}{true}%
      {%
```

Do nothing if the shortaccess key hasn't been set.

```
        \ifdefempty\@gls@shortaccess
        {}%
        {%
           \eappto\ExtraCustomAbbreviationFields{%
              access={\expandonce\@gls@shortaccess},%
           }%
        }%
      }%
      {}%
    }%
    {}%
```

If textaccess key hasn't been set, check if the textshortaccess attribute has been set.

```
    \ifdefempty\@gls@textaccess
    {%
      \glsifcategoryattribute{\glscategorylabel}{textshortaccess}{true}%
      {%
```

Do nothing if the shortaccess key hasn't been set.

```
        \ifdefempty\@gls@shortaccess
        {}%
        {%
           \eappto\ExtraCustomAbbreviationFields{%
              textaccess={\expandonce\@gls@shortaccess},%
           }%
        }%
      }%
      {}%
    }%
    {}%
```

If firstaccess key hasn't been set, check if the firstshortaccess attribute has been set.

```
    \ifdefempty\@gls@firstaccess
    {%
      \glsifcategoryattribute{\glscategorylabel}{firstshortaccess}{true}%
      {%
```

Do nothing if the shortaccess key hasn't been set.

```
        \ifdefempty\@gls@shortaccess
        {}%
        {%
           \eappto\ExtraCustomAbbreviationFields{%
              firstaccess={\expandonce\@gls@shortaccess},%
           }%
        }%
      }%
      {}%
    }%
    {}%
  }
```

End of if accsupp part

```
}
{
```

No accessibility support. Just define these commands to do \glsentry⟨*xxx*⟩

\glsaccessname Display the name value (no link and no check for existence).

```
  \newcommand*{\glsaccessname}[1]{\glsentryname{#1}}
```

\Glsaccessname Display the name value (no link and no check for existence) with the first letter converted to upper case.

```
  \newcommand*{\Glsaccessname}[1]{\Glsentryname{#1}}
```

\GLSaccessname Display the name value (no link and no check for existence). converted to upper case.

```
  \newcommand*{\GLSaccessname}[1]{%
   \protect\mfirstucMakeUppercase{\glsentryname{#1}}}
```

\glsaccesstext Display the text value (no link and no check for existence).

```
  \newcommand*{\glsaccesstext}[1]{\glsentrytext{#1}}
```

\Glsaccesstext Display the text value (no link and no check for existence) with the first letter converted to upper case.

```
  \newcommand*{\Glsaccesstext}[1]{\Glsentrytext{#1}}
```

\GLSaccesstext Display the text value (no link and no check for existence). converted to upper case.

```
  \newcommand*{\GLSaccesstext}[1]{%
   \protect\mfirstucMakeUppercase{\glsentrytext{#1}}}
```

glsaccessplural Display the plural value (no link and no check for existence).

```
  \newcommand*{\glsaccessplural}[1]{\glsentryplural{#1}}
```

Glsaccessplural Display the plural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessplural}[1]{\Glsentryplural{#1}}
```

GLSaccessplural Display the plural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessplural}[1]{%
 \protect\mfirstucMakeUppercase{\glsentryplural{#1}}}
```

\glsaccessfirst Display the first value (no link and no check for existence).

```
\newcommand*{\glsaccessfirst}[1]{\glsentryfirst{#1}}
```

\Glsaccessfirst Display the first value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessfirst}[1]{\Glsentryfirst{#1}}
```

\GLSaccessfirst Display the first value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessfirst}[1]{%
 \protect\mfirstucMakeUppercase{\glsentryfirst{#1}}}
```

cessfirstplural Display the firstplural value (no link and no check for existence).

```
\newcommand*{\glsaccessfirstplural}[1]{\glsentryfirstplural{#1}}
```

cessfirstplural Display the firstplural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessfirstplural}[1]{\Glsentryfirstplural{#1}}
```

cessfirstplural Display the firstplural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessfirstplural}[1]{%
 \protect\mfirstucMakeUppercase{\glsentryfirstplural{#1}}}
```

glsaccesssymbol Display the symbol value (no link and no check for existence).

```
\newcommand*{\glsaccesssymbol}[1]{\glsentrysymbol{#1}}
```

Glsaccesssymbol Display the symbol value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccesssymbol}[1]{\Glsentrysymbol{#1}}
```

GLSaccesssymbol Display the symbol value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccesssymbol}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrysymbol{#1}}}
```

esssymbolplural Display the symbolplural value (no link and no check for existence).

```
\newcommand*{\glsaccesssymbolplural}[1]{\glsentrysymbolplural{#1}}
```

esssymbolplural Display the symbolplural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccesssymbolplural}[1]{\Glsentrysymbolplural{#1}}
```

esssymbolplural Display the symbolplural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccesssymbolplural}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrysymbolplural{#1}}}
```

\glsaccessdesc Display the desc value (no link and no check for existence).

```
\newcommand*{\glsaccessdesc}[1]{\glsentrydesc{#1}}
```

\Glsaccessdesc Display the desc value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessdesc}[1]{\Glsentrydesc{#1}}
```

\GLSaccessdesc Display the desc value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessdesc}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrydesc{#1}}}
```

ccessdescplural Display the descplural value (no link and no check for existence).

```
\newcommand*{\glsaccessdescplural}[1]{\glsentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence) with the first letter converted to upper case.

```
\newcommand*{\Glsaccessdescplural}[1]{\Glsentrydescplural{#1}}
```

ccessdescplural Display the descplural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessdescplural}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrydescplural{#1}}}
```

\glsaccessshort Display the short form (no link and no check for existence).

```
\newcommand*{\glsaccessshort}[1]{\glsentryshort{#1}}
```

\Glsaccessshort Display the short form with first letter converted to uppercase (no link and no check for existence).

```
\newcommand*{\Glsaccessshort}[1]{\Glsentryshort{#1}}
```

\GLSaccessshort Display the short value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessshort}[1]{%
 \protect\mfirstucMakeUppercase{\glsentryshort{#1}}}
```

lsaccessshortpl Display the short plural form (no link and no check for existence).

```
\newcommand*{\glsaccessshortpl}[1]{\glsentryshortpl{#1}}
```

lsaccessshortpl Display the short plural form with first letter converted to uppercase (no link and no check for existence).

```
\newcommand*{\Glsaccessshortpl}[1]{\Glsentryshortpl{#1}}
```

LSaccessshortpl Display the shortplural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccessshortpl}[1]{%
 \protect\mfirstucMakeUppercase{\glsentryshortpl{#1}}}
```

\glsaccesslong Display the long form (no link and no check for existence).

```
\newcommand*{\glsaccesslong}[1]{\glsentrylong{#1}}
```

\Glsaccesslong Display the long form (no link and no check for existence).

```
\newcommand*{\Glsaccesslong}[1]{\Glsentrylong{#1}}
```

\GLSaccesslong Display the long value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccesslong}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrylong{#1}}}
```

glsaccesslongpl Display the long plural form (no link and no check for existence).

```
\newcommand*{\glsaccesslongpl}[1]{\glsentrylongpl{#1}}
```

Glsaccesslongpl Display the long plural form (no link and no check for existence).

```
\newcommand*{\Glsaccesslongpl}[1]{\Glsentrylongpl{#1}}
```

GLSaccesslongpl Display the longplural value (no link and no check for existence). converted to upper case.

```
\newcommand*{\GLSaccesslongpl}[1]{%
 \protect\mfirstucMakeUppercase{\glsentrylongpl{#1}}}
```

@initaccesskeys This does nothing if there's no accessibility support.

```
\newcommand*{\@gls@initaccesskeys}{}
```

lt@short@access This does nothing if there's no accessibility support.

```
\newcommand{\@gls@setup@default@short@access}[1]{}%
```

End of else part

```
}
```

1.6 Categories

\glscategory Add a new storage key that can be used to indicate a category. The default category is general.

```
\glsaddstoragekey{category}{general}{\glscategory}
```

\glsifcategory Convenient shortcut to determine if an entry has the given category.

```
\newcommand{\glsifcategory}[4]{%
 \ifglsfieldeq{#1}{category}{#2}{#3}{#4}%
}
```

Categories can have attributes.

ategoryattribute `\glssetcategoryattribute{⟨category⟩}{⟨attribute-label⟩}{⟨value⟩}`

Set (or override if already set) an attribute for the given category.

```
\newcommand*{\glssetcategoryattribute}[3]{%
  \csdef{@glsxtr@categoryattr@@#1@#2}{#3}%
}
```

ategoryattribute `\glsgetcategoryattribute{⟨category⟩}{⟨attribute-label⟩}`

Get the value of the given attribute for the given category. Does nothing if the attribute isn't defined.

```
\newcommand*{\glsgetcategoryattribute}[2]{%
  \csuse{@glsxtr@categoryattr@@#1@#2}%
}
```

ategoryattribute `\glshascategoryattribute{⟨category⟩}{⟨attribute-label⟩}{⟨true⟩}{⟨false⟩}`

Tests if the category has the given attribute set.

```
\newcommand*{\glshascategoryattribute}[4]{%
  \ifcsvoid{@glsxtr@categoryattr@@#1@#2}{#4}{#3}%
}
```

\glssetattribute `\glssetattribute{⟨entry label⟩}{⟨attribute-label⟩}{⟨value⟩}`

Short cut where the category label is obtained from the entry information.

```
\newcommand*{\glssetattribute}[3]{%
  \glssetcategoryattribute{\glscategory{#1}}{#2}{#3}%
}
```

\glsgetattribute `\glsgetattribute{⟨entry label⟩}{⟨attribute-label⟩}`

Short cut where the category label is obtained from the entry information.

```
\newcommand*{\glsgetattribute}[2]{%
  \glsgetcategoryattribute{\glscategory{#1}}{#2}%
}
```

\glshasattribute `\glshasattribute{⟨entry label⟩}{⟨attribute-label⟩}{⟨true⟩}{⟨false⟩}`

Short cut to test if the given attribute has been set where the category label is obtained from the entry information.

```
\newcommand*{\glshasattribute}[4]{%
  \ifglsentryexists{#1}%
  {\glshascategoryattribute{\glscategory{#1}}{#2}{#3}{#4}}%
  {#4}%
}
```

ategoryattribute `\glsifcategoryattribute{⟨category⟩}{⟨attribute-label⟩}{⟨value⟩}{⟨true part⟩}{⟨false part⟩}`

True if category has the attribute with the given value.

```
\newcommand{\glsifcategoryattribute}[5]{%
 \ifcsundef{@glsxtr@categoryattr@@#1@#2}%
 {#5}%
 {\ifcsstring{@glsxtr@categoryattr@@#1@#2}{#3}{#4}{#5}}%
}
```

\glsifattribute `\glsifattribute{⟨entry label⟩}{⟨attribute-label⟩}{⟨value⟩}{⟨true part⟩}{⟨false part⟩}`

Short cut to determine if the given entry has a category with the given attribute set.

```
\newcommand{\glsifattribute}[5]{%
  \ifglsentryexists{#1}%
  {\glsifcategoryattribute{\glscategory{#1}}{#2}{#3}{#4}{#5}}%
  {#5}%
}
```

Set attributes for the default general category:

```
\glssetcategoryattribute{general}{regular}{true}
```

Acronyms are regular by default, since they're typically just treated like normal words.

```
\glssetcategoryattribute{acronym}{regular}{true}
```

regularcategory Convenient shortcut to create add the regular attribute.

```
\newcommand*{\glssetregularcategory}[1]{%
 \glssetcategoryattribute{#1}{regular}{true}%
}
```

fregularcategory `\glsifregularcategory{⟨category⟩}{⟨true part⟩}{⟨false part⟩}`

Short cut to determine if a category has the regular attribute explicitly set to true.

```
\newcommand{\glsifregularcategory}[3]{%
  \glsifcategoryattribute{#1}{regular}{true}{#2}{#3}%
}
```

tregularcategory `\glsifnotregularcategory{⟨category⟩}{⟨true part⟩}{⟨false part⟩}`

Short cut to determine if a category has the regular attribute explicitly set to false.

```
\newcommand{\glsifnotregularcategory}[3]{%
  \glsifcategoryattribute{#1}{regular}{false}{#2}{#3}%
}
```

\glsifregular `\glsifregular{⟨entry label⟩}{⟨true part⟩}{⟨false part⟩}`

Short cut to determine if an entry has a regular attribute set to true.

```
\newcommand{\glsifregular}[3]{%
  \glsifregularcategory{\glscategory{#1}}{#2}{#3}%
}
```

\glsifnotregular `\glsifnotregular{⟨entry label⟩}{⟨true part⟩}{⟨false part⟩}`

Short cut to determine if an entry has a regular attribute set to false.

```
\newcommand{\glsifnotregular}[3]{%
  \glsifnotregularcategory{\glscategory{#1}}{#2}{#3}%
}
```

oreachincategory `\glsforeachincategory[⟨glossary labels⟩]{⟨category-label⟩}{⟨glossary-cs⟩}{⟨label-cs⟩}{⟨body⟩}`

Iterates through all entries in all the glossaries (or just those listed in ⟨*glossary labels*⟩) and does ⟨*body*⟩ if the category matches ⟨*category-label*⟩. The control sequences ⟨*glossary-cs*⟩ and ⟨*label-cs*⟩ may be used in ⟨*body*⟩ to access the glossary label and entry label for the current iteration.

```
\newcommand{\glsforeachincategory}[5][\@glo@types]{%
```

```
  \forallglossaries[#1]{#3}%
  {%
     \forglsentries[#3]{#4}%
     {%
       \glsifcategory{#4}{#2}{#5}{}%
     }%
  }%
}
```

achwithattribute

\glsforeachwithattribute[⟨*glossary labels*⟩]{⟨*attribute-label*⟩}{⟨*attribute-value*⟩}{⟨*glossary-cs*⟩}{⟨*label-cs*⟩}{⟨*body*⟩}

Iterates through all entries in all the glossaries (or just those listed in ⟨*glossary labels*⟩) and does ⟨*body*⟩ if the category attribute ⟨*attribute-label*⟩ matches ⟨*attribute-value*⟩. The control sequences ⟨*glossary-cs*⟩ and ⟨*label-cs*⟩ may be used in ⟨*body*⟩ to access the glossary label and entry label for the current iteration.

```
\newcommand{\glsforeachwithattribute}[6][\@glo@types]{%
  \forallglossaries[#1]{#4}%
  {%
     \forglsentries[#4]{#5}%
     {%
       \glsifattribute{#5}{#2}{#3}{#6}{}%
     }%
  }%
}
```

If \newterm has been defined, redefine it so that it automatically sets the category label to index and add \glsxtrpostdescription.

```
\ifdef\newterm
{%
```

\newterm

```
  \renewcommand*{\newterm}[2][]{%
    \newglossaryentry{#2}%
    {type={index},category=index,name={#2},%
     description={\glsxtrpostdescription\nopostdesc},#1}%
  }
```

Indexed terms are regular by default.

```
  \glssetcategoryattribute{index}{regular}{true}
```

trpostdescindex

```
  \newcommand*{\glsxtrpostdescindex}{}
```

```
}
{}
```

If the symbols package option was used, define a similar command for symbols, but set the default sort to the label rather than the name as the symbols will typically contain commands that will confuse makeindex and xindy.

```
\ifdef\printsymbols
{%
```

glsxtrnewsymbol Unlike \newterm, this has a separate argument for the label (since the symbol will likely contain commands).

```
  \newcommand*{\glsxtrnewsymbol}[3][]{%
    \newglossaryentry{#2}{name={#3},sort={#2},type=symbols,category=symbol,#1}%
  }
```

Symbols are regular by default.

```
  \glssetcategoryattribute{symbol}{regular}{true}
```

rpostdescsymbol

```
  \newcommand*{\glsxtrpostdescsymbol}{}

}
{}
```

Similar for the numbers option.

```
\ifdef\printnumbers
{%
```

glsxtrnewnumber

```
\ifdef\printnumbers
  \newcommand*{\glsxtrnewnumber}[3][]{%
    \newglossaryentry{#2}{name={#3},sort={#2},type=numbers,category=number,#1}%
  }
```

Numbers are regular by default.

```
  \glssetcategoryattribute{number}{regular}{true}
```

rpostdescnumber

```
  \newcommand*{\glsxtrpostdescnumber}{}

}
{}
```

sxtrsetcategory Set the category for all listed labels. The first argument is the list of entry labels and the second argument is the category label.

```
\newcommand*{\glsxtrsetcategory}[2]{%
  \@for\@glsxtr@label:=#1\do
  {%
    \glsfieldxdef{\@glsxtr@label}{category}{#2}%
  }%
}
```

tcategoryforall Set the category for all entries in the listed glossaries. The first argument is the list of glossary labels and the second argument is the category label.

```
\newcommand*{\glsxtrsetcategoryforall}[2]{%
  \forallglossaries[#1]{\@glsxtr@type}{%
    \forglsentries[\@glsxtr@type]{\@glsxtr@label}%
    {%
      \glsfieldxdef{\@glsxtr@label}{category}{#2}%
    }%
  }%
}
```

trfieldtitlecase `\glsxtrfieldtitlecase{⟨label⟩}{⟨field⟩}`

Apply title casing to the contents of the given field.

```
\newcommand*{\glsxtrfieldtitlecase}[2]{%
  \expandafter\glsxtrfieldtitlecasecs\expandafter
    {\csname glo@\glsdetoklabel{#1}@#2\endcsname}%
}
```

ieldtitlecasecs The command used by `\glsxtrfieldtitlecase`. May be redefined to use a different command, for example, `\xcapitalisefmtwords`.

```
\newcommand*{\glsxtrfieldtitlecasecs}[1]{\xcapitalisewords{#1}}
```

Provide a convenient way to modify glossary styles without having to define a new style just to convert the first letter of fields to upper case.

\glossentrydesc If the glossdesc attribute is “firstuc” convert first letter to upper case. If the attribute is “title” use title case.

```
\@ifpackageloaded{glossaries-accsupp}
{
  \renewcommand*{\glossentrydesc}[1]{%
    \glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossdescfont attribute to determine the font applied.

```
      \glshasattribute{#1}{glossdescfont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossdescfont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          `\@glsxtr@attrval' supplied in glossdescfont attribute
```

```
6022          for entry ‘#1’. Ignoring}%
6023          \let\@glsxtr@glossdescfont\@firstofone
6024        }%
6025      }%
6026      {\let\@glsxtr@glossdescfont\@firstofone}%
6027      \glsifattribute{#1}{glossdesc}{firstuc}%
6028      {%
6029        \@glsxtr@glossdescfont{\Glsaccessdesc{#1}}%
6030      }%
6031      {%
6032        \glsifattribute{#1}{glossdesc}{title}%
6033        {%
6034          \@glsxtr@do@titlecaps@warn
6035          \glsdescriptionaccessdisplay
6036          {%
6037            \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
6038          }%
6039          {#1}%
6040        }%
6041        {%
6042          \@glsxtr@glossdescfont{\glsaccessdesc{#1}}%
6043        }%
6044      }%
6045    }%
6046  }
6047 }
6048 {
6049  \renewcommand*{\glossentrydesc}[1]{%
6050    \glsdoifexistsorwarn{#1}%
6051    {%
6052      \glssetabbrvfmt{\glscategory{#1}}%
6053      \glshasattribute{#1}{glossdescfont}%
6054      {%
6055        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossdescfont}}%
6056        \ifcsdef{\@glsxtr@attrval}%
6057        {%
6058          \letcs{\@glsxtr@glossdescfont}{\@glsxtr@attrval}%
6059        }%
6060        {%
6061          \GlossariesExtraWarning{Unknown control sequence name
6062          ‘\@glsxtr@attrval’ supplied in glossdescfont attribute
6063          for entry ‘#1’. Ignoring}%
6064          \let\@glsxtr@glossdescfont\@firstofone
6065        }%
6066      }%
6067      {\let\@glsxtr@glossdescfont\@firstofone}%
6068      \glsifattribute{#1}{glossdesc}{firstuc}%
6069      {%
6070        \@glsxtr@glossdescfont{\Glsentrydesc{#1}}%
```

```
      }%
      {%
        \glsifattribute{#1}{glossdesc}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \@glsxtr@glossdescfont{\glsxtrfieldtitlecase{#1}{desc}}%
        }%
        {%
          \@glsxtr@glossdescfont{\glsentrydesc{#1}}%
        }%
      }%
    }%
  }
}
```

\glossentryname If the glossname attribute is "firstuc" convert first letter to upper case. If the attribute is "title" use title case.

```
\@ifpackageloaded{glossaries-accsupp}
{
  \renewcommand*{\glossentryname}[1]{%
    \@glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```
      \glshasattribute{#1}{glossnamefont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          `\@glsxtr@attrval' supplied in glossnamefont attribute
          for entry `#1'. Reverting to default \string\glsnamefont}%
          \let\@glsxtr@glossnamefont\glsnamefont
        }%
      }%
      {\let\@glsxtr@glossnamefont\glsnamefont}%
      \glsifattribute{#1}{glossname}{firstuc}%
      {%
        \glsnameaccessdisplay
        {%
          \@glsxtr@glossnamefont{\Glsentryname{#1}}%
        }%
        {#1}%
      }%
      {%
        \glsifattribute{#1}{glossname}{title}%
```

```
        {%
          \@glsxtr@do@titlecaps@warn
          \glsnameaccessdisplay
          {%
            \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
          }%
          {#1}%
        }%
        {%
          \glsifattribute{#1}{glossname}{uc}%
          {%
            \glsnameaccessdisplay
            {%
```

Hide the label from the upper-casing command.

```
               \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
               \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
            }%
            {#1}%
          }%
          {%
            \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
            \glsnameaccessdisplay
            {%
              \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
            }%
            {#1}%
          }%
        }%
      }%
```

Do post-name hook:

```
      \glsxtrpostnamehook{#1}%
    }%
  }
}
{
  \renewcommand*{\glossentryname}[1]{%
    \@glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glshasattribute{#1}{glossnamefont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
```

```
          `\@glsxtr@attrval' supplied in glossnamefont attribute
          for entry `#1'. Reverting to default \string\glsnamefont}%
          \let\@glsxtr@glossnamefont\glsnamefont
        }%
      }%
      {\let\@glsxtr@glossnamefont\glsnamefont}%
      \glsifattribute{#1}{glossname}{firstuc}%
      {%
        \@glsxtr@glossnamefont{\Glsentryname{#1}}%
      }%
      {%
        \glsifattribute{#1}{glossname}{title}%
        {%
          \@glsxtr@do@titlecaps@warn
          \@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{name}}%
        }%
        {%
          \glsifattribute{#1}{glossname}{uc}%
          {%
```

Hide the label from the upper-casing command.

