ACRO

v3.5 2021/01/16

Typeset Acronyms and other Abbreviations

Clemens Niederberger

https://github.com/cgnieder/acro/

contact@mychemistry.eu

acro allows you to define and manage acronyms and abbreviations. It can also be used for glossaries or nomenclatures.

Table of contents

Part I.
Get started with ACRO

1. Licence

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3c or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."

2. Glossary

articles Articles are prefixes to acronyms, usually separated with a blank. *Different types of articles are mutually exclusive.*

endings Endings are postfixes to acronyms, usually not separated from the acronym. *Different types of endings are mutually exclusive.*

load-time option A load-time `option` is a package option of ACRO which *must* be set as option to `\usepackage[⟨options⟩]{acro}`.

option An `option` is a package option of ACRO which must set with `\acsetup`. It *cannot* be set as option to `\usepackage`. Options usually also can be set in the optional argument of `\ac` and friends.

property A `property` is an option to the second argument of the `\DeclareAcroym` command. They are options of an individual acronym if you will.

template A template determines how different objects of ACRO are printed. This includes the acronyms themselves but also for example the list of acronyms as a whole.

translations Localisation strings which can be modified.

3. ACRO for the impatient

Acronyms are defined in the preamble via the command

`\DeclareAcronym{⟨id⟩}{⟨properties⟩}`
where ⟨*id*⟩ is a unique string to identify the acronym and ⟨*properties*⟩ is a key/value list of acronym properties. These include:

`short = {⟨text⟩}` (required)
The short form of the acronym. *This property is required*: an acronym must have a short form.

`long = {⟨text⟩}` (required)
The long form of the acronym. *This property is required*: an acronym must have a description.

! In its simplest form an acronym needs a short and a long form. Please note that both properties *must* be set.

In the document acronyms are used with these commands:

`\ac{⟨id⟩}` `\Ac{⟨id⟩}`
`\ac` prints the acronym ⟨*id*⟩, the first time with full description and every subsequent use only the abbreviated form. `\Ac` does the same but uppercases the first letter – this may be needed at the beginning of a sentence.

`\acs{⟨id⟩}` `\Acs{⟨id⟩}`
`\acs` prints the short form of the acronym ⟨*id*⟩. `\Acs` does the same but uppercases the first letter.

`\acl{⟨id⟩}` `\Acl{⟨id⟩}`
`\acl` prints the long form of the acronym ⟨*id*⟩. `\Acl` does the same but uppercases the first letter.

`\acf{⟨id⟩}` `\Acf{⟨id⟩}`
`\acf` prints the full form of the acronym ⟨*id*⟩. `\Acf` does the same but uppercases the first letter.

Let's say you defined CD as follows:

```
\DeclareAcronym{cd}{
  short = CD ,
  long  = compact disc
}
```

Then the usage is

```
\begin{tabular}{ll}
  first  & \ac{cd} \\
  second & \ac{cd} \\
  long   & \acl{cd} \\
  short  & \acs{cd} \\
  full   & \acf{cd}
\end{tabular}
```

first	compact disc (CD)
second	CD
long	compact disc
short	CD
full	compact disc (CD)

4. Setting options

4.1. Load-time options

ACRO knows only a small set of load-time options which can be used as argument to `\usepackage`:

`version` = `2|3` Initial: `3`

The option allows you to use the last version prior to the update to version 3. This may help if you don't have the time to fix issues after upgrading to the new version.

`upgrade` = `true|false` Initial: `true`

When this option is used ACRO tries to give as much helpful and meaningful warning or error messages when a deprecated or removed command, property, or option is used. This is especially useful if you are upgrading from version 2. The option will initially be true for a few months after the upgrade to version 3.

4.2. Setup command

All options of ACRO that have *not* been mentioned in section 4.1 have to be set up either with this command

`\acsetup{`⟨*options*⟩`}`

or as option to other commands. If the latter is possible then it is described when the corresponding commands are explained. Options usually follow a key/value syntax and are always described in the following way:

`option`

An option without a value. Those options are very rare if there are any.

`option` = `{`⟨*value*⟩`}` Initial: `preset`

An option where a value can be given. The pre-set value is given to the right.

`option` = `choiceA|choiceB|choiceC` Initial: `choiceB`

An option with a determined set of choices. The underlined value is chosen if the option is given without value.

`option` = `true|false`

A boolean option with only the choices `true` and `false`.

`module/option`

An option at a deeper level belonging to the module `module`.

All of the above is probably clear from an example (using real options):

```
\acsetup{
  make-links = true ,   % boolean
  index ,               % boolean
  format = \emph ,      % standard
  list / local ,        % boolean option of the list module
  list / display = all  % choice option of the list module
}
```

Part II.
Comprehensive description of creation and usage of acronyms

5. Declaring acronyms and other abbreviations

All acronyms have to be declared in the preamble with the following command in order to be used in the document. Any usage of an acronym which has not been declared leads to an error message.

`\DeclareAcronym{⟨id⟩}{⟨list of properties⟩}`

The basic command for declaring an acronym where ⟨*id*⟩ is a unique string identifying the acronym. Per default this is case sensitive which means `id` is different from `ID`, for example.

The command understands a number of properties which are listed in the following sections. This is a comprehensive overview over the existing properties. Many properties are also explained in more detail in later sections of this manual.

`case-sensitive = true|false` Initial: `false`

When this is set you can write the ID of the acronym upper- or lower- or mixed case and it is recognized by ACRO as the same. This might be useful when the acronym appears in the page header, for example.

! In its simplest form an acronym needs a short and a long form. Please note that both properties *must* be set.

5.1. Basic properties

`short = {⟨text⟩}` (required)

The short form of the acronym. *This property is required*: an acronym must have a short form.

Maybe you mostly have simple acronyms where the ID and short form are the same. In that case you can use

`use-id-as-short = true|false` Initial: `false`

to use the ID of the acronym as short form. For more complicated cases this would still allow you to set the short form.

`long = {⟨text⟩}` (required)

The long form of the acronym. *This property is required*: an acronym must have a description.

`alt = {⟨text⟩}` (initially empty)

Alternative short form.

`extra` = {⟨*text*⟩} (initially empty)
Extra information to be added in the list of acronyms.

`foreign` = {⟨*long form in foreign language*⟩} (initially empty)
Can be useful when dealing with acronyms in foreign languages, see section 14 on page 23 for details.

