xcref – Extension of cleveref for non-English languages*

Florent Rougon†

Released 2019-09-19

Abstract

In non-English languages, words such as prepositions and articles need to be adapted in function of the noun they are used with, as well as possibly in function of "cases" such as nominative, accusative, dative, and genitive as used in German. As far as I know, these are things that cleveref does not allow to do in any practical way, short of programming it. This package works on top of cleveref and, given language-specific knowledge as well as additional user input (at least to select a preposition), it makes it possible to produce references that cope with the various forms of ancillary words such as articles and prepositions, and don't break when the underlying noun is changed (for instance, from chapter to section or theorem to proposition).

Contents

*This file describes xcref v0.1a, last revised 2019-09-19.

†E-mail: mailto:f.rougon@free.fr

WARNING: this is an alpha release. Some interfaces may change if they prove to be suboptimal. The documentation isn't finished. The code has received reasonable testing but might still have bugs.

1 Introduction

Generating cross-references to chapters, sections, theorems, lemmas, etc. in English is easy, because it can be done in a very simple and mechanical way ("see chapter 1", "as we saw in section 2", "an important consequence of theorems 3 and 4", etc.). However, it can be more difficult in other languages, especially when you expect that the types of your references may change: a section might become a chapter after some reorganization of your work; a theorem could be downgraded to a proposition, etc.

The cleveref package makes it easy to change reference types and have the names adapted, however its approach is too simple to work well with languages such as French and German. For instance, in French, "la section" would have to become "le chapitre", but you can't just use `\crefname` to tell `\cref` to use `la section`, `le chapitre`, etc. as the *names* associated to the `section` and `chapter` reference types, because depending on the preposition that is used, you need various forms: "à la section", "au chapitre", "de la section", "du chapitre", "d'après la section", "d'après le chapitre", etc. Exactly the same happens with "théorème" and "proposition" (these are French words). Fortunately, there is enough regularity in how these various forms are produced to allow writing code that automatically produces correct French with input such as `\xcref[french/preposition=d'après]{some-label}`.

Note that convenience wrappers are provided to shorten the previous command (see section 3.4 and the example files) and that several labels can be used in the same command, when one wishes cleveref to combine them; this works especially well when all such labels are of the same type.

Other languages have rules of the same kind and can also be nicely accomodated. For instance, with the `ngerman` module, one can reference two propositions with the "nach" preposition using code such as:

```
\xcref[ngerman/preposition=nach]{some-prop,other-prop}
```

(again, there are convenience wrappers to make this shorter). xcref knows that "nach" is always followed by the dative case, so it will produce reference text such as "nach den Propositionen 3 und 4". With the `/xcref/ngerman/form` option (which has its counterpart in the `french` module), one can obtain variant forms: in the same conditions,

```
\xcref[ngerman/form=article+noun]{some-prop,other-prop}
```

would yield "den Propositionen 3 und 4" and

```
\xcref[ngerman/form=noun]{some-prop,other-prop}
```

would produce "Propositionen 3 und 4" (exactly the same options are available in the `french` module along with the initial value `prep+article+noun`, however this may not fit all languages, hence the use of the `/xcref/ngerman` and `/xcref/french` namespaces for the `form` option). In German, it is also possible, and very often necessary, to indicate which case to use between nominative, accusative, dative and genitive, since in many cases, understanding the sentence is required to choose the appropriate case. This can be done using the `/xcref/ngerman/case` option which can take the values `nom`, `acc`, `dat`, and `gen`.

2 Installation

2.1 Code

Installation of xcref can be done with the following steps:

1. Go to the directory containing `build.lua` and `xcref.dtx`.
2. (a) If you have `l3build` and agree to install xcref in the appropriate place under `TEXMFHOME`, run the command `l3build install`.

   If you want to use a different base directory, you can pass the `--texmfhome` option (e.g., `--texmfhome=`⟨*your choice*⟩) after `l3build install`. In order to see which files would be installed without actually installing them, run `l3build install --dry-run`. See `texdoc l3build` for more details.

   (b) Otherwise, run `tex xcref.ins` (or `pdftex xcref.ins`, etc.) in order to extract `xcref.sty` and the `xcref-*.tex` language modules from `xcref.dtx` to the current directory (unless you have a `docstrip.cfg` file that says otherwise, of course). Then, install the extracted files as usual in a place listed in your `TEXINPUTS` and refresh the "filename database" of your TEX distribution if needed.

Once these steps have been performed, you should be able to build this document yourself (see below) and compile the `*.tex` files in the `examples` directory.

2.2 Documentation

In order to build the documentation (`xcref.pdf`), you need to have `xcref.sty` somewhere in your `TEXINPUTS`. How to do so is explained in the previous section. We'll assume you've already done that. There are essentially two ways to build he documentation. In either case, the first thing to do is to go to the directory containing `build.lua` and `xcref.dtx`.

With `l3build`

This is the easiest way: simply run `l3build doc`.

Without `l3build`

Run a LATEX-based engine on `xcref.dtx`. You also need to use `makeindex` to generate the index. For instance, the following command sequence is likely to produce a correct xcref manual:

```
pdflatex xcref.dtx
makeindex -s gind.ist -o xcref.ind xcref.idx
pdflatex xcref.dtx
pdflatex xcref.dtx
makeindex -s gind.ist -o xcref.ind xcref.idx
pdflatex xcref.dtx
```

Choosing a different paper size

Note: the following won't work unless your `l3doc.cls` is from 2019-09-18 or later (see LaTeX3 commit ef39a40f586f).

In case you want the documentation to be produced in A4 format, create a file named `l3doc.cfg` in the directory containing `xcref.dtx`, with the following contents:

```
\PassOptionsToClass{a4paper}{l3doc}
```

then run `l3build doc` as usual.

Removing generated files

You can remove generated files (not only from the documentation build) with the `l3build clean` command. Note: this will remove `xcref.pdf` too.

3 Usage

3.1 Package options

The xcref package accepts only one fixed option, namely `languages`. The other options are names of language modules, such as `french` and `ngerman`. The `languages` option accepts a comma-separated list of languages, therefore the following are equivalent:

- `\usepackage[french, ngerman]{xcref}`
- `\usepackage[languages={french, ngerman}]{xcref}`
- `\usepackage{xcref}`
  `\xcrefusemodules{french, ngerman}`

Language modules are loaded in the specified order in each of these three examples, but no module is loaded twice. Therefore,

```
\usepackage[ngerman,french,ngerman]{xcref}
```

will load the `ngerman` module followed by the `french` module and stop there.

Unless you intend to write xcref-managed cross-references in several languages, you only need to load one language module, and its name can very well be picked from the `\documentclass` options. The language in which cross-references are written in the document is not determined by the load order of language modules, but by the current babel language at that point in the document or by the `/xcref/lang` PGF key if specified (which then takes precedence over the current babel language).

3.2 PGF keys

The options defined in this section are managed by pgfkeys. In what follows, they will be referred to as *PGF keys*, or simply *keys*. These options can be used in:

- the optional argument of `\xcref` (first argument of `\xcref:nn`);
- the argument of `\xcrefset` (and thus of `\xcref_set:n`).

3.2.1 Generic

`/xcref/capitalize=true|false` (default `true`, initially `false`)

When set to `true`, causes references to start with a capital letter. Capitalization can be inhibited for specific parts of a reference text by means of macros listed via `/xcref/functions for preventing auto case change`.

`/xcref/hyperlinks=true|false` (default `true`, initially `true`)

When hyperref is loaded, the value `true` causes xcref to use hyperlinks in references.

`/xcref/lang=`⟨*language*⟩ (no default, initially empty)

Determine the language in which references are to be typeset. This doesn't cause xcref to change the current babel or polyglossia language—this aspect is entirely up to you. ⟨*language*⟩ should be either empty or the name of an xcref language module, such as `french` or `ngerman` (so far, these are babel language names and this will remain so unless there is a good reason to depart from this naming scheme). If ⟨*language*⟩ is empty, the language to use is autodetected whenever xcref is asked to typeset a reference. This is done with the iflang package and uses the language that is current at the point where the reference is typeset.

Except for the empty value, only languages corresponding to an existing xcref language module can be used as ⟨*language*⟩. Language modules are stored in files named `xcref-`⟨*language*⟩`.tex` next to `xcref.sty` in `$TEXMF/tex/latex/xcref`. Their source code can be found by looking for markers such as `%<*french-module>` and `%<*ngerman-module>` in `xcref.dtx`. When xcref is loaded, the `seq` variable `\g_xcref_available_language_modules_seq` contains all available language modules, and `\g_xcref_loaded_language_modules_seq` contains all those that have been loaded so far (see l3seq in `interface3.pdf`).

`/xcref/functions for preventing auto case change=`⟨*macros*⟩
(no default, initially `\xcrefNoCaseChange`)

List of macros that prevent automatic case change on particular pieces of text when references are capitalized (see `/xcref/capitalize`). ⟨*macros*⟩ should be one or more control sequence tokens. Each of these tokens should be the name of a macro that accepts one argument. When `\tl_mixed_case:nn` is called to capitalize a reference, each such macro present in the reference text protects its argument from changes by `\tl_mixed_case:nn`.

Note that `\tl_mixed_case:nn` doesn't expand the macros listed in ⟨*macros*⟩; it only uses them to detect which parts of a reference text mustn't be modified. The macro calls will be left as is in the reference text; thus, a reasonable behavior for a macro listed in ⟨*macros*⟩ is to simply expand to its only argument. This is precisely what `\xcrefNoCaseChange` does, which is defined to behave exactly like `\use:n`. That said, you are free to define and use macros that do something else with their argument: you might want them to typeset it in a special font, use a particular color, etc.

`/xcref/lang for capitalization func=`⟨*macro*⟩
(no default, initially `\xcrefDefaultMappingForTlMixedCase`)

Change the function used by xcref to map language module names to language codes suitable for use with `\tl_mixed_case:nn`. The function in question is called

when producing a capitalized reference. See the documentation of `\xcref_lang_-for_tl_mixed_case:n` for the requirements that apply to such a function.

3.2.2 `french` langage module

`/xcref/french/form`=⟨*form*⟩ (no default, initially `prep+article+noun`)

Specifies which grammatical form to use for references. The available choices are `noun`, `article+noun` and `prep+article+noun`, where `prep` stands for "preposition".

`/xcref/french/preposition`=⟨*prep*⟩ (no default, initially empty)

Preposition to use in references. When non-empty, ⟨*prep*⟩ overrides the default preposition associated to the reference type.

`/xcref/french/names table`=⟨*data*⟩ (no default)

Define how to write each reference type in French. ⟨*data*⟩ must be a comma-separated list of (key, value) pairs, where each pair has the form ⟨*reftype*⟩=⟨*value*⟩. Each ⟨*reftype*⟩ must be a reference type and the corresponding ⟨*value*⟩ should consist in four items (see below), the first two of which must be in LICR form.[1] In each ⟨*value*⟩, the first item is the *default preposition* for reference type ⟨*reftype*⟩. Here is an example where, for instance, the default preposition for reference type `assertion` is defined to be `d'apr\'es` (which is typeset as "d'après"):

```
\xcrefset{
  french/.cd,
  names table = {
    problem     = {\`a}{le}{problème}{problèmes},
    proposition = {\`a}{la}{proposition}{propositions},
    assertion   = {d'apr\'es}{l'}{assertion}{assertions},
    theorem     = {\`a}{le}{théorème}{théorèmes},
    lemma       = {\`a}{le}{lemme}{lemmes},
    definition  = {dans}{la}{définition}{définitions},
  }
}
```

Note: xcref knows that in French, `\`a` + `le` results in "au" in singular form and in "aux" in plural form, that the plural form of definite article `l'` is `les`, etc.