```
            \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
            \@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}%
          }%
          {%
```

This little trick is used by glossaries to allow the user to redefine \glsnamefont to use \makefirstuc. Support it even though they can now use the firstuc attribute.

```
            \letcs{\glo@name}{glo@\glsdetoklabel{#1}@name}%
            \expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}%
          }%
        }%
      }%
```

Do post-name hook.

```
      \glsxtrpostnamehook{#1}%
    }%
  }
}
```

\Glossentryname Redefine to set the abbreviation format and accessibility support.

```
\@ifpackageloaded{glossaries-accsupp}
{
  \renewcommand*{\Glossentryname}[1]{%
    \@glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
```

As from version 1.04, allow the glossnamefont attribute to determine the font applied.

```
      \glshasattribute{#1}{glossnamefont}%
      {%
```

```
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          `\@glsxtr@attrval' supplied in glossnamefont attribute
          for entry `#1'. Reverting to default \string\glsnamefont}%
          \let\@glsxtr@glossnamefont\glsnamefont
        }%
      }%
      {\let\@glsxtr@glossnamefont\glsnamefont}%
      \glsnameaccessdisplay
      {%
        \@glsxtr@glossnamefont{\Glsentryname{#1}}%
      }%
      {#1}%
```

Do post-name hook:

```
      \glsxtrpostnamehook{#1}%
    }%
  }
}
{
  \renewcommand*{\Glossentryname}[1]{%
    \@glsdoifexistsorwarn{#1}%
    {%
      \glssetabbrvfmt{\glscategory{#1}}%
      \glshasattribute{#1}{glossnamefont}%
      {%
        \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
        \ifcsdef{\@glsxtr@attrval}%
        {%
          \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
        }%
        {%
          \GlossariesExtraWarning{Unknown control sequence name
          `\@glsxtr@attrval' supplied in glossnamefont attribute
          for entry `#1'. Reverting to default \string\glsnamefont}%
          \let\@glsxtr@glossnamefont\glsnamefont
        }%
      }%
      {\let\@glsxtr@glossnamefont\glsnamefont}%
      \@glsxtr@glossnamefont{\Glsentryname{#1}}%
```

Do post-name hook:

```
      \glsxtrpostnamehook{#1}%
    }%
  }
```

```
}
```

Provide a convenient way to also index the entries using the standard `\index` mechanism. This may use different actual, encap and escape characters to those used for the glossaries.

`xtrpostnamehook` Hook to append stuff after the name is displayed in the glossary. The argument is the entry's label.

```
\newcommand*{\glsxtrpostnamehook}[1]{%
  \let\@glsnumberformat\@glsxtr@defaultnumberformat
  \glsxtrdoautoindexname{#1}{indexname}%
```

Allow additional code regardless of category:

```
  \glsextrapostnamehook{#1}%
```

Allow categories to hook in here.

```
  \csuse{glsxtrpostname\glscategory{#1}}%
}
```

`trapostnamehook`

```
\newcommand*{\glsextrapostnamehook}[1]{}%
```

`\glsdefpostname` Provide a convenient command for defining the post-name hook for the given category.

```
\newcommand*{\glsdefpostname}[2]{%
  \csdef{glsxtrpostname#1}{#2}%
}
```

`etaccessdisplay`

```
\@ifpackageloaded{glossaries-accsupp}
{
  \newcommand*{\glsxtr@setaccessdisplay}[1]{%
     \ifcsdef{gls#1accessdisplay}%
     {\letcs\@glsxtr@accessdisplay{gls#1accessdisplay}}%
     {%
```

This is essentially the reverse of `\@gls@fetchfield`, since the field supplied to `\glossentryname` has to be the internal label, but the `\gls`⟨*field*⟩`accessdisplay` commands use the key name.

```
       \edef\@gls@thisval{#1}%
       \@for\@gls@map:=\@gls@keymap\do{%
        \edef\@this@key{\expandafter\@secondoftwo\@gls@map}%
        \ifdefequal{\@this@key}{\@gls@thisval}%
        {%
          \edef\@gls@thisval{\expandafter\@firstoftwo\@gls@map}%
          \@endfortrue
        }%
        {}%
       }%
       \ifcsdef{gls\@gls@thisval accessdisplay}%
       {\letcs\@glsxtr@accessdisplay{gls\@gls@thisval accessdisplay}}%
```

```
      {\let\@glsxtr@accessdisplay\@firstoftwo}%
    }%
  }
}
{%
  \newcommand*{\glsxtr@setaccessdisplay}[1]{%
   \let\@glsxtr@accessdisplay\@firstoftwo}
}
```

sentrynameother Provide a command that works like \glossentryname but accesses a different field (which must be supplied using its internal field label).

```
\newrobustcmd*{\glossentrynameother}[2]{%
  \@glsdoifexistsorwarn{#1}%
  {%
```

Accessibility support:

```
    \glsxtr@setaccessdisplay{#2}%
```

Set the abbreviation format:

```
    \glssetabbrvfmt{\glscategory{#1}}%
    \glshasattribute{#1}{glossnamefont}%
    {%
      \edef\@glsxtr@attrval{\glsgetattribute{#1}{glossnamefont}}%
      \ifcsdef{\@glsxtr@attrval}%
      {%
        \letcs{\@glsxtr@glossnamefont}{\@glsxtr@attrval}%
      }%
      {%
        \GlossariesExtraWarning{Unknown control sequence name
        `\@glsxtr@attrval' supplied in glossnamefont attribute
        for entry `#1'. Reverting to default \string\glsnamefont}%
        \let\@glsxtr@glossnamefont\glsnamefont
      }%
    }%
    {\let\@glsxtr@glossnamefont\glsnamefont}%
    \glsifattribute{#1}{glossname}{firstuc}%
    {%
      \@glsxtr@accessdisplay
      {\@glsxtr@glossnamefont{\@Gls@entry@field{#1}{#2}}}%
      {#1}%
    }%
    {%
      \glsifattribute{#1}{glossname}{title}%
      {%
        \@glsxtr@do@titlecaps@warn
        \@glsxtr@accessdisplay
        {\@glsxtr@glossnamefont{\glsxtrfieldtitlecase{#1}{#2}}}%
        {#1}%
      }%
      {%
```

```
        \glsifattribute{#1}{glossname}{uc}%
        {%
          \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
          \@glsxtr@accessdisplay
          {\@glsxtr@glossnamefont{\mfirstucMakeUppercase{\glo@name}}}%
          {#1}%
        }%
        {%
          \letcs{\glo@name}{glo@\glsdetoklabel{#1}@#2}%
          \@glsxtr@accessdisplay
          {\expandafter\@glsxtr@glossnamefont\expandafter{\glo@name}}%
          {#1}%
        }%
      }%
    }%
```

Do post-name hook.

```
      \glsxtrpostnamehook{#1}%
  }%
}
```

`format@override` Determines if the format key should override the indexing attribute value.

```
\newif\if@glsxtr@format@override
\@glsxtr@format@overridefalse
```

If overriding is enabled, the `\glshypernumber` command will have to be redefined in the index to use `\hyperpage` instead.

`xFormatOverride`

```
\@ifpackageloaded{hyperref}
{
```

If hyperref's hyperindex option is on, then hyperref will automatically add `\hyperpage`, so don't add it.

```
  \ifHy@hyperindex
    \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
      \@glsxtr@format@overridetrue
      \appto\theindex{\let\glshypernumber\@firstofone}%
    }
  \else
    \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
      \@glsxtr@format@overridetrue
      \appto\theindex{\let\glshypernumber\hyperpage}%
    }
  \fi
}
{
  \newcommand*{\GlsXtrEnableIndexFormatOverride}{%
    \@glsxtr@format@overridetrue
  }
```

```
}
\@onlypreamble\GlsXtrEnableIndexFormatOverride
```

doautoindexname

```
\newcommand*{\glsxtrdoautoindexname}[2]{%
  \glshasattribute{#1}{#2}%
  {%
```

Escape any makeindex/xindy characters in the value of the name field. Take care with babel as this won't work if the category code has changed for those characters.

```
    \@glsxtr@autoindex@setname{#1}%
```

If the attribute value is simply "true" don't add an encap, otherwise use the value as the encap.

```
    \protected@edef\@glsxtr@attrval{\glsgetattribute{#1}{#2}}%
    \if@glsxtr@format@override
      \ifx\@glsnumberformat\@glsxtr@defaultnumberformat
      \else
        \let\@glsxtr@attrval\@glsnumberformat
      \fi
    \fi
    \ifdefstring{\@glsxtr@attrval}{true}%
    {}%
    {\eappto\@glo@name{\@glsxtr@autoindex@encap\@glsxtr@attrval}}%
    \expandafter\glsxtrautoindex\expandafter{\@glo@name}%
  }%
  {}%
}
```

glsxtrautoindex

```
\newcommand*{\glsxtrautoindex}{\index}
```

xtrautoindexesc

```
\newcommand{\glsxtrautoindexesc}{%
  \@gls@checkmkidxchars\@glo@sort
  \@glsxtr@autoindex@doextra@esc\@glo@sort
}
```

toindex@setname Assign \@glo@name for use with indexname attribute.

```
\newcommand*{\@glsxtr@autoindex@setname}[1]{%
  \protected@edef\@glo@name{\glsxtrautoindexentry{#1}}%
  \glsxtrautoindexassignsort{\@glo@sort}{#1}%
  \glsxtrautoindexesc
  \epreto\@glo@name{\@glo@sort\@glsxtr@autoindex@at}%
}
```

rautoindexentry Command used for the actual part when auto-indexing.

```
\newcommand*{\glsxtrautoindexentry}[1]{\string\glsentryname{#1}}
```

indexassignsort Used to assign the sort value when auto-indexing.

```
\newcommand*{\glsxtrautoindexassignsort}[2]{%
  \glsletentryfield{#1}{#2}{sort}%
}
```

dex@doextra@esc

```
\newcommand*{\@glsxtr@autoindex@doextra@esc}[1]{%
```

Escape the escape character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@esc\@gls@quotechar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@escquote\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@esc\noexpand\@nnil
        \@glsxtr@autoindex@esc\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
```

Escape actual character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@at\@gls@actualchar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@escat\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@at\noexpand\@nnil
        \@glsxtr@autoindex@at\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
```

Escape level character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@level\@gls@levelchar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@esclevel\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@level\noexpand\@nnil
        \@glsxtr@autoindex@level\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
```

Escape encap character unless it has already been escaped.

```
  \ifx\@glsxtr@autoindex@encap\@gls@encapchar
  \else
    \def\@gls@checkedmkidx{}%
    \edef\@@glsxtr@checkspch{%
      \noexpand\@glsxtr@autoindex@escencap\expandonce{#1}%
        \noexpand\@empty\@glsxtr@autoindex@encap\noexpand\@nnil
```

```
        \@glsxtr@autoindex@encap\noexpand\@empty\noexpand\@glsxtr@endescspch}%
    \@@glsxtr@checkspch
    \let#1\@gls@checkedmkidx\relax
  \fi
}
```

The user commands here have a preamble-only restriction to ensure they are set before required and also to reduce the chances of complications caused by babel's shorthands.

`tr@autoindex@at` Actual character for use with `\index`.

```
\newcommand*{\@glsxtr@autoindex@at}{}
```

`trSetActualChar` Set the actual character.

```
\newcommand*{\GlsXtrSetActualChar}[1]{%
  \gdef\@glsxtr@autoindex@at{#1}%
  \def\@glsxtr@autoindex@escat##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escat}{##1}{##2}{##3}%
  }%
}
\@onlypreamble\GlsXtrSetActualChar
\makeatother
\GlsXtrSetActualChar{@}
\makeatletter
```

`autoindex@encap` Encap character for use with `\index`.

```
\newcommand*{\@glsxtr@autoindex@encap}{}
```

`XtrSetEncapChar` Set the encap character.

```
\newcommand*{\GlsXtrSetEncapChar}[1]{%
  \gdef\@glsxtr@autoindex@encap{#1}%
  \def\@glsxtr@autoindex@escencap##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escencap}{##1}{##2}{##3}%
  }%
}
\GlsXtrSetEncapChar{|}
\@onlypreamble\GlsXtrSetEncapChar
```

`autoindex@level` Level character for use with `\index`.

```
\newcommand*{\@glsxtr@autoindex@level}{}
```

`XtrSetLevelChar` Set the encap character.

```
\newcommand*{\GlsXtrSetLevelChar}[1]{%
  \gdef\@glsxtr@autoindex@level{#1}%
  \def\@glsxtr@autoindex@esclevel##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@esclevel}{##1}{##2}{##3}%
  }%
}
\GlsXtrSetLevelChar{!}
\@onlypreamble\GlsXtrSetLevelChar
```

r@autoindex@esc Escape character for use with \index.

```
\newcommand*{\@glsxtr@autoindex@esc}{"}
```

lsXtrSetEscChar Set the escape character.

```
\newcommand*{\GlsXtrSetEscChar}[1]{%
  \gdef\@glsxtr@autoindex@esc{#1}%
  \def\@glsxtr@autoindex@escquote##1#1##2#1##3\@glsxtr@endescspch{%
    \@@glsxtr@autoindex@escspch{#1}{\@glsxtr@autoindex@escquote}{##1}{##2}{##3}%
  }%
}
\GlsXtrSetEscChar{"}
\@onlypreamble\GlsXtrSetEscChar
```

Set if defined. (For example, if doc package has been loaded.) Actual character \actualchar:

```
\ifdef\actualchar
 {\expandafter\GlsXtrSetActualChar\expandafter{\actualchar}}
 {}
```

Quote character \quotechar:

```
\ifdef\quotechar
 {\expandafter\GlsXtrSetEscChar\expandafter{\quotechar}}
 {}
```

Level character \levelchar:

```
\ifdef\levelchar
 {\expandafter\GlsXtrSetLevelChar\expandafter{\levelchar}}
 {}
```

Encap character \encapchar:

```
\ifdef\encapchar
 {\expandafter\GlsXtrSetEncapChar\expandafter{\encapchar}}
 {}
```

leto@endescspch

```
\def\@glsxtr@gobbleto@endescspch#1\@glsxtr@endescspch{}
```

toindex@esc@spch `\@@glsxtr@autoindex@escspch{⟨char⟩}{⟨cs⟩}{⟨pre⟩}{⟨mid⟩}{⟨post⟩}`

```
\newcommand*{\@@glsxtr@autoindex@escspch}[5]{%
  \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
  \toks@={#3}%
  \ifx\@nnil#3\relax
    \def\@@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch#5\@glsxtr@endescspch}%
  \else
    \ifx\@nnil#4\relax
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
     \def\@@glsxtr@checkspch{\@glsxtr@gobbleto@endescspch
```

```
        #4#5\@glsxtr@endescspch}%
    \else
      \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
       \@glsxtr@autoindex@esc#1}%
      \def\@@glsxtr@checkspch{#2#5#1\@nnil#1\@glsxtr@endescspch}%
    \fi
  \fi
  \@@glsxtr@checkspch
}
```

\Glossentrydesc Redefine to set the abbreviation format and accessibility support.

```
\renewcommand*{\Glossentrydesc}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \glssetabbrvfmt{\glscategory{#1}}%
    \Glsaccessdesc{#1}%
  }%
}
```

lossentrysymbol Redefine to set the abbreviation format and accessibility support.

```
\renewcommand*{\glossentrysymbol}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \glssetabbrvfmt{\glscategory{#1}}%
    \glsaccesssymbol{#1}%
  }%
}
```

lossentrysymbol Redefine to set the abbreviation format and accessibility support.

```
\renewcommand*{\Glossentrysymbol}[1]{%
  \glsdoifexistsorwarn{#1}%
  {%
    \glssetabbrvfmt{\glscategory{#1}}%
    \Glsaccesssymbol{#1}%
  }%
}
```

Allow initials to be marked but only use the formatting for the tag in the glossary.

eInitialTagging Allow initial tagging. The first argument is a list of categories to apply this to. The second argument is the name of the command to use to tag the initials. This can't already be defined for safety unless the starred version is used.

```
\newcommand*{\GlsXtrEnableInitialTagging}{%
  \@ifstar\s@glsxtr@enabletagging\@glsxtr@enabletagging
}
\@onlypreamble\GlsXtrEnableInitialTagging
```

r@enabletagging Starred version undefines command.

```
\newcommand*{\s@glsxtr@enabletagging}[2]{%
  \undef#2%
  \@glsxtr@enabletagging{#1}{#2}%
}
```

r@enabletagging Internal command.

```
\newcommand*{\@glsxtr@enabletagging}[2]{%
```

Set attributes for categories given in the first argument.

```
  \@for\@glsxtr@cat:=#1\do
  {%
    \ifdefempty\@glsxtr@cat
    {}%
    {\glssetcategoryattribute{\@glsxtr@cat}{tagging}{true}}%
  }%
  \newrobustcmd*#2[1]{##1}%
  \def\@glsxtr@taggingcs{#2}%
  \renewcommand*\@glsxtr@activate@initialtagging{%
    \let#2\@glsxtr@tag
  }%
  \ifundef\@gls@preglossaryhook
  {\GlossariesExtraWarning{Initial tagging requires at least
    glossaries.sty v4.19 to work correctly}}%
  {}%
}
```

Are we using an old version of mfirstuc that has a bug in `\capitalisewords`? If so, patch it so we don't have a problem with a combination of tagging and title case.

fu@checkword@do If this command hasn't been defined, then we have pre v2.02 of mfirstuc

```
\ifundef\mfu@checkword@do
{
  \newcommand*{\mfu@checkword@do}[1]{%
   \ifdefstring{\mfu@checkword@arg}{#1}%
   {%
     \let\@mfu@domakefirstuc\@firstofone
     \listbreak
   }%
   {}%
  }
```

\mfu@checkword `\capitalisewords` was introduced in mfirstuc v1.06. If `\mfu@checkword` hasn't been defined mfirstuc is too old to support the title case attribute.

```
  \ifundef\mfu@checkword
  {
    \newcommand{\@glsxtr@do@titlecaps@warn}{%
     \GlossariesExtraWarning{mfirstuc.sty too old. Title Caps
      support not available}%
```

One warning should suffice.

```
      \let\@glsxtr@do@titlecaps@warn\relax
    }
  }
  {
    \renewcommand*{\mfu@checkword}[1]{%
      \def\mfu@checkword@arg{#1}%
      \let\@mfu@domakefirstuc\makefirstuc
      \forlistloop\mfu@checkword@do\@mfu@nocaplist
    }
  }
}
{}% no patch required
```

`@titlecaps@warn` Do warning if title case not supported.

```
\newcommand*{\@glsxtr@do@titlecaps@warn}{}
```

`@initialtagging` Used in `\printglossary` but at least v4.19 of glossaries required.

```
\newcommand*\@glsxtr@activate@initialtagging{}
```

`\@glsxtr@tag` Definition of tagging command when used in glossary.

```
\newrobustcmd*{\@glsxtr@tag}[1]{%
  \glsifattribute{\glscurrententrylabel}{tagging}{true}%
  {\glsxtrtagfont{#1}}{#1}%
}
```

`\glsxtrtagfont` Used in the glossary.

```
\newcommand*{\glsxtrtagfont}[1]{\underline{#1}}
```

`preglossaryhook` This macro was introduced in glossaries version 4.19, so it may not be defined. If it hasn't been defined this feature is unavailable. A check is added for the entry's existence to prevent errors from occurring if the user removes an entry or changes the label, which can interrupt the build process.

```
\ifdef\@gls@preglossaryhook
{
  \renewcommand*{\@gls@preglossaryhook}{%
    \@glsxtr@activate@initialtagging
```

Since the glossaries are automatically scoped, `\@glsxtr@org@postdescription` shouldn't already be defined, but check anyway just as a precautionary measure.

```
    \ifundef\@glsxtr@org@postdescription
    {%
      \let\@glsxtr@org@postdescription\glspostdescription
      \renewcommand*{\glspostdescription}{%
        \ifglsentryexists{\glscurrententrylabel}%
        {%
          \glsxtrpostdescription
          \@glsxtr@org@postdescription
```

```
          }%
          {}%
        }%
      }%
      {}%
```

Enable the options used by `\@@glsxtrp`:

```
    \glossxtrsetpopts
  }%
}
{}
```

`postdescription` This command will only be used if `\@gls@preglossaryhook` is available *and* the glossary style uses `\glspostdescription` without modifying it. (`\nopostdesc` will suppress this.) The glossaries-extra-stylemods package will add the post description hook to all the predefined styles that don't include it.

```
\newcommand*{\glsxtrpostdescription}{%
  \csuse{glsxtrpostdesc\glscategory{\glscurrententrylabel}}%
}
```

`postdescgeneral`

```
\newcommand*{\glsxtrpostdescgeneral}{}
```

`xtrpostdescterm`

```
\newcommand*{\glsxtrpostdescterm}{}
```

`postdescacronym`

```
\newcommand*{\glsxtrpostdescacronym}{}
```

`escabbreviation`

```
\newcommand*{\glsxtrpostdescabbreviation}{}
```

`\glsdefpostdesc` Provide a convenient command for defining the post-description hook for the given category.

```
\newcommand*{\glsdefpostdesc}[2]{%
  \csdef{glsxtrpostdesc#1}{#2}%
}
```

`glspostlinkhook` Redefine the post link hook used by commands like `\gls` to make it easier for categories or attributes to modify this action. Since this hook occurs outside the existence check of commands like `\gls`, this needs to be checked again here. Do nothing if the entry hasn't been defined.

```
\renewcommand*{\glspostlinkhook}{%
 \ifglsentryexists{\glslabel}{\glsxtrpostlinkhook}{}%
}
```