`long-post` = {⟨*text*⟩} (initially empty)
⟨*text*⟩ is appended to the long form of the acronym in the text but not in the list of acronyms.

`post` = {⟨*text*⟩} (initially empty)
⟨*text*⟩ is appended to the acronym in the text but not in the list of acronyms.

`single` = {⟨*text*⟩} if unused then equal to `long`
If provided ⟨*text*⟩ will be used instead of the long form if the acronym is only used a single time *and* the option `single` has been set, see section 9 on page 14.

`sort` = {⟨*text*⟩} if unused then equal to `short`
If used the acronym will be sorted according to this property instead of its short form.

`tag` = {⟨*csv list*⟩} (initially empty)
The tag(s) of an acronym.

`cite` = [⟨*prenote*⟩][⟨*postnote*⟩]{⟨*citation keys*⟩} (initially empty)
A citation that is printed to the acronym according to an option explained later.

`before-citation` = {⟨*text*⟩} (initially empty)
⟨*text*⟩ is prepended to the citation of the acronym when and where the citation is printed.

`index` = {⟨*text*⟩} (initially empty)
This property allows to overwrite the automatic index entry with an arbitrary one. See section 16.2 on page 27 for details.

`index-sort` = {⟨*text*⟩} if unused then equal to `sort`
If you use the option `index` every occurrence of an acronym is recorded to the index and sorted by its short form or (if set) by the value of the `sort` property. This property allows to set an individual sorting option for the index. See section 16.2 on page 27 for details.

`index-cmd` = {⟨*index command*⟩} (initially empty)
This sets the indexing command for the acronym. If unused then the command set by the corresponding option is used. See section 16.2 on page 27 for details.

5.2. Properties related to plural and indefinite forms

`short-plural` = {⟨*text*⟩} Initial: s
The plural ending appended to the short form.

`short-plural-form` = {⟨*text*⟩} (initially empty)
The plural short form of the acronym; replaces the short form when used instead of appending the plural ending.

`long-plural` = {⟨*text*⟩} Initial: `s`
The plural ending appended to the long form.

`long-plural-form` = {⟨*text*⟩} (initially empty)
Plural long form of the acronym; replaces the long form when used instead of appending the plural ending.

`alt-plural` = {⟨*text*⟩} Initial: `s`
The plural ending appended to the alternative form.

`alt-plural-form` = {⟨*text*⟩} (initially empty)
The plural alternative form of the acronym; replaces the alternative form when used instead of appending the plural ending.

`foreign-plural` = {⟨*text*⟩} Initial: `s`
The plural ending appended to the foreign form.

`foreign-plural-form` = {⟨*text*⟩} (initially empty)
Plural foreign form of the acronym; replaces the foreign form when used instead of appending the plural ending.

`short-indefinite` = {⟨*text*⟩} Initial: `a`
Indefinite article for the short form.

`long-indefinite` = {⟨*text*⟩} Initial: `a`
Indefinite article for the long form.

`alt-indefinite` = {⟨*text*⟩} Initial: `a`
Indefinite article for the alternative form.

5.3. Properties related to formatting

`format` = {⟨*code*⟩} (initially empty)
The format used for both short and long form of the acronym.

`short-format` = {⟨*code*⟩} if unused then equal to `format`
The format used for the short form of the acronym.

`long-format` = {⟨*code*⟩} if unused then equal to `format`
The format used for the long form of the acronym.

`first-long-format` = {⟨*code*⟩} if unused then equal to `long-format`
The format used for the first appearance of the long form of the acronym.

`alt-format = {⟨code⟩}` if unused then equal to `short-format`
The format used for the alternative form of the acronym. If this is not given the short format will be used.

`extra-format = {⟨code⟩}` (initially empty)
The format used for the additional information of the acronym.

`foreign-format = {⟨code⟩}` (initially empty)
The format used for the foreign form of the acronym.

`list-format = {⟨code⟩}` if unused then equal to `long-format`
The format used for the long form of the acronym in the list if the list template supports it. All pre-defined list templates *do* support it.

`first-style = long-short|short-long|short|long|footnote` (initially empty)
The style of the first appearance of the acronym, see also section 8 on page 13.

`subsequent-style = long-short|short-long|short|long|footnote` (initially empty)
The style of the appearance of the acronym after the first time.

Introduced in version v3.4 (2020/12/25)

`single-style = long-short|short-long|short|long|footnote` (initially empty)
The style of a single appearance of the acronym, see also section 9 on page 14.

5.4. Properties related to the created PDF file

`pdfstring = {⟨pdfstring⟩}` if unused then equal to `short`
Used as PDF string replacement in bookmarks when used together with the hyperref [ORT20] or the bookmark package [Obe19].

`pdfcomment = {⟨text⟩}`
Sets a tooltip description for an acronym. For actually getting tooltips you also need an appropriate setting of the options `pdfcomment/cmd` and `pdfcomment/use`, see also section 20.3 on page 32.

`short-acc = {⟨text⟩}` if unused then equal to `short`
Sets the `ActualText` property as presented by the accsupp package for the short form of the acronym.

`long-acc = {⟨text⟩}` if unused then equal to `long`
Sets the `ActualText` property as presented by the accsupp package for the long form of the acronym.

`alt-acc = {⟨text⟩}` if unused then equal to `alt`
Sets the `ActualText` property as presented by the accsupp package for the alternative short form of the acronym.

`foreign-acc = {⟨text⟩}` if unused then equal to `foreign`
Sets the `ActualText` property as presented by the accsupp package for the foreign form of the acronym.

`extra-acc = {⟨text⟩}` if unused then equal to `extra`

Sets the `ActualText` property as presented by the accsupp package for the extra information of the acronym.

`single-acc = {⟨text⟩}` if unused then equal to `long-acc`

Sets the `ActualText` property as presented by the accsupp package for a single appearance of the acronym.

`list-acc = {⟨text⟩}` if unused then equal to `list`

Sets the `ActualText` property as presented by the accsupp package for the appearance in the list of acronyms.

5.5. Futher properties

`list = {⟨text⟩}` if unused then equal to `long`

If specified this will be written in the list as description instead of the long form if the corresponding list template supports it.