`/xcref/french/composition table for prepositions and articles`=⟨*data*⟩ (no default)

Define how a given preposition combines with the definite articles `l'`, `le` , `la` and `les`. ⟨*data*⟩ must be a comma-separated list of (key, value) pairs, where each pair has the form ⟨*prep*⟩=⟨*value*⟩. Each ⟨*prep*⟩ must be a preposition in LICR form (see above) and the corresponding ⟨*value*⟩ should consist in four items. These items respectively indicate how to combine preposition ⟨*prep*⟩ with the definite articles `l'`, `le` , `la` and `les`. Example:

[1] This implies that in the first two items of each ⟨*value*⟩, you should use syntax such as `\'e` or `\'e` instead of `é` or `è`.

```
\xcrefset{
  french/.cd,
  composition table for prepositions and articles = {
    \‘a    = {à l’}{au }{à la }{aux },
    de     = {de l’}{du }{de la }{des },
  }
}
```

3.2.3 ngerman langage module

/xcref/ngerman/form=⟨*form*⟩ (no default, initially `prep+article+noun`)

Specifies which grammatical form to use for references. The available choices are `noun`, `article+noun`, `prep+noun` and `prep+article+noun`, where `prep` stands for “preposition”.

/xcref/ngerman/preposition=⟨*prep*⟩ (no default, initially empty)

Preposition to use in references. When non-empty, ⟨*prep*⟩ overrides the default preposition associated to the reference type.

/xcref/ngerman/case=⟨*case*⟩ (no default, no initial value)

Specify which case to use among nominative, accusative, dative and genitive. The corresponding values for ⟨*case*⟩ are respectively `nom`, `acc`, `dat` and `gen`. Since there is no initial value, you must set this key one way or another when typesetting references with xcref’s `ngerman` module. An easy way to do that is to decide on a default case for some part or all of your document (say, dative) and declare it like this (`\xcrefset` respects TEX’s grouping rules):

```
\xcrefset{ngerman/case=dat}
```

/xcref/ngerman/names table=⟨*data*⟩ (no default)

Define how to write each reference type in German. ⟨*data*⟩ must be a comma-separated list of (key, value) pairs, where each pair has the form ⟨`reftype`⟩=⟨`value`⟩. Each ⟨*reftype*⟩ must be a reference type and the corresponding ⟨*value*⟩ should consist in five items. More precisely, each ⟨*value*⟩ should be of the form:

`{`⟨*default-prep*⟩`}{`⟨*nominative*⟩`}{`⟨*accusative*⟩`}{`⟨*dative*⟩`}{`⟨*genitive*⟩`}`

where ⟨*default-prep*⟩ defines the default preposition[2] for reference type ⟨*reftype*⟩ and each of the ⟨*nominative*⟩, ⟨*accusative*⟩, ⟨*dative*⟩ and ⟨*genitive*⟩ placeholders should be of the form `{`⟨*article-sing*⟩`}{`⟨*noun-sing*⟩`}{`⟨*article-plur*⟩`}{`⟨*noun-plur*⟩`}`, where the four metasyntactic variables represent the definite article and noun to use for reference type ⟨*reftype*⟩ in singular and plural forms, respectively. Like the default preposition ⟨*default-prep*⟩, each of the eight definite articles must be in LICR form. Here is an example where, for instance, the default preposition for reference type `proposition` is defined to be `nach`:

```
\xcrefset{
  ngerman/.cd,
```

[2] In LICR form: for instance, `f\"ur`.

```
  names table = {
    problem    = {in}{{das}{Problem}{die}{Probleme}}%
                   {{das}{Problem}{die}{Probleme}}%
                   {{dem}{Problem}{den}{Problemen}}%
                   {{des}{Problems}{der}{Probleme}},
    proposition= {nach}{{die}{Proposition}{die}{Propositionen}}%
                   {{die}{Proposition}{die}{Propositionen}}%
                   {{der}{Proposition}{den}{Propositionen}}%
                   {{der}{Proposition}{der}{Propositionen}},
    satz       = {nach}{{der}{Satz}{die}{Sätze}}%
                   {{den}{Satz}{die}{Sätze}}%
                   {{dem}{Satz}{den}{Sätzen}}%
                   {{des}{Satzes}{der}{Sätze}},
  }
}
```

Note: xcref knows that in German, `in` + `dem` results in `im`, etc. (if something like this has been forgotten by the package author, it just needs to be added to `\l_xcref_ngerman_article_and_prep_prop`).

There is an *initial* value, but I'm not sure it is a great idea to set it in stone now by documenting it here.

/xcref/ngerman/prepositions always followed by the same case=⟨*data*⟩
(no default)

⟨*data*⟩ must be a comma-separated list of (key, value) pairs, where each pair has the form ⟨*prep*⟩=⟨*case*⟩. Each such pair declares that preposition ⟨*prep*⟩ is always followed by ⟨*case*⟩ (this overrides /xcref/ngerman/case). Each ⟨*prep*⟩ must be given in LICR form and each ⟨*case*⟩ must be one of `nominative`, `accusative`, `dative` and `genitive`.

There is an *initial* value, but I'm not sure it is a great idea to set it in stone now by documenting it here.

/xcref/ngerman/composition table for prepositions and articles=⟨*data*⟩
(no default)

Define how a given preposition combines with the definite article, in the cases where simple concatenation with a space doesn't give the correct result. For instance, `in dem = im` must be declared here if one wants to have this contraction instead of `in dem`; on the other hand, since `in der` can be obtained by simple concatenation, it doesn't need to (and should not) be specified here.

⟨*data*⟩ must be a comma-separated list of (key, value) pairs, where each pair has the form ⟨*key*⟩=⟨*value*⟩. Each ⟨*key*⟩ must be written in LICR form.[3] As an example, the initial value of /xcref/ngerman/composition table for prepositions and articles at the time of this writing is equivalent to the following setting:

```
\xcrefset{
  ngerman/.cd,
  composition table for prepositions and articles = {
```

[3] For instance, should you want to use the preposition "für" inside a ⟨*key*⟩, it would have to be written as `f\"ur`, not `für`.

```
    an dem = am,
    in dem = im,
    von dem = vom,
    zu dem = zum,
    zu der = zur,
    bei dem = beim,
  }
}
```

While this is currently the initial value, it may be unwise to expect it to always remain the same—I wrote this initial setting and am not a native German speaker, thus the initial setting might receive improvements in the future. If you want to be preserved from future changes, specify the desired value yourself in your documents (possibly in a personal style file) using an `\xcrefset` call as shown above.

3.3 Generic macros

`\xcrefusemodules` `\xcrefusemodules{⟨language$_1$⟩,...,⟨language$_n$⟩}`
`\xcref_use_modules:n` `\xcref_use_modules:n {⟨language$_1$⟩,...,⟨language$_n$⟩}`
`\xcref_use_module:n` `\xcref_use_module:n {⟨language⟩}`
`\xcref_use_module:V`

Add the specified languages to the list of requested language-specific modules. Example values for the language metasyntactic variables are `french` and `ngerman`. These commands can only be used in the preamble.

`\xcref` `\xcref[⟨options⟩]{⟨comma-separated list of references⟩}`
`\xcref:nn` `\xcref:nn {⟨options⟩} {⟨comma-separated list of references⟩}`

xcref's equivalent of `\cref`, `\Cref` and starred forms of these.

The ⟨*options*⟩ are processed with pgfkeys with a default path of `/xcref`. The references are passed as is to `\cref` or `\Cref`, depending on the value of `/xcref/capitalize` (a starred form is called if the `/xcref/hyperlinks` option is `false` and hyperref is loaded).

Spaces around unbraced commas and equal signs are ignored in ⟨*options*⟩ because this argument is processed by pgfkeys, but they are *not* ignored in ⟨*comma-separated list of references*⟩ (this is the behavior of cleveref).

`\xcrefset` `\xcrefset{⟨key-value assignments⟩}`
`\xcref_set:n` `\xcref_set:n {⟨key-value assignments⟩}`

Set PGF keys related to xcref. For now, these keys only influence `\xcref`. The ⟨*key-value assignments*⟩ are comma-separated assignments such as `ngerman/case=acc`. These commands are analogous to `\tikzset` from the Ti*k*Z package, `\forestset` from forest, `\pgfplotsset` from pgfplots, etc. For instance, the command

```
\xcrefset{ngerman/case=acc}
```

is equivalent to

```
\pgfkeys{/xcref/ngerman/case=acc}
```

and has the effect of calling the `/xcref/ngerman/case` PGF key with the value `acc`.

\xcrefNoCaseChange `\xcrefNoCaseChange{⟨text⟩}`

Locally inhibit conversion in the second argument of `\tl_mixed_case:nn`. When references are automatically captitalized, `\tl_mixed_case:nn` is called and can cause various, *a priori* non-first characters in the texts eventually passed to `\Crefname`, to be changed to lowercase. In some cases, this is undesirable (for instance, in German, nouns always start with a capital letter; converting a noun to lowercase would be a mistake). For such situations, `\xcrefNoCaseChange`, or any user-specified macro passed to `/xcref/functions for preventing auto case change`, can be used in the texts that are destined to be passed to `\Crefname`: it "protects" its argument against any action from `\tl_mixed_case:nn`. For example, `xcref-ngeman.tex` uses this technique to protect everything after the first word of a capitalized reference.

The macro call isn't removed by `\tl_mixed_case:nn`, therefore if you want such protection macros that do "funny things" (contrary to `\xcrefNoCaseChange` which returns its argument unchanged), it is possible by passing your own macro names to `/xcref/functions for preventing auto case change`.

\xcref@tabacckludge `\xcref@tabacckludge⟨accent⟩⟨character token⟩`

Variant of `\@tabacckludge` that doesn't expand inside `\edef`, `\tl_mixed_case:nn`, etc. (it is `\protected`). Apart from this aspect, it works like `\@tabacckludge` and avoids problems due to the fact that the `tabbing` environment redefines `\'`, `\‘` and `\=` (see the inputenc documentation).

This command is useful in `xcref.sty` and its language-specific modules, but in documents, users can use inputenc and for instance accented characters encoded in UTF-8 instead; this works as well and is usually more readable (the reason why we need `\xcref@tabacckludge` here is that we refrain from loading inputenc ourselves).

3.4 Language-specific convenience wrappers

3.4.1 `french` langage module

\xcreffrenchgenericwrapper `\xcreffrenchgenericwrapper {⟨generic options⟩} {⟨language-specific options⟩}`

XXX documentation needed

\xcreffrenchwrapper `\xcreffrenchwrapper {⟨language-specific options⟩}`
\xcreffrenchWrapper `\xcreffrenchWrapper {⟨language-specific options⟩}`

XXX documentation needed

3.4.2 `ngerman` langage module

\xcrefngermangenericwrapper `\xcrefngermangenericwrapper {⟨generic options⟩} {⟨lang-specific options⟩}`

XXX documentation needed

\xcrefngermanwrapper `\xcrefngermanwrapper {⟨language-specific options⟩}`
\xcrefngermanWrapper `\xcrefngermanWrapper {⟨language-specific options⟩}`

XXX documentation needed

3.5 Defining how references are to be printed

The commands in this section can be seen as analogous to `\crefname` and `\Crefname`; their arguments are usually a bit more complex than those of `\crefname` and `\Crefname`, though, since they can do more subtle language-specific work than what the cleveref commands allow out of the box. Another important difference with the cleveref commands is that the commands presented in this section can only be used in the preamble.