`xtrpostlinkhook` The entry label should already be stored in `\glslabel` by `\@gls@link`.

```
\newcommand*{\glsxtrpostlinkhook}{%
 \glsxtrdiscardperiod{\glslabel}%
```

```
 {\glsxtrpostlinkendsentence}%
 {\glsxtrifcustomdiscardperiod
  {\glsxtrifperiod{\glsxtrpostlinkendsentence}{\glsxtrpostlink}}%
  {\glsxtrpostlink}%
 }%
}
```

omdiscardperiod Allow user to provide a custom check. Should expand to #2 if no check is required otherwise expand to #1.

```
\newcommand*{\glsxtrifcustomdiscardperiod}[2]{#2}
```

\glsxtrpostlink

```
\newcommand*{\glsxtrpostlink}{%
 \csuse{glsxtrpostlink\glscategory{\glslabel}}%
}
```

\glsdefpostlink Provide a convenient command for defining the post-link hook for the given category. Doesn't allow an empty argument (which) would overwrite \glsxtrpostlink.

```
\newcommand*{\glsdefpostlink}[2]{%
```

\ifthenelse is used to ensure that the expanded value is tested. (The category label must be fully expandable.)

```
  \ifthenelse{\equal{#1}{}}%
  {\PackageError{glossaries-extra}
    {Invalid empty category label in \string\glsdefpostlink}{}}%
  {\csdef{glsxtrpostlink#1}{#2}}%
}
```

linkendsentence Done by \glsxtrpostlinkhook if a full stop is discarded.

```
\newcommand*{\glsxtrpostlinkendsentence}{%
 \ifcsdef{glsxtrpostlink\glscategory{\glslabel}}
 {%
   \csuse{glsxtrpostlink\glscategory{\glslabel}}%
```

Put the full stop back.

```
   .\spacefactor\sfcode`\. \relax
 }%
 {%
```

Assume the full stop was discarded because the entry ends with a period, so adjust the space-factor.

```
   \spacefactor\sfcode`\. \relax
 }%
}
```

dDescOnFirstUse Provide a command for appending the description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
\newcommand*{\glsxtrpostlinkAddDescOnFirstUse}{%
  \glsxtrifwasfirstuse{\space\glsxtrparen{\glsaccessdesc{\glslabel}}}{}%
}
```

ymbolOnFirstUse Provide a command for appending the symbol (if defined) in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
\newcommand*{\glsxtrpostlinkAddSymbolOnFirstUse}{%
  \glsxtrifwasfirstuse
  {%
    \ifglshassymbol{\glslabel}%
    {\space\glsxtrparen{\glsaccesssymbol{\glslabel}}}%
    {}%
  }%
  {}%
}
```

lDescOnFirstUse Provide a command for appending the symbol (if defined) and description in parentheses on first use, for the convenience of users wanting to add this to the post link hook.

```
\newcommand*{\glsxtrpostlinkAddSymbolDescOnFirstUse}{%
  \glsxtrifwasfirstuse
  {%
    \space\glsxtrparen
    {%
      \ifglshassymbol{\glslabel}%
      {\glsaccesssymbol{\glslabel}, }%
      {}%
      \glsaccessdesc{\glslabel}%
    }%
  }%
  {}%
}
```

trdiscardperiod Discard following period (if present) if the discardperiod attribute is true. If a period is discarded, do the second argument otherwise do the third argument. The entry label is in the first argument. Since this is designed for abbreviations that end with a period, check if the plural form was used (which typically won't end with a period).

```
\newcommand*{\glsxtrdiscardperiod}[3]{%
 \glsxtrifwasfirstuse
 {%
   \glsifattribute{#1}{retainfirstuseperiod}{true}%
   {#3}%
   {%
     \glsifattribute{#1}{discardperiod}{true}%
     {%
       \glsifplural
       {%
         \glsifattribute{#1}{pluraldiscardperiod}{true}%
         {\glsxtrifperiod{#2}{#3}}%
         {#3}%
       }%
       {%
         \glsxtrifperiod{#2}{#3}%
```

```
      }%
    }%
    {#3}%
  }%
}%
{%
  \glsifattribute{#1}{discardperiod}{true}%
  {%
    \glsifplural
    {%
      \glsifattribute{#1}{pluraldiscardperiod}{true}%
      {\glsxtrifperiod{#2}{#3}}%
      {#3}%
    }%
    {%
      \glsxtrifperiod{#2}{#3}%
    }%
  }%
  {#3}%
}%
}
```

\glsxtrifperiod Make a convenient user command to check if the next character is a full stop (period). Works like \@ifstar but uses \new@ifnextchar rather than \@ifnextchar

```
\newcommand*{\glsxtrifperiod}[1]{\new@ifnextchar.{\@firstoftwo{#1}}}
```

Sometimes it's useful to test if there's a punctuation character following the glossary entry.

\glsxtr@punclist List of characters identified as punctuation marks. (Be careful of babel shorthands!) This doesn't allow for punctuation marks made up from multiple characters (such as ''').

```
\newcommand*{\glsxtr@punclist}{.,:;?!}
```

punctuationmark Add character to punctuation list.

```
\newcommand*{\glsxtraddpunctuationmark}[1]{\appto\glsxtr@punclist{#1}}
```

unctuationmarks Reset the punctuation list.

```
\newcommand*{\glsxtrsetpunctuationmarks}[1]{\def\glsxtr@punclist{#1}}
```

\glsxtrifpunc `\glsxtrifnextpunc{⟨true part⟩}{⟨false part⟩}`

Test if this is followed by a punctuation mark. (Adapted from \new@ifnextchar.)

```
\newcommand*{\glsxtrifnextpunc}[2]{%
  \def\reserved@a{#1}%
  \def\reserved@b{#2}%
  \futurelet\@glspunc@token\glsxtr@ifnextpunc
}
```

sxtr@ifnextpunc

```
\newcommand*{\glsxtr@ifnextpunc}{%
 \glsxtr@ifpunctoken{\@glspunc@token}{\let\reserved@b\reserved@a}{}%
 \reserved@b
}
```

xtr@ifpunctoken Test if the token given in the first argument is in the punctuation list.

```
\newcommand*{\glsxtr@ifpunctoken}[1]{%
  \expandafter\@glsxtr@ifpunctoken\expandafter#1\glsxtr@punclist\@nnil
}
```

xtr@ifpunctoken

```
\def\@glsxtr@ifpunctoken#1#2{%
  \let\reserved@d=#2%
  \ifx\reserved@d\@nnil
    \let\glsxtr@next\@glsxtr@notfoundinlist
  \else
    \ifx#1\reserved@d
     \let\glsxtr@next\@glsxtr@foundinlist
    \else
     \let\glsxtr@next\@glsxtr@ifpunctoken
    \fi
  \fi
  \glsxtr@next#1%
}
```

xtr@foundinlist

```
\def\@glsxtr@foundinlist#1\@nnil{\@firstoftwo}
```

@notfoundinlist

```
\def\@glsxtr@notfoundinlist#1{\@secondoftwo}
```

glsxtrdopostpunc `\glsxtrdopostpunc{⟨code⟩}`

If this is followed be a punctuation character, do ⟨*code*⟩ after the character otherwise do ⟨*code*⟩ before whatever comes next.

```
\newcommand{\glsxtrdopostpunc}[1]{%
  \glsxtrifnextpunc{\@glsxtr@swaptwo{#1}}{#1}%
}
```

@glsxtr@swaptwo

```
\newcommand{\@glsxtr@swaptwo}[2]{#2#1}
```

1.7 Abbreviations

The "acronym" code from glossaries is misnamed as it's more often used for other forms of abbreviations. This code corrects this inconsistency, but rather than just having synonyms, provide commands for abbreviations that have a similar, but not identical, underlying mechanism to acronyms.

If there's a style for the given category, apply it.

```
\define@key{glsxtrabbrv}{category}{%
 \edef\glscategorylabel{#1}%
 \ifcsdef{@glsabbrv@current@#1}%
 {%
```

Warning should already have been issued.

```
   \let\@glsxtr@orgwarndep\GlsXtrWarnDeprecatedAbbrStyle
   \let\GlsXtrWarnDeprecatedAbbrStyle\@gobbletwo
   \glsxtr@applyabbrvstyle{\csname @glsabbrv@current@#1\endcsname}%
   \let\GlsXtrWarnDeprecatedAbbrStyle\@glsxtr@orgwarndep
 }%
 {}%
}
```

Save the short plural form. This may be needed before the entry is defined.

```
\define@key{glsxtrabbrv}{shortplural}{%
  \def\@gls@shortpl{#1}%
}
```

Similarly for the long plural form.

```
\define@key{glsxtrabbrv}{longplural}{%
  \def\@gls@longpl{#1}%
}
```

Token registers for the short plural and long plural, provided for use in the abbreviation style definitions.

\glsshortpltok

```
\newtoks\glsshortpltok
```

\glslongpltok

```
\newtoks\glslongpltok
```

sxtr@insertdots Provided in case user wants to automatically insert dots between each letter of the abbreviation. This should be applied before defining the abbreviation to optimise the document build. (Otherwise, it would have to be done each time the short form is required, which is an unnecessary waste of time.) For this to work the short form must be expanded when passed to \newabbreviation. Note that explicitly using the short or shortplural keys will override this.

```
\newcommand*{\@glsxtr@insertdots}[2]{%
  \def#1{}%
  \@glsxtr@insert@dots#1#2\@nnil
}
```

xtr@insert@dots

```
\newcommand*{\@glsxtr@insert@dots}[2]{%
  \ifx\@nnil#2\relax
   \let\@glsxtr@insert@dots@next\@gobble
  \else
   \ifx\relax#2\relax
   \else
     \appto#1{#2.}%
   \fi
   \let\@glsxtr@insert@dots@next\@glsxtr@insert@dots
  \fi
  \@glsxtr@insert@dots@next#1%
}
```

Similarly provide a way of replacing spaces with \glsxtrwordsep, which first needs to be defined:

\glsxtrwordsep

```
\newcommand*{\glsxtrwordsep}{\space}
```

Each word is marked with

\glsxtrword

```
\newcommand*{\glsxtrword}[1]{#1}
```

tr@markwordseps

```
\newcommand*{\@glsxtr@markwordseps}[2]{%
  \def#1{}%
  \@glsxtr@mark@wordseps#1#2 \@nnil
}
```

r@mark@wordseps

```
\def\@glsxtr@mark@wordseps#1#2 #3{%
  \ifdefempty{#1}%
  {\def#1{\protect\glsxtrword{#2}}}%
  {\appto#1{\protect\glsxtrwordsep\protect\glsxtrword{#2}}}%
  \ifx\@nnil#3\relax
   \let\@glsxtr@mark@wordseps@next\relax
  \else
   \def\@glsxtr@mark@wordseps@next{%
     \@glsxtr@mark@wordseps#1#3}%
  \fi
  \@glsxtr@mark@wordseps@next
}
```

newabbreviation Define a new generic abbreviation.

```
\newcommand*{\newabbreviation}[4][]{%
  \glsxtr@newabbreviation{#1}{#2}{#3}{#4}%
}
```

newabbreviation Internal macro. (bib2gls has an option that needs to temporarily redefine \newabbreviation. This is just makes it easier to save and restore the original definition.)

```
\newcommand*{\glsxtr@newabbreviation}[4]{%
  \glskeylisttok{#1}%
  \glslabeltok{#2}%
  \glsshorttok{#3}%
  \glslongtok{#4}%
```

Save the original short and long values (before attribute settings modify them).

```
  \def\glsxtrorgshort{#3}%
  \def\glsxtrorglong{#4}%
```

Provide extra settings for hooks (if modified, this command must end with a comma).

```
  \def\ExtraCustomAbbreviationFields{}%
```

Initialise accessibility settings if required.

```
  \@gls@initaccesskeys
```

Get the category.

```
  \def\glscategorylabel{abbreviation}%
  \glsxtr@applyabbrvstyle{\@glsabbrv@current@abbreviation}%
```

Ignore the shortplural and longplural keys.

```
  \setkeys*{glsxtrabbrv}[shortplural,longplural]{#1}%
```

Set the default long plural

```
  \def\@gls@longpl{#4\glspluralsuffix}%
  \let\@gls@default@longpl\@gls@longpl
```

Has the markwords attribute been set?

```
  \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
  {%
    \@glsxtr@markwordseps\@gls@long{#4}%
    \expandafter\def\expandafter\@gls@longpl\expandafter
     {\@gls@long\glspluralsuffix}%
    \let\@gls@default@longpl\@gls@longpl
```

Update \glslongtok.

```
    \expandafter\glslongtok\expandafter{\@gls@long}%
  }%
  {}%
```

Has the markshortwords attribute been set? (Not compatible with insertdots.)

```
  \glsifcategoryattribute{\glscategorylabel}{markshortwords}{true}%
  {%
    \@glsxtr@markwordseps\@gls@short{#3}%
  }%
  {%
```

Has the insertdots attribute been set?

```
    \glsifcategoryattribute{\glscategorylabel}{insertdots}{true}%
    {%
      \@glsxtr@insertdots\@gls@short{#3}%
```

```
      \expandafter\glsshorttok\expandafter{\@gls@short\spacefactor1000 \relax}%
    }%
    {\def\@gls@short{#3}}%
  }%
```

Has the aposplural attribute been set? (Not compatible with noshortplural.)

```
  \glsifcategoryattribute{\glscategorylabel}{aposplural}{true}%
  {%
    \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
      '\abbrvpluralsuffix}%
  }%
  {%
```

Has the noshortplural attribute been set?

```
    \glsifcategoryattribute{\glscategorylabel}{noshortplural}{true}%
    {%
      \let\@gls@shortpl\@gls@short
    }%
      {%
      \expandafter\def\expandafter\@gls@shortpl\expandafter{\@gls@short
        \abbrvpluralsuffix}%
    }%
  }%
```

Update \glsshorttok:

```
  \expandafter\glsshorttok\expandafter{\@gls@short}%
```

Hook for further customisation if required:

```
  \glsxtrnewabbrevpresetkeyhook{#1}{#2}{#3}%
```

Get the short and long plurals provided by user in optional argument to override defaults, if necessary. Ignore the category key (already obtained).

```
  \setkeys*{glsxtrabbrv}[category]{#1}%
```

Has the plural been explicitly set?

```
  \ifx\@gls@default@longpl\@gls@longpl
  \else
```

Has the markwords attribute been set?

```
    \glsifcategoryattribute{\glscategorylabel}{markwords}{true}%
    {%
      \expandafter\@glsxtr@markwordseps\expandafter\@gls@longpl\expandafter
       {\@gls@longpl}%
    }%
    {}%
  \fi
```

Set the plural token registers so the values can be accessed by the abbreviation styles.

```
  \expandafter\glsshortpltok\expandafter{\@gls@shortpl}%
  \expandafter\glslongpltok\expandafter{\@gls@longpl}%
```

Hook for accessibility support (does nothing if glossaries-accsupp hasn't been loaded).

```
  \@gls@setup@default@short@access{#3}%
```

Do any extra setup provided by hook:

```
\newabbreviationhook
```

Define this entry:

```
\protected@edef\@do@newglossaryentry{%
  \noexpand\newglossaryentry{\the\glslabeltok}%
  {%
    type=\glsxtrabbrvtype,%
    category=abbreviation,%
    short={\the\glsshorttok},%
    shortplural={\the\glsshortpltok},%
    long={\the\glslongtok},%
    longplural={\the\glslongpltok},%
    name={\the\glsshorttok},%
    \CustomAbbreviationFields,%
```

Hook may override abbreviation style default settings (this hook must end with a comma if set).

```
    \ExtraCustomAbbreviationFields
```

Any explicit fields set in the optional argument override all other settings.

```
    \the\glskeylisttok
  }%
}%
\@do@newglossaryentry
\GlsXtrPostNewAbbreviation
}
```

`evpresetkeyhook` Hook for extra stuff in `\newabbreviation`

```
\newcommand*{\glsxtrnewabbrevpresetkeyhook}[3]{}
```

`NewAbbreviation` Hook used by abbreviation styles.

```
\newcommand*{\GlsXtrPostNewAbbreviation}{}
```

`bbreviationhook` Hook for use with `\newabbreviation`.

```
\newcommand*{\newabbreviationhook}{}
```

`reviationFields`

```
\newcommand*{\CustomAbbreviationFields}{}
```

`\glsxtrparen` For the parenthetical styles.

```
\newcommand*{\glsxtrparen}[1]{(#1)}
```

`lsxtrfullformat` Full format without case change.

```
\newcommand*{\glsxtrfullformat}[2]{%
  \glsfirstlongfont{\glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
  \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
}
```

lsxtrfullformat Full format with case change.

```
\newcommand*{\Glsxtrfullformat}[2]{%
  \glsfirstlongfont{\Glsaccesslong{#1}}#2\glsxtrfullsep{#1}%
  \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{#1}}}%
}
```

xtrfullplformat Plural full format without case change.

```
\newcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstlongfont{\glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
  \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}}}%
}
```

xtrfullplformat Plural full format with case change.

```
\newcommand*{\Glsxtrfullplformat}[2]{%
  \glsfirstlongfont{\Glsaccesslongpl{#1}}#2\glsxtrfullsep{#1}%
  \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{#1}}}%
}
```

\glsxtrfullsep Separator used by full format is a space by default. The argument is the entry's label.

```
\newcommand*{\glsxtrfullsep}[1]{\space}
```

In-line formats in case first use isn't compatible with \glsentryfull (for example, first use suppresses the long form or uses a footnote).

nlinefullformat Full format without case change.

```
\newcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}
```

nlinefullformat Full format with case change.

```
\newcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}
```

xtrfullplformat Plural full format without case change.

```
\newcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}
```

inefullplformat Plural full format with case change.

```
\newcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}
```

Redefine \glsentryfull etc to use the inline format. Since these commands as supposed to be expandable, they can only use the currently applied style. If there are mixed styles, you'll need to use the \glsxtrfull set of commands instead.

\glsentryfull

```
\renewcommand*{\glsentryfull}[1]{\glsxtrinlinefullformat{#1}{}}
```

\Glsentryfull

```
\renewcommand*{\Glsentryfull}[1]{\Glsxtrinlinefullformat{#1}{}}
```

\glsentryfullpl

```
\renewcommand*{\glsentryfullpl}[1]{\glsxtrinlinefullplformat{#1}{}}
```

\Glsentryfullpl

```
\renewcommand*{\Glsentryfullpl}[1]{\Glsxtrinlinefullplformat{#1}{}}
```

sfirstabbrvfont Font changing command used for the abbreviation on first use or in the full format.

```
\newcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{#1}}
```

bbrvdefaultfont Font changing command used for the abbreviation on first use or in the full format.

```
\newcommand*{\glsfirstabbrvdefaultfont}[1]{\glsabbrvfont{#1}}
```

\glsabbrvfont Font changing command used for the abbreviation on subsequent use.

```
\newcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{#1}}
```

bbrvdefaultfont

```
\newcommand*{\glsabbrvdefaultfont}[1]{#1}
```

\glslongfont Font changing command used for the long form in commands like \glsxtrlong.

```
\newcommand*{\glslongfont}[1]{\glslongdefaultfont{#1}}
```

longdefaultfont Default font changing command used for the long form in commands like \glsxtrlong.

```
\newcommand*{\glslongdefaultfont}[1]{#1}
```

lsfirstlongfont Font changing command used for the long form on first use or in the full format.

```
\newcommand*{\glsfirstlongfont}[1]{\glslongfont{#1}}
```

longdefaultfont

```
\newcommand*{\glsfirstlongdefaultfont}[1]{\glslongdefaultfont{#1}}
```

brvpluralsuffix Default plural suffix. Allow an alternative default suffix for abbreviations.

```
\newcommand*{\glsxtrabbrvpluralsuffix}{\glspluralsuffix}
```

brvpluralsuffix Default plural suffix.

```
\newcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtrfull Full form (no case-change).

```
\newrobustcmd*{\glsxtrfull}{\@gls@hyp@opt\ns@glsxtrfull}
\newcommand*\ns@glsxtrfull[2][]{%
  \new@ifnextchar[{\@glsxtr@full{#1}{#2}}%
                  {\@glsxtr@full{#1}{#2}[]}%
}
```

\@glsxtr@full Low-level macro:

```
\def\@glsxtr@full#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\glsxtrinlinefullformat{#2}{#3}}%
```

What should \glsxtrifwasfirstuse be set to here? Where the inline and display full forms are the same, this is essentially emulating first use, to it make sense for the postlink hook to pretend it was a first use instance. It makes less sense if the inline and display forms are different. Provide a hook to make it easier to reconfigure.

```
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

trsetupfulldefs

```
\newcommand*{\glsxtrsetupfulldefs}{%
  \let\glsxtrifwasfirstuse\@firstoftwo
}
```

\Glsxtrfull Full form (first letter uppercase).

```
\newrobustcmd*{\Glsxtrfull}{\@gls@hyp@opt\ns@Glsxtrfull}
\newcommand*\ns@Glsxtrfull[2][]{%
  \new@ifnextchar[{\@Glsxtr@full{#1}{#2}}%
                  {\@Glsxtr@full{#1}{#2}[]}%
}
```

\@Glsxtr@full Low-level macro:

```
\def\@Glsxtr@full#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\Glsxtrinlinefullformat{#2}{#3}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\GLSxtrfull Full form (all uppercase).

```
\newrobustcmd*{\GLSxtrfull}{\@gls@hyp@opt\ns@GLSxtrfull}
\newcommand*\ns@GLSxtrfull[2][]{%
  \new@ifnextchar[{\@GLSxtr@full{#1}{#2}}%
                  {\@GLSxtr@full{#1}{#2}[]}%
}
```

\@GLSxtr@full Low-level macro:

```
\def\@GLSxtr@full#1#2[#3]{%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\mfirstucMakeUppercase{\glsxtrinlinefullformat{#2}{#3}}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\glsxtrfullpl Plural full form (no case-change).

```
\newrobustcmd*{\glsxtrfullpl}{\@gls@hyp@opt\ns@glsxtrfullpl}
\newcommand*\ns@glsxtrfullpl[2][]{%
  \new@ifnextchar[{\@glsxtr@fullpl{#1}{#2}}%
                  {\@glsxtr@fullpl{#1}{#2}[]}%
}
```

\@glsxtr@fullpl Low-level macro:

```
\def\@glsxtr@fullpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\glsxtrinlinefullplformat{#2}{#3}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\Glsxtrfullpl Plural full form (first letter uppercase).