`foreign-babel = {⟨language⟩}` (initially empty)

The babel [Bra19] or polyglossia [Cha19] language of the foreign form. This language is used to wrap the entry with `\foreignlanguage{⟨language⟩}` if either babel or polyglossia is loaded. You'll need to take care that the corresponding language is loaded by babel or polyglossia.

`foreign-locale = {⟨language⟩}` (initially empty)

The language name that is output when the option `locale/display` is used. If this property is not set then the appropriate value might be derived from `foreign-babel`. See section 14 on page 23 for details.

`preset = {⟨set name⟩}` (initially empty)

☆ New Enables to load a set of properties that has been defined earlier with `\NewAcroPreset`, siehe section 5.6.

`uselist = {⟨csv list of acronym ids⟩}` (initially empty)

☆ New If this property is given and all acronyms specified in this property have been used before the first time the current acronym is used it behaves as if it has been used before.

5.6. Presets

☆ New Sometimes it can be useful to have different kinds of acronyms or abbreviations or similar which share a common set of properties. Such sets can be defined with these commands:

`\NewAcroPreset{⟨set name⟩}{⟨csv list of properties⟩}`

Defines the property set ⟨*set name*⟩. Any valid property can be set in ⟨*csv list of properties*⟩.

`\RenewAcroPreset{⟨set name⟩}{⟨csv list of properties⟩}`

Redefines the property set ⟨*set name*⟩.

`\DeclareAcroPreset{⟨set name⟩}{⟨csv list of properties⟩}`

Defines or redefines the property set ⟨*set name*⟩ without checking.

6. Using acronyms

There are a number of commands to use acronyms with. Their names always follow the same pattern which should make their usage intuitive immediately.

All of these commands have a starred form which means "don't count this as usage". All of these commands also have an optional argument that allows to set options for that usage only.

`\acrocommand*[⟨options⟩]{⟨id⟩}`

This is the general syntax of all of the commands listed below. The star and the optional argument is left way for the sake of readability. *A command `\acrocommand` does not actually exist.*

`\ac{⟨id⟩}` `\Ac{⟨id⟩}` `\acp{⟨id⟩}` `\Acp{⟨id⟩}` `\iac{⟨id⟩}` `\Iac{⟨id⟩}`

`\ac` prints the acronym ⟨*id*⟩, the first time with full description and every subsequent use only the abbreviated form. `\Ac` does the same but uppercases the first letter – this may be needed at the beginning of a sentence. The commands `\acp` and `\Acp`, resp., print the corresponding plural forms. The commands `\iac` and `\Iac`, resp., print indefinite forms.

`\acs{⟨id⟩}` `\Acs{⟨id⟩}` `\acsp{⟨id⟩}` `\Acsp{⟨id⟩}` `\iacs{⟨id⟩}` `\Iacs{⟨id⟩}`

`\acs` prints the short form of the acronym ⟨*id*⟩. `\Acs` does the same but uppercases the first letter. The commands `\acsp` and `\Acsp`, resp., print the corresponding plural forms. The commands `\iacs` and `\Iacs`, resp., print indefinite forms.

`\acl{⟨id⟩}` `\Acl{⟨id⟩}` `\aclp{⟨id⟩}` `\Aclp{⟨id⟩}` `\iacl{⟨id⟩}` `\Iacl{⟨id⟩}`

`\acl` prints the long form of the acronym ⟨*id*⟩. `\Acl` does the same but uppercases the first letter. The commands `\aclp` and `\Aclp`, resp., print the corresponding plural forms. The commands `\iacl` and `\Iacl`, resp., print indefinite forms.

`\aca{⟨id⟩}` `\Aca{⟨id⟩}` `\acap{⟨id⟩}` `\Acap{⟨id⟩}` `\iaca{⟨id⟩}` `\Iaca{⟨id⟩}`

`\aca` prints the alternative short form of the acronym ⟨*id*⟩. `\Aca` does the same but uppercases the first letter. The commands `\acap` and `\Acap`, resp., print the corresponding plural forms. The commands `\iaca` and `\Iaca`, resp., print indefinite forms.

`\acf{⟨id⟩}` `\Acf{⟨id⟩}` `\acfp{⟨id⟩}` `\Acfp{⟨id⟩}` `\iacf{⟨id⟩}` `\Iacf{⟨id⟩}`

`\acf` prints the full form of the acronym ⟨*id*⟩. `\Acf` does the same but uppercases the first letter. The commands `\acfp` and `\Acfp`, resp., print the corresponding plural forms. The commands `\iacf` and `\Iacf`, resp., print indefinite forms.

The usage should be clear. Let's assume you have defined an acronym UFO like this:

```
\DeclareAcronym{ufo}{
  short = UFO ,
  long = unidentified flying object ,
  foreign = unbekanntes Flugobjekt ,
  foreign-plural-form = unbekannte Flugobjekte ,
  foreign-babel = ngerman ,
  long-indefinite = an
}
```

The typical outputs look like this:

```
\ac{ufo} \\
\iac{ufo} \\
\iacl{ufo} \\
\Iacf{ufo} \\
\acfp{ufo}
```

unidentified flying object (unbekanntes Flugobjekt, UFO)
a UFO
an unidentified flying object
An unidentified flying object (unbekanntes Flugobjekt, UFO)
unidentified flying objects (unbekannte Flugobjekte, UFOs)

! In a number of contexts all acronym commands act as if their starred form is used: in the table of contents, in the list of figures, and in the list of tables. The same is true for floats and the measuring phase of common table environments like `tabularx` or `ltxtable`.

7. Alternative short forms

Sometimes expressions have two different short forms. And example might be JPEG which also often is JPG. This is what the property `alt` is there for.

`alt` = {⟨*text*⟩}
Alternative short form.

Let's define JPEG:

```
\DeclareAcronym{jpg}{
  short = JPEG ,
  sort  = jpeg ,
  alt   = JPG ,
  long  = Joint Photographic Experts Group
}
```

And let's see how to use it:

```
\ac{jpg} \\
\ac{jpg} \\
\aca{jpg}
```

Joint Photographic Experts Group (JPEG or JPG)
JPEG
JPG

As you can see the full form shows both short forms of the acronym. This could be changed by altering the template for the full form, see section 24 on page 37 and section 8. The alternative form is also printed in the list of acronyms, see section A on page 56. This can also be changed by altering the template for the list, again see section 24.