3.5.1 `french` langage module

`\xcreffrenchSetNamesTableEntry` `\xcreffrenchSetNamesTableEntry {⟨reftype⟩} {⟨prep⟩} {⟨article⟩} {⟨singular⟩} {⟨plural⟩}`

Define how to print references of type ⟨*reftype*⟩. This function associates four token lists to reference type ⟨*reftype*⟩, which could be something such as `section`, `chapter` or `theorem`:

- a default preposition in LICR form, for instance `d'apr\‘es`;
- a definite article for the singular form, among `l'`, `le` and `la`;
- the singular form of the corresponding noun (e.g., `théorème`);
- the plural form of the same noun (e.g., `théorèmes`).

This function can only be used in the preamble and overrides any previous setting for the same reference type.

3.5.2 `ngerman` langage module

`\xcrefngermanSetNamesTableEntry` `\xcrefngermanSetNamesTableEntry {⟨reftype⟩} {⟨prep⟩} {⟨nominative⟩} {⟨accusative⟩} {⟨dative⟩} {⟨genitive⟩}`

Define how to print references of type ⟨*reftype*⟩. This function associates five token lists to reference type ⟨*reftype*⟩, which could be something such as `section`, `chapter` or `theorem`:

- a default preposition in LICR form, for instance `f\"ur`;
- ⟨*nominative*⟩, ⟨*accusative*⟩, ⟨*dative*⟩ and ⟨*genitive*⟩ forms as specified below.

Each of the ⟨*nominative*⟩, ⟨*accusative*⟩, ⟨*dative*⟩ and ⟨*genitive*⟩ arguments should be of the form `{`⟨*article-sing*⟩`}{`⟨*noun-sing*⟩`}{`⟨*article-plur*⟩`}{`⟨*noun-plur*⟩`}`, where the four metasyntactic variables represent the definite article and noun to use for reference type ⟨*reftype*⟩ in singular and plural forms, respectively.

This function can only be used in the preamble and overrides any previous setting for the same reference type.

4 Code-level functions

4.1 Generic

`\xcref_lang_for_tl_mixed_case:n`	`\xcref_lang_for_tl_mixed_case:n {⟨language⟩}`
`\xcref_lang_for_tl_mixed_case:V`	`\xcref_lang_for_tl_mixed_case_default:n {⟨language⟩}`
`\xcref_lang_for_tl_mixed_case_default:n`	`\xcrefDefaultMappingForTlMixedCase {⟨language⟩}`
`\xcrefDefaultMappingForTlMixedCase`	

When references have to be capitalized (cf. /xcref/capitalize), the expl3 function `\tl_mixed_case:nn` is used. This function accepts a "language code" argument to adapt to each language, because this process can vary depending on the language (for instance, it appears that in Dutch, capitalisation of `ij` at the beginning of mixed cased text produces `IJ` rather than `Ij`). However, the language codes used by `\tl_mixed_case:nn` don't coincide with xcref's language-specific module names—which currently are identical to babel language names. Therefore, xcref uses a function, namely `\xcref_lang_for_tl_mixed_case:n`, to map xcref language module names to suitable language codes for use with `\tl_mixed_case:nn`. By default, this function is `\let`-equal to `\xcref_lang_for_tl_mixed_case_default:n`, or `\xcrefDefaultMappingForTlMixedCase` in the LaTeX 2_ε naming style. It accepts one argument which must be an xcref language module name (e.g., `ngerman`) and expands (as in `\edef`) to a language code suitable for use with `\tl_mixed_case:nn`.

Should the mapping provided by `\xcref_lang_for_tl_mixed_case_default:n` be incomplete, you should probably contact the package maintainer and suggest an update. However, it is also possible to change the meaning of `\xcref_lang_for_tl_mixed_case:n` (*i.e.,* redefine it) in order to use the mapping function of your choice. For this, simply use the /xcref/lang for capitalization func option to point to the expandable macro of your choice.

4.2 Language-specific

4.2.1 From the `french` module

`\xcref_french_set_names_table_entry:nnnnn`	`\xcref_french_set_names_table_entry:nnnnn {⟨reftype⟩} {⟨prep⟩} {⟨article⟩} {⟨singular⟩} {⟨plural⟩}`

This is the code-level counterpart of `\xcreffrenchSetNamesTableEntry`.

`\xcref_french_assemble_prep_and_article:nnN`	`\xcref_french_assemble_prep_and_article:nnN {⟨preposition⟩} {⟨article⟩} ⟨tl var⟩`
`\xcref_french_assemble_prep_and_article:(VnN|VVN)`	

Assemble a preposition and a definite article. In some cases, this means combining them into a single word. Any accented characters in ⟨*preposition*⟩ must be in LICR form. ⟨*article*⟩ has to be one of `l'`, `le`, `la` and `les`. The result is stored in ⟨*tl var*⟩; in case it is incorrect, it is likely that more exceptions need to be added to `\g__xcref_french_prep_and_article_prop`.

4.2.2 From the ngerman module

`\xcref_ngerman_set_names_table_entry:nnnnnn` `\xcref_ngerman_set_names_table_entry:nnnnnn {⟨reftype⟩} {⟨prep⟩} {⟨nominative⟩} {⟨accusative⟩} {⟨dative⟩} {⟨genitive⟩}`

This is the code-level counterpart of `\xcrefngermanSetNamesTableEntry`.

`\xcref_ngerman_assemble_prep_and_article:nnN` `\xcref_ngerman_assemble_prep_and_article:VVN` `\xcref_ngerman_assemble_prep_and_article:nnN {⟨preposition⟩} {⟨article⟩} ⟨tl var⟩`

Assemble a preposition and a definite article. In some cases, this means combining them into a single word. In case ⟨*preposition*⟩␣⟨*article*⟩ is a key of `\l__xcref_ngerman_article_and_prep_prop` (with eventual accented characters in LICR form), the result is the corresponding value; otherwise, the result is ⟨*preposition*⟩␣⟨*article*⟩. In any case, it is stored in ⟨*tl var*⟩.

5 Public variables

`\g_xcref_available_language_modules_seq`

List of the available language-specific modules. Example items are `french` and `ngerman`. Each of these is made of character tokens with category code 12 only (other).

`\g_xcref_loaded_language_modules_seq`

List of the language-specific modules that have been loaded. Each element is a babel language name, is not allowed to contain any space and only contains character tokens of category code 12.

6 Development repository

The Git repository containing the xcref source code is located at https://github.com/frougon/xcref.

7 xcref Implementation

7.1 `xcref.sty`

DocStrip start guard for `xcref.sty`.

```
1 ⟨*package⟩
```

Identify the internal prefix (LaTeX3 DocStrip convention).

```
2 ⟨@@=xcref⟩
3 \NeedsTeXFormat{LaTeX2e}[1995/12/01]
4 \RequirePackage{expl3}
5 \RequirePackage{l3keys2e}
6 \RequirePackage{xparse}
7 \RequirePackage{etoolbox}
```

```
\RequirePackage{iflang}
\RequirePackage{pgfkeys}
\RequirePackage{cleveref}
```

xcref is an expl3-based package, declare its metadata.

```
\ProvidesExplPackage{xcref}{2019-09-19}{0.1a}
                    {Extension of cleveref for non-English languages}
```

7.1.1 expl3 messages

Messages emitted by `xcref.sty`:

```
\msg_new:nnn { xcref } { loading-language-module }
  { Loading~language~module~'\exp_not:n {#1}'. }
\msg_new:nnn { xcref } { unable-to-determine-language }
  { Unable~to~determine~the~language. }
```

7.1.2 Variables

\g_xcref_available_language_modules_seq

List of the available language-specific modules. For consistency with `\g_xcref_loaded_language_modules_seq`, each item is obtained with `\tl_to_str:n`.

```
\seq_new:N \g_xcref_available_language_modules_seq

\clist_map_inline:nn { french, ngerman }
  {
    \seq_gput_right:Nx \g_xcref_available_language_modules_seq
      { \tl_to_str:n {#1} }
  }
```

(*End definition for* `\g_xcref_available_language_modules_seq`. *This variable is documented on page* 14.)

\g_xcref_loaded_language_modules_seq

List of the loaded language-specific modules.

```
\seq_new:N \g_xcref_loaded_language_modules_seq
```

(*End definition for* `\g_xcref_loaded_language_modules_seq`. *This variable is documented on page* 14.)

\l__xcref_selected_lang_str

Name of the currently-selected language for xcref. It must be the name of a language-specific module.

```
\str_new:N \l__xcref_selected_lang_str
```

(*End definition for* `\l__xcref_selected_lang_str`.)

\l__xcref_use_hyperlinks_bool

Whether to use hyperlinks in references generated by xcref when hyperref is loaded.

```
\bool_new:N \l__xcref_use_hyperlinks_bool
```

(*End definition for* `\l__xcref_use_hyperlinks_bool`.)

\l__xcref_case_change_exclude_tl

List of commands (control sequence tokens) that indicate contents which mustn't be affected by an automatic case change (when `\tl_mixed_case:nn` is used to automatically capitalize text, before passing it to `\Crefname`). See the documentation of `\xcrefNoCaseChange`.

```
\tl_new:N \l__xcref_case_change_exclude_tl
```

(*End definition for* `\l__xcref_case_change_exclude_tl`.)

\l__xcref_capitalize_bool Whether to write references in capitalized form (first letter in uppercase, the rest in lowercase[4]).

```
\bool_new:N \l__xcref_capitalize_bool
```

(*End definition for* `\l__xcref_capitalize_bool`.)

\g__xcref_hyperref_loaded_bool Whether hyperref is loaded.

```
\bool_new:N \g__xcref_hyperref_loaded_bool
```

(*End definition for* `\g__xcref_hyperref_loaded_bool`.)

7.1.3 Interaction with hyperref

Remember if hyperref is loaded: if so and if the `/xcref/hyperlinks` option is false, we'll have to use the starred form of `\cref` and `\Cref`.

```
\AtBeginDocument
  {
    \@ifpackageloaded { hyperref }
      { \bool_gset_true:N \g__xcref_hyperref_loaded_bool }
      { \bool_gset_false:N \g__xcref_hyperref_loaded_bool }
  }
```

7.1.4 Loading of langage-specific modules

\xcref_use_module:n
\xcref_use_module:V

`#1` : name of a language-specific module

Load a langage-specific module unless it is already loaded. One subtlety here: if a language name such as `french` is passed as a `\documentclass` option, it arrives here in `#1` with character tokens of category code 12. However, the same option passed to the xcref package would cause `#1` to contain character tokens of category code 11. Thus, we use `\tl_to_str:n` on `#1` to normalize this input (the elements of `\g_xcref_loaded_language_modules_seq` are therefore made of character tokens of category code 12, since they can't contain any space).

```
\cs_new_protected:Npn \xcref_use_module:n #1
  {
    \tl_set:Nx \l_tmpa_tl { \tl_to_str:n {#1} }

    \seq_if_in:NVF \g_xcref_loaded_language_modules_seq \l_tmpa_tl
      {
        \__xcref_load_module:n {#1}
        \seq_gput_right:NV \g_xcref_loaded_language_modules_seq \l_tmpa_tl
      }
  }

\@onlypreamble \xcref_use_module:n
\cs_generate_variant:Nn \xcref_use_module:n { V }
```

(*End definition for* `\xcref_use_module:n`. *This function is documented on page 10.*)

[4] Well, this can be subtler than that depending on the language—see the documentation of `\tl_mixed_case:nn`.

\xcref_use_modules:n
\xcrefusemodules

#1 : comma-separated list of language-specific module names

```
\cs_new_protected:Npn \xcref_use_modules:n #1
  {
    \clist_map_inline:nn {#1} { \xcref_use_module:n {##1} }
  }

\NewDocumentCommand \xcrefusemodules { m }
  {
    \xcref_use_modules:n {#1}
  }
```

(*End definition for* `\xcref_use_modules:n` *and* `\xcrefusemodules`. *These functions are documented on page* 10.)

__xcref_load_module:n

`\input` the specified language-specific module. This is done under the "traditional" category code régime for LaTeX 2_ε packages.