```
\newrobustcmd*{\Glsxtrfullpl}{\@gls@hyp@opt\ns@Glsxtrfullpl}
\newcommand*\ns@Glsxtrfullpl[2][]{%
  \new@ifnextchar[{\@Glsxtr@fullpl{#1}{#2}}%
                  {\@Glsxtr@fullpl{#1}{#2}[]}%
}
```

\@Glsxtr@fullpl Low-level macro:

```
\def\@Glsxtr@fullpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{\Glsxtrinlinefullplformat{#2}{#3}}%
    \glsxtrsetupfulldefs
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\GLSxtrfullpl Plural full form (all upper case).

```
\newrobustcmd*{\GLSxtrfullpl}{\@gls@hyp@opt\ns@GLSxtrfullpl}
\newcommand*\ns@GLSxtrfullpl[2][]{%
  \new@ifnextchar[{\@GLSxtr@fullpl{#1}{#2}}%
                  {\@GLSxtr@fullpl{#1}{#2}[]}%
}
```

\@GLSxtr@fullpl Low-level macro:

```
\def\@GLSxtr@fullpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase{\glsxtrinlinefullplformat{#2}{#3}}}%
    \glsxtrsetupfulldefs
```

```
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

The short and long forms work in a similar way to acronyms.

\glsxtrshort

```
\newrobustcmd*{\glsxtrshort}{\@gls@hyp@opt\ns@glsxtrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@glsxtrshort}[2][]{%
  \new@ifnextchar[{\@glsxtrshort{#1}{#2}}{\@glsxtrshort{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@glsxtrshort#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
```

Need to make sure \glsabbrvfont is set correctly.

```
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glsabbrvfont{\glsaccessshort{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\Glsxtrshort

```
\newrobustcmd*{\Glsxtrshort}{\@gls@hyp@opt\ns@Glsxtrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@Glsxtrshort}[2][]{%
  \new@ifnextchar[{\@Glsxtrshort{#1}{#2}}{\@Glsxtrshort{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Glsxtrshort#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glsabbrvfont{\Glsaccessshort{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\GLSxtrshort

```
\newrobustcmd*{\GLSxtrshort}{\@gls@hyp@opt\ns@GLSxtrshort}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@GLSxtrshort}[2][]{%
  \new@ifnextchar[{\@GLSxtrshort{#1}{#2}}{\@GLSxtrshort{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@GLSxtrshort#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase
      {\glsabbrvfont{\glsaccessshort{#2}\ifglsxtrinsertinside#3\fi}%
        \ifglsxtrinsertinside\else#3\fi
      }%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
```

```
  \glspostlinkhook
}
```

\glsxtrlong

```
\newrobustcmd*{\glsxtrlong}{\@gls@hyp@opt\ns@glsxtrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@glsxtrlong}[2][]{%
  \new@ifnextchar[{\@glsxtrlong{#1}{#2}}{\@glsxtrlong{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@glsxtrlong#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glslongfont{\glsaccesslong{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\Glsxtrlong

```
\newrobustcmd*{\Glsxtrlong}{\@gls@hyp@opt\ns@Glsxtrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@Glsxtrlong}[2][]{%
  \new@ifnextchar[{\@Glsxtrlong{#1}{#2}}{\@Glsxtrlong{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Glsxtrlong#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
```

```
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glslongfont{\Glsaccesslong{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\GLSxtrlong

```
\newrobustcmd*{\GLSxtrlong}{\@gls@hyp@opt\ns@GLSxtrlong}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@GLSxtrlong}[2][]{%
  \new@ifnextchar[{\@GLSxtrlong{#1}{#2}}{\@GLSxtrlong{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@GLSxtrlong#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@secondoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
     \mfirstucMakeUppercase
     {\glslongfont{\glsaccesslong{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
     }%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

Plural short forms:

\glsxtrshortpl

```
\newrobustcmd*{\glsxtrshortpl}{\@gls@hyp@opt\ns@glsxtrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@glsxtrshortpl}[2][]{%
```

```
  \new@ifnextchar[{\@glsxtrshortpl{#1}{#2}}{\@glsxtrshortpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@glsxtrshortpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\Glsxtrshortpl

```
\newrobustcmd*{\Glsxtrshortpl}{\@gls@hyp@opt\ns@Glsxtrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@Glsxtrshortpl}[2][]{%
  \new@ifnextchar[{\@Glsxtrshortpl{#1}{#2}}{\@Glsxtrshortpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Glsxtrshortpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glsabbrvfont{\Glsaccessshortpl{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
```

```
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\GLSxtrshortpl

```
\newrobustcmd*{\GLSxtrshortpl}{\@gls@hyp@opt\ns@GLSxtrshortpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@GLSxtrshortpl}[2][]{%
  \new@ifnextchar[{\@GLSxtrshortpl{#1}{#2}}{\@GLSxtrshortpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@GLSxtrshortpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \glssetabbrvfmt{\glscategory{#2}}%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase
      {\glsabbrvfont{\glsaccessshortpl{#2}\ifglsxtrinsertinside#3\fi}%
       \ifglsxtrinsertinside\else#3\fi
      }%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

Plural long forms:

\glsxtrlongpl

```
\newrobustcmd*{\glsxtrlongpl}{\@gls@hyp@opt\ns@glsxtrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@glsxtrlongpl}[2][]{%
  \new@ifnextchar[{\@glsxtrlongpl{#1}{#2}}{\@glsxtrlongpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@glsxtrlongpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@firstofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glslongfont{\glsaccesslongpl{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\Glsxtrlongpl

```
\newrobustcmd*{\Glsxtrlongpl}{\@gls@hyp@opt\ns@Glsxtrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@Glsxtrlongpl}[2][]{%
  \new@ifnextchar[{\@Glsxtrlongpl{#1}{#2}}{\@Glsxtrlongpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@Glsxtrlongpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@secondofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \glslongfont{\Glsaccesslongpl{#2}\ifglsxtrinsertinside#3\fi}%
      \ifglsxtrinsertinside\else#3\fi
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

\GLSxtrlongpl

```
\newrobustcmd*{\GLSxtrlongpl}{\@gls@hyp@opt\ns@GLSxtrlongpl}
```

Define the un-starred form. Need to determine if there is a final optional argument

```
\newcommand*{\ns@GLSxtrlongpl}[2][]{%
  \new@ifnextchar[{\@GLSxtrlongpl{#1}{#2}}{\@GLSxtrlongpl{#1}{#2}[]}%
}
```

Read in the final optional argument:

```
\def\@GLSxtrlongpl#1#2[#3]{%
```

If the record option has been used, the information needs to be written to the aux file regardless of whether the entry exists (unless indexing has been switched off).

```
  \@glsxtr@record{#1}{#2}{glslink}%
  \glsdoifexists{#2}%
  {%
    \let\do@gls@link@checkfirsthyper\@gls@link@nocheckfirsthyper
    \let\glsxtrifwasfirstuse\@secondoftwo
    \let\glsifplural\@firstoftwo
    \let\glscapscase\@thirdofthree
    \let\glsinsert\@empty
    \def\glscustomtext{%
      \mfirstucMakeUppercase
      {\glslongfont{\glsaccesslongpl{#2}\ifglsxtrinsertinside#3\fi}%
       \ifglsxtrinsertinside\else#3\fi
      }%
    }%
    \@gls@link[#1]{#2}{\csname gls@\glstype @entryfmt\endcsname}%
  }%
  \glspostlinkhook
}
```

`\glssetabbrvfmt` Set the current format for the given category (or the abbreviation category if unset).

```
\newcommand*{\glssetabbrvfmt}[1]{%
  \ifcsdef{@glsabbrv@current@#1}%
  {\glsxtr@applyabbrvfmt{\csname @glsabbrv@current@#1\endcsname}}%
  {\glsxtr@applyabbrvfmt{\@glsabbrv@current@abbreviation}}%
}
```

`glsuseabbrvfont` Provide a way to use the abbreviation font for a given category for arbitrary text.

```
\newrobustcmd*{\glsuseabbrvfont}[2]{{\glssetabbrvfmt{#2}\glsabbrvfont{#1}}}
```

`\glsuselongfont` Provide a way to use the long font for a given category for arbitrary text.

```
\newrobustcmd*{\glsuselongfont}[2]{{\glssetabbrvfmt{#2}\glslongfont{#1}}}
```

`sxtrgenabbrvfmt` Similar to `\glsgenacfmt`, but for abbreviations.

```
\newcommand*{\glsxtrgenabbrvfmt}{%
  \ifdefempty\glscustomtext
  {%
    \ifglsused\glslabel
    {%
```

```
  Subsequent use:
      \glsifplural
      {%
  Subsequent plural form:
        \glscapscase
        {%
  Subsequent plural form, don't adjust case:
          \glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
        }%
        {%
  Subsequent plural form, make first letter upper case:
          \Glsxtrsubsequentplfmt{\glslabel}{\glsinsert}%
        }%
        {%
  Subsequent plural form, all caps:
          \mfirstucMakeUppercase
            {\glsxtrsubsequentplfmt{\glslabel}{\glsinsert}}%
        }%
      }%
      {%
  Subsequent singular form
        \glscapscase
        {%
  Subsequent singular form, don't adjust case:
          \glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
        }%
        {%
  Subsequent singular form, make first letter upper case:
          \Glsxtrsubsequentfmt{\glslabel}{\glsinsert}%
        }%
        {%
  Subsequent singular form, all caps:
          \mfirstucMakeUppercase
            {\glsxtrsubsequentfmt{\glslabel}{\glsinsert}}%
        }%
      }%
    }%
    {%
  First use:
      \glsifplural
      {%
  First use plural form:
        \glscapscase
        {%
```

First use plural form, don't adjust case:

```
            \glsxtrfullplformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use plural form, make first letter upper case:

```
            \Glsxtrfullplformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use plural form, all caps:

```
            \mfirstucMakeUppercase
              {\glsxtrfullplformat{\glslabel}{\glsinsert}}%
          }%
        }%
        {%
```

First use singular form

```
          \glscapscase
          {%
```

First use singular form, don't adjust case:

```
            \glsxtrfullformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use singular form, make first letter upper case:

```
            \Glsxtrfullformat{\glslabel}{\glsinsert}%
          }%
          {%
```

First use singular form, all caps:

```
            \mfirstucMakeUppercase
             {\glsxtrfullformat{\glslabel}{\glsinsert}}%
          }%
        }%
      }%
    }%
    {%
```

User supplied text.

```
      \glscustomtext
    }%
}
```

`trsubsequentfmt` Subsequent use format (singular no case change).

```
\newcommand*{\glsxtrsubsequentfmt}[2]{%
  \glsabbrvfont{\glsaccessshort{#1}\ifglsxtrinsertinside #2\fi}%
  \ifglsxtrinsertinside \else#2\fi
}
\let\glsxtrdefaultsubsequentfmt\glsxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural no case change).

```
\newcommand*{\glsxtrsubsequentplfmt}[2]{%
  \glsabbrvfont{\glsaccessshortpl{#1}\ifglsxtrinsertinside #2\fi}%
  \ifglsxtrinsertinside \else#2\fi
}
\let\glsxtrdefaultsubsequentplfmt\glsxtrsubsequentplfmt
```

trsubsequentfmt Subsequent use format (singular, first letter uppercase).

```
\newcommand*{\Glsxtrsubsequentfmt}[2]{%
  \glsabbrvfont{\Glsaccessshort{#1}\ifglsxtrinsertinside #2\fi}%
  \ifglsxtrinsertinside \else#2\fi
}
\let\Glsxtrdefaultsubsequentfmt\Glsxtrsubsequentfmt
```

subsequentplfmt Subsequent use format (plural, first letter uppercase).

```
\newcommand*{\Glsxtrsubsequentplfmt}[2]{%
  \glsabbrvfont{\Glsaccessshortpl{#1}\ifglsxtrinsertinside #2\fi}%
  \ifglsxtrinsertinside \else#2\fi
}
\let\Glsxtrdefaultsubsequentplfmt\Glsxtrsubsequentplfmt
```

1.7.1 Abbreviation Styles Setup

breviationstyle

```
\newcommand*{\setabbreviationstyle}[2][abbreviation]{%
  \ifcsundef{@glsabbrv@dispstyle@setup@#2}
  {%
    \PackageError{glossaries-extra}{Undefined abbreviation style `#2'}{}%
  }%
  {%
```

Have abbreviations already been defined for this category?

```
    \ifcsstring{@glsabbrv@current@#1}{#2}%
    {%
```

Style already set.

```
    }%
    {%
      \def\@glsxtr@dostylewarn{}%
      \glsforeachincategory{#1}{\@gls@type}{\@gls@label}%
      {%
        \def\@glsxtr@dostylewarn{\GlossariesWarning{Abbreviation
          style has been switched \MessageBreak
          for category `#1', \MessageBreak
          but there have already been entries \MessageBreak
          defined for this category. Unwanted \MessageBreak
          side-effects may result}}%
        \@endfortrue
      }%
      \@glsxtr@dostylewarn
```

Set up the style for the given category.

```
      \csdef{@glsabbrv@current@#1}{#2}%
      \glsxtr@applyabbrvstyle{#2}%
    }%
  }%
}
```

applyabbrvstyle Apply the abbreviation style without existence check.

```
\newcommand*{\glsxtr@applyabbrvstyle}[1]{%
  \csuse{@glsabbrv@dispstyle@setup@#1}%
  \csuse{@glsabbrv@dispstyle@fmts@#1}%
}
```

r@applyabbrvfmt Only apply the style formats.

```
\newcommand*{\glsxtr@applyabbrvfmt}[1]{%
  \csuse{@glsabbrv@dispstyle@fmts@#1}%
}
```

breviationstyle This is different from \newacronymstyle. The first argument is the label, the second argument sets the information required when defining the new abbreviation and the third argument sets the commands used to display the full format.

```
\newcommand*{\newabbreviationstyle}[3]{%
  \ifcsdef{@glsabbrv@dispstyle@setup@#1}
  {%
    \PackageError{glossaries-extra}{Abbreviation style `#1' already
     defined}{}%
  }%
  {%
    \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
      \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
      #2}%
    \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
    \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
    \renewcommand*{\Glsxtrinlinefullformat}{\Glsxtrfullformat}%
    \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
    \renewcommand*{\Glsxtrinlinefullplformat}{\Glsxtrfullplformat}%
```

Reset \glsxtrsubsequentfmt etc in case a style changes this.

```
    \let\glsxtrsubsequentfmt\glsxtrdefaultsubsequentfmt
    \let\glsxtrsubsequentplfmt\glsxtrdefaultsubsequentplfmt
    \let\Glsxtrsubsequentfmt\Glsxtrdefaultsubsequentfmt
    \let\Glsxtrsubsequentplfmt\Glsxtrdefaultsubsequentplfmt
    #3}%
  }%
}
```

breviationstyle

```
\newcommand*{\renewabbreviationstyle}[3]{%
  \ifcsundef{@glsabbrv@dispstyle@setup@#1}
  {%
    \PackageError{glossaries-extra}{Abbreviation style `#1' not defined}{}%
  }%
  {%
    \csdef{@glsabbrv@dispstyle@setup@#1}{%
```

Initialise hook to do nothing. The style may change this.

```
     \renewcommand*{\GlsXtrPostNewAbbreviation}{}%
     #2}%
    \csdef{@glsabbrv@dispstyle@fmts@#1}{%
```

Assume in-line form is the same as first use. The style may change this.

```
    \renewcommand*{\glsxtrinlinefullformat}{\glsxtrfullformat}%
    \renewcommand*{\GlsXtrinlinefullformat}{\GlsXtrfullformat}%
    \renewcommand*{\glsxtrinlinefullplformat}{\glsxtrfullplformat}%
    \renewcommand*{\GlsXtrinlinefullplformat}{\GlsXtrfullplformat}%
    #3}%
  }%
}
```

breviationstyle Define a synonym for an abbreviation style. The first argument is the new name. The second argument is the original style's name.

```
\newcommand*{\letabbreviationstyle}[2]{%
  \csletcs{@glsabbrv@dispstyle@setup@#1}{@glsabbrv@dispstyle@setup@#2}%
  \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
}
```

ecated@abbrstyle `\@glsxtr@deprecated@abbrstyle{⟨old-name⟩}{⟨new-name⟩}`

Define a synonym for a deprecated abbreviation style.

```
\newcommand*{\@glsxtr@deprecated@abbrstyle}[2]{%
  \csdef{@glsabbrv@dispstyle@setup@#1}{%
    \GlsXtrWarnDeprecatedAbbrStyle{#1}{#2}%
    \csuse{@glsabbrv@dispstyle@setup@#2}%
  }%
  \csletcs{@glsabbrv@dispstyle@fmts@#1}{@glsabbrv@dispstyle@fmts@#2}%
}
```

ecatedAbbrStyle Generate warning for deprecated style use.

```
\newcommand*{\GlsXtrWarnDeprecatedAbbrStyle}[2]{%
  \GlossariesExtraWarning{Deprecated abbreviation style name `#1',
  use `#2' instead}%
}
```

eAbbrStyleSetup

```
\newcommand*{\GlsXtrUseAbbrStyleSetup}[1]{%
  \ifcsundef{@glsabbrv@dispstyle@setup@#1}%
  {%
     \PackageError{glossaries-extra}%
     {Unknown abbreviation style definitions `#1'}{}%
  }%
  {%
     \csname @glsabbrv@dispstyle@setup@#1\endcsname
  }%
}
```

seAbbrStyleFmts

```
\newcommand*{\GlsXtrUseAbbrStyleFmts}[1]{%
  \ifcsundef{@glsabbrv@dispstyle@fmts@#1}%
  {%
     \PackageError{glossaries-extra}%
     {Unknown abbreviation style formats `#1'}{}%
  }%
  {%
     \csname @glsabbrv@dispstyle@fmts@#1\endcsname
  }%
}
```

1.7.2 Predefined Styles (Default Font)

Define some common styles. These will set the first, firstplural, text and plural keys, even if the regular attribute isn't set to "true". If this attribute is set, commands like \gls will use them as per a regular entry, otherwise those keys will be ignored unless explicitly invoked by the user with commands like \glsfirst. In order for the first letter uppercase versions to work correctly, \glsxtrfullformat needs to be expanded when those keys are set. The final optional argument of \glsfirst will behave differently to the final optional argument of \gls with some styles.

xtrinsertinside Switch to determine if the insert text should be inside or outside the font changing command. The default is outside.

```
\newif\ifglsxtrinsertinside
\glsxtrinsertinsidefalse
```

trlongshortname

```
\newcommand*{\glsxtrlongshortname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

long-short

```
\newabbreviationstyle{long-short}%
{%
```

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}},%
    description={\the\glslongtok}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
}
```

Set this as the default style for general abbreviations:

```
\setabbreviationstyle{long-short}
```

ngshortdescsort

```
\newcommand*{\glsxtrlongshortdescsort}{%
 \expandonce\glsxtrorglong\space (\expandonce\glsxtrorgshort)%
}
```

ngshortdescname

```
\newcommand*{\glsxtrlongshortdescname}{%
  \protect\glslongfont{\the\glslongtok}
  \glsxtrparen{\protect\glsabbrvfont{\the\glsshorttok}}%
}
```

long-short-desc User supplies description. The long form is included in the name.

```
\newabbreviationstyle{long-short-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvfont{\the\glsshortpltok}}},%
```

The text key should only have the short form.

```
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-short}%
}
```

trshortlongname

```
\newcommand*{\glsxtrshortlongname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

short-long Short form followed by long form in parenthesis on first use.

```
\newabbreviationstyle{short-long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
```

```
    \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
  }%
}
```

ortlongdescsort

```
\newcommand*{\glsxtrshortlongdescsort}{\the\glsshorttok}
```

ortlongdescname

```
\newcommand*{\glsxtrshortlongdescname}{%
  \protect\glsabbrvfont{\the\glsshorttok}
  \glsxtrparen{\protect\glslongfont{\the\glslongtok}}%
}
```

short-long-desc User supplies description. The long form is included in the name.

```
\newabbreviationstyle{short-long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

```
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-long}%
}
```

ongfootnotefont Only used by the "footnote" styles.

```
\newcommand*{\glsfirstlongfootnotefont}[1]{\glslongfootnotefont{#1}}%
```

ongfootnotefont Only used by the "footnote" styles.

```
\newcommand*{\glslongfootnotefont}[1]{\glslongdefaultfont{#1}}%
```

xtrabbrvfootnote `\glsxtrabbrvfootnote{⟨label⟩}{⟨long⟩}`

Command used by footnote abbreviation styles. The default definition ignores the first argument. The second argument ⟨*long*⟩ includes the font changing command and may be the singular or plural form, depending on the command that was used (for example, `\gls` or `\glspl`).

```
\newcommand*{\glsxtrabbrvfootnote}[2]{\footnote{#2}}
```

xtrfootnotename

```
\newcommand*{\glsxtrfootnotename}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

footnote Short form followed by long form in footnote on first use.

```
\newabbreviationstyle{footnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
    \glshasattribute{\the\glslabeltok}{regular}%
```

```
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
```

```
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

short-footnote

```
\letabbreviationstyle{short-footnote}{footnote}
```

postfootnote

Similar to the above but the footnote is placed afterwards, outside the link. This avoids nested links and can also move the footnote marker after any following punctuation mark. Pre v1.07 included `\footnote` in the first keys, which was incorrect as it caused duplicate footnotes.

```
\newabbreviationstyle{postfootnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}}%
```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
        \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
        {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

The footnote needs to be suppressed in the inline form, so \glsxtrfull must set the first use switch off.