8. The first or full appearance

If an acronym is used for the first time with `\ac` (after any number of usages with the starred forms of the usage commands listed in section 6 on page 11) or if an acronym is used `\acf`, then the first or full appearance of the acronym is printed.[1]

The first or full appearance of an acronym is determined by this option:

`first-style = long-short|short-long|short|long|footnote` Initial: `long-short`
The style of the first appearance of the acronym. This options sets the appearance for all acronyms. Available options in reality are the names of all defined templates of the type `acronym`. All pre-defined templates can be found in section 24.1 on page 37.

`subsequent-style = long-short|short-long|short|long|footnote` Initial: `short`
Introduced in version v3.4
The style of the appearance of the acronym after the first time. This options sets the appearance for all acronyms. Available options in reality are the names of all defined templates of the type `acronym`. All pre-defined templates can be found in section 24.1 on page 37.

It might be desirable to set the first appearance of an acronym individually. This is possible by setting the corresponding property:

`first-style = long-short|short-long|short|long|footnote` (initially empty)
The style of the first appearance of the acronym.

Let's again look at an example:

```
\acf[first-style=long-short]{cd} \\
\acf[first-style=short-long]{cd} \\
\acf[first-style=footnote]{cd} \\
\acf[first-style=long]{cd} \\
\acf[first-style=short]{cd}
```

compact disc (CD)
CD (compact disc)
CD[a]
compact disc
CD

a. compact disc

1. This usually requires at least two LaTeX runs until it is stable.

This also demonstrates the use of the optional argument.

An example of an abbreviation that should have `long` as first appearance might be "*etc.*", defined like this

```
\DeclareAcronym{etc}{
  short = etc\acdot ,
  long = et cetera ,
  format = \textit ,
  first-style = long ,
  plural =
}
```

and output like this:

```
\ac{etc}, \ac{etc} \ac{etc}.
```

et cetera, etc. etc.

The command `\acdot` is explained in section 18 on page 29. Basically it checks if a dot follows and outputs a dot if not.

9. Single appearances of an acronym

If an acronym is used only once (not counting usages with the starred forms of the usage commands listed in section 6 on page 11), then the single appearance of the acronym is printed.[2]

The single appearance of an acronym is determined by this option:

`single` = `true`|`false`|⟨*number*⟩ Initial: `false`

This option determines wether a single appearance of an acronym counts as *usage*. It might be desirable in such cases that an acronym is simply printed as long form and not added to the list of acronym. This is what this option does. With ⟨*number*⟩ the minimal number of usages can be given that needs to be exceeded. `single = {1}` is the same as `single = {true}`.

`single-style` = `long-short`|`short-long`|`short`|`long`|`footnote` Initial: `long`

The style of the single appearance of an acronym. Can be used to determine how a single appearance is printed if the option `single` has been set. This option sets the appearance for all acronyms. Available options in reality are the names of all defined templates of the type `acronym`. All pre-defined templates can be found in section 24.1 on page 37.

If you like you can also set the single appearance of an acronym individually:

`single` = {⟨*text*⟩} (initially empty)

If provided ⟨*text*⟩ will be used instead of whatever template ("style") has been set for the single appearance if the acronym is only used a single time *and* the option `single` has been set.[3]

2. This usually requires at least two LaTeX runs until it is stable.

3. Actually the template *`single`* is used which typesets the `single` property.

`single-style = long-short|short-long|short|long|footnote` (initially empty)
The style of the single appearance of the acronym.

Let's again look at an example. The acronym PNG is defined as follows:

```
\DeclareAcronym{png}{
  short = PNG ,
  long  = Portable Network Graphics ,
  first-style = short-long ,
  single-style = short
}
```

And it is used only once in this manual[4]:

```
\ac{png}
```

PNG

Please be aware that `\acf` would still print the full form, of course.

10. Printing the list

10.1. The main command and its options

The main idea is simple: just place

`\printacronyms[`*⟨options⟩*`]`
where you want the list to appear. It may require several (most times two) LaTeX runs for it to stabilize so look out for any warnings from ACRO requiring to re-run.

The options controlling the list are these:

`list/template = description|table|longtable|lof|toc` Initial: `description`
Choose the template to create the list with. See more on this in sections 24 on page 37 and A on page 52.

`list/sort = true|false` Initial: `true`
Decide wether to sort the list of acronyms alphabetically or to print it in order of definition.

`list/display = all|used` Initial: `used`
Decide wether to print only the acronyms actually used in the document or all acronyms which have been declared in the preamble.

`list/exclude = {`*⟨csv list of tags⟩*`}` (initially empty)
Set a list of tags to exclude from the list. Only acronyms not belonging to one of these tags will be included.

4. You will find it in the list of acronyms in section A nonetheless as this document does `list/display = {all}`.

`list/include = {⟨csv list of tags⟩}` (initially empty)
Set a list of tags to include in the list. Only acronyms belonging to one of these tags will be included.

`list/add = true|false` (initially empty)
Set a list of tags to include in the list. These acronyms will be included in any case.

Introduced in version v3.4
`list/heading = none|section|section*|chapter|chapter*`
Choose the heading template for the list of acronyms.

This only has an effect if the list template supports it. All pre-defined templates *do* support it.

`list/name = {⟨text⟩}` Initial: `\acrotranslate{list-name}`
Overwrites the text which is used in the heading.

This only has an effect if the list template supports heading templates *and* the heading templates support it. All pre-defined heading templates *do* support this.

`list/preamble = {⟨text⟩}` (initially empty)
Set a preamble to be placed between heading and actual list.

This only has an effect if the list template supports it. All pre-defined templates *do* support it.

`list/locale/display = true|false` Initial: `false`
This options determines wether the language of the foreign form is printed or not.

This only has an effect if the list template supports foreign forms. All pre-defined templates *do* support them.

All these options can be set with `\acsetup` globally or locally as options to `\printacronyms`. In the latter case omit the leading `list`:

```
\acsetup{list/display=all,list/exclude=units}
or
\printacronyms[display=all,exclude=units]
```

10.2. Add page numbers to the list

If you want to include the page numbers where the acronyms have been used in the list of acronym you can use these options:

`pages/display = first|all|none` Initial: `none`
Decide wether to include page numbers in the list of acroynms and wether to add the first page or every page. When you choose `first` and have hyperref loaded you will also get a backlink to that page.