#1 : language name corresponding to the module

```
\cs_new_protected:Npn \__xcref_load_module:n #1
  {
    \makeatletter
    \msg_info:nnn { xcref } { loading-language-module } {#1}
    \input { xcref-#1.tex }
    \makeatother
  }
```

(*End definition for* `\__xcref_load_module:n`.)

7.1.5 Main code

__xcref_set_to_licr_form:Nn

Get the LICR form of a token list representing text.

#1 : token list variable

#2 : token list representing text

Set #1 to the LICR form of #2 (the assignment is local). This is very important when comparing strings using non-ASCII characters, because for instance, `über` is not the same token list in the `latin1` and `utf8` encodings (inputenc side). To overcome this difficulty, when we need to compare pure-text user input that could contain non-ASCII characters to keys (e.g., of a mapping as implemented in l3prop), we use the LICR as a normalized form.

```
\cs_new_protected:Npn \__xcref_set_to_licr_form:Nn #1#2
  {
    \group_begin:
```

Remove the `\IeC` inputenc macro (only used to protect control words written to files such as `.aux` and `.toc`) in order to retrieve the LICR object corresponding to #2.

```
    \cs_set_eq:NN \IeC \@firstofone % see the inputenc documentation (or code)
    \protected@xdef \xcref@tmpa {#2}
    \group_end:
```

Not using `\tl_set_eq:NN` here, because that would mean relying on implementation details of token list variables.

```
    \tl_set:No #1 { \xcref@tmpa }
  }
```

(*End definition for* `\__xcref_set_to_licr_form:Nn`.)

\xcref@tabaccкludge

\protected variant of \@tabaccкludge. We need this for text that is going to be transformed by \tl_mixed_case:nn.

```
\cs_set_eq:NN \xcref@tabaccкludge \@tabaccкludge
\robustify { \xcref@tabaccкludge }
```

(*End definition for* \xcref@tabaccкludge. *This function is documented on page* 11.)

__xcref_protect_accent_macros_from_tabbing:N

Replace \', \` and \= for safe use inside tabbing.

#1 : token list variable

Replace \', \` and \= in #1 with \xcref@tabaccкludge followed by ', ` or =, respectively. This is necessary in case the replaced commands were meant to produce accented characters and the material in #1 is going to occur inside a tabbing environment. Indeed, tabbing redefines \', \` and \= to do completely different things from the corresponding TeX commands (see the inputenc documentation and the LaTeX book).

```
\cs_new_protected:Npn \__xcref_protect_accent_macros_from_tabbing:N #1
  {
    \tl_replace_all:Nnn #1 { \' } { \xcref@tabaccкludge ' }
    \tl_replace_all:Nnn #1 { \` } { \xcref@tabaccкludge ` }
    \tl_replace_all:Nnn #1 { \= } { \xcref@tabaccкludge = }
  }
```

(*End definition for* __xcref_protect_accent_macros_from_tabbing:N.)

__xcref_call_lang_setup_func:nN
__xcref_call_lang_setup_func:VN

Call the main function of a language-specific module.

#1 : name of a language-specific module

#2 : sequence variable containing the reference types to configure

```
\cs_new_protected:Npn \__xcref_call_lang_setup_func:nN #1#2
  {
    \use:c { __xcref_#1_setup_cref_names:N } #2
  }
```

Define a variant of this function:

```
\cs_generate_variant:Nn \__xcref_call_lang_setup_func:nN { V }
```

(*End definition for* __xcref_call_lang_setup_func:nN.)

__xcref_autodetect_lang_unless_explicit:

Use the current language if not explicitly selected for xcref. May be used to map several babel languages to the same xcref language-specific (backend) module (note: the mapping is only used when the language is autodetected; explicit selection with xcref doesn't use it).

```
\cs_new_protected:Npn \__xcref_autodetect_lang_unless_explicit:
  {
    \str_if_empty:NT \l__xcref_selected_lang_str
      {
        \str_set:Nx \l__xcref_selected_lang_str
          {
            \IfLanguageName { english } { english } { }
            \IfLanguageName { french } { french } { }
            \IfLanguageName { ngerman } { ngerman } { }
          }
        \str_if_empty:NT \l__xcref_selected_lang_str
          { \msg_error:nn { xcref } { unable-to-determine-language } }
      }
  }
```

(*End definition for* `\__xcref_autodetect_lang_unless_explicit:`.)

`\l__xcref_xcref_reference_types_seq` Types of all references in the second argument of `\xcref:nn`. Duplicates are not added; thus, the number of elements stored in this variable is less than or equal to the number of references in the second argument of `\xcref:nn`.

```
\seq_new:N \l__xcref_xcref_reference_types_seq
```

(*End definition for* `\l__xcref_xcref_reference_types_seq`.)

`\xcref:nn` xcref's wrapper for `\cref`, `\Cref`, `\cref*` and `\Cref*`.
`\xcref` `#1` : options processed with pgfkeys using a default path of `/xcref`
`#2` : comma-separated list of references passed as is to `\cref`, `\Cref`, `\cref*` or `\Cref*` depending on the values of the `/xcref/capitalize` and `/xcref/hyperlinks` options

Basically, this first calls `\crefname`, `\Crefname` and `\@crefdefineallformats` for each type pertaining to a reference in `#2`. Then the appropriate `\cref`-like command is called. All this is done inside a group (`\crefname` and friends act locally).

```
\cs_new_protected:Npn \xcref:nn #1#2
  {
    \group_begin:
    \xcref_set:n {#1}
    \__xcref_autodetect_lang_unless_explicit:

    \__xcref_set_to_reference_types:Nn \l__xcref_xcref_reference_types_seq {#2}
    \__xcref_call_lang_setup_func:VN \l__xcref_selected_lang_str
                                     \l__xcref_xcref_reference_types_seq

    \__xcref_call_cref:Nxn \l__xcref_capitalize_bool
      {
        \bool_lazy_and:nnT { \g__xcref_hyperref_loaded_bool }
                           { ! \l__xcref_use_hyperlinks_bool }
                           { * }
      }
      {#2}
    \group_end:
  }

\NewDocumentCommand \xcref { O{} m }
  {
    \xcref:nn {#1} {#2}
  }
```

(*End definition for* `\xcref:nn` *and* `\xcref`. *These functions are documented on page* 10.)

`\__xcref_call_cref:Nnn` Call `\cref`, `\Cref` or their starred forms.
`\__xcref_call_cref:Nxn` `#1` : boolean variable indicating whether to capitalize
`#2` : star or empty
`#3` : list of references in the format supported by `\cref` and friends (spaces around the comma separators are *not* ignored)

```
\cs_new_protected:Npn \__xcref_call_cref:Nnn #1#2#3
  {
    \use:c { \bool_if:NTF #1 { C } { c } ref } #2 {#3}
  }
```

```
\cs_generate_variant:Nn \__xcref_call_cref:Nnn { Nx }
```

(*End definition for* `\__xcref_call_cref:Nnn`.)

`\__xcref_call_crefname:nnn` Simple expl3 wrapper for `\crefname`.
`\__xcref_call_crefname:nVV` `#1` : reference type (e.g., `chapter`, `section`, `theorem`, `lemma`, etc.)
`#2` : singular form
`#3` : plural form

```
\cs_new_protected:Npn \__xcref_call_crefname:nnn #1#2#3
  {
    \crefname {#1} {#2} {#3}
  }

\cs_generate_variant:Nn \__xcref_call_crefname:nnn { nVV }
```

(*End definition for* `\__xcref_call_crefname:nnn`.)

`\__xcref_call_Crefname:nnn` Simple expl3 wrapper for `\Crefname`.
`\__xcref_call_Crefname:nVV` `#1` : reference type (see `\xcref_call_crefname:nnn` for examples)
`#2` : singular form
`#3` : plural form

```
\cs_new_protected:Npn \__xcref_call_Crefname:nnn #1#2#3
  {
    \Crefname {#1} {#2} {#3}
  }

\cs_generate_variant:Nn \__xcref_call_Crefname:nnn { nVV }
```

(*End definition for* `\__xcref_call_Crefname:nnn`.)

`\__xcref_call_crefdefineallformats:n` Call `\@crefdefineallformats` for the specified type.
`#1` : reference type

```
\cs_new_protected:Npn \__xcref_call_crefdefineallformats:n #1
  {
    \@crefdefineallformats {#1}
  }
```

(*End definition for* `\__xcref_call_crefdefineallformats:n`.)

`\__xcref_ref_defined_p:n` Test whether the specified reference is defined for cleveref.
`\__xcref_ref_defined:nTF` `#1` : reference (*i.e.*, a label)

```
\prg_new_conditional:Npnn \__xcref_ref_defined:n #1 { p, T, F, TF }
  {
    \exp_after:wN \if_meaning:w \cs:w r@#1@cref \cs_end: \relax
      \prg_return_false:
    \else:
      \prg_return_true:
    \fi:
  }
```

(*End definition for* `\__xcref_ref_defined:nTF`.)

`\__xcref_call_cref_gettype:nN` Use `\cref@gettype` to retrieve the type of a reference.

#1 : reference (*i.e.*, a label)
#2 : token list variable (receives the result)
This can only be used for references that have already been defined by `\newlabel` (*i.e.*, they have been read from the `.aux` file: `\r@`⟨*reference*⟩`@cref` is not `\let`-equal to `\relax`).

```
\cs_new_protected:Npn \__xcref_call_cref_gettype:nN #1#2
  {
    \cref@gettype {#1} {#2}
  }
```

(*End definition for* `\__xcref_call_cref_gettype:nN`.)

`\__xcref_set_to_reference_types:Nn` Extract the list of reference types from a comma list of references.
#1 : sequence variable
#2 : comma list of references
Store in #1 the types of all references listed in #2, eliminating duplicates. #1 is assigned locally.

```
\cs_new_protected:Npn \__xcref_set_to_reference_types:Nn #1#2
  {
    \seq_clear:N #1
    \clist_map_inline:nn {#2}
      {
        \__xcref_ref_defined:nT {##1}
          {
            \__xcref_call_cref_gettype:nN {##1} \l_tmpa_tl
            \seq_if_in:NVF #1 \l_tmpa_tl
              { \seq_put_right:NV #1 \l_tmpa_tl }
          }
      }
  }
```

(*End definition for* `\__xcref_set_to_reference_types:Nn`.)

`\xcref_set:n` `\xcrefset` Set PGF keys related to xcref.
#1 : comma-separated list of PGF keys

```
\cs_new_protected:Npn \xcref_set:n #1
  {
    \pgfqkeys { /xcref } {#1}
  }

\NewDocumentCommand \xcrefset { m }
  {
    \xcref_set:n {#1}
  }
```

(*End definition for* `\xcref_set:n` *and* `\xcrefset`. *These functions are documented on page* 10.)

7.1.6 Helper macros used with `\tl_mixed_case:nn`

`\xcrefNoCaseChange` Locally inhibit conversion when used in the second argument of `\tl_mixed_case:nn`.
#1 : a token list (*a priori* representing text) to "protect" from changes by `\tl_mixed_case:nn`

This function (the control sequence token) can be passed to /xcref/functions for preventing auto case change in order to tell the underlying \tl_mixed_case:nn operation not to modify some particular tokens (of a reference name for instance). We set it to the identity function.

```
\cs_set_eq:NN \xcrefNoCaseChange \use:n
```

(*End definition for* \xcrefNoCaseChange. *This function is documented on page 11.*)

\xcref_lang_for_tl_mixed_case:n
\xcref_lang_for_tl_mixed_case:V
\xcref_lang_for_tl_mixed_case_default:n
\xcrefDefaultMappingForTlMixedCase

\xcref_lang_for_tl_mixed_case_default:n is the default mapping from xcref language names to language codes suitable for use with \tl_mixed_case:nn. Users may use /xcref/lang for capitalization func in order to point to a different mapping function in case \xcref_lang_for_tl_mixed_case_default:n doesn't do what they want. Calling this key simply redefines \xcref_lang_for_tl_mixed_case:n to point to the specified macro. At any time, the current mapping is given by \xcref_lang_for_tl_mixed_case:n. \xcref_lang_for_tl_mixed_case_default:n is the initial ("default") mapping function. \xcrefDefaultMappingForTlMixedCase is just another name for this function that complies with the LaTeX 2_ε naming conventions.