```
  \renewcommand*{\glsxtrsetupfulldefs}{%
    \let\glsxtrifwasfirstuse\@secondoftwo
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
```

```
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

rt-postfootnote

```
\letabbreviationstyle{short-postfootnote}{postfootnote}
```

shortnolongname

```
\newcommand*{\glsxtrshortnolongname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

short Provide a style that only displays the short form on first use, but the short and long form can be displayed with the "full" commands that use the inline format. If the user supplies a description, the long form won't be displayed in the predefined glossary styles, but the post description hook can be employed to automatically insert it.

```
\newabbreviationstyle{short}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortnolongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
    text={\protect\glsabbrvfont{\the\glsshorttok}},
    plural={\protect\glsabbrvfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}%
```

```
    \ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullformat}[2]{%
  \protect\glsfirstabbrvfont{\glsaccessshort{##1}%
    \ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\glsfirstlongfont{\Glsaccesslong{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullplformat}[2]{%
  \protect\glsfirstabbrvfont{\glsaccessshortpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\glsfirstlongfont{\Glsaccesslongpl{##1}}}%
}%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
\renewcommand*{\glsxtrfullformat}[2]{%
  \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullformat}[2]{%
  \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullplformat}[2]{%
  \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
}
```

Set this as the default style for acronyms:

```
\setabbreviationstyle[acronym]{short}
```

short-nolong

```
\letabbreviationstyle{short-nolong}{short}
```

rt-nolong-noreg Like short-nolong but doesn't set the regular attribute.

```
\newabbreviationstyle{short-nolong-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
```

```
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-nolong}%
}
```

trshortdescname

```
\newcommand*{\glsxtrshortdescname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

short-desc The user must supply the description in this style. The long form is added to the name. The short style (possibly with the post-description hooks set) might be a better option.

```
\newabbreviationstyle{short-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortdescname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvfont{\the\glsshortpltok}},
    text={\protect\glsabbrvfont{\the\glsshorttok}},
    plural={\protect\glsabbrvfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
```

```
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
}
```

ort-nolong-desc

```
\letabbreviationstyle{short-nolong-desc}{short-desc}
```

long-desc-noreg Like short-nolong-desc but doesn't set the regular attribute.

```
\newabbreviationstyle{short-nolong-desc-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong-desc}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

```
  \GlsXtrUseAbbrStyleFmts{short-nolong-desc}%
}
```

nolong-short Similar to short-nolong but the full form shows the long form followed by the short form in parentheses.

```
\newabbreviationstyle{nolong-short}%
{%
  \GlsXtrUseAbbrStyleSetup{short-nolong}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\Glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongfont{\glsaccesslongpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvfont{\Glsaccessshortpl{##1}}}%
  }%
}
```

ong-short-noreg Like nolong-short but doesn't set the regular attribute.

```
\newabbreviationstyle{nolong-short-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{nolong-short}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
```

```
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{nolong-short}%
}
```

noshortdescname

```
\newcommand*{\glsxtrlongnoshortdescname}{%
  \protect\glslongfont{\the\glslongtok}%
}
```

long-desc Provide a style that only displays the long form, but the long and short form can be displayed with the "full" commands that use the inline format. The predefined glossary styles won't show the short form. The user must supply a description for this style.

```
\newabbreviationstyle{long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortdescname},
    sort={\the\glslongtok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
    text={\glslongfont{\the\glslongtok}},
    plural={\glslongfont{\the\glslongpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

```
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`ng-noshort-desc` Provide a synonym that matches similar styles.

```
\letabbreviationstyle{long-noshort-desc}{long-desc}
```

hort-desc-noreg Like long-noshort-desc but doesn't set the regular attribute.

```
\newabbreviationstyle{long-noshort-desc-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-noshort-desc}%
}
```

longnoshortname

```
\newcommand*{\glsxtrlongnoshortname}{%
  \protect\glsabbrvfont{\the\glsshorttok}%
}
```

long It doesn't really make a great deal of sense to have a long-only style that doesn't have a description (unless no glossary is required), but the best course of action here is to use the short form as the name and the long form as the description.

```
\newabbreviationstyle{long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongfont{\the\glslongpltok}},
    text={\glslongfont{\the\glslongtok}},
    plural={\glslongfont{\the\glslongpltok}},%
    description={\the\glslongtok}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-desc}%
}
```

long-noshort Provide a synonym that matches similar styles.

```
\letabbreviationstyle{long-noshort}{long}
```

g-noshort-noreg Like long-noshort but doesn't set the regular attribute.

```
\newabbreviationstyle{long-noshort-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-noshort}%
}
```

1.7.3 Predefined Styles (Small Capitals)

These styles use `\textsc` for the short form.

`\glsxtrscfont` Maintained for backward-compatibility.

```
\newcommand*{\glsxtrscfont}[1]{\textsc{#1}}
```

`\glsabbrvscfont` Added for consistent naming.

```
\newcommand*{\glsabbrvscfont}{\glsxtrscfont}
```

`sxtrfirstscfont` Maintained for backward-compatibility.

```
\newcommand*{\glsxtrfirstscfont}[1]{\glsabbrvscfont{#1}}
```

`irstabbrvscfont` Added for consistent naming.

```
\newcommand*{\glsfirstabbrvscfont}{\glsxtrfirstscfont}
```

and for the default short form suffix:

`\glsxtrscsuffix`

```
\newcommand*{\glsxtrscsuffix}{\glstextup{\glsxtrabbrvpluralsuffix}}
```

`long-short-sc`

```
\newabbreviationstyle{long-short-sc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
```

```
      \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}},%
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
```

Use the default long fonts.

```
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
}
```

g-short-sc-desc

```
\newabbreviationstyle{long-short-sc-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
```

```
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvscfont{\the\glsshortpltok}}},%
    text={\protect\glsabbrvscfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

As long-short-sc style:

```
  \GlsXtrUseAbbrStyleFmts{long-short-sc}%
}
```

Now the short (long) version

```
\newabbreviationstyle{short-sc-long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
```

```
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
}
```

As before but user provides description

```
\newabbreviationstyle{short-sc-long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvscfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

As short-sc-long style:

```
  \GlsXtrUseAbbrStyleFmts{short-sc-long}%
}
```

short-sc

```
\newabbreviationstyle{short-sc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortnolongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
    text={\protect\glsabbrvscfont{\the\glsshorttok}},
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvscfont{\glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

```
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvscfont{\Glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvscfont{\Glsaccessshortpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

short-sc-nolong

```
\letabbreviationstyle{short-sc-nolong}{short-sc}
```

short-sc-desc

```
\newabbreviationstyle{short-sc-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortdescname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},
    text={\protect\glsabbrvscfont{\the\glsshorttok}},
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
```

```
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
}
```

-sc-nolong-desc

```
\letabbreviationstyle{short-sc-nolong-desc}{short-sc-desc}
```

nolong-short-sc

```
\newabbreviationstyle{nolong-short-sc}%
{%
  \GlsXtrUseAbbrStyleSetup{short-sc-nolong}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-sc-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
}
```

long-noshort-sc The smallcaps font will only be used if the short form is explicitly invoked through commands like \glsxtrshort.

```
\newabbreviationstyle{long-noshort-sc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
    text={\protect\glslongdefaultfont{\the\glslongtok}},
    plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
```

```
    description={\the\glslongtok}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
    \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`long-sc` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-sc}{long-noshort-sc}
```

`noshort-sc-desc` The smallcaps font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```
\newabbreviationstyle{long-noshort-sc-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

```
\renewcommand*{\Glsxtrsubsequentfmt}[2]{%
  \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
  \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
```

The inline full form displays the long format followed by the short form in parentheses.

```
\renewcommand*{\glsxtrinlinefullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
}%
\renewcommand*{\glsxtrinlinefullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullformat}[2]{%
  \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshort{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullplformat}[2]{%
  \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvscfont{\glsaccessshortpl{##1}}}%
}%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
\renewcommand*{\glsxtrfullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
}
```

long-desc-sc Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-desc-sc}{long-noshort-sc-desc}
```

ort-sc-footnote

```
\newabbreviationstyle{short-sc-footnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
```

```
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

`footnote-sc` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{footnote-sc}{short-sc-footnote}
```

`sc-postfootnote`

```
\newabbreviationstyle{short-sc-postfootnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvscfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvscfont{\the\glsshortpltok}},%
    plural={\protect\glsabbrvscfont{\the\glsshortpltok}}}%
```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
\renewcommand*{\GlsXtrPostNewAbbreviation}{%
  \csdef{glsxtrpostlink\glscategorylabel}{%
    \glsxtrifwasfirstuse
    {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
      \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
      {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
    }%
    {}%
  }%
  \glshasattribute{\the\glslabeltok}{regular}%
  {%
    \glssetattribute{\the\glslabeltok}{regular}{false}%
  }%
  {}%
}%
```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```
\renewcommand*{\glsxtrsetupfulldefs}{%
  \let\glsxtrifwasfirstuse\@secondoftwo
}%
}%
{%
```

Use smallcaps and adjust the plural suffix to revert to upright.

```
\renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrscsuffix}%
\renewcommand*\glsabbrvfont[1]{\glsabbrvscfont{##1}}%
\renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvscfont{##1}}%
\renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
\renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
\renewcommand*{\glsxtrfullformat}[2]{%
  \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullformat}[2]{%
  \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullplformat}[2]{%
```

```
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvscfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

postfootnote-sc Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{postfootnote-sc}{short-sc-postfootnote}
```

1.7.4 Predefined Styles (Fake Small Capitals)

These styles require the relsize package, which must be loaded by the user. These styles all use:

\glsxtrsmfont Maintained for backward compatibility.

```
\newcommand*{\glsxtrsmfont}[1]{\textsmaller{#1}}
```

\glsabbrvsmfont Added for consistent naming.

```
\newcommand*{\glsabbrvsmfont}{\glsxtrsmfont}
```

sxtrfirstsmfont Maintained for backward compatibility.

```
\newcommand*{\glsxtrfirstsmfont}[1]{\glsabbrvsmfont{#1}}
```

irstabbrvsmfont Added for consistent naming.

```
\newcommand*{\glsfirstabbrvsmfont}{\glsxtrfirstsmfont}
```

and for the default short form suffix:

\glsxtrsmsuffix

```
\newcommand*{\glsxtrsmsuffix}{\glsxtrabbrvpluralsuffix}
```

long-short-sm

```
\newabbreviationstyle{long-short-sm}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},%
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
```

Use the default long fonts.

```
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
```

```
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
}
```

g-short-sm-desc

```
\newabbreviationstyle{long-short-sm-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvsmfont{\the\glsshortpltok}}},%
    text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

As long-short-sm style:

```
  \GlsXtrUseAbbrStyleFmts{long-short-sm}%
}
```

short-sm-long Now the short (long) version

```
\newabbreviationstyle{short-sm-long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
```

```
      \protect\glsxtrfullsep{\the\glslabeltok}%
      \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
}
```

`rt-sm-long-desc` As before but user provides description

```
\newabbreviationstyle{short-sm-long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
```

```
    sort={\glsxtrshortlongdescsort},
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvsmfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

As short-sm-long style:

```
  \GlsXtrUseAbbrStyleFmts{short-sm-long}%
}
```

short-sm

```
\newabbreviationstyle{short-sm}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortnolongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
    text={\protect\glsabbrvsmfont{\the\glsshorttok}},
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvsmfont{\glsaccessshort{##1}%
```

```
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%

  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvsmfont{\Glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvsmfont{\Glsaccessshortpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

short-sm-nolong

```
\letabbreviationstyle{short-sm-nolong}{short-sm}
```

short-sm-desc

```
\newabbreviationstyle{short-sm-desc}%
```

```
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortdescname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},
    text={\protect\glsabbrvsmfont{\the\glsshorttok}},
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
```

```
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi
  }%
}
```

-sm-nolong-desc

```
\letabbreviationstyle{short-sm-nolong-desc}{short-sm-desc}
```

nolong-short-sm

```
\newabbreviationstyle{nolong-short-sm}%
{%
  \GlsXtrUseAbbrStyleSetup{short-sm-nolong}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-sm-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
}
```

long-noshort-sm The smallcaps font will only be used if the short form is explicitly invoked through commands like \glsshort.

```
\newabbreviationstyle{long-noshort-sm}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
    text={\protect\glslongdefaultfont{\the\glslongtok}},
    plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
    description={\the\glslongtok}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
    \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`long-sm` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-sm}{long-noshort-sm}
```

`noshort-sm-desc` The smaller font will only be used if the short form is explicitly invoked through commands like `\glsshort`.

```
\newabbreviationstyle{long-noshort-sm-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
\renewcommand*{\glsxtrsubsequentfmt}[2]{%
  \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\glsxtrsubsequentplfmt}[2]{%
  \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\Glsxtrsubsequentfmt}[2]{%
  \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
  \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
```

The inline full form displays the long format followed by the short form in parentheses.

```
\renewcommand*{\glsxtrinlinefullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
}%
\renewcommand*{\glsxtrinlinefullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullformat}[2]{%
  \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshort{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullplformat}[2]{%
  \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvsmfont{\glsaccessshortpl{##1}}}%
}%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
\renewcommand*{\glsxtrfullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullformat}[2]{%
```

```
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

long-desc-sm Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-desc-sm}{long-noshort-sm-desc}
```

ort-sm-footnote

```
\newabbreviationstyle{short-sm-footnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
```

```
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

footnote-sm Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{footnote-sm}{short-sm-footnote}
```

sm-postfootnote

```
\newabbreviationstyle{short-sm-postfootnote}%
{%
```

```
\renewcommand*{\CustomAbbreviationFields}{%
  name={\glsxtrfootnotename},
  sort={\the\glsshorttok},
  description={\the\glslongtok},%
  first={\protect\glsfirstabbrvsmfont{\the\glsshorttok}},%
  firstplural={\protect\glsfirstabbrvsmfont{\the\glsshortpltok}},%
  plural={\protect\glsabbrvsmfont{\the\glsshortpltok}}}%
```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
\renewcommand*{\GlsXtrPostNewAbbreviation}{%
  \csdef{glsxtrpostlink\glscategorylabel}{%
    \glsxtrifwasfirstuse
    {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
      \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
      {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
    }%
    {}%
  }%
  \glshasattribute{\the\glslabeltok}{regular}%
  {%
    \glssetattribute{\the\glslabeltok}{regular}{false}%
  }%
  {}%
}%
```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```
  \renewcommand*{\glsxtrsetupfulldefs}{%
    \let\glsxtrifwasfirstuse\@secondoftwo
  }%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvsmfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvsmfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtrsmsuffix}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

```
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvsmfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvsmfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

postfootnote-sm Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{postfootnote-sm}{short-sm-postfootnote}
```

1.7.5 Predefined Styles (Emphasized)

These styles use \emph for the short form.

\glsabbrvemfont

```
\newcommand*{\glsabbrvemfont}[1]{\emph{#1}}%
```

irstabbrvemfont

```
\newcommand*{\glsfirstabbrvemfont}[1]{\glsabbrvemfont{#1}}%
```

The default short form suffix:

\glsxtremsuffix

```
\newcommand*{\glsxtremsuffix}{\glsxtrabbrvpluralsuffix}
```

firstlongemfont Only used by the “long-em” styles.

```
\newcommand*{\glsfirstlongemfont}[1]{\glslongemfont{#1}}%
```

\glslongemfont Only used by the “long-em” styles.

```
\newcommand*{\glslongemfont}[1]{\emph{#1}}%
```

long-short-em The long form is just set in the default long font.

```
\newabbreviationstyle{long-short-em}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
```

Use the default long fonts.

```
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
```

```
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
}
```

g-short-em-desc

```
\newabbreviationstyle{long-short-em-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

As long-short-em style:

```
  \GlsXtrUseAbbrStyleFmts{long-short-em}%
}
```

ong-em-short-em

```
\newabbreviationstyle{long-em-short-em}%
{%
```

`\glslongemfont` is used in the description since `\glsdesc` doesn't set the style.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
```

```
    first={\protect\glsfirstlongemfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}},%
    description={\protect\glslongemfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
}
```

m-short-em-desc

```
\newabbreviationstyle{long-em-short-em-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlongemfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvemfont{\the\glsshortpltok}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-em-short-em}%
}
```

short-em-long Now the short (long) version

```
\newabbreviationstyle{short-em-long}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
```

```
    {}%
  }%
}%
{%
```

Mostly as short-long style:

```
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*\glsfirstabbrvfont[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
}
```

`rt-em-long-desc` As before but user provides description

```
\newabbreviationstyle{short-em-long-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongdefaultfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
```

```
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-em-long}%
}
```

hort-em-long-em

```
\newabbreviationstyle{short-em-long-em}%
{%
```

`\glslongemfont` is used in the description since `\glsdesc` doesn't set the style.

```
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    description={\protect\glslongemfont{\the\glslongtok}},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongemfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongemfont{\glsaccesslongpl{##1}}}%
  }%
}
```

em-long-em-desc

```
\newabbreviationstyle{short-em-long-em-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},%
    sort={\glsxtrshortlongdescsort},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlongemfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvemfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

```
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-em-long-em}%
}
```

short-em

```
\newabbreviationstyle{short-em}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortnolongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
    text={\protect\glsabbrvemfont{\the\glsshorttok}},
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short form followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvemfont{\glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%

  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstabbrvemfont{\Glsaccessshort{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstabbrvemfont{\Glsaccessshortpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
```

```
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

short-em-nolong

```
\letabbreviationstyle{short-em-nolong}{short-em}
```

short-em-desc

```
\newabbreviationstyle{short-em-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortdescname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},
    text={\protect\glsabbrvemfont{\the\glsshorttok}},
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}},
    description={\the\glslongtok}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The inline full form displays the short format followed by the long form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
```

```
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongdefaultfont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form only displays the short form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

-em-nolong-desc

```
\letabbreviationstyle{short-em-nolong-desc}{short-em-desc}
```

nolong-short-em

```
\newabbreviationstyle{nolong-short-em}%
{%
  \GlsXtrUseAbbrStyleSetup{short-em-nolong}%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-em-nolong}%
```

The inline full form displays the long form followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\glsaccesslongpl{##1}%
     \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslong{##1}%
      \ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \protect\glsfirstlongdefaultfont{\Glsaccesslongpl{##1}%
       \ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
}
```

long-noshort-em The short form is explicitly invoked through commands like \glsshort.

```
\newabbreviationstyle{long-noshort-em}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongdefaultfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongdefaultfont{\the\glslongpltok}},
    text={\protect\glslongdefaultfont{\the\glslongtok}},
    plural={\protect\glslongdefaultfont{\the\glslongpltok}},%
    description={\the\glslongtok}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
\renewcommand*{\glsxtrsubsequentfmt}[2]{%
  \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\glsxtrsubsequentplfmt}[2]{%
  \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\Glsxtrsubsequentfmt}[2]{%
  \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
\renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
  \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
  \ifglsxtrinsertinside \else##2\fi
}%
```

The inline full form displays the long format followed by the short form in parentheses.

```
\renewcommand*{\glsxtrinlinefullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
}%
\renewcommand*{\glsxtrinlinefullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullformat}[2]{%
  \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
}%
\renewcommand*{\Glsxtrinlinefullplformat}[2]{%
  \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
   \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
  \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
}%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
\renewcommand*{\glsxtrfullformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\glsxtrfullplformat}[2]{%
  \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
  \ifglsxtrinsertinside\else##2\fi
}%
\renewcommand*{\Glsxtrfullformat}[2]{%
```

```
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

long-em Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-em}{long-noshort-em}
```

g-em-noshort-em The short form is explicitly invoked through commands like \glsshort.

```
\newabbreviationstyle{long-em-noshort-em}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongemfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
    text={\protect\glslongemfont{\the\glslongtok}},
    plural={\protect\glslongemfont{\the\glslongpltok}},%
    description={\protect\glslongemfont{\the\glslongtok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
```

```
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`oshort-em-noreg` Like long-em-noshort-em but doesn't set the regular attribute.

```
\newabbreviationstyle{long-em-noshort-em-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{long-em-noshort-em}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-em-noshort-em}%
}
```

noshort-em-desc The emphasized font will only be used if the short form is explicitly invoked through commands like \glsshort.

```
\newabbreviationstyle{long-noshort-em-desc}%
{%
  \GlsXtrUseAbbrStyleSetup{long-noshort-desc}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongdefaultfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongdefaultfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
```

```
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongdefaultfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongdefaultfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

`long-desc-em` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{long-desc-em}{long-noshort-em-desc}
```

`noshort-em-desc` The short form is explicitly invoked through commands like `\glsshort`. The long form is emphasized.

```
\newabbreviationstyle{long-em-noshort-em-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortdescname},
    sort={\the\glslongtok},
    first={\protect\glsfirstlongemfont{\the\glslongtok}},
    firstplural={\protect\glsfirstlongemfont{\the\glslongpltok}},
    text={\glslongemfont{\the\glslongtok}},
    plural={\glslongemfont{\the\glslongpltok}}%
```

```
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{regular}{true}}%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongemfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongemfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glslongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glslongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glslongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glslongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside ##2\fi}%
    \ifglsxtrinsertinside \else##2\fi
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongemfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongemfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvemfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form, but it typically won't be used as the regular attribute is set by this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlongemfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongemfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
}
```

t-em-desc-noreg Like long-em-noshort-em-desc but doesn't set the regular attribute.

```
\newabbreviationstyle{long-em-noshort-em-desc-noreg}%
{%
  \GlsXtrUseAbbrStyleSetup{long-em-noshort-em-desc}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-em-noshort-em-desc}%
}
```

ort-em-footnote

```
\newabbreviationstyle{short-em-footnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}%
     \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
       {\protect\glsfirstlongfootnotefont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}%
```

```
    \protect\glsxtrabbrvfootnote{\the\glslabeltok}%
      {\protect\glsfirstlongfootnotefont{\the\glslongpltok}}},%
  plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
```

Switch off hyperlinks on first use to prevent nested hyperlinks, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glssetattribute{\the\glslabeltok}{nohyperfirst}{true}%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form followed by the long form as a footnote.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \protect\glsxtrabbrvfootnote{##1}%
      {\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
```

```
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

`footnote-em` Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{footnote-em}{short-em-footnote}
```

`em-postfootnote`

```
\newabbreviationstyle{short-em-postfootnote}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrfootnotename},
    sort={\the\glsshorttok},
    description={\the\glslongtok},%
    first={\protect\glsfirstabbrvemfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvemfont{\the\glsshortpltok}},%
    plural={\protect\glsabbrvemfont{\the\glsshortpltok}}}%
```

Make this category insert a footnote after the link if this was the first use, and unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
```

Needs the specific font command here as the style may have been lost by the time the footnote occurs.