`pages/seq/use = true|false` Initial: `true`
Turns a two-page range into ⟨*num*⟩ f. (*sequens*) and a three-page range into ⟨*num*⟩ ff. (*sequentes*) when set to `true`.

`pages/seq/pre = {⟨code⟩}` Initial: `\,`

⟨*code*⟩ is inserted between the page number and the sequens or sequentes symbol.

`pages/seq/threshold = {⟨num⟩}` Initial: 3

The threshold for a page range to be turned into *sequentes*. A page range above the threshold is still typeset as a range: ⟨*num1*⟩–⟨*num2*⟩.

`pages/fill = {⟨code⟩}` Initial: `\acrdotfill`

This is the code that is placed between acronym description and actual page numbers.

`pages/name = true|false` Initial: `false`

If set to true the page numbers are preceded with p. or pp.

`\acrodotfill`

☆ New

Creates a dotted line like those in the table of contents. If the macro `\cftdotfill` is defined it is equal to `\cftdotfill{\cftdotsep}`.

Introduced in version v3.3 (2020/11/21)

Additionally to setting these options with `\acsetup` they can be set as options to `\printacronyms`:

```
\printacronyms[pages={display=all,seq/use=false}]
```

10.3. Filter lists using tags

With the property `tag` you can assign one or more tags to an acronym. These tags can be used to filter the list of acronyms.

`tag = {⟨csv list⟩}` (initially empty)

The tag(s) of an acronym.

`list/exclude = {⟨csv list of tags⟩}` (initially empty)

Set a list of tags to exclude from the list. Only acronyms not belonging to one of these tags will be included.

`list/include = {⟨csv list of tags⟩}` (initially empty)

Set a list of tags to include in the list. Only acronyms belonging to one of these tags will be included.

Let's look at an example. This manual declares these two acronyms with the tag `city`:

```
\DeclareAcronym{la}{
  short = LA ,
  long = Los Angeles,
  plural = ,
  tag = city
}
```

```
\DeclareAcronym{ny}{
  short = NY ,
  long = New York ,
  plural = ,
  tag = city
}
```

We can now use this to either print a list *without* these acronyms by saying

```
\printacronyms[exclude=city]
```

or print a list *with only* these acronyms with

```
\printacronyms[include=city,heading=none]
```

LA Los Angeles

NY New York 26

! If you use both `exclude` and `include` and list a tag in both `exclude` takes precedence over `include`.

```
\printacronyms[exclude={a,b},include={b,c}]
```

would only print acronyms with tag `c`.

10.4. Local lists

Maybe you like a list of acronyms for each chapter in a book which only lists the acronyms used within this chapter. You need to do three things: set

`barriers/use = true|false` Initial: `false`

this option to `true`, place

`\acbarrier`

before a new chapter starts (this is not necessary for the first chapter), and use `\printacronyms` with the option

`list/local = true|false` Initial: `false`

or set this option once in the preamble with `\acsetup` so it is applied to every list.

Please read more on barriers in section 17 on page 28.

> **!** Please don't use page numbers together with local lists for the time being. If an acronym appears in more than one list both lists would contain the *same* page numbers anstead of only the ones local to barriers.
> For the similar reasons please also don't use `make-links` together with local lists.
> This *might* be resolved on day.

11. Formatting

ACRO has a number of options and parameters which can be used to influence the formatting of acronyms.

`format` = {⟨*code*⟩} (initially empty)
Sets the format for both the short and the long form.

`format/short` = {⟨*code*⟩} (initially empty)
Sets the format for the short form.

`format/long` = {⟨*code*⟩} (initially empty)
Sets the format for the long form.

`format/first-long` = {⟨*code*⟩} (initially empty)
Sets the format for the first appearance of the long form.

`format/alt` = {⟨*code*⟩} (initially empty)
Sets the format for the alternative form.

`format/extra` = {⟨*code*⟩} (initially empty)
Sets the format for the extra information.

`format/foreign` = {⟨*code*⟩} (initially empty)
Sets the format for the foreign form.

`format/list` = {⟨*code*⟩} (initially empty)
Sets the format for the long form in the list form.

While this options influence the formatting of the acronyms globally you can also give each acronym its own formatting individually:

`format` = {⟨*code*⟩} (initially empty)
The format used for both short and long form of the acronym.

`short-format` = {⟨*code*⟩} if unused then equal to `format`
The format used for the short form of the acronym.

`long-format` = {⟨*code*⟩} if unused then equal to `format`
The format used for the long form of the acronym.

`first-long-format = {⟨code⟩}` if unused then equal to `long-format`
The format used for the first appearance of the long form of the acronym.

`alt-format = {⟨code⟩}` if unused then equal to `short-format`
The format used for the alternative form of the acronym. If this is not given the short format will be used.

`extra-format = {⟨code⟩}` (initially empty)
The format used for the additional information of the acronym.

`foreign-format = {⟨code⟩}` (initially empty)
The format used for the foreign form of the acronym.

`single-format = {⟨code⟩}` if unused then equal to `long-format`
The format used for the acronym if the acronym is only used a single time.

`list-format = {⟨code⟩}` if unused then equal to `long-format`
The format used for the long form of the acronym in the list if the list template supports it. All pre-defined list templates *do* support it.

`first-style = long-short|short-long|short|long|footnote` (initially empty)
The style of the first appearance of the acronym, see also section 8 on page 13.

`single-style = long-short|short-long|short|long|footnote` (initially empty)
The style of a single appearance of the acronym, see also section 9 on page 14.

Changed in version v3.3

Per default the individual formatting instructions *replace* the global ones. This can be changed through the option

`format/replace = true|false` Initial: `true`
With this option active local options will *replace* the global ones.

Let's see an example:

```
\DeclareAcronym{pdf}{
  short = pdf ,
  long = Portable Document Format ,
  short-format = \scshape
}
```

```
\acsetup{format = \itshape}
\acf{pdf} \par
\acsetup{format/replace=false}
\acf{pdf}
```

Portable Document Format (PDF)
Portable Document Format (*PDF*)

12. Plural forms and other endings

12.1. The plural ending and the plural form

Not in all languages plural forms are as easy as always appending an “s”. Not even English. Sometimes there’s other endings instead.[5] This is why ACRO has quite a number of different properties related to plural forms or endings:

`short-plural` = {⟨*text*⟩} Initial: s
The plural ending appended to the short form.

`short-plural-form` = {⟨*text*⟩} (initially empty)
The plural short form of the acronym; replaces the short form when used instead of appending the plural ending.