#1 : an xcref language name

```
\cs_new:Npn \xcref_lang_for_tl_mixed_case_default:n #1
  {
    \str_case:nnF {#1}
      {
        { azerbaijani } { az }
        { dutch } { nl }
        { lithuanian } { lt }
        { ngerman } { de-alt }
        { turkish } { tr }
      }
      {#1}                      % pass through for all other cases
  }

\cs_new_eq:NN \xcref_lang_for_tl_mixed_case:n
              \xcref_lang_for_tl_mixed_case_default:n
\cs_new_eq:NN \xcrefDefaultMappingForTlMixedCase
              \xcref_lang_for_tl_mixed_case_default:n

\cs_generate_variant:Nn \xcref_lang_for_tl_mixed_case:n { V }
```

(*End definition for* \xcref_lang_for_tl_mixed_case:n, \xcref_lang_for_tl_mixed_case_default:n, *and* \xcrefDefaultMappingForTlMixedCase. *These functions are documented on page 13.*)

7.1.7 Factory for convenience \xcref front-ends

__xcref_create_language_specific_wrapper:Nn

Create convenience front-end macros to \xcref for the specified language. See the language-specific modules for examples of use.

#1 : name of the language-specific module used by the created macros—as specified by the following arguments
#2 : name of the generic language-specific code-level macro to create
#3 : name of its document-level counterpart (it can be useful to users who wish to create custom front-ends \xcref)
#4 : name of the language-specific document-level front-end macro that uses capitalize=false
#5 : name of the language-specific document-level front-end macro that uses capitalize=true

```
\cs_new_protected:Npn \__xcref_create_language_specific_wrapper:nNNNN #1#2#3#4#5
  {
    \cs_new_protected:Npn #2 ##1##2
      {
        \xcref:nn { ##1, lang=#1, #1/.cd, ##2 }
      }

    \NewDocumentCommand #3 { m m }
      {
        #2 {##1} {##2}
      }

    \NewDocumentCommand #4 { O{} }
      {
        #2 {capitalize=false} {##1}
      }

    \NewDocumentCommand #5 { O{} }
      {
        #2 {capitalize=true} {##1}
      }
  }
```

(*End definition for* `\__xcref_create_language_specific_wrapper:Nn`.)

7.1.8 PGF key definitions

The following PGF keys can be set with `\xcrefset` as will as in the first argument of `\xcref:nn`.

```
\xcref_set:n
  {
    capitalize/.is~choice,
    capitalize/true/.code = { \bool_set_true:N \l__xcref_capitalize_bool },
    capitalize/false/.code = { \bool_set_false:N \l__xcref_capitalize_bool },
    capitalize/.default = true, % when the key is used with no value
    capitalize = false,         % initial value
    %
    hyperlinks/.is~choice,
    hyperlinks/true/.code = { \bool_set_true:N \l__xcref_use_hyperlinks_bool },
    hyperlinks/false/.code = { \bool_set_false:N \l__xcref_use_hyperlinks_bool },
    hyperlinks/.default = true, % when the key is used with no value
    hyperlinks = true,          % initial value
```

Used with `\tl_mixed_case:nn` which prepares the arguments of `\Crefname`

```
    functions~for~preventing~auto~case~change/.value~required,
    functions~for~preventing~auto~case~change/.code =
      { \tl_set:Nn \l__xcref_case_change_exclude_tl {#1} },
```

The initial setting only contains one macro:

```
    functions~for~preventing~auto~case~change = \xcrefNoCaseChange,
```

The language to use will be autodetected with iflang, unless it is explicitly specified using this option.

```
    lang/.value~required,
    lang/.code = { \str_set:Nn \l__xcref_selected_lang_str {#1} },
    lang = {},                  % initial value: autodetect
```

Function mapping xcref language module names to language codes used by `\tl_mixed_case:nn`.

```
    lang~for~capitalization~func/.value~required,
    lang~for~capitalization~func/.code = {
      \cs_set_eq:NN \xcref_lang_for_tl_mixed_case:n #1 },
```

It would be nice to be able to use `.unknown/.code` so that unrecognized options `foo` are interpreted as being `preposition=foo`, however this is not satisfactory because `\pgfkeys` appears to expand tokens when looking for key names. There is no such problem with l3keys, but pgfkeys has other advantages...

```
  }
```

7.1.9 Package options

Define the package options using l3keys.

```
\keys_define:nn { xcref }
  {
    languages .code:n = \xcref_use_modules:n {#1},
  }
```

Explicitly define the name of each supported language module as a package option. This way, if such an option (e.g., `ngerman`) was given to `\documentclass`, `\ProcessKeysOptions` will pass it to xcref.

```
\seq_map_inline:Nn \g_xcref_available_language_modules_seq
  {
    \keys_define:nn { xcref }
      { #1 .code:n = \xcref_use_module:V \l_keys_key_tl }
  }
```

Process LaTeX 2ε-style package options. This uses l3keys2e, see https://tex.stackexchange.com/a/371754/73317 for explanations.

```
\ProcessKeysOptions { xcref }
```

This ends the code for `xcref.sty`.

```
⟨/package⟩
```

7.2 Module `xcref-french.tex`

DocStrip start guard for `xcref-french.tex`.

```
⟨*french-module⟩
```

Switch to the category code régime suitable for expl3 code.

```
\ExplSyntaxOn
```

7.2.1 expl3 messages

```
\msg_new:nnn { xcref } { french-requested-use-of-unknown-definite-article }
  { (french)~Unknown~definite~article:~'\exp_not:n {#1}'. }
\msg_new:nnn { xcref }
  { french-definite-article-not-in-c__xcref_french_article_prop }
  { (french)~Bug:~definite~article~not~in~\token_to_str:N
    \c__xcref_french_article_prop \c_colon_str \space '\exp_not:n {#1}'. }
\msg_new:nnn { xcref } { french-requested-use-of-undefined-type }
  { (french)~Requested~use~of~undefined~reference~type:~'\exp_not:n {#1}'. }
```

```
\msg_new:nnn { xcref } { french-unknown-prefix-form }
  { (french)~Unknown~form~for~the~part~preceding~the~
    \token_to_str:N \namecref \c_colon_str \space '\exp_not:n {#1}'. }
```

7.2.2 Variables

\l__xcref_french_selected_form_str

Specifies how references are to be produced, in grammatical terms. There are three possible values: `noun`, `article+noun` and `prep+article+noun`, where `prep` stands for "preposition".

```
\str_new:N \l__xcref_french_selected_form_str
```

(*End definition for* `\l__xcref_french_selected_form_str`.)

\l__xcref_french_selected_preposition_tl

The selected preposition, if any. When non-empty, this variable overrides the default preposition associated to the reference type.

```
\tl_new:N \l__xcref_french_selected_preposition_tl
```

(*End definition for* `\l__xcref_french_selected_preposition_tl`.)

\g__xcref_french_names_prop

French language-specific data on reference names: for each name, the default preposition, definite article to use, singular and plural forms. The first two ⟨*items*⟩ of each value of the property list must use the LICR representation.

We use `\xcref@tabacckludge` in *output text* instead of `\@tabacckludge` because things can go wrong when `\tl_mixed_case:nn` expands `\@tabacckludge` in its second argument (try with `\@tabacckludge‘a`). Regarding the use of `\@tabacckludge` itself, see the documentation of inputenc. Note that it is not really necessary here, because we are going to automatically do the corresponding replacements downstream to be on the safe side, but it will help in case someone uses data from this variable and forgets to systematically replace `\’`, `\‘` and `\=` by commands that behave as expected inside a `tabbing` environment.

```
\prop_new:N \g__xcref_french_names_prop

\prop_gset_from_keyval:Nn \g__xcref_french_names_prop
  {
    problem    = {\‘a}{le}{probl\xcref@tabacckludge‘eme}
                 {probl\xcref@tabacckludge‘emes},
    proposition= {\‘a}{la}{proposition}{propositions},
    assertion  = {d’apr\‘es}{l’}{assertion}{assertions},
    theorem    = {\‘a}{le}{th\xcref@tabacckludge’eor\xcref@tabacckludge‘eme}
                 {th\xcref@tabacckludge’eor\xcref@tabacckludge‘emes},
    lemma      = {\‘a}{le}{lemme}{lemmes},
    definition = {dans}{la}{d\xcref@tabacckludge’efinition}
                 {d\xcref@tabacckludge’efinitions},
  }
```

(*End definition for* `\g__xcref_french_names_prop`.)

\g__xcref_french_types_seq

`\g__xcref_french_types_seq` will be set up to contain the list of keys in `\g__xcref_french_names_prop` (this will be done from the `\AtBeginDocument` hook).

```
\seq_new:N \g__xcref_french_types_seq
```

(*End definition for* `\g__xcref_french_types_seq`.)

Change the category code of spaces inside the following group so that they behave as in plain TEX and LATEX 2ε.

```
\group_begin:
  \char_set_catcode_space:n { `~ }
```

`\g__xcref_french_prep_and_article_prop` Mapping describing how prepositions and definite articles combine in French. The keys must use the LICR representation. `\xcref@tabacckludge` is not really necessary here, because we are going to do the corresponding replacements downstream to be on the safe side—but it is safer in case someone uses data from this variable and forgets to systematically replace `\'`, `\`` and `\=` by commands that behave as expected inside a `tabbing` environment.

```
\prop_new:N \g__xcref_french_prep_and_article_prop
\prop_gset_from_keyval:Nn \g__xcref_french_prep_and_article_prop
  {
    \`a    = {\xcref@tabacckludge`a l'}{au }{\xcref@tabacckludge`a la }{aux }
  }
```

(*End definition for* `\g__xcref_french_prep_and_article_prop`.)

`\c__xcref_french_article_prop` This mapping allows one to always get proper spacing between an article and the following noun—which sometimes is "no space" (cf. the case of `l'`).

```
\prop_const_from_keyval:Nn \c__xcref_french_article_prop
  { l'  = {l'}, le  = {le }, la  = {la }, les = {les } }
```

(*End definition for* `\c__xcref_french_article_prop`.)

```
\group_end:
```

7.2.3 PGF keys

Note that the `/xcref/french/names table` key can only be set in the preamble. The initial value of `\l__xcref_french_selected_preposition_tl` is empty (see above).

```
\pgfqkeys { /xcref/french }
  {
    form/.is~choice,
    form/noun/.code = { \str_set:Nn \l__xcref_french_selected_form_str {noun} },
    form/article+noun/.code =
             { \str_set:Nn \l__xcref_french_selected_form_str {article+noun} },
    form/prep+article+noun/.code =
             { \str_set:Nn \l__xcref_french_selected_form_str {prep+article+noun} },
    form = prep+article+noun,   % set the initial value
    %
    preposition/.value~required,
    preposition/.code = {
      \__xcref_set_to_licr_form:Nn \l__xcref_french_selected_preposition_tl {#1} },
    %
    names~table/.code = { \__xcref_french_set_names_table:n {#1} },
    %
    composition~table~for~prepositions~and~articles/.code = {
      \prop_gset_from_keyval:Nn \g__xcref_french_prep_and_article_prop {#1} },
  }
```

7.2.4 Main code

```
\int_new:N \l__xcref_fapaa_index_int
\tl_new:N \l__xcref_fapaa_prepart_tl
```

\xcref_french_assemble_prep_and_article:nnN
\xcref_french_assemble_prep_and_article:VnN
\xcref_french_assemble_prep_and_article:VVN

Assemble a preposition and a definite article.