```
        \glsxtrdopostpunc{\protect\glsxtrabbrvfootnote{\glslabel}%
        {\glsfirstlongfootnotefont{\glsentrylong{\glslabel}}}}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
```

```
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
```

The footnote needs to be suppressed in the inline form, so `\glsxtrfull` must set the first use switch off.

```
  \renewcommand*{\glsxtrsetupfulldefs}{%
    \let\glsxtrifwasfirstuse\@secondoftwo
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtremsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvemfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvemfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongfootnotefont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongfootnotefont{##1}}%
```

The full format displays the short form. The long form is deferred.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

The first use full form and the inline full form use the short (long) style.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslong{##1}}}%
```

```
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvemfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
     \ifglsxtrinsertinside\else##2\fi\glsxtrfullsep{##1}%
    \glsxtrparen{\glsfirstlongfootnotefont{\glsaccesslongpl{##1}}}%
  }%
}
```

postfootnote-em Backward compatibility:

```
\@glsxtr@deprecated@abbrstyle{postfootnote-em}{short-em-postfootnote}
```

1.7.6 Predefined Styles (User Parentheses Hook)

These styles allow the user to adjust the parenthetical forms. These styles all test for the existence of the field given by:

glsxtruserfield Default is the useri field.

```
\newcommand*{\glsxtruserfield}{useri}
```

glsxtruserparen The format of the parenthetical information. The first argument is the long/short form. The second argument is the entry's label. If \glscurrentfieldvalue has been defined, then we have at least glossaries v4.23, which makes it easier for the user to adjust this.

```
\ifdef\glscurrentfieldvalue
{
  \newcommand*{\glsxtruserparen}[2]{%
    \glsxtrfullsep{#2}%
    \glsxtrparen
     {#1\ifglshasfield{\glsxtruserfield}{#2}{, \glscurrentfieldvalue}{}}%
  }
}
{
  \newcommand*{\glsxtruserparen}[2]{%
    \glsxtrfullsep{#2}%
    \glsxtrparen
      {#1\ifglshasfield{\glsxtruserfield}{#2}{, \@glo@thisvalue}{}}%
  }
}
```

Font used for short form:

lsabbrvuserfont

```
\newcommand*{\glsabbrvuserfont}[1]{\glsabbrvdefaultfont{#1}}
```

Font used for short form on first use:

stabbrvuserfont

```
\newcommand*{\glsfirstabbrvuserfont}[1]{\glsabbrvuserfont{#1}}
```

Font used for long form:

glslonguserfont

```
\newcommand*{\glslonguserfont}[1]{\glslongdefaultfont{#1}}
```

Font used for long form on first use:

rstlonguserfont

```
\newcommand*{\glsfirstlonguserfont}[1]{\glslonguserfont{#1}}
```

The default short form suffix:

lsxtrusersuffix

```
\newcommand*{\glsxtrusersuffix}{\glsxtrabbrvpluralsuffix}
```

Description encapsulator.

userdescription The first argument is the description. The second argument is the label.

```
\newcommand*{\glsuserdescription}[2]{\glslonguserfont{#1}}
```

long-short-user

```
\newabbreviationstyle{long-short-user}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}%
     \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
      {\the\glslabeltok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
     \protect\glsxtruserparen
      {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
    description={\protect\glsuserdescription{\the\glslongtok}%
     {\the\glslabeltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
  }%
}
```

-postshort-user Like long-short-user but defers the parenthetical matter to after the link.

```
\newabbreviationstyle{long-postshort-user}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
    description={\protect\glsuserdescription{\the\glslongtok}%
     {\the\glslabeltok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
```

```
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

In-line format:

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlonguserfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshort{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlonguserfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstabbrvuserfont{\glsaccessshortpl{##1}}}{##1}%
```

```
  }%
}
```

ortuserdescname

```
\newcommand*{\glsxtrlongshortuserdescname}{%
  \protect\glslonguserfont{\the\glslongtok}%
  \protect\glsxtruserparen
   {\protect\glsabbrvuserfont{\the\glsshorttok}}{\the\glslabeltok}%
}
```

short-user-desc Like long-postshort-user but the user supplies the description.

```
\newabbreviationstyle{long-postshort-user-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortuserdescname},
    sort={\the\glslongtok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
    text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstabbrvuserfont{\glsentryshort{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-postshort-user}%
}
```

t-postlong-user Like short-long-user but defers the parenthetical matter to after the link.

```
\newabbreviationstyle{short-postlong-user}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
```

```
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}},%
    description={\protect\glsuserdescription{\the\glslongtok}%
     {\the\glslabeltok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
          {\glslabel}%
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

First use full form:

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

In-line format:

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
  }%
}
```

onguserdescname

```
\newcommand*{\glsxtrshortlonguserdescname}{%
 \protect\glsabbrvuserfont{\the\glsshorttok}%
 \protect\glsxtruserparen
   {\protect\glslonguserfont{\the\glslongpltok}}%
   {\the\glslabeltok}%
}
```

tlong-user-desc Like short-postlong-user but leaves the user to specify the description.

```
\newabbreviationstyle{short-postlong-user-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlonguserdescname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonguserfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}},%
    text={\protect\glsabbrvuserfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtruserparen
          {\glsfirstlonguserfont{\glsentrylong{\glslabel}}}%
          {\glslabel}%
```

```
      }%
      {}%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-postlong-user}%
}
```

short-user-desc

```
\newabbreviationstyle{long-short-user-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortuserdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlonguserfont{\the\glslongtok}%
     \protect\glsxtruserparen{\protect\glsfirstabbrvuserfont{\the\glsshorttok}}%
      {\the\glslabeltok}},%
    firstplural={\protect\glsfirstlonguserfont{\the\glslongpltok}%
     \protect\glsxtruserparen
      {\protect\glsfirstabbrvuserfont{\the\glsshortpltok}}{\the\glslabeltok}},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-short-user}%
}
```

short-long-user

```
\newabbreviationstyle{short-long-user}%
{%
```

`\glslonguserfont` is used in the description since `\glsdesc` doesn't set the style. (Now in `\glsuserdescription`.)

```
\renewcommand*{\CustomAbbreviationFields}{%
  name={\glsxtrshortlongname},
  sort={\the\glsshorttok},
  description={\protect\glsuserdescription{\the\glslongtok}%
    {\the\glslabeltok}},%
  first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
   \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
     {\the\glslabeltok}},%
  firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
   \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
     {\the\glslabeltok}},%
  plural={\protect\glsabbrvuserfont{\the\glsshortpltok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrusersuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvuserfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonguserfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonguserfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstabbrvuserfont{\glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshort{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslong{##1}}}{##1}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstabbrvuserfont{\Glsaccessshortpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
```

```
    \glsxtruserparen{\glsfirstlonguserfont{\glsaccesslongpl{##1}}}{##1}%
  }%
}
```

-long-user-desc

```
\newabbreviationstyle{short-long-user-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlonguserdescname},
    sort={\glsxtrshortlongdescsort},%
    first={\protect\glsfirstabbrvuserfont{\the\glsshorttok}%
     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongtok}}%
       {\the\glslabeltok}},%
    firstplural={\protect\glsfirstabbrvuserfont{\the\glsshortpltok}%
     \protect\glsxtruserparen{\protect\glsfirstlonguserfont{\the\glslongpltok}}%
       {\the\glslabeltok}},%
    text={\protect\glsabbrvfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-long-user}%
}
```

1.7.7 Predefined Styles (Hyphen)

These styles are designed to work with the markwords attribute. They check if the inserted material (provided by the final optional argument of commands like \gls) starts with a hyphen. If it does, the insert is added to the parenthetical material. Note that commands like \glsxtrlong set \glsinsert to empty with the entire link-text stored in \glscustomtext.

trifhyphenstart Checks if the argument starts with a hyphen. The argument may be \glsinsert so check for that and expand.

```
\newrobustcmd*{\glsxtrifhyphenstart}[3]{%
  \ifx\glsinsert#1\relax
   \expandafter\@glsxtrifhyphenstart#1\relax\relax
     \@end@glsxtrifhyphenstart{#2}{#3}%
  \else
   \@glsxtrifhyphenstart#1\relax\relax\@end@glsxtrifhyphenstart{#2}{#3}%
```

```
  \fi
}
```

trifhyphenstart

```
\def\@glsxtrifhyphenstart#1#2\@end@glsxtrifhyphenstart#3#4{%
  \ifx-#1\relax#3\else #4\fi
}
```

rlonghyphenshort

\glsxtrlonghyphenshort{⟨*label*⟩}{⟨*long*⟩}{⟨*short*⟩}{⟨*insert*⟩}

The ⟨*long*⟩ and ⟨*short*⟩ arguments may be the plural form. The ⟨*long*⟩ argument may also be the first letter uppercase form.

```
\newcommand*{\glsxtrlonghyphenshort}[4]{%
```

Grouping is needed to localise the redefinitions.

```
 {%
```

If ⟨*insert*⟩ starts with a hyphen, redefine \glsxtrwordsep to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to \glsxtrwordsep if ⟨*insert*⟩ doesn't start with a hyphen.

```
   \glsxtrifhyphenstart{#4}{\def\glsxtrwordsep{-}}{}%
   \glsfirstlonghyphenfont{#2\ifglsxtrinsertinside{#4}\fi}%
   \ifglsxtrinsertinside\else{#4}\fi
   \glsxtrfullsep{#1}%
   \glsxtrparen{\glsfirstabbrvhyphenfont{#3\ifglsxtrinsertinside{#4}\fi}%
    \ifglsxtrinsertinside\else{#4}\fi}%
 }%
}
```

abbrvhyphenfont

```
\newcommand*{\glsabbrvhyphenfont}{\glsabbrvdefaultfont}%
```

abbrvhyphenfont

```
\newcommand*{\glsfirstabbrvhyphenfont}{\glsabbrvhyphenfont}%
```

slonghyphenfont

```
\newcommand*{\glslonghyphenfont}{\glslongdefaultfont}%
```

tlonghyphenfont

```
\newcommand*{\glsfirstlonghyphenfont}{\glslonghyphenfont}%
```

The default short form suffix:

xtrhyphensuffix

```
\newcommand*{\glsxtrhyphensuffix}{\glsxtrabbrvpluralsuffix}
```

en-short-hyphen Designed for use with the markwords attribute.

```
\newabbreviationstyle{long-hyphen-short-hyphen}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
    description={\protect\glslonghyphenfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrhyphensuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsxtrlonghyphenshort{##1}{\glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsxtrlonghyphenshort{##1}{\glsaccesslongpl{##1}}%
     {\glsaccessshortpl{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsxtrlonghyphenshort{##1}{\Glsaccesslong{##1}}{\glsaccessshort{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsxtrlonghyphenshort{##1}{\Glsaccesslongpl{##1}}%
     {\glsaccessshortpl{##1}}{##2}%
  }%
}
```

ort-hyphen-desc Like long-hyphen-short-hyphen but the description must be supplied by the user.

```
\newabbreviationstyle{long-hyphen-short-hyphen-desc}%
```

```
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
}
```

onghyphennoshort

\glsxtrlonghyphennoshort{⟨*label*⟩}{⟨*long*⟩}{⟨*insert*⟩}

```
\newcommand*{\glsxtrlonghyphennoshort}[3]{%
```

Grouping is needed to localise the redefinitions.

```
 {%
```

If ⟨*insert*⟩ starts with a hyphen, redefine \glsxtrwordsep to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.) No change is made to \glsxtrwordsep if ⟨*insert*⟩ doesn't start with a hyphen.

```
    \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
    \glsfirstlonghyphenfont{#2\ifglsxtrinsertinside{#3}\fi}%
    \ifglsxtrinsertinside\else{#3}\fi
 }%
}
```

hort-desc-noreg

This version doesn't show the short form (except explicitly with \glsxtrshort). Since \glsxtrshort doesn't support the hyphen switch, the short form just uses the default short-form font command. This style won't work with the regular as the regular form isn't flexible enough.

```
\newabbreviationstyle{long-hyphen-noshort-desc-noreg}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortdescname},
    sort={\expandonce\glsxtrorglong},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
    plural={\protect\glslonghyphenfont{\the\glslongpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-hyphen-short-hyphen}%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*\glsabbrvfont[1]{\glsabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvdefaultfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The format for subsequent use (not used when the regular attribute is set).

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
  }%
```

The inline full form displays the long format followed by the short form in parentheses.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
```

```
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvfont{\glsaccessshortpl{##1}}}%
  }%
```

The first use full form only displays the long form.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\glsaccesslongpl{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsxtrlonghyphennoshort{##1}{\Glsaccesslongpl{##1}}{##2}%
  }%
}
```

n-noshort-noreg

It doesn't really make a great deal of sense to have a long-only style that doesn't have a description (unless no glossary is required), but the best course of action here is to use the short form as the name and the long form as the description.

```
\newabbreviationstyle{long-hyphen-noshort-noreg}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongnoshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
    text={\protect\glslonghyphenfont{\the\glslongtok}},%
    plural={\protect\glslonghyphenfont{\the\glslongpltok}},%
    description={\the\glslongtok}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
```

```
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-desc}%
}
```

glsxtrlonghyphen

\glsxtrlonghyphen{⟨*long*⟩}{⟨*label*⟩}{⟨*insert*⟩}

Used by long-hyphen-postshort-hyphen. The ⟨*insert*⟩ is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```
\newcommand*{\glsxtrlonghyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```
 {%
   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
   \glsfirstlonghyphenfont{#1}%
 }%
}
```

rposthyphenshort

\glsxtrposthyphenshort{⟨*label*⟩}{⟨*insert*⟩}

Used in the post-link hook for the long-hyphen-postshort-hyphen style. Much like \glsxtrlonghyphenshort but omits the ⟨*long*⟩ part. This always uses the singular short form.

```
\newcommand*{\glsxtrposthyphenshort}[2]{%
 {%
   \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
   \ifglsxtrinsertinside{\glsfirstlonghyphenfont{#2}}\else{#2}\fi
   \glsxtrfullsep{#1}%
   \glsxtrparen
   {\glsfirstabbrvhyphenfont{\glsentryshort{#1}\ifglsxtrinsertinside{#2}\fi}%
    \ifglsxtrinsertinside\else{#2}\fi
   }%
 }%
}
```

hyphensubsequent

\glsxtrposthyphensubsequent{⟨*label*⟩}{⟨*insert*⟩}

Format in the post-link hook for subsequent use. The label is ignored by default.

```
\newcommand*{\glsxtrposthyphensubsequent}[2]{%
  \glsabbrvfont{\ifglsxtrinsertinside {#2}\fi}%
```

```
  \ifglsxtrinsertinside \else{#2}\fi
}
```

ostshort-hyphen Like long-hyphen-short-hyphen but shifts the insert and parenthetical material to the post-link hook.

```
\newabbreviationstyle{long-hyphen-postshort-hyphen}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
    description={\protect\glslonghyphenfont{\the\glslongtok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtrposthyphenshort{\glslabel}{\glsinsert}%
      }%
      {%
```

Put the insertion into the post-link:

```
        \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
      }%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

Subsequent use needs to omit the insertion:

```
  \renewcommand*{\glsxtrsubsequentfmt}[2]{%
    \glsabbrvfont{\glsaccessshort{##1}}%
  }%
  \renewcommand*{\glsxtrsubsequentplfmt}[2]{%
    \glsabbrvfont{\glsaccessshortpl{##1}}%
  }%
  \renewcommand*{\Glsxtrsubsequentfmt}[2]{%
```

```
    \glsabbrvfont{\Glsaccessshort{##1}}%
  }%
  \renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
    \glsabbrvfont{\Glsaccessshortpl{##1}}%
  }%
```

First use full form:

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsxtrlonghyphen{\glsaccesslong{##1}}{##1}{##2}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsxtrlonghyphen{\glsaccesslongpl{##1}}{##1}{##2}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsxtrlonghyphen{\Glsaccesslong{##1}}{##1}{##2}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsxtrlonghyphen{\Glsaccesslongpl{##1}}{##1}{##2}%
  }%
```

In-line format.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlonghyphenfont{\glsaccesslong{##1}%
      \ifglsxtrinsertinside{##2}\fi}%
    \ifglsxtrinsertinside \else{##2}\fi
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlonghyphenfont{\glsaccesslongpl{##1}%
      \ifglsxtrinsertinside{##2}\fi}%
    \ifglsxtrinsertinside \else{##2}\fi
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlonghyphenfont{\Glsaccesslong{##1}%
      \ifglsxtrinsertinside{##2}\fi}%
    \ifglsxtrinsertinside \else{##2}\fi
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlonghyphenfont{\Glsaccesslongpl{##1}%
      \ifglsxtrinsertinside{##2}\fi}%
    \ifglsxtrinsertinside \else{##2}\fi
  }%
}
```

`ort-hyphen-desc` Like long-hyphen-postshort-hyphen but the description must be supplied by the user.

```
\newabbreviationstyle{long-hyphen-postshort-hyphen-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrlongshortdescname},
    sort={\glsxtrlongshortdescsort},%
    first={\protect\glsfirstlonghyphenfont{\the\glslongtok}},%
```

```
    firstplural={\protect\glsfirstlonghyphenfont{\the\glslongpltok}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtrposthyphenshort{\glslabel}{\glsinsert}%
      }%
      {%
```

Put the insertion into the post-link:

```
        \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
      }%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-hyphen-postshort-hyphen}%
}
```

rshorthyphenlong `\glsxtrshorthyphenlong{⟨label⟩}{⟨short⟩}{⟨long⟩}{⟨insert⟩}`

The ⟨*long*⟩ and ⟨*short*⟩ arguments may be the plural form. The ⟨*long*⟩ argument may also be the first letter uppercase form.

```
\newcommand*{\glsxtrshorthyphenlong}[4]{%
```

Grouping is needed to localise the redefinitions.

```
 {%
```

If ⟨*insert*⟩ starts with a hyphen, redefine `\glsxtrwordsep` to a hyphen. The inserted material is also inserted into the parenthetical part. (The inserted material is grouped as a precautionary measure.)

```
    \glsxtrifhyphenstart{#4}{\def\glsxtrwordsep{-}}{}%
    \glsfirstabbrvhyphenfont{#2\ifglsxtrinsertinside{#4}\fi}%
    \ifglsxtrinsertinside\else{#4}\fi
    \glsxtrfullsep{#1}%
    \glsxtrparen{\glsfirstlonghyphenfont{#3\ifglsxtrinsertinside{#4}\fi}%
     \ifglsxtrinsertinside\else{#4}\fi}%
 }%
}
```

hen-long-hyphen Designed for use with the markwords attribute.

```
\newabbreviationstyle{short-hyphen-long-hyphen}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
    description={\protect\glslonghyphenfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrhyphensuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

The first use full form and the inline full form are the same for this style.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsxtrshorthyphenlong{##1}%
     {\glsaccessshortpl{##1}}{\glsaccesslongpl{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
    \glsxtrshorthyphenlong{##1}{\glsaccessshort{##1}}{\Glsaccesslong{##1}}{##2}%
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsxtrshorthyphenlong{##1}%
     {\glsaccessshortpl{##1}}{\Glsaccesslongpl{##1}}{##2}%
  }%
}
```

ong-hyphen-desc Like short-hyphen-long-hyphen but the description must be supplied by the user.

```
\newabbreviationstyle{short-hyphen-long-hyphen-desc}%
```

```
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},
    first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongtok}}},%
    firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}%
     \protect\glsxtrfullsep{\the\glslabeltok}%
     \glsxtrparen{\protect\glsfirstlonghyphenfont{\the\glslongpltok}}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-hyphen-long-hyphen}%
}
```

lsxtrshorthyphen `\glsxtrshorthyphen{⟨short⟩}{⟨label⟩}{⟨insert⟩}`

Used by short-hyphen-postlong-hyphen. The ⟨*insert*⟩ is check to determine if it starts with a hyphen but isn't used here as it's moved to the post-link hook.

```
\newcommand*{\glsxtrshorthyphen}[3]{%
```

Grouping is needed to localise the redefinitions.

```
 {%
   \glsxtrifhyphenstart{#3}{\def\glsxtrwordsep{-}}{}%
   \glsfirstabbrvhyphenfont{#1}%
 }%
}
```

trposthyphenlong `\glsxtrposthyphenlong{⟨label⟩}{⟨insert⟩}`

Used in the post-link hook for the short-hyphen-postlong-hyphen style. Much like `\glsxtrshorthyphenlong` but omits the ⟨*short*⟩ part. This always uses the singular long form.

```
\newcommand*{\glsxtrposthyphenlong}[2]{%
 {%
   \glsxtrifhyphenstart{#2}{\def\glsxtrwordsep{-}}{}%
   \ifglsxtrinsertinside{\glsfirstabbrvhyphenfont{#2}}\else{#2}\fi
   \glsxtrfullsep{#1}%
   \glsxtrparen
   {\glsfirstlonghyphenfont{\glsentrylong{#1}\ifglsxtrinsertinside{#2}\fi}%
    \ifglsxtrinsertinside\else{#2}\fi
   }%
 }%
}
```

postlong-hyphen Like short-hyphen-long-hyphen but shifts the insert and parenthetical material to the post-link hook.

```
\newabbreviationstyle{short-hyphen-postlong-hyphen}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}},%
    description={\protect\glslonghyphenfont{\the\glslongtok}}}%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtrposthyphenlong{\glslabel}{\glsinsert}%
      }%
      {%
```

Put the insertion into the post-link:

```
        \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
      }%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
```

In case the user wants to mix and match font styles, these are redefined here.

```
  \renewcommand*{\abbrvpluralsuffix}{\glsxtrabbrvpluralsuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvhyphenfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlonghyphenfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslonghyphenfont{##1}}%
```