`long-plural` = {⟨*text*⟩} Initial: s
The plural ending appended to the long form.

`long-plural-form` = {⟨*text*⟩} (initially empty)
Plural long form of the acronym; replaces the long form when used instead of appending the plural ending.

`alt-plural` = {⟨*text*⟩} Initial: s
The plural ending appended to the alternative form.

`alt-plural-form` = {⟨*text*⟩} (initially empty)
The plural alternative form of the acronym; replaces the alternative form when used instead of appending the plural ending.

`foreign-plural` = {⟨*text*⟩} Initial: s
The plural ending appended to the foreign form.

`foreign-plural-form` = {⟨*text*⟩} (initially empty)
Plural foreign form of the acronym; replaces the foreign form when used instead of appending the plural ending.

There are two options which allow to change the default values for the whole document:

`short-plural-ending` = {⟨*text*⟩} Initial: s
Defines the plural ending for the short forms to be ⟨*text*⟩.

5. German is full of such examples.

`long-plural-ending` = {⟨*text*⟩} Initial: s

Defines the plural ending for the long forms to be ⟨*text*⟩.

Now let's see two simple examples demonstrating the two different kinds of plural settings:

```
\DeclareAcronym{sw}{
  short = SW ,
  long = Sammelwerk ,
  long-plural = e
}
\DeclareAcronym{MP}{
  short = MP ,
  long  = Member of Parliament ,
  plural-form = Members of Parliament
}
```

The first one has another plural ending than the usual "s". The second one has a different plural form altogether because appending an "s" would give a wrong form:

```
\acfp{sw} \par
\acfp{MP}
```

Sammelwerke (SWs)
Members of Parliament (MPs)

12.2. Other endings

Besides plural endings there are other ones like the genitive case, for example. This is why ACRO generalized the concept. Section 25 on page 45 explains in detail how to define and use additional endings.

13. Articles

13.1. Indefinite forms

Indefinite forms can be a problem if the short and the long form of acronyms have different indefinite articles.[6]

```
\acreset{ufo}%
a \ac{ufo} \par
an \ac{ufo}
```

a unidentified flying object (unbekanntes Flugobjekt, UFO)
an UFO

And what good would it be to use a package like ACRO if you have to keep track of first and second uses, anyway? This is why UFO should be defined like we did on page 11. We then can just use the dedicated commands and let them decide for us:

6. This may very well be a language specific issue.

```
\acreset{ufo}%
\iac{ufo} \par
\iac{ufo}
```

an unidentified flying object (unbekanntes Flugobjekt, UFO)
a UFO

The commands which also output the indefinite article all start with an "i" and have all been described in section 6 on page 11 already: `\iac`, `\Iac`, `\iacs`, `\Iacs`, `\iacl`, `\Iacl`, `\iaca`, `\Iaca`, `\iacf`, and `\Iacf`.

13.2. Other articles

There might be cases – most likely depending on your language – when you would like to have other articles behaving similar to the indefinite ones. Section 26 explains in detail how to define and use additional articles.

14. Foreign language acronyms

Sometimes and in some fields more often than in others abbreviations are used that are derived from another language. ACRO provides a number of properties for such cases:

`foreign = {`⟨*long form in foreign language*⟩`}` (initially empty)
Can be useful when dealing with acronyms in foreign languages, see section 14 for details.

`foreign-plural = {`⟨*text*⟩`}` Initial: `s`
The plural ending appended to the foreign form.

`foreign-plural-form = {`⟨*text*⟩`}` (initially empty)
Plural foreign form of the acronym; replaces the foreign form when used instead of appending the plural ending.

`foreign-format = {`⟨*code*⟩`}` (initially empty)
The format used for the foreign form of the acronym.

`foreign-babel = {`⟨*language*⟩`}` (initially empty)
The babel or polyglossia language of the foreign form. This language is used to wrap the entry with `\foreignlanguage{`⟨*language*⟩`}` if either babel or polyglossia is loaded. You'll need to take care that the corresponding language is loaded by babel or polyglossia.

`foreign-locale = {`⟨*language*⟩`}` (initially empty)
The language name that is output when the option `locale/display` is used. If this property is not set then the appropriate value might be derived from `foreign-babel`.

There are also some options:

`foreign/display = true|false` Initial: `true`
Determine wether to hide or display the foreign form.

`list/foreign/display = true|false` Initial: `true`

Introduced in version v3.2 (2020/05/09)

Determine wether to hide or display the foreign form in the list of acronyms.

`locale/display = true|false` Initial: `false`

This options determines wether the language of the foreign form is printed or not when the full form of the acronym is printed.

`list/locale/display = true|false` Initial: `false`

The same but for the list of acronyms.

`locale/format = {⟨code⟩}` Initial: `\em\text_titlecase_first:n`

Determines how said language is formatted when printed. The last command in ⟨*code*⟩ may take a mandatory argument.

Let's say you are writing a German document and are using the abbreviation ECU for Steuergerät which stems from the English "Electronic Control Unit". Then you can define it as follows:

```
\DeclareAcronym{ecu}{
  short   = ECU ,
  long    = Steuergerät ,
  foreign = Electronic Control Unit ,
  foreign-babel = english ,
  foreign-locale = englisch
}
```

Now the abbreviation is introduced so that everyone understands the confusion:

```
\ac{ecu} \par
\acsetup{locale/display,locale/format=\emph}
\acf{ecu}
```

Steuergerät (Electronic Control Unit, ECU)
Steuergerät (*englisch*: Electronic Control Unit, ECU)

The property `foreign-babel` is used for ensuring correct hyphenation as long as you use babel or polyglossia and load the corresponding language, too. If you are writing your document in English then ACRO is able to deduce the language used for the "locale" field by itself:

```
\DeclareAcronym{eg}{
  short = e.g\acdot ,
  long  = for example ,
  foreign = exempli gratia ,
  foreign-babel  = latin ,
  short-format = \textit ,
  foreign-format = \textit
}
```

```
\acsetup{locale/display,first-style=short-long}
\acf{eg}
```

e.g. (*Latin*: *exempli gratia*: for example)

15. Uppercasing

Depending on the kind of abbreviations you have and depending on their definition and maybe also depending on your language the long and sometimes also the short forms need to start with an uppercase letter at the beginning of a sentence while it starts with a lowercase letter otherwise.