#1 : preposition (key of `\g__xcref_french_prep_and_article_prop`)
#2 : definite article among `l'`, `le`, `la` and `les`
#3 : token list variable where the result is to be stored

```
\cs_new_protected:Npn \xcref_french_assemble_prep_and_article:nnN #1#2#3
  {
    \prop_get:NnNTF \g__xcref_french_prep_and_article_prop {#1}
                    \l__xcref_fapaa_prepart_tl
      {
        \seq_set_split:NnV \l_tmpa_seq { } \l__xcref_fapaa_prepart_tl
        \int_set:Nn \l__xcref_fapaa_index_int
          {
            \str_case:nnF {#2}
              {
                { l' } { 1 }
                { le } { 2 }
                { la } { 3 }
                { les } { 4 }
              }
              {
                \msg_error:nnn { xcref }
                  { french-requested-use-of-unknown-definite-article } {#2}
              }
          }

        \tl_set:Nx #3 { \seq_item:Nn \l_tmpa_seq { \l__xcref_fapaa_index_int } }
      }
      {
        \prop_get:NnNF \c__xcref_french_article_prop {#2} \l_tmpa_tl
          {
            \msg_error:nnn { xcref }
              { french-definite-article-not-in-c__xcref_french_article_prop } {#2}
          }

        \tl_set:Nn #3 { #1~ }
        \tl_put_right:NV #3 \l_tmpa_tl
      }

    \__xcref_protect_accent_macros_from_tabbing:N #3
    \__xcref_protect_accent_macros_from_tabbing:N #3
  }

\cs_generate_variant:Nn \xcref_french_assemble_prep_and_article:nnN { V }
\cs_generate_variant:Nn \xcref_french_assemble_prep_and_article:nnN { VV }
```

(*End definition for* `\xcref_french_assemble_prep_and_article:nnN`. *This function is documented on page* 13.)

__xcref_french_set_names_table:n

Set `\g__xcref_french_names_prop` to the given value.

#1 : valid value for an l3prop object

This function can only be used in the preamble because the list of keys is computed at `\begin{document}` time and stored in `\g__xcref_french_types_seq`.

```
\cs_new_protected:Npn \__xcref_french_set_names_table:n #1
  {
    \prop_gset_from_keyval:Nn \g__xcref_french_names_prop {#1}
  }

\@onlypreamble \__xcref_french_set_names_table:n
```

(*End definition for* `\__xcref_french_set_names_table:n`.)

`\xcref_french_set_names_table_entry:nnnnn`
`\xcreffrenchSetNamesTableEntry`

Set one (key, value) pair in `\g__xcref_french_names_prop`.
`#1` : reference type
`#2` : default preposition, in LICR form
`#3` : definite article for the singular form, among `l'`, `le` and `la`
`#4` : singular form of the noun corresponding to the reference type
`#5` : plural form of the same noun
This function can only be used in the preamble because it modifies `\g__xcref_french_names_prop` and the list of keys in this property list is computed at `\begin{document}` time (stored in `\g__xcref_french_types_seq`).

```
\cs_new_protected:Npn \xcref_french_set_names_table_entry:nnnnn #1#2#3#4#5
  {
    \prop_gput:Nnn \g__xcref_french_names_prop {#1} { {#2} {#3} {#4} {#5} }
  }

\@onlypreamble \xcref_french_set_names_table_entry:nnnnn

\NewDocumentCommand \xcreffrenchSetNamesTableEntry { m m m m m }
  {
    \xcref_french_set_names_table_entry:nnnnn {#1} {#2} {#3} {#4} {#5}
  }
```

(*End definition for* `\xcref_french_set_names_table_entry:nnnnn` *and* `\xcreffrenchSetNamesTableEntry`. *These functions are documented on page* 13.)

`\__xcref_french_setup_cref_names:N`

Call `\crefname`, `\Crefname` and `\@crefdefineallformats` for each reference type in `#1`.
`#1` : sequence variable containing reference types
This is the main entry point of the language-specific module.

```
\cs_new_protected:Npn \__xcref_french_setup_cref_names:N #1
  {
    \seq_map_function:NN #1 \__xcref_french_setup_cref_names_mapfunc:n
  }
```

(*End definition for* `\__xcref_french_setup_cref_names:N`.)

Prepare the terrain for `\__xcref_french_setup_cref_names_mapfunc:n`.

```
\cs_generate_variant:Nn \tl_mixed_case:nn { VV }
\cs_generate_variant:Nn \msg_error:nnn { nnV }
\cs_generate_variant:Nn \str_case:nnF { V }
```

Used to store data extracted from `\g__xcref_french_names_prop` corresponding to the selected reference type.

```
\tl_new:N \l__xcref_fscnm_type_data_tl
\seq_new:N \l__xcref_fscnm_type_data_seq
```

Selected preposition.

```
\tl_new:N \l__xcref_fscnm_prep_tl
```

Singular form of the definite article normally used with the selected reference type.

```
\tl_new:N \l__xcref_fscnm_article_tl
```

What to insert before a reference name (these contain the whole prefix: empty, article or preposition plus article, depending on the value of /xcref/french/form).

```
\tl_new:N \l__xcref_fscnm_prefix_sing_tl
\tl_new:N \l__xcref_fscnm_prefix_plur_tl
```

The whole strings passed to \crefname and \Crefname.

```
\tl_new:N \l__xcref_fscnm_singular_tl
\tl_new:N \l__xcref_fscnm_plural_tl
```

Language code for use with \tl_mixed_case:nn.

```
\tl_new:N \l__xcref_fscnm_lang_tl
```

__xcref_french_setup_cref_names_mapfunc:n

Call \crefname, \Crefname and \@crefdefineallformats for the specified reference type.

#1 : reference type (e.g., chapter, section, theorem, etc.)

```
\cs_new_protected:Npn \__xcref_french_setup_cref_names_mapfunc:n #1
  {
    \prop_get:NnNF \g__xcref_french_names_prop {#1} \l__xcref_fscnm_type_data_tl
      {
        \msg_error:nnn { xcref } { french-requested-use-of-undefined-type } {#1}
      }

    \seq_set_split:NnV \l__xcref_fscnm_type_data_seq { }
                       \l__xcref_fscnm_type_data_tl
    \seq_pop_left:NN \l__xcref_fscnm_type_data_seq \l__xcref_fscnm_prep_tl
```

Override the default preposition for reference type #1 if one was explicitly specified.

```
    \tl_if_empty:NF \l__xcref_french_selected_preposition_tl
      {
        \tl_set_eq:NN \l__xcref_fscnm_prep_tl \l__xcref_french_selected_preposition_tl
      }
    \seq_pop_left:NN \l__xcref_fscnm_type_data_seq \l__xcref_fscnm_article_tl
    \seq_pop_left:NN \l__xcref_fscnm_type_data_seq \l__xcref_fscnm_singular_tl
    \seq_pop_left:NN \l__xcref_fscnm_type_data_seq \l__xcref_fscnm_plural_tl

    \str_case:VnF \l__xcref_french_selected_form_str
      {
        { prep+article+noun }
        {
          \xcref_french_assemble_prep_and_article:VVN \l__xcref_fscnm_prep_tl
            \l__xcref_fscnm_article_tl \l__xcref_fscnm_prefix_sing_tl
          \xcref_french_assemble_prep_and_article:VnN \l__xcref_fscnm_prep_tl
            { les } \l__xcref_fscnm_prefix_plur_tl
        }
        { article+noun }
        {
          \prop_get:NVNF \c__xcref_french_article_prop \l__xcref_fscnm_article_tl
                         \l__xcref_fscnm_prefix_sing_tl
            {
```

```
            \msg_error:nnV { xcref }
              { french-definite-article-not-in-c__xcref_french_article_prop }
              \l__xcref_fscnm_article_tl
          }
        \tl_set:Nn \l__xcref_fscnm_prefix_plur_tl { les~ }
      }
      { noun }
      {
        \tl_clear:N \l__xcref_fscnm_prefix_sing_tl
        \tl_clear:N \l__xcref_fscnm_prefix_plur_tl
      }
    }
    {
      \msg_error:nnV { xcref } { french-unknown-prefix-form }
        \l__xcref_french_selected_form_str
    }

  \tl_put_left:NV \l__xcref_fscnm_singular_tl \l__xcref_fscnm_prefix_sing_tl
  \tl_put_left:NV \l__xcref_fscnm_plural_tl \l__xcref_fscnm_prefix_plur_tl
```

Make sure `\xcref@tabacckludge` is used to protect `\'`, `\`` and `\=` from the `tabbing` environment, which redefines them (see the inputenc documentation).

```
  \__xcref_protect_accent_macros_from_tabbing:N \l__xcref_fscnm_singular_tl
  \__xcref_protect_accent_macros_from_tabbing:N \l__xcref_fscnm_plural_tl
```

Set up the non-capitalized forms.

```
  \__xcref_call_crefname:nVV {#1} \l__xcref_fscnm_singular_tl \l__xcref_fscnm_plural_tl
```

This influences the behavior of `\tl_mixed_case:nn`.

```
  \tl_set_eq:NN \l_tl_case_change_exclude_tl \l__xcref_case_change_exclude_tl
```

Prepare for the language-specific case conversion.

```
  \tl_set:Nx \l__xcref_fscnm_lang_tl
    { \xcref_lang_for_tl_mixed_case:V \l__xcref_selected_lang_str }
  \tl_set:Nx \l__xcref_fscnm_singular_tl
    { \tl_mixed_case:VV \l__xcref_fscnm_lang_tl \l__xcref_fscnm_singular_tl }
  \tl_set:Nx \l__xcref_fscnm_plural_tl
    { \tl_mixed_case:VV \l__xcref_fscnm_lang_tl \l__xcref_fscnm_plural_tl }
```

Set up the capitalized forms.

```
  \__xcref_call_Crefname:nVV {#1} \l__xcref_fscnm_singular_tl \l__xcref_fscnm_plural_tl
```

Call `\@crefdefineallformats` to define all things derived from the `\crefname` and `\Crefname` calls we just did (for instance, this will call `\crefmultiformat`, which in turn will define `\cref@`⟨*type*⟩`@format@first`, among others). This is likely to be slow.

```
  \__xcref_call_crefdefineallformats:n {#1}
}
```

(*End definition for* `\__xcref_french_setup_cref_names_mapfunc:n`.)