Subsequent use needs to omit the insertion:

```
\renewcommand*{\glsxtrsubsequentfmt}[2]{%
  \glsabbrvfont{\glsaccessshort{##1}}%
}%
\renewcommand*{\glsxtrsubsequentplfmt}[2]{%
  \glsabbrvfont{\glsaccessshortpl{##1}}%
}%
\renewcommand*{\Glsxtrsubsequentfmt}[2]{%
  \glsabbrvfont{\Glsaccessshort{##1}}%
}%
\renewcommand*{\Glsxtrsubsequentplfmt}[2]{%
  \glsabbrvfont{\Glsaccessshortpl{##1}}%
}%
```

First use full form:

```
\renewcommand*{\glsxtrfullformat}[2]{%
  \glsxtrshorthyphen{\glsaccessshort{##1}}{##1}{##2}%
}%
\renewcommand*{\glsxtrfullplformat}[2]{%
  \glsxtrshorthyphen{\glsaccessshortpl{##1}}{##1}{##2}%
}%
\renewcommand*{\Glsxtrfullformat}[2]{%
  \glsxtrshorthyphen{\Glsaccessshort{##1}}{##1}{##2}%
}%
\renewcommand*{\Glsxtrfullplformat}[2]{%
  \glsxtrshorthyphen{\Glsaccessshortpl{##1}}{##1}{##2}%
}%
```

In-line format. Commands like `\glsxtrfull` set `\glsinsert` to empty. The entire link-text (provided by the following commands) is stored in `\glscustomtext`.

```
\renewcommand*{\glsxtrinlinefullformat}[2]{%
  \glsfirstabbrvhyphenfont{\glsaccessshort{##1}%
    \ifglsxtrinsertinside{##2}\fi}%
  \ifglsxtrinsertinside \else{##2}\fi
}%
\renewcommand*{\glsxtrinlinefullplformat}[2]{%
  \glsfirstabbrvhyphenfont{\glsaccessshortpl{##1}%
    \ifglsxtrinsertinside{##2}\fi}%
  \ifglsxtrinsertinside \else{##2}\fi
}%
\renewcommand*{\Glsxtrinlinefullformat}[2]{%
  \glsfirstabbrvhyphenfont{\Glsaccessshort{##1}%
    \ifglsxtrinsertinside{##2}\fi}%
  \ifglsxtrinsertinside \else{##2}\fi
}%
\renewcommand*{\Glsxtrinlinefullplformat}[2]{%
  \glsfirstabbrvhyphenfont{\Glsaccessshortpl{##1}%
    \ifglsxtrinsertinside{##2}\fi}%
  \ifglsxtrinsertinside \else{##2}\fi
}%
```

```
}
```

ong-hyphen-desc Like short-hyphen-postlong-hyphen but the description must be supplied by the user.

```
\newabbreviationstyle{short-hyphen-postlong-hyphen-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtrshortlongdescname},
    sort={\glsxtrshortlongdescsort},%
    first={\protect\glsfirstabbrvhyphenfont{\the\glsshorttok}},%
    firstplural={\protect\glsfirstabbrvhyphenfont{\the\glsshortpltok}},%
    text={\protect\glsabbrvhyphenfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvhyphenfont{\the\glsshortpltok}}%
  }%
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \csdef{glsxtrpostlink\glscategorylabel}{%
      \glsxtrifwasfirstuse
      {%
        \glsxtrposthyphenlong{\glslabel}{\glsinsert}%
      }%
      {%
```

Put the insertion into the post-link:

```
        \glsxtrposthyphensubsequent{\glslabel}{\glsinsert}%
      }%
    }%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{short-hyphen-postlong-hyphen}%
}
```

1.7.8 Predefined Styles (No Short on First Use)

These styles show only the long form on first use and only the short form on subsequent use.

lsabbrvonlyfont

```
\newcommand*{\glsabbrvonlyfont}{\glsabbrvdefaultfont}%
```

stabbrvonlyfont

```
\newcommand*{\glsfirstabbrvonlyfont}{\glsabbrvonlyfont}%
```

glslongonlyfont

```
\newcommand*{\glslongonlyfont}{\glslongdefaultfont}%
```

rstlongonlyfont

```
\newcommand*{\glsfirstlongonlyfont}{\glslongonlyfont}%
```

The default short form suffix:

lsxtronlysuffix

```
\newcommand*{\glsxtronlysuffix}{\glsxtrabbrvpluralsuffix}
```

\glsxtronlyname The default name format for this style.

```
\newcommand*{\glsxtronlyname}{%
  \protect\glsabbrvonlyfont{\the\glsshorttok}%
}
```

only-short-only

```
\newabbreviationstyle{long-only-short-only}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtronlyname},
    sort={\the\glsshorttok},
    first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
    plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}},%
    description={\protect\glslongonlyfont{\the\glslongtok}}}%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \renewcommand*{\abbrvpluralsuffix}{\protect\glsxtronlysuffix}%
  \renewcommand*{\glsabbrvfont}[1]{\glsabbrvonlyfont{##1}}%
  \renewcommand*{\glsfirstabbrvfont}[1]{\glsfirstabbrvonlyfont{##1}}%
  \renewcommand*{\glsfirstlongfont}[1]{\glsfirstlongonlyfont{##1}}%
  \renewcommand*{\glslongfont}[1]{\glslongonlyfont{##1}}%
```

The first use full form doesn't show the short form.

```
  \renewcommand*{\glsxtrfullformat}[2]{%
    \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\glsxtrfullplformat}[2]{%
    \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullformat}[2]{%
```

```
    \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
  \renewcommand*{\Glsxtrfullplformat}[2]{%
    \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
  }%
```

The inline full form does show the short form.

```
  \renewcommand*{\glsxtrinlinefullformat}[2]{%
    \glsfirstlongonlyfont{\glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshort{##1}}}%
  }%
  \renewcommand*{\glsxtrinlinefullplformat}[2]{%
    \glsfirstlongonlyfont{\glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullformat}[2]{%
    \glsfirstlongonlyfont{\Glsaccesslong{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvonlyfont{\glsaccessshortpl{##1}}}%
  }%
  \renewcommand*{\Glsxtrinlinefullplformat}[2]{%
    \glsfirstlongonlyfont{\Glsaccesslongpl{##1}\ifglsxtrinsertinside##2\fi}%
    \ifglsxtrinsertinside\else##2\fi
    \glsxtrfullsep{##1}%
    \glsxtrparen{\protect\glsfirstabbrvonlyfont{\Glsaccessshortpl{##1}}}%
  }%
}
```

xtronlydescsort

```
\newcommand*{\glsxtronlydescsort}{\the\glslongtok}
```

xtronlydescname

```
\newcommand*{\glsxtronlydescname}{%
  \protect\glslongfont{\the\glslongtok}%
}
```

short-only-desc

```
\newabbreviationstyle{long-only-short-only-desc}%
{%
  \renewcommand*{\CustomAbbreviationFields}{%
    name={\glsxtronlydescname},
    sort={\glsxtronlydescsort},%
```

```
    first={\protect\glsfirstlongonlyfont{\the\glslongtok}},%
    firstplural={\protect\glsfirstlongonlyfont{\the\glslongpltok}},%
    text={\protect\glsabbrvonlyfont{\the\glsshorttok}},%
    plural={\protect\glsabbrvonlyfont{\the\glsshortpltok}}%
  }%
```

Unset the regular attribute if it has been set.

```
  \renewcommand*{\GlsXtrPostNewAbbreviation}{%
    \glshasattribute{\the\glslabeltok}{regular}%
    {%
      \glssetattribute{\the\glslabeltok}{regular}{false}%
    }%
    {}%
  }%
}%
{%
  \GlsXtrUseAbbrStyleFmts{long-only-short-only}%
}
```

1.8 Using Entries in Headings

There are four main problems with using entries in sectioning commands: they can mess with the first use flag if they end up in the table of contents, they can add unwanted numbers to the entry's location list, the label is corrupted if used inside `\MakeUppercase` (which is used by the default headings style) and they need to be expandable for PDF bookmarks. The glossaries package therefore recommends the use of the expandable commands, such as `\glsentryshort`, instead but this doesn't reflect the formatting since it doesn't include `\glsabbrvfont`. The commands below are an attempt to get around these problems.

The PDF bookmark issue can easily be fixed with hyperref's `\texorpdfstring` which can simply use the expandable command in the PDF string case. The TEX string case can now use `\glsxtrshort` with the noindex key set, which prevents the unwanted additions to the location list, and the hyper key set to false, which prevents the problem of nested links. This just leaves one thing left that needs to be dealt with, and that's what to do if the heading style uses `\MakeUppercase`.

Note that glossaries automatically loads textcase, so the label can be protected from case change with textcase's `\NoCaseChange`. This means that we don't have a problem provided the page style uses `\MakeTextUppercase`, but the default heading page style uses `\MakeUppercase`.

To get around this, save the original definition of `\markboth` and `\markright` and adjust it so that `\MakeUppercase` is temporarily redefined to `\MakeTextUppercase`. Some packages or classes redefine these commands, so we can't just assume they still have the original kernel definition.

`\markright` Save original definition:

```
\let\@glsxtr@org@markright\markright
```

Redefine (grouping not added in case it interferes with the original code):

```
\renewcommand*{\markright}[1]{%
 \glsxtrmarkhook
 \@glsxtr@org@markright{\@glsxtrinmark#1\@glsxtrnotinmark}%
 \glsxtrrestoremarkhook
}
```

`\markboth` Save original definition:

```
\let\@glsxtr@org@markboth\markboth
```

Redefine (grouping not added in case it interferes with the original code):

```
\renewcommand*{\markboth}[2]{%
 \glsxtrmarkhook
 \@glsxtr@org@markboth
   {\@glsxtrinmark#1\@glsxtrnotinmark}%
   {\@glsxtrinmark#2\@glsxtrnotinmark}%
 \glsxtrrestoremarkhook
}
```

Also do this for `\@starttoc`

`\@starttoc` Save original definition:

```
\let\@glsxtr@org@@starttoc\@starttoc
```

Redefine:

```
\renewcommand*{\@starttoc}[1]{%
 \glsxtrmarkhook
 \@glsxtrinmark
 \@glsxtr@org@@starttoc{#1}%
 \@glsxtrnotinmark
 \glsxtrrestoremarkhook
}
```

If this causes a problem provide a simple way of switching back to the original definitions:

`sxtrRevertMarks`

```
\newcommand*{\glsxtrRevertMarks}{%
  \let\markright\@glsxtr@org@markright
  \let\markboth\@glsxtr@org@markboth
  \let\@starttoc\@glsxtr@org@@starttoc
}
```

`rRevertTocMarks` Just restores `\@starttoc`.

```
\newcommand*{\glsxtrRevertTocMarks}{%
  \let\@starttoc\@glsxtr@org@@starttoc
}
```

`\glsxtrifinmark`

```
\newcommand*{\glsxtrifinmark}[2]{#2}
```

\@glsxtrinmark

```
\newrobustcmd*{\@glsxtrinmark}{%
  \let\glsxtrifinmark\@firstoftwo
}
```

glsxtrnotinmark

```
\newrobustcmd*{\@glsxtrnotinmark}{%
  \let\glsxtrifinmark\@secondoftwo
}
```

eorpdforheading

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsxtrtitleorpdforheading}[3]{\texorpdfstring{#1}{#2}}
}
{
  \newcommand*{\glsxtrtitleorpdforheading}[3]{#1}
}
```

\glsxtrmarkhook Hook used in new definition of \markboth and \markright to make some changes to apply to the marks:

```
\newcommand*{\glsxtrmarkhook}{%
```

Save current definitions:

```
  \let\@glsxtr@org@MakeUppercase\MakeUppercase
  \let\@glsxtr@org@glsxtrtitleorpdforheading\glsxtrtitleorpdforheading
  \let\@glsxtr@org@glsxtrtitleshort\glsxtrtitleshort
  \let\@glsxtr@org@glsxtrtitleshortpl\glsxtrtitleshortpl
  \let\@glsxtr@org@Glsxtrtitleshort\Glsxtrtitleshort
  \let\@glsxtr@org@Glsxtrtitleshortpl\Glsxtrtitleshortpl
  \let\@glsxtr@org@glsxtrtitlename\glsxtrtitlename
  \let\@glsxtr@org@Glsxtrtitlename\Glsxtrtitlename
  \let\@glsxtr@org@glsxtrtitletext\glsxtrtitletext
  \let\@glsxtr@org@Glsxtrtitletext\Glsxtrtitletext
  \let\@glsxtr@org@glsxtrtitleplural\glsxtrtitleplural
  \let\@glsxtr@org@Glsxtrtitleplural\Glsxtrtitleplural
  \let\@glsxtr@org@glsxtrtitlefirst\glsxtrtitlefirst
  \let\@glsxtr@org@Glsxtrtitlefirst\Glsxtrtitlefirst
  \let\@glsxtr@org@glsxtrtitlefirstplural\glsxtrtitlefirstplural
  \let\@glsxtr@org@Glsxtrtitlefirstplural\Glsxtrtitlefirstplural
  \let\@glsxtr@org@glsxtrtitlelong\glsxtrtitlelong
  \let\@glsxtr@org@glsxtrtitlelongpl\glsxtrtitlelongpl
  \let\@glsxtr@org@Glsxtrtitlelong\Glsxtrtitlelong
  \let\@glsxtr@org@Glsxtrtitlelongpl\Glsxtrtitlelongpl
  \let\@glsxtr@org@glsxtrtitlefull\glsxtrtitlefull
  \let\@glsxtr@org@glsxtrtitlefullpl\glsxtrtitlefullpl
  \let\@glsxtr@org@Glsxtrtitlefull\Glsxtrtitlefull
  \let\@glsxtr@org@Glsxtrtitlefullpl\Glsxtrtitlefullpl
```

New definitions

```
  \let\glsxtrifinmark\@firstoftwo
  \let\MakeUppercase\MakeTextUppercase
  \let\glsxtrtitleorpdforheading\@thirdofthree
  \let\glsxtrtitleshort\glsxtrheadshort
  \let\glsxtrtitleshortpl\glsxtrheadshortpl
  \let\Glsxtrtitleshort\Glsxtrheadshort
  \let\Glsxtrtitleshortpl\Glsxtrheadshortpl
  \let\glsxtrtitlename\glsxtrheadname
  \let\Glsxtrtitlename\Glsxtrheadname
  \let\glsxtrtitletext\glsxtrheadtext
  \let\Glsxtrtitletext\Glsxtrheadtext
  \let\glsxtrtitleplural\glsxtrheadplural
  \let\Glsxtrtitleplural\Glsxtrheadplural
  \let\glsxtrtitlefirst\glsxtrheadfirst
  \let\Glsxtrtitlefirst\Glsxtrheadfirst
  \let\glsxtrtitlefirstplural\glsxtrheadfirstplural
  \let\Glsxtrtitlefirstplural\Glsxtrheadfirstplural
  \let\glsxtrtitlelong\glsxtrheadlong
  \let\glsxtrtitlelongpl\glsxtrheadlongpl
  \let\Glsxtrtitlelong\Glsxtrheadlong
  \let\Glsxtrtitlelongpl\Glsxtrheadlongpl
  \let\glsxtrtitlefull\glsxtrheadfull
  \let\glsxtrtitlefullpl\glsxtrheadfullpl
  \let\Glsxtrtitlefull\Glsxtrheadfull
  \let\Glsxtrtitlefullpl\Glsxtrheadfullpl
}
```

`restoremarkhook` Hook used in new definition of `\markboth` and `\markright` to restore the modified definitions. (This is in case the original `\markboth` and `\markright` shouldn't be grouped for some reason. There already is some grouping within those original definitions, but some of the code lies outside that grouping, and possibly there's a reason for it.)

```
\newcommand*{\glsxtrrestoremarkhook}{%
  \let\glsxtrifinmark\@secondoftwo
  \let\MakeUppercase\@glsxtr@org@MakeUppercase
  \let\glsxtrtitleorpdforheading\@glsxtr@org@glsxtrtitleorpdforheading
  \let\glsxtrtitleshort\@glsxtr@org@glsxtrtitleshort
  \let\glsxtrtitleshortpl\@glsxtr@org@glsxtrtitleshortpl
  \let\Glsxtrtitleshort\@glsxtr@org@Glsxtrtitleshort
  \let\Glsxtrtitleshortpl\@glsxtr@org@Glsxtrtitleshortpl
  \let\glsxtrtitlename\@glsxtr@org@glsxtrtitlename
  \let\Glsxtrtitlename\@glsxtr@org@Glsxtrtitlename
  \let\glsxtrtitletext\@glsxtr@org@glsxtrtitletext
  \let\Glsxtrtitletext\@glsxtr@org@Glsxtrtitletext
  \let\glsxtrtitleplural\@glsxtr@org@glsxtrtitleplural
  \let\Glsxtrtitleplural\@glsxtr@org@Glsxtrtitleplural
  \let\glsxtrtitlefirst\@glsxtr@org@glsxtrtitlefirst
  \let\Glsxtrtitlefirst\@glsxtr@org@Glsxtrtitlefirst
  \let\glsxtrtitlefirstplural\@glsxtr@org@glsxtrtitlefirstplural
```

```
  \let\Glsxtrtitlefirstplural\@glsxtr@org@Glsxtrtitlefirstplural
  \let\glsxtrtitlelong\@glsxtr@org@glsxtrtitlelong
  \let\glsxtrtitlelongpl\@glsxtr@org@glsxtrtitlelongpl
  \let\Glsxtrtitlelong\@glsxtr@org@Glsxtrtitlelong
  \let\Glsxtrtitlelongpl\@glsxtr@org@Glsxtrtitlelongpl
  \let\glsxtrtitlefull\@glsxtr@org@glsxtrtitlefull
  \let\glsxtrtitlefullpl\@glsxtr@org@glsxtrtitlefullpl
  \let\Glsxtrtitlefull\@glsxtr@org@Glsxtrtitlefull
  \let\Glsxtrtitlefullpl\@glsxtr@org@Glsxtrtitlefullpl
}
```

Instead of using one document-wide conditional, use headuc attribute to determine whether or not to use the all upper case form.

`glsxtrheadshort` Command used to display short form in the page header.

```
\newcommand*{\glsxtrheadshort}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshort[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsxtrshort[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`lsxtrtitleshort` Command to display short form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitleshort}[1]{%
  \glsxtrshort[noindex,hyper=false]{#1}[]%
}
```

`sxtrheadshortpl` Command used to display plural short form in the page header. If you want the text converted to upper case, this needs to be redefined to use `\GLSxtrshortpl` instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
\newcommand*{\glsxtrheadshortpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
      \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
   {%
      \glsxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitleshortpl}[1]{%
  \glsxtrshortpl[noindex,hyper=false]{#1}[]%
}
```

Glsxtrheadshort Command used to display short form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadshort}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshort[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrshort[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

lsxtrtitleshort Command to display short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitleshort}[1]{%
  \Glsxtrshort[noindex,hyper=false]{#1}[]%
}
```

sxtrheadshortpl Command used to display plural short form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadshortpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

xtrtitleshortpl Command to display plural short form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitleshortpl}[1]{%
  \Glsxtrshortpl[noindex,hyper=false]{#1}[]%
}
```

\glsxtrheadname As above but for the name value.

```
\newcommand*{\glsxtrheadname}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSname[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsname[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`glsxtrtitlename` Command to display name value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlename}[1]{%
  \glsname[noindex,hyper=false]{#1}[]%
}
```

`\Glsxtrheadname` First letter converted to upper case

```
\newcommand*{\Glsxtrheadname}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSname[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsname[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`Glsxtrtitlename` Command to display name value in section title and table of contents with the first letter changed to upper case.

```
%\changes{1.21}{2017-11-03}{new}
\newrobustcmd*{\Glsxtrtitlename}[1]{%
  \Glsname[noindex,hyper=false]{#1}[]%
}
```

`\glsxtrheadtext` As above but for the text value.

```
\newcommand*{\glsxtrheadtext}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLStext[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glstext[noindex,hyper=false]{#1}[]%
```

```
    }%
  }%
}
```

`glsxtrtitletext` Command to display text value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitletext}[1]{%
  \glstext[noindex,hyper=false]{#1}[]%
}
```

`\Glsxtrheadtext` First letter converted to upper case

```
\newcommand*{\Glsxtrheadtext}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLStext[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glstext[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`Glsxtrtitletext` Command to display text value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitletext}[1]{%
  \Glstext[noindex,hyper=false]{#1}[]%
}
```

`lsxtrheadplural` As above but for the plural value.

```
\newcommand*{\glsxtrheadplural}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSplural[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsplural[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`sxtrtitleplural` Command to display plural value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitleplural}[1]{%
  \glsplural[noindex,hyper=false]{#1}[]%
}
```

lsxtrheadplural Convert first letter to upper case.

```
\newcommand*{\Glsxtrheadplural}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSplural[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsplural[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

sxtrtitleplural Command to display plural value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitleplural}[1]{%
  \Glsplural[noindex,hyper=false]{#1}[]%
}
```

glsxtrheadfirst As above but for the first value.

```
\newcommand*{\glsxtrheadfirst}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSfirst[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsfirst[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

lsxtrtitlefirst Command to display first value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefirst}[1]{%
  \glsfirst[noindex,hyper=false]{#1}[]%
}
```

Glsxtrheadfirst First letter converted to upper case

```
\newcommand*{\Glsxtrheadfirst}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSfirst[noindex,hyper=false]{#1}[]%
   }%
   {%
```

```
      \Glsfirst[noindex,hyper=false]{#1}[]%
    }%
  }%
}
```

lsxtrtitlefirst Command to display first value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitlefirst}[1]{%
  \Glsfirst[noindex,hyper=false]{#1}[]%
}
```

headfirstplural As above but for the firstplural value.