For this ACRO provides uppercase versions for all predefined acronym commands listed in section 6. The usage is self-explaining:

```
There was \iacl{ufo} hovering \dots \par
\Aclp{ufo} were hovering \dots
```

There was an unidentified flying object hovering …
Unidentified flying objects were hovering …

If you defined them with uppercase letters to begin with then these commands have no effect, of course.

```
\DeclareAcronym{ufo}{
  short = UFO ,
  long = Unidentified Flying Object
}
```

There are a number of options to control the uppercasing behavior:

`uppercase/first`
The default setting. Converts the first letter to uppercase.

`uppercase/title`
This is just a synonym of `first`.

`uppercase/all`
Converts *all* letters to uppercase.

`uppercase/none`
Converts *all* letters to *lowercase*

`uppercase/cmd = {⟨command⟩}`
All of the above options just choose the right command using this option internally. This means you can choose a different behavior altogether by setting this option to something else. For example you could use `\capitalisewords` from the package mfirstuc [Tal17]. The command needs to have one mandatory argument.

There may be reasons to exclude short forms from being uppercased. This can be controlled by this option:

`uppercase/short = true|false` Initial: `true`
It allows you to disable the mechanism for the `short` and `alt` properties.

16. Citing and indexing

16.1. Citing

Acronyms can be given cite keys. This makes it possible to add a citation reference automatically when the acronym is used for the first time.

Let's see an example first. NY has been defined like this:

```
\DeclareAcronym{ny}{
  short = NY ,
  long = New York ,
  plural = ,
  tag = city ,
  cite = NewYork
}
```

The property `cite` will now trigger ACRO to input `\cite{NewYork}` after the acronym:

```
\ac{ny}
```

New York (NY) [Wik20]

Depending on the citation style (and probably other factors, too) it might be desirable to add the citation rather inside the parentheses together with the short form of the acronym and even cited with a different command. For cases like these ACRO offers a number of options:

`cite/cmd = {⟨citation command⟩}` Initial: `\cite`
Choose the command with which citations ar printed.

`cite/group = true|false` Initial: `false`

Decide wether to group citations with the short form in the parentheses. The template must support this. ACRO's pre-defined templates *do* support it.

`cite/display = first|all|none` Initial: `first`

Decide wether to output the citation in the first/full usage only or always or never.

`cite/pre = {⟨text⟩}` Initial: `\nobreakspace`

Arbitrary code directly output before the citation.

`cite/group/cmd = {⟨citation command⟩}` Initial: `\cite`

Choose the command with which grouped citations are printed.

`cite/group/pre = {⟨text⟩}` Initial: `,␣`

Arbitrary code directly output before the citation in the grouped case.

If for example you use biblatex's `authoryear` style [LKW19] you might want to have settings like these:

```
\acsetup{
  cite/group = true ,
  cite/cmd = \parencite ,
  cite/group/cmd = \cite
}
```

```
\acsetup{cite/display = all}
\acf{ny} \\
\ac{ny}
```

New York (NY, Wikipedia 2020)
NY (Wikipedia 2020)

16.2. Indexing

Maybe you want to add your acronyms to an index. In that case it is probably desirable to let ACRO make this automatically. In the simplest case just enable it:

`index/use = true|false|indexed` Initial: `false`

Changed in version v3.5 (2020/01/16)

Enable indexing. If `indexed` is chosen only the acronyms for which the property `index` has been set are indexed. With `true` *all* acronyms are indexed.

`index/cmd = {⟨index command⟩}` Initial: `\index`

Choose a command for indexing.

`index/disable = {⟨code⟩}` Initial: `\def\@{}`

Sometimes it is desirable to change the meaning of a command inside an index entry. For the entries created by ACRO this can be achieved with this option.

`index/clear`

This option clears the disable list.

While these options set global behavior there are also properties to set them for an acronym individually.

`index = {⟨text⟩}` (initially empty)

This property allows to overwrite the automatic index entry with an arbitrary one.

`index-sort = {⟨text⟩}` if unused then equal to `sort`

If you use the option `index` every occurrence of an acronym is recorded to the index and sorted by its short form or (if set) by the value of the `sort` property. This property allows to set an individual sorting option for the index.

`no-index = true|false` Initial: `true`

This property allows to exclude an acronym from being indexed.

This manual is an example for the indexing feature. Each acronym from section A on page 56 that has been used in this manual is also listed in the index.

17. Barriers

The main purpose of the concept of barriers is to be able to have *local* lists of acronyms. This concept does a little bit more than that, though, which should become clear from the following options:

`barriers/use = true|false` Initial: `false`

Activate usage of barriers. Otherwise the command `\acbarrier` just does nothing except writing a warning in the log.

`barriers/reset = true|false` Initial: `false`

When set to `true` the acronym usage is reset for all acronyms at a barrier. The first use of `\ac` after a barrier will again look like the `\acf`.

`barriers/single = true|false` Initial: `false`

When set to `true` a single usage of an acronym between two barriers with `\ac` will look according to the chosen style as explained in section 9 on page 14. This option only has an effect when the option `single` is used as well.

There are two natural barriers in a document: `\begin{document}` and `\end{document}`. You can add an arbitrary number of additional barriers with

`\acbarrier`

For this command to have any effect you must set `barriers/use` to `true`!

! It usually takes two or even three LaTeX runs until acronym usages between barriers are properly counted.

18. Trailing tokens

18.1. What is it about?

ACRO has the possibility to look ahead for certain tokens and switch a boolean variable if it finds them. Per default ACRO knows about three tokens: the "`dot`" (`.`), the "`dash`" (`-`) and the "`babel-hyphen`" (`\babelhyphen`).

You have seen an example for this already:

```
\DeclareAcronym{etc}{
  short = etc\acdot ,
  long = et cetera ,
  format = \textit ,
  first-style = long ,
  plural =
}
```

The macro `\acdot` recognizes if a dot is directly following. It only prints a dot if it doesn't.

```
\ac{etc} and \ac{etc}.
```

etc. and *etc.*

Another example: let's say you're a German scientist, you have

```
\DeclareAcronym{PU}{
  short = PU ,
  long = Polyurethan ,
  long-plural = e
}
```

and you use it the first time like this:

```
\ac{PU}-Hartschaum
```

then according to German orthography and typesetting rules this should be printed as

"Polyurethan(PU)-Hartschaum"

i. e., with *no* space between long and short form.

```
\acf{PU}-Hartschaum
```

Polyurethan(PU)-Hartschaum

This works because the template `long-short`[7] uses `\acspace` at the appropriate place and the manual setup does

```
\acsetup{trailing/activate = dash}
```

`\acspace` looks ahead for a trailing dash and adds a space it it doesn't find it.