`\__xcref_french_generic_wrapper:nn`
`\xcreffrenchgenericwrapper`
`\xcreffrenchwrapper`
`\xcreffrenchWrapper`

Create convenience `\xcref` wrappers for `french`.

```
\__xcref_create_language_specific_wrapper:nNNNN { french }
  \__xcref_french_generic_wrapper:nn \xcreffrenchgenericwrapper
  \xcreffrenchwrapper \xcreffrenchWrapper
```

(*End definition for* `\__xcref_french_generic_wrapper:nn` *and others. These functions are documented on page 11.*)

Fill the list of known types (`\g__xcref_french_types_seq`) with the keys of `\g__xcref_french_names_prop`. Since we only do this here, this implies that `/xcref/french/names table` can only be set in the preamble (otherwise, `\g__xcref_french_types_seq` and `\g__xcref_french_names_prop` would become out of sync).

```
\AtBeginDocument
  {
    \seq_gclear:N \g__xcref_french_types_seq
    \prop_map_inline:Nn \g__xcref_french_names_prop
      { \seq_gput_right:Nn \g__xcref_french_types_seq {#1} }
  }
```

Switch off the expl3 category code régime.

```
\ExplSyntaxOff
```

DocStrip end guard for `xcref-french.tex`.

```
⟨/french-module⟩
```

7.3 Module `xcref-ngerman.tex`

DocStrip start guard for `xcref-ngerman.tex`.

```
⟨*ngerman-module⟩
```

Switch to the category code régime suitable for expl3 code.

```
\ExplSyntaxOn
```

7.3.1 expl3 messages

```
\msg_new:nnn { xcref } { ngerman-requested-use-of-undefined-type }
  { (ngerman)~Requested~use~of~undefined~reference~type:~'\exp_not:n {#1}'. }

\msg_new:nnn { xcref } { ngerman-unable-to-determine-the-case }
  { (ngerman)~Unable~to~determine~the~case~to~use~(nominative,~etc.) }

\msg_new:nnn { xcref }
  { ngerman-unexpected-case-name-when-trying-to-find-index }
  { (ngerman)~Bug:~unexpected~case~name~when~doing~lookup~in~\token_to_str:N
    \c__xcref_ngerman_case_idx_prop \c_colon_str \space '\exp_not:n {#1}'. }

\msg_new:nnn { xcref } { ngerman-unknown-prefix-form }
  { (ngerman)~Unknown~form~for~the~part~preceding~the~
    \token_to_str:N \namecref \c_colon_str \space '\exp_not:n {#1}'. }
```

7.3.2 Variables

`\l__xcref_ngerman_selected_form_str`

Specifies how references are to be produced, in grammatical terms. There are four possible values: `noun`, `article+noun`, `prep+noun` and `prep+article+noun`, where `prep` stands for "preposition".

```
\str_new:N \l__xcref_ngerman_selected_form_str
```

(*End definition for* `\l__xcref_ngerman_selected_form_str`.)

\l__xcref_ngerman_selected_preposition_tl The selected preposition, if any. When non-empty, this variable overrides the default preposition associated to the reference type.

```
\tl_new:N \l__xcref_ngerman_selected_preposition_tl
```

(*End definition for* `\l__xcref_ngerman_selected_preposition_tl`.)

Indices corresponding to the order of the cases in `\g__xcref_ngerman_names_prop`

```
\prop_const_from_keyval:Nn \c__xcref_ngerman_case_idx_prop
  {
    nominative = 1,
    accusative = 2,
    dative     = 3,
    genitive   = 4
  }
```

Can be changed by users via `/xcref/ngerman/names table`. This is global because we don't want to compute `\g__xcref_ngerman_types_seq` (containing the list of keys in `\g__xcref_ngerman_names_prop`) all the time. The prepositions and articles in this variable must be written in LICR representation (for instance, use `\"uber` but not `über` nor `{\"u}ber`).

In case you would need `\'`, `\‘` or `\=` in the nouns ("Problem", "Problème", etc.), you may want to use `\xcref@tabacckludge` followed by `'`, `‘` or `=` instead (see `\g__xcref_french_names_prop` in the `french` module for examples).

```
\prop_new:N \g__xcref_ngerman_names_prop
\prop_gset_from_keyval:Nn \g__xcref_ngerman_names_prop
  {
    problem    = {in}{{das}{Problem}{die}{Probleme}}%
                    {{das}{Problem}{die}{Probleme}}%
                    {{dem}{Problem}{den}{Problemen}}%
                    {{des}{Problems}{der}{Probleme}},
    proposition= {nach}{{die}{Proposition}{die}{Propositionen}}%
                    {{die}{Proposition}{die}{Propositionen}}%
                    {{der}{Proposition}{den}{Propositionen}}%
                    {{der}{Proposition}{der}{Propositionen}},
    satz       = {nach}{{der}{Satz}{die}{S\"atze}}%
                    {{den}{Satz}{die}{S\"atze}}%
                    {{dem}{Satz}{den}{S\"atzen}}%
                    {{des}{Satzes}{der}{S\"atze}},
    theorem    = {nach}{{das}{Theorem}{die}{Theoreme}}%
                    {{das}{Theorem}{die}{Theoreme}}%
                    {{dem}{Theorem}{den}{Theoremen}}%
                    {{des}{Theorems}{der}{Theoreme}},
    lemma      = {nach}{{das}{Lemma}{die}{Lemmata}}%
                    {{das}{Lemma}{die}{Lemmata}}%
                    {{dem}{Lemma}{den}{Lemmata}}%
                    {{des}{Lemmas}{der}{Lemmata}},
    definition = {nach}{{die}{Definition}{die}{Definitionen}}%
                    {{die}{Definition}{die}{Definitionen}}%
                    {{der}{Definition}{den}{Definitionen}}%
                    {{der}{Definition}{der}{Definitionen}},
  }
```

Will contain the list of keys in `\g__xcref_ngerman_names_prop` (computed `\AtBeginDocument`)

```
\seq_new:N \g__xcref_ngerman_types_seq
```

Authorized values: `nominative`, `accusative`, `dative`, `genitive`.

```
\str_new:N \l__xcref_ngerman_selected_case_str
```

With some prepositions, the case is perfectly determined. This mapping thus takes precedence over whatever was set via the PGF key `/xcref/ngerman/case`. It can be modified by users via `/xcref/ngerman/prepositions always followed by the same case`. The keys must be written in LICR representation.

```
\prop_new:N \l__xcref_ngerman_case_for_prep_prop
\prop_set_from_keyval:Nn \l__xcref_ngerman_case_for_prep_prop
  {
    bis   = accusative,
    durch = accusative,
    f\"ur = accusative,
    gegen = accusative,
    ohne  = accusative,
    um    = accusative,
    aus   = dative,
    bei   = dative,
    mit   = dative,
    nach  = dative,
    seit  = dative,
    von   = dative,
    zu    = dative,
```

Prepositions normally followed by the genitive case seem to often have exceptions; thus, I prefer letting a native German speaker decide what is correct to include here.

```
  }
```

Some prepositions and articles can merge. This mapping can be modified by users via `/xcref/ngerman/composition table for prepositions and articles`. The keys must be written in LICR representation.

```
\prop_new:N \l__xcref_ngerman_article_and_prep_prop
\prop_set_from_keyval:Nn \l__xcref_ngerman_article_and_prep_prop
  {
    an~dem = am,
    in~dem = im,
    von~dem = vom,
    zu~dem = zum,
    zu~der = zur,
    bei~dem = beim,
  }
```

7.3.3 PGF keys

Note that the `/xcref/ngerman/names table` key can only be set in the preamble. The initial value of `\l__xcref_ngerman_selected_preposition_tl` is empty (see above).

```
\pgfqkeys { /xcref/ngerman }
  {
    form/.is~choice,
    form/noun/.code = { \str_set:Nn \l__xcref_ngerman_selected_form_str {noun} },
    form/article+noun/.code =
            { \str_set:Nn \l__xcref_ngerman_selected_form_str {article+noun} },
    form/prep+noun/.code =
            { \str_set:Nn \l__xcref_ngerman_selected_form_str {prep+noun} },
```

```
    form/prep+article+noun/.code =
            { \str_set:Nn \l__xcref_ngerman_selected_form_str {prep+article+noun} },
    form = prep+article+noun,   % set the initial value
    %
    preposition/.value~required,
    preposition/.code = {
      \__xcref_set_to_licr_form:Nn \l__xcref_ngerman_selected_preposition_tl {#1} },
    case/.is~choice,
    case/nom/.code = { \str_set:Nn \l__xcref_ngerman_selected_case_str
                                   { nominative } },
    case/acc/.code = { \str_set:Nn \l__xcref_ngerman_selected_case_str
                                   { accusative } },
    case/dat/.code = { \str_set:Nn \l__xcref_ngerman_selected_case_str
                                   { dative } },
    case/gen/.code = { \str_set:Nn \l__xcref_ngerman_selected_case_str
                                   { genitive } },
    names~table/.code = { \__xcref_ngerman_set_names_table:n {#1} },
    prepositions~always~followed~by~the~same~case/.code = {
      \prop_set_from_keyval:Nn \l__xcref_ngerman_case_for_prep_prop {#1} },
    composition~table~for~prepositions~and~articles/.code = {
      \prop_set_from_keyval:Nn \l__xcref_ngerman_article_and_prep_prop {#1} },
  }
```

7.3.4 Main code

\xcref_ngerman_assemble_prep_and_article:nnN
\xcref_ngerman_assemble_prep_and_article:VVN

Assemble a preposition and an article.

#1 : preposition
#2 : article
#3 : token list variable where the result is stored

```
\cs_new_protected:Npn \xcref_ngerman_assemble_prep_and_article:nnN #1#2#3
  {
    \prop_get:NnNF \l__xcref_ngerman_article_and_prep_prop { #1~#2 } #3
      {
        \tl_set:Nn #3 { #1~#2 }
      }

    \__xcref_protect_accent_macros_from_tabbing:N #3
    \__xcref_protect_accent_macros_from_tabbing:N #3
  }

\cs_generate_variant:Nn \xcref_ngerman_assemble_prep_and_article:nnN { VV }
```

(*End definition for* `\xcref_ngerman_assemble_prep_and_article:nnN`. *This function is documented on page* 14.)

__xcref_ngerman_set_names_table:n

Set the `\g__xcref_ngerman_names_prop` mapping to the given value.

#1 : valid value for an l3prop object

This function can only be used in the preamble because the list of keys is computed at `\begin{document}` time and stored in `\g__xcref_ngerman_types_seq`.

```
\cs_new_protected:Npn \__xcref_ngerman_set_names_table:n #1
  {
    \prop_gset_from_keyval:Nn \g__xcref_ngerman_names_prop {#1}
  }
```

```
\@onlypreamble \__xcref_ngerman_set_names_table:n
```

(*End definition for* `\__xcref_ngerman_set_names_table:n`.)

`\xcref_ngerman_set_names_table_entry:nnnnnn`
`\xcrefngermanSetNamesTableEntry`

Set one (key, value) pair in `\g__xcref_ngerman_names_prop`.
`#1` : reference type
`#2` : default preposition, in LICR form
`#3` : ⟨*nominative*⟩
`#4` : ⟨*accusative*⟩
`#5` : ⟨*dative*⟩
`#6` : ⟨*genitive*⟩
Each of the ⟨*nominative*⟩, ⟨*accusative*⟩, ⟨*dative*⟩ and ⟨*genitive*⟩ arguments should be of the form `{`⟨*article-sing*⟩`}{`⟨*noun-sing*⟩`}{`⟨*article-plur*⟩`}{`⟨*noun-plur*⟩`}`, where the four meta-syntactic variables represent the definite article and noun to use for reference type `#1` in singular and plural forms, respectively.

This function can only be used in the preamble because it modifies `\g__xcref_ngerman_names_prop` and the list of keys in this property list is computed at `\begin{document}` time (stored in `\g__xcref_ngerman_types_seq`).

```
\cs_new_protected:Npn \xcref_ngerman_set_names_table_entry:nnnnnn #1#2#3#4#5#6
  {
    \prop_gput:Nnn \g__xcref_ngerman_names_prop {#1} { {#2} {#3} {#4} {#5} {#6} }
  }

\@onlypreamble \xcref_ngerman_set_names_table_entry:nnnnnn

\NewDocumentCommand \xcrefngermanSetNamesTableEntry { m m m m m m }
  {
    \xcref_ngerman_set_names_table_entry:nnnnnn {#1} {#2} {#3} {#4} {#5} {#6}
  }
```

(*End definition for* `\xcref_ngerman_set_names_table_entry:nnnnnn` *and* `\xcrefngermanSetNamesTableEntry`. *These functions are documented on page 14.*)

`\__xcref_ngerman_setup_cref_names:N`

Call `\crefname`, `\Crefname` and `\@crefdefineallformats` for each reference type in `#1`.
`#1` : sequence variable containing reference types
This is the main entry point of the language-specific module.

```
\cs_new_protected:Npn \__xcref_ngerman_setup_cref_names:N #1
  {
    \seq_map_function:NN #1 \__xcref_ngerman_setup_cref_names_mapfunc:n
  }
```

(*End definition for* `\__xcref_ngerman_setup_cref_names:N`.)