```
\newcommand*{\glsxtrheadfirstplural}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSfirstplural[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsfirstplural[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

itlefirstplural Command to display firstplural value in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefirstplural}[1]{%
  \glsfirstplural[noindex,hyper=false]{#1}[]%
}
```

headfirstplural First letter converted to upper case

```
\newcommand*{\Glsxtrheadfirstplural}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSfirstplural[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsfirstplural[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

itlefirstplural Command to display first value in section title and table of contents with the first letter changed to upper case.

```
\newrobustcmd*{\Glsxtrtitlefirstplural}[1]{%
  \Glsfirstplural[noindex,hyper=false]{#1}[]%
}
```

\glsxtrheadlong Command used to display long form in the page header.

```
\newcommand*{\glsxtrheadlong}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrlong[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsxtrlong[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

glsxtrtitlelong Command to display long form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlelong}[1]{%
  \glsxtrlong[noindex,hyper=false]{#1}[]%
}
```

lsxtrheadlongpl Command used to display plural long form in the page header. If you want the text converted to upper case, this needs to be redefined to use \GLSxtrlongpl instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
\newcommand*{\glsxtrheadlongpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
      \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
   {%
      \glsxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

sxtrtitlelongpl Command to display plural long form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlelongpl}[1]{%
  \glsxtrlongpl[noindex,hyper=false]{#1}[]%
}
```

\Glsxtrheadlong Command used to display long form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadlong}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrlong[noindex,hyper=false]{#1}[]%
```

```
    }%
    {%
      \Glsxtrlong[noindex,hyper=false]{#1}[]%
    }%
  }%
}
```

`Glsxtrtitlelong` Command to display long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlelong}[1]{%
  \Glsxtrlong[noindex,hyper=false]{#1}[]%
}
```

`lsxtrheadlongpl` Command used to display plural long form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadlongpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

`sxtrtitlelongpl` Command to display plural long form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlelongpl}[1]{%
  \Glsxtrlongpl[noindex,hyper=false]{#1}[]%
}
```

`\glsxtrheadfull` Command used to display full form in the page header.

```
\newcommand*{\glsxtrheadfull}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrfull[noindex,hyper=false]{#1}[]%
   }%
   {%
     \glsxtrfull[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefull}[1]{%
  \glsxtrfull[noindex,hyper=false]{#1}[]%
}
```

lsxtrheadfullpl Command used to display plural full form in the page header. If you want the text converted to upper case, this needs to be redefined to use \GLSxtrfullpl instead. If you are using a smallcaps style, the default fonts don't provide italic smallcaps.

```
\newcommand*{\glsxtrheadfullpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
      \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
   }%
   {%
      \glsxtrfullpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents.

```
\newrobustcmd*{\glsxtrtitlefullpl}[1]{%
  \glsxtrfullpl[noindex,hyper=false]{#1}[]%
}
```

\Glsxtrheadfull Command used to display full form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadfull}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrfull[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrfull[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

Glsxtrtitlefull Command to display full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlefull}[1]{%
  \Glsxtrfull[noindex,hyper=false]{#1}[]%
}
```

lsxtrheadfullpl Command used to display plural full form in the page header with the first letter converted to upper case.

```
\newcommand*{\Glsxtrheadfullpl}[1]{%
 \protect\NoCaseChange
 {%
   \glsifattribute{#1}{headuc}{true}%
   {%
     \GLSxtrfullpl[noindex,hyper=false]{#1}[]%
   }%
   {%
     \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
   }%
 }%
}
```

sxtrtitlefullpl Command to display plural full form of abbreviation in section title and table of contents with the first letter converted to upper case.

```
\newrobustcmd*{\Glsxtrtitlefullpl}[1]{%
  \Glsxtrfullpl[noindex,hyper=false]{#1}[]%
}
```

\glsfmtshort Provide a way of using the formatted short form in section headings. If hyperref has been loaded, use \texorpdfstring for convenience in PDF bookmarks.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtshort}[1]{%
    \texorpdfstring
      {\glsxtrtitleshort{#1}}%
      {\glsentryshort{#1}}%
  }
}
{
  \newcommand*{\glsfmtshort}[1]{%
   \glsxtrtitleshort{#1}}
}
```

Similarly for the plural version.

\glsfmtshortpl

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtshortpl}[1]{%
    \texorpdfstring
      {\glsxtrtitleshortpl{#1}}%
      {\glsentryshortpl{#1}}%
  }
}
{
  \newcommand*{\glsfmtshortpl}[1]{%
```

```
    \glsxtrtitleshortpl{#1}}
}
```

The case-changing version isn't suitable for PDF bookmarks, so the PDF alternative uses the non-case-changing version.

`\Glsfmtshort` Singular form (first letter uppercase).

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtshort}[1]{%
    \texorpdfstring
      {\Glsxtrtitleshort{#1}}%
      {\glsentryshort{#1}}%
  }
}
{
  \newcommand*{\Glsfmtshort}[1]{%
   \Glsxtrtitleshort{#1}}
}
```

`\Glsfmtshortpl` Plural form (first letter uppercase).

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtshortpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitleshortpl{#1}}%
    {\glsentryshortpl{#1}}%
  }
}
{
  \newcommand*{\Glsfmtshortpl}[1]{%
   \Glsxtrtitleshortpl{#1}}
}
```

`\glsfmtname` As above but for the name value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtname}[1]{%
    \texorpdfstring
    {\glsxtrtitlename{#1}}%
    {\glsentryname{#1}}%
  }
}
{
  \newcommand*{\glsfmtname}[1]{%
   \glsxtrtitlename{#1}}
}
```

`\Glsfmtname` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtname}[1]{%
    \texorpdfstring
    {\Glsxtrtitlename{#1}}%
    {\glsentryname{#1}}%
  }
}
{
  \newcommand*{\Glsfmtname}[1]{%
   \Glsxtrtitlename{#1}}
}
```

`\glsfmttext` As above but for the text value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmttext}[1]{%
    \texorpdfstring
    {\glsxtrtitletext{#1}}%
    {\glsentrytext{#1}}%
  }
}
{
  \newcommand*{\glsfmttext}[1]{%
   \glsxtrtitletext{#1}}
}
```

`\Glsfmttext` First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmttext}[1]{%
    \texorpdfstring
    {\Glsxtrtitletext{#1}}%
    {\glsentrytext{#1}}%
  }
}
{
  \newcommand*{\Glsfmttext}[1]{%
   \Glsxtrtitletext{#1}}
}
```

`\glsfmtplural` As above but for the plural value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtplural}[1]{%
    \texorpdfstring
    {\glsxtrtitleplural{#1}}%
    {\glsentryplural{#1}}%
  }
```

```
}
{
  \newcommand*{\glsfmtplural}[1]{%
   \glsxtrtitleplural{#1}}
}
```

\Glsfmtplural First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtplural}[1]{%
    \texorpdfstring
    {\Glsxtrtitleplural{#1}}%
    {\glsentryplural{#1}}%
  }
}
{
  \newcommand*{\Glsfmtplural}[1]{%
   \Glsxtrtitleplural{#1}}
}
```

\glsfmtfirst As above but for the first value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfirst}[1]{%
    \texorpdfstring
    {\glsxtrtitlefirst{#1}}%
    {\glsentryfirst{#1}}%
  }
}
{
  \newcommand*{\glsfmtfirst}[1]{%
   \glsxtrtitlefirst{#1}}
}
```

\Glsfmtfirst First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfirst}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefirst{#1}}%
    {\glsentryfirst{#1}}%
  }
}
{
  \newcommand*{\Glsfmtfirst}[1]{%
   \Glsxtrtitlefirst{#1}}
}
```

\glsfmtfirstpl As above but for the firstplural value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfirstpl}[1]{%
    \texorpdfstring
    {\glsxtrtitlefirstplural{#1}}%
    {\glsentryfirstplural{#1}}%
  }
}
{
  \newcommand*{\glsfmtfirstpl}[1]{%
   \glsxtrtitlefirstplural{#1}}
}
```

\Glsfmtfirstpl First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfirstpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefirstplural{#1}}%
    {\glsentryfirstplural{#1}}%
  }
}
{
  \newcommand*{\Glsfmtfirstpl}[1]{%
   \Glsxtrtitlefirstplural{#1}}
}
```

\glsfmtlong As above but for the long value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtlong}[1]{%
    \texorpdfstring
    {\glsxtrtitlelong{#1}}%
    {\glsentrylong{#1}}%
  }
}
{
  \newcommand*{\glsfmtlong}[1]{%
   \glsxtrtitlelong{#1}}
}
```

\Glsfmtlong First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtlong}[1]{%
    \texorpdfstring
    {\Glsxtrtitlelong{#1}}%
    {\glsentrylong{#1}}%
  }
```

```
}
{
  \newcommand*{\Glsfmtlong}[1]{%
   \Glsxtrtitlelong{#1}}
}
```

\glsfmtlongpl As above but for the longplural value.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtlongpl}[1]{%
    \texorpdfstring
    {\glsxtrtitlelongpl{#1}}%
    {\glsentrylongpl{#1}}%
  }
}
{
  \newcommand*{\glsfmtlongpl}[1]{%
   \glsxtrtitlelongpl{#1}}
}
```

\Glsfmtlongpl First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtlongpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitlelongpl{#1}}%
    {\glsentrylongpl{#1}}%
  }
}
{
  \newcommand*{\Glsfmtlongpl}[1]{%
   \Glsxtrtitlelongpl{#1}}
}
```

\glsfmtfull In-line full format.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfull}[1]{%
    \texorpdfstring
    {\glsxtrtitlefull{#1}}%
    {\glsxtrinlinefullformat{#1}{}}%
  }
}
{
  \newcommand*{\glsfmtfull}[1]{%
   \glsxtrtitlefull{#1}}
}
```

\Glsfmtfull First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfull}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefull{#1}}%
    {\Glsxtrinlinefullformat{#1}{}}%
  }
}
{
  \newcommand*{\Glsfmtfull}[1]{%
   \Glsxtrtitlefull{#1}}
}
```

\glsfmtfullpl In-line full plural format.

```
\ifdef\texorpdfstring
{
  \newcommand*{\glsfmtfullpl}[1]{%
    \texorpdfstring
    {\glsxtrtitlefullpl{#1}}%
    {\glsxtrinlinefullplformat{#1}{}}%
  }
}
{
  \newcommand*{\glsfmtfullpl}[1]{%
   \glsxtrtitlefullpl{#1}}
}
```

\Glsfmtfullpl First letter converted to upper case.

```
\ifdef\texorpdfstring
{
  \newcommand*{\Glsfmtfullpl}[1]{%
    \texorpdfstring
    {\Glsxtrtitlefullpl{#1}}%
    {\Glsxtrinlinefullplformat{#1}{}}%
  }
}
{
  \newcommand*{\Glsfmtfullpl}[1]{%
   \Glsxtrtitlefullpl{#1}}
}
```

1.9 Multi-Lingual Support

Add the facility to load language modules, if they are installed, but none are provided with this package.

sariesExtraLang

```
\newcommand*{\RequireGlossariesExtraLang}[1]{%
```

```
  \@ifundefined{ver@glossariesxtr-#1.ldf}{\input{glossariesxtr-#1.ldf}}{}%
}
```

sariesExtraLang

```
\newcommand*{\ProvidesGlossariesExtraLang}[1]{%
  \ProvidesFile{glossariesxtr-#1.ldf}%
}
```

Load any required language modules that are available. This doesn't generate any warning if none are found, since they're not essential. (The only command that really needs defining for the document is `\abbreviationsname`, which can simply be redefined. However, with `bib2gls` it might be useful to provide custom rules for a particular locale.)

xtr@loaddialect The dialect label should be stored in `\this@dialect` before using this command.

```
\newcommand{\glsxtr@loaddialect}{%
  \IfTrackedLanguageFileExists{\this@dialect}%
  {glossariesxtr-}% prefix
  {.ldf}%
  {%
    \RequireGlossariesExtraLang{\CurrentTrackedTag}%
  }%
  {}% not found
```

If glossaries-extra-bib2gls has been loaded, `\@glsxtrdialecthook` will check for the associated script, otherwise it will do nothing.

```
  \@glsxtrdialecthook
}

\@ifpackageloaded{tracklang}
{%
  \AnyTrackedLanguages
  {%
    \ForEachTrackedDialect{\this@dialect}{\glsxtr@loaddialect}%
  }%
  {}%
}
{}
```

Load glossaries-extra-stylemods if required.

```
\@glsxtr@redefstyles
```

and set the style:

```
\@glsxtr@do@style
```

1.10 glossaries-extra-bib2gls.sty

This package provides additional support for `bib2gls` and is automatically loaded by the record option.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{glossaries-extra-bib2gls}[2018/11/30 1.37 (NLCT)]
```

These are some convenient macros for use with custom rules.

\glshex

```
\newcommand*{\glshex}{\string\u}
```

lscapturedgroup

```
\newcommand*{\glscapturedgroup}{\string\$}
```

nZeroChildCount For use with bib2gls's save-child-count resource option.

```
\newcommand*{\GlsXtrIfHasNonZeroChildCount}[3]{%
  \GlsXtrIfFieldNonZero{childcount}{#1}{#2}{#3}%
}
```

rprovidecommand For use in @preamble, this behaves like \providecommand in the document but like \renewcommand in bib2gls.

```
\newcommand*{\glsxtrprovidecommand}{\providecommand}
```

glsrenewcommand Like \renewcommand but only generates a warning rather than an error if the command isn't defined.

```
\newcommand*{\glsrenewcommand}{\@star@or@long\glsxtr@renewcommand}
```

tr@renewcommand

```
\newcommand*{\glsxtr@renewcommand}[1]{%
 \begingroup \escapechar\m@ne\xdef\@gtempa{{\string#1}}\endgroup
 \expandafter\@ifundefined\@gtempa
   {%
     \GlossariesExtraWarning{can't redefine \noexpand#1(not already defined)}%
   }%
   \relax
  \relax
 \let\@ifdefinable\@rc@ifdefinable
 \new@command#1%
}
```

lossarylocation For use with indexcounter and bib2gls.

```
\newcommand*{\glsxtr@wrglossarylocation}[2]{#1}
```

IndexCounterLink `\GlsXtrIndexCounterLink{⟨text⟩}{⟨label⟩}`

For use with indexcounter and bib2gls.

```
\ifdef\hyperref
{%
  \newcommand*{\GlsXtrIndexCounterLink}[2]{%
```

```
    \glsxtrifhasfield{indexcounter}{#2}%
    {\hyperref[wrglossary.\glscurrentfieldvalue]{#1}}%
    {#1}%
  }
}
{
  \newcommand*{\GlsXtrIndexCounterLink}[2]{#1}
}
```

\GlsXtrDualField

\GlsXtrDualField

The internal field used to store the dual label. The `dual-field` defaults to `dual` if no value is supplied so that's used as the default.

```
\newcommand*{\GlsXtrDualField}{dual}
```

sXtrDualBackLink

\GlsXtrDualBackLink{⟨*text*⟩}{⟨*label*⟩}

Adds a hyperlink to the dual entry.

```
\newcommand*{\GlsXtrDualBackLink}[2]{%
  \glsxtrifhasfield{\GlsXtrDualField}{#2}%
  {\glshyperlink[#1]{\glscurrentfieldvalue}}%
  {#2}%
}
```

TeXEntryAliases

Convenient shortcut for use with `entry-type-aliases` to alias standard BibTeX entry types to `@bibtexentry`.

```
\newcommand*{\GlsXtrBibTeXEntryAliases}{%
  article=bibtexentry,
  book=bibtexentry,
  booklet=bibtexentry,
  conference=bibtexentry,
  inbook=bibtexentry,
  incollection=bibtexentry,
  inproceedings=bibtexentry,
  manual=bibtexentry,
  mastersthesis=bibtexentry,
  misc=bibtexentry,
  phdthesis=bibtexentry,
  proceedings=bibtexentry,
  techreport=bibtexentry,
  unpublished=bibtexentry
}
```

ideBibTeXFields Convenient shortcut to define the standard BibTeX fields.

```
\newcommand*{\GlsXtrProvideBibTeXFields}{%
  \glsaddstoragekey{address}{}{\glsxtrbibaddress}%
  \glsaddstoragekey{author}{}{\glsxtrbibauthor}%
  \glsaddstoragekey{booktitle}{}{\glsxtrbibbooktitle}%
  \glsaddstoragekey{chapter}{}{\glsxtrbibchapter}%
  \glsaddstoragekey{edition}{}{\glsxtrbibedition}%
  \glsaddstoragekey{howpublished}{}{\glsxtrbibhowpublished}%
  \glsaddstoragekey{institution}{}{\glsxtrbibinstitution}%
  \glsaddstoragekey{journal}{}{\glsxtrbibjournal}%
  \glsaddstoragekey{month}{}{\glsxtrbibmonth}%
  \glsaddstoragekey{note}{}{\glsxtrbibnote}%
  \glsaddstoragekey{number}{}{\glsxtrbibnumber}%
  \glsaddstoragekey{organization}{}{\glsxtrbiborganization}%
  \glsaddstoragekey{pages}{}{\glsxtrbibpages}%
  \glsaddstoragekey{publisher}{}{\glsxtrbibpublisher}%
  \glsaddstoragekey{school}{}{\glsxtrbibschool}%
  \glsaddstoragekey{series}{}{\glsxtrbibseries}%
  \glsaddstoragekey{title}{}{\glsxtrbibtitle}%
  \glsaddstoragekey{bibtextype}{}{\glsxtrbibtype}%
  \glsaddstoragekey{volume}{}{\glsxtrbibvolume}%
}
```

Multiple supplementary references are only supported with `bib2gls`.

ltisupplocation This is like `\glsxtrsupphypernumber` but the second argument is the external file name (which isn't obtained from the externallocation attribute). The third argument is the formatting (encap) control sequence *name*. This is ignored by default, but is set by `bib2gls` to the original encap in case it's required.

```
\newcommand*{\glsxtrmultisupplocation}[3]{%
 {%
   \def\glsxtrsupplocationurl{#2}%
   \glshypernumber{#1}%
 }%
}
```

trdisplaysupploc `\glsxtrdisplaysupploc{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨src⟩}{⟨location⟩}`

This is like `\glsnoidxdisplayloc` but is used for supplementary locations and so requires an extra argument.

```
\newcommand*\glsxtrdisplaysupploc[5]{%
  \setentrycounter[#1]{#2}%
  \glsxtrmultisupplocation{#5}{#4}{#3}%
}
```

splaylocnameref `\glsxtrdisplaylocnameref{⟨prefix⟩}{⟨counter⟩}{⟨format⟩}{⟨location⟩}{⟨name⟩}{⟨href⟩}{⟨hcounter⟩}{⟨external file⟩}` Used with the [nameref]record package option. The ⟨*href*⟩ argument was obtained from `\@currentHref` and the ⟨*hcounter*⟩ argument was obtained from `\theHentrycounter`, which is more reliable. If hyperref hasn't been loaded, this just behaves like `\glsnoidxdisplayloc`.

```
\ifundef\hyperlink
{
  \newcommand*{\glsxtrdisplaylocnameref}[8]{%
    \glsnoidxdisplayloc{#1}{#2}{#3}{#4}%
  }
}
{
```

Default action uses ⟨*hcounter*⟩. Equations and pages typically don't have a title, so check the counter name.

```
  \newcommand*{\glsxtrdisplaylocnameref}[8]{%
    \ifstrequal{#2}{equation}%
    {\glsxtrnamereflink{#3}{(#4)}{#2.#7}{#8}}%
    {%
      \ifstrempty{#5}%
      {%
```

No title, so just use the location as the link text.

```
        \glsxtrnamereflink{#3}{#4}{#2.#7}{#8}%
      }%
      {%
        \ifstrequal{#2}{page}%
        {\glsxtrnamereflink{#3}{#4}{#2.#7}{#8}}%
        {\glsxtrnamereflink{#3}{#5}{#2.#7}{#8}}%
      }%
    }%
  }
}
```

lsxtrnamereflink `\glsxtrfmtnamereflink{⟨format⟩}{⟨title⟩}{⟨href⟩}{⟨external file⟩}`

```
\newcommand*{\glsxtrnamereflink}[4]{%
```

Locally change `\glshypernumber` to `\@firstofone` to remove the normal location hyperlink.

```
  \begingroup
    \let\glshypernumber\@firstofone
```

If the ⟨*external file*⟩ argument is empty, an internal link is used, otherwise an external one is needed.

```
    \ifstrempty{#4}%
    {\glsxtrfmtinternalnameref{#3}{#1}{#2}}%
```

```
    {\glsxtrfmtexternalnameref{#3}{#1}{#2}{#4}}%
  \endgroup
}
```

lsxtrnameloclink

\glsxtrnamerefloclink{⟨*prefix*⟩}{⟨*counter*⟩}{⟨*format*⟩}{⟨*location*⟩}{⟨*text*⟩}{⟨*external file*⟩}

Like \@gls@numberlink, this creates a hyperlink to the target obtained from the prefix, counter and location but uses ⟨*text*⟩ as the hyperlink text. As with regular indexing, this will fail if the target name can't be formed by prefixing the location value.

```
\newcommand{\glsxtrnameloclink}[6]{%
 \begingroup
  \setentrycounter[#1]{#2}%
  \def\glsxtr@locationhypertext{#5}%
  \let\glshypernumber\@firstofone
  \def\@glsnumberformat{#3}%
  \def\glsxtrsupplocationurl{#6}%
  \toks@={}%
  \@glsxtr@bibgls@removespaces#4 \@nil
 \endgroup
}
```

ls@removespaces

```
\def\@glsxtr@bibgls@removespaces#1 #2\@nil{%
 \toks@=\expandafter{\the\toks@#1}%
 \ifx\\#2\\%
   \edef\x{\the\toks@}%
   \ifx\x\empty
   \else
     \protected@edef\x{\glsentrycounter\@glo@counterprefix\the\toks@}%
     \ifdefvoid\glsxtrsupplocationurl
     {%
       \expandafter\glsxtrfmtinternalnameref\expandafter{\x}%
       {\@glsnumberformat}{\glsxtr@locationhypertext}%
     }%
     {%
       \expandafter\glsxtrfmtexternalnameref\expandafter{\x}%
       {\@glsnumberformat}{\glsxtr@locationhypertext}{\glsxtrsupplocationurl}%
     }%
   \fi
 \else
   \@gls@ReturnAfterFi{%
     \@glsxtr@bibgls@removespaces#2\@nil
   }%
 \fi
}
```