Prepare the terrain for `\__xcref_ngerman_setup_cref_names_mapfunc:n`.

```
\cs_generate_variant:Nn \tl_mixed_case:nn { VV }
\cs_generate_variant:Nn \msg_error:nnn { nnV }
\cs_generate_variant:Nn \str_case:nnF { V }
```

Used to store data extracted from `\g__xcref_ngerman_names_prop` corresponding to the selected reference type.

```
\tl_new:N \l__xcref_gscnm_type_data_tl
\seq_new:N \l__xcref_gscnm_type_data_seq
```

Used to store data for the selected case (one ⟨*item*⟩ with index 2, 3, 4, or 5 in the values from `\g__xcref_ngerman_names_prop`).

```
\tl_new:N \l__xcref_gscnm_case_data_tl
\seq_new:N \l__xcref_gscnm_case_data_seq
```

Selected preposition.

```
\tl_new:N \l__xcref_gscnm_prep_tl
```

Value obtained from `\c__xcref_ngerman_case_idx_prop`.

```
\tl_new:N \l__xcref_gscnm_case_idx_tl
```

Singular and plural forms of the article.

```
\tl_new:N \l__xcref_gscnm_article_sing_tl
\tl_new:N \l__xcref_gscnm_article_plur_tl
```

What to insert before a reference name such as Abschnitt, Theorem, etc. (these contain the whole prefix: empty, article or preposition plus article, depending on the value of `/xcref/ngerman/form`).

```
\tl_new:N \l__xcref_gscnm_prefix_sing_tl
\tl_new:N \l__xcref_gscnm_prefix_plur_tl
```

The whole strings passed to `\crefname` and `\Crefname`.

```
\tl_new:N \l__xcref_gscnm_singular_tl
\tl_new:N \l__xcref_gscnm_plural_tl
```

Language for use with `\tl_mixed_case:nn`.

```
\tl_new:N \l__xcref_gscnm_lang_tl
```

`\__xcref_ngerman_setup_cref_names_mapfunc:n` Call `\crefname`, `\Crefname` and `\@crefdefineallformats` for the specified reference type.

`#1` : reference type (e.g., `chapter`, `section`, `theorem`, etc.)

```
\cs_new_protected:Npn \__xcref_ngerman_setup_cref_names_mapfunc:n #1
  {
    \prop_get:NnNF \g__xcref_ngerman_names_prop {#1} \l__xcref_gscnm_type_data_tl
      {
        \msg_error:nnn { xcref } { ngerman-requested-use-of-undefined-type }
                       {#1}
      }

    \seq_set_split:NnV \l__xcref_gscnm_type_data_seq { } \l__xcref_gscnm_type_data_tl
    \seq_pop_left:NN \l__xcref_gscnm_type_data_seq \l__xcref_gscnm_prep_tl
```

Override the default preposition for type `#1` if one was explicitly specified.

```
    \tl_if_empty:NF \l__xcref_ngerman_selected_preposition_tl
      {
        \tl_set_eq:NN \l__xcref_gscnm_prep_tl \l__xcref_ngerman_selected_preposition_tl
      }
```

If the selected form has a preposition and that preposition is always followed by a particular case (e.g., `dative`), use it. This may seem strange to store the "new" case in `\l__xcref_ngerman_selected_case_str`, but this variable will recover its previous value as soon as the group opened by `\xcref:nn` ends.

```
    \bool_lazy_any:nT
      {
        { \str_if_eq_p:Vn \l__xcref_ngerman_selected_form_str { prep+noun } }
        { \str_if_eq_p:Vn \l__xcref_ngerman_selected_form_str { prep+article+noun } }
```

```
    }
    {
      \prop_get:NVNT \l__xcref_ngerman_case_for_prep_prop
                     \l__xcref_gscnm_prep_tl \l_tmpa_tl
        { \str_set:Nx \l__xcref_ngerman_selected_case_str { \l_tmpa_tl } }
    }

  \prop_get:NVNF \c__xcref_ngerman_case_idx_prop
    \l__xcref_ngerman_selected_case_str \l__xcref_gscnm_case_idx_tl
    {
      \str_if_empty:NTF \l__xcref_ngerman_selected_case_str
        { \msg_error:nn { xcref } { ngerman-unable-to-determine-the-case } }
        {
          \msg_error:nnV { xcref }
            { ngerman-unexpected-case-name-when-trying-to-find-index }
            \l__xcref_ngerman_selected_case_str
        }
    }
```

Get data (initially coming from `\g__xcref_ngerman_names_prop`) corresponding to the selected case (`nominative`, `accusative`, etc.).

```
  \tl_set:Nx \l__xcref_gscnm_case_data_tl
    { \seq_item:Nn \l__xcref_gscnm_type_data_seq { \l__xcref_gscnm_case_idx_tl } }

  \seq_set_split:NnV \l__xcref_gscnm_case_data_seq { } \l__xcref_gscnm_case_data_tl
  \seq_pop_left:NN \l__xcref_gscnm_case_data_seq \l__xcref_gscnm_article_sing_tl
  \seq_pop_left:NN \l__xcref_gscnm_case_data_seq \l__xcref_gscnm_singular_tl
  \seq_pop_left:NN \l__xcref_gscnm_case_data_seq \l__xcref_gscnm_article_plur_tl
  \seq_pop_left:NN \l__xcref_gscnm_case_data_seq \l__xcref_gscnm_plural_tl
```

Compute the "prefix" token list (what comes before the noun).

```
  \str_case:VnF \l__xcref_ngerman_selected_form_str
    {
      { prep+article+noun }
      {
        \xcref_ngerman_assemble_prep_and_article:VVN \l__xcref_gscnm_prep_tl
          \l__xcref_gscnm_article_sing_tl \l__xcref_gscnm_prefix_sing_tl
        \xcref_ngerman_assemble_prep_and_article:VVN \l__xcref_gscnm_prep_tl
          \l__xcref_gscnm_article_plur_tl \l__xcref_gscnm_prefix_plur_tl
      }
      { prep+noun }
      {
        \tl_set_eq:NN \l__xcref_gscnm_prefix_sing_tl \l__xcref_gscnm_prep_tl
        \tl_set_eq:NN \l__xcref_gscnm_prefix_plur_tl \l__xcref_gscnm_prep_tl
      }
      { article+noun }
      {
        \tl_set_eq:NN \l__xcref_gscnm_prefix_sing_tl \l__xcref_gscnm_article_sing_tl
        \tl_set_eq:NN \l__xcref_gscnm_prefix_plur_tl \l__xcref_gscnm_article_plur_tl
      }
      { noun }
      {
        \tl_clear:N \l__xcref_gscnm_prefix_sing_tl
        \tl_clear:N \l__xcref_gscnm_prefix_plur_tl
      }
```

```
      }
      {
        \msg_error:nnV { xcref } { ngerman-unknown-prefix-form }
          \l__xcref_ngerman_selected_form_str
      }
```

Make sure that when we use `\tl_mixed_case:nn` to prepare for `\Crefname`, the case of the noun won't be affected (we don't want nouns to become all-lowercase in German!).

```
    \tl_set:Nx \l__xcref_gscnm_singular_tl
      { \exp_not:N \xcrefNoCaseChange
        { \exp_args:No \exp_not:n { \l__xcref_gscnm_singular_tl } } }
    \tl_set:Nx \l__xcref_gscnm_plural_tl
      { \exp_not:N \xcrefNoCaseChange
        { \exp_args:No \exp_not:n { \l__xcref_gscnm_plural_tl } } }
```

Assemble the prefix and the noun.

```
    \tl_if_empty:NF \l__xcref_gscnm_prefix_sing_tl
      {
        \tl_put_left:Nn \l__xcref_gscnm_singular_tl { ~ }
        \tl_put_left:NV \l__xcref_gscnm_singular_tl \l__xcref_gscnm_prefix_sing_tl
      }
    \tl_if_empty:NF \l__xcref_gscnm_prefix_plur_tl
      {
        \tl_put_left:Nn \l__xcref_gscnm_plural_tl { ~ }
        \tl_put_left:NV \l__xcref_gscnm_plural_tl \l__xcref_gscnm_prefix_plur_tl
      }
```

Make sure `\xcref@tabacckludge` is used to protect `\'`, `\`` and `\=` from the `tabbing` environment, which redefines them (see the inputenc documentation).

```
    \__xcref_protect_accent_macros_from_tabbing:N \l__xcref_gscnm_singular_tl
    \__xcref_protect_accent_macros_from_tabbing:N \l__xcref_gscnm_plural_tl
```

Set up the non-capitalized forms.

```
    \__xcref_call_crefname:nVV {#1} \l__xcref_gscnm_singular_tl \l__xcref_gscnm_plural_tl
```

This influences the behavior of `\tl_mixed_case:nn`.

```
    \tl_set_eq:NN \l_tl_case_change_exclude_tl \l__xcref_case_change_exclude_tl
```

Prepare for the language-specific case conversion.

```
    \tl_set:Nx \l__xcref_gscnm_lang_tl
      { \xcref_lang_for_tl_mixed_case:V \l__xcref_selected_lang_str }
    \tl_set:Nx \l__xcref_gscnm_singular_tl
      { \tl_mixed_case:VV \l__xcref_gscnm_lang_tl \l__xcref_gscnm_singular_tl }
    \tl_set:Nx \l__xcref_gscnm_plural_tl
      { \tl_mixed_case:VV \l__xcref_gscnm_lang_tl \l__xcref_gscnm_plural_tl }
```

Set up the capitalized forms.

```
    \__xcref_call_Crefname:nVV {#1} \l__xcref_gscnm_singular_tl \l__xcref_gscnm_plural_tl
```

Call `\@crefdefineallformats` to define all things derived from the `\crefname` and `\Crefname` calls we just did (for instance, this will call `\crefmultiformat`, which in turn will define `\cref@⟨type⟩@format@first`, among others). This is likely to be slow.

```
    \__xcref_call_crefdefineallformats:n {#1}
  }
```

(*End definition for* `\__xcref_ngerman_setup_cref_names_mapfunc:n`.)

__xcref_ngerman_generic_wrapper:nn
\xcrefngermangenericwrapper
\xcrefngermanwrapper
\xcrefngermanWrapper

Create convenience \xcref wrappers for ngerman.

```
\__xcref_create_language_specific_wrapper:nNNNN { ngerman }
  \__xcref_ngerman_generic_wrapper:nn \xcrefngermangenericwrapper
  \xcrefngermanwrapper \xcrefngermanWrapper
```

(*End definition for* `\__xcref_ngerman_generic_wrapper:nn` *and others. These functions are documented on page 11.*)

Fill the list of known types (\g__xcref_ngerman_types_seq) with the keys of \g__xcref_ngerman_names_prop. Since we only do this here, this implies that /xcref/ngerman/names table can only be set in the preamble (otherwise, \g__xcref_ngerman_types_seq and \g__xcref_ngerman_names_prop would become out of sync).

```
\AtBeginDocument
  {
    \seq_gclear:N \g__xcref_ngerman_types_seq
    \prop_map_inline:Nn \g__xcref_ngerman_names_prop
      { \seq_gput_right:Nn \g__xcref_ngerman_types_seq {#1} }
  }
```

Switch off the expl3 category code régime.

```
\ExplSyntaxOff
```

DocStrip end guard for xcref-ngerman.tex.

```
⟨/ngerman-module⟩
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

S

T

U

X