18.2. How does it work?

Tokens to look for can be defined and activated through the following options:

`trailing/define` = ⟨*token*⟩{⟨*name*⟩}
: Defines token ⟨*name*⟩ and tells ACRO look for ⟨*token*⟩ if ⟨*name*⟩ is activated.

`trailing/activate` = {⟨*csv list of token names*⟩}
: Tell ACRO to look for trailing tokens. This is done by giving a csv list of the internal *names* of the tokens. Per default only `dot` is activated.

`trailing/deactivate` = {⟨*csv list of token names*⟩}
: Tell ACRO not to look for trailing tokens. This is done by giving a csv list of the internal *names* of the tokens.

The package itself does this:

```
\acsetup{
  trailing/define   = . {dot} ,
  trailing/define   = {, {comma}} ,
  trailing/define   = - {dash}
  trailing/define   = \babelhyphen {babel-hyphen} ,
  trailing/activate = {dot,comma}
}
```

In order to make use of this mechanism there is the following command:

* `\aciftrailing`{⟨*csv list of token names*⟩}{⟨*true*⟩}{⟨*false*⟩}
: Check if one of the tokens listed in ⟨*csv list of token names*⟩ is following and either place ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

This command is used to define the two commands you already know:

7. The template that is used by default for the first appearance.

* `\acdot`
Inserts `\abbrdot` if no `dot` follows.

* `\acspace`
Inserts a `\space` if no `dash` or `babel-hyphen` follows.

* `\abbrdot`
Inserts `.\@`

The definitions are equivalent to the following code:

```
\newcommand*\acdot{\aciftrailing{dot}{}{\abbrdot}}
\newcommand*\acspace{\aciftrailing{dash,babel-hyphen}{}{\space}}
```

You are of course free to redefine them according to your needs.

19. Using or resetting acronyms

Sometimes it is necessary to mark an acronym as used before it actually has been used or to mark an acronym as unused even though it *has* been used. You have already seen one of the commands which make it possible:

`\acuse{`⟨*csv list of acronym ids*⟩`}`
Every acronym given in the list will be marked as used.

`\acuseall`
Every acronym is marked as used.

`\acreset{`⟨*csv list of acronym ids*⟩`}`
Every acronym given in the list will be reset.

`\acresetall`
Every acronym will be reset.

! In a number of contexts all acronym commands act as if their starred form is used: in the table of contents, in the list of figures, and in the list of tables. The same is true for floats and the measuring phase of common table environments like `tabularx` or `ltxtable`.

20. Bookmarks, backlinks and accessibility support

20.1. Backlinks

When ACRO is used together with the package hyperref [ORT20] then you can make use of the following option:

`make-links` = true|false Initial: `false`

If this is activated then every short or alternative appearance of an acronym will be linked to its description in the list of acronyms.

`link-only-first` = true|false Initial: `false`

☆ New

If this is activated in addition to `make-links` then *only the first* short or alternative appearance of an acronym will be linked to its description in the list of acronyms.

! This will fail miserably together with local lists if an acronym appears in more than one list. This *might* be resolved on day.

20.2. Bookmarks

Since bookmarks (which are created by the hyperref or the bookmark packages [Obe19]) can only contain simple text ACRO simplifies the output of the acronym commands when they appear in a bookmark. Although the output can be modified with a dedicated template-mechanism there is no user interface at the moment. Contact me at `https://github.com/cgnieder/acro/issues` if you need it.

Acronyms have the property `pdfstring`:

`pdfstring` = {⟨*pdfstring*⟩} if unused then equal to `short`

Used as PDF string replacement for the short form in bookmarks when used together with the hyperref [ORT20] or the bookmark package [Obe19].

This is for acronyms like

```
\DeclareAcronym{pdf}{
  short = pdf ,
  long = Portable Document Format ,
  short-format = \scshape ,
  pdfstring = PDF
}
```

where the bookmark would write "pdf" instead of "PDF" if the property where not set.

20.3. PDF comments

Some people like see comments in the PDF when they're hovering with the mouse over the short form of an acronym. This can be achieved.

`pdfcomments/use` = true|false Initial: `false`

This enables the creation of PDF comments.

`pdfcomments/cmd` = {⟨*code*⟩} Initial: `\pdftooltip{#1}{#2}`

Chooses the command for actually creating the comment. You must refer to the printed output in the PDF with #1 and to the comment with #2. The default command \pdftooltip is provided by the package pdfcomment [Kle18]. You must load it in order to use it.

Only acronyms where the corresponding property has been set will get comments:

`pdfcomment = {⟨text⟩}`
Sets a tooltip description for an acronym.

20.4. Accessibility support

ACRO supports the accsupp package [Obe18] when you *also load hyperref*. Then ACRO uses

```
\BeginAccSupp{ method = pdfstringdef ,  ActualText = {PDF} }
  \textsc{pdf}%
\EndAccSupp{}%
```

for an acronym defined like this:

```
\DeclareAcronym{pdf}{
  short = pdf ,
  long = Portable Document Format ,
  short-format = \scshape ,
  pdfstring = PDF ,
  short-acc = PDF
}
```

Without accessibility support when a string like "PDF" is copied from the PDF and pasted you get "pdf". If you don't care about that simply don't load accsupp and ignore this section.

You have a few options to be able to manipulate what ACRO does here but I recommend to stay with the default settings:

`accsupp/use = true|false` Initial: `true`
When this is true and the package accsupp is loaded then accessibility support is used.

`accsupp/options = {⟨text⟩}` (initially empty)
Additional option to be passed to \BeginAccSupp. See the accsupp manual for possible settings.

`accsupp/method = {⟨method⟩}` Initial: `pdfstringdef`
The method used by \BeginAccSupp. See the accsupp manual for possible values.

The "ActualText" that is used by ACRO always defaults to the values of the acronym properties themselves. You can choose these values individually by setting the corresponding properties:

`short-acc = {⟨text⟩}` if unused then equal to `short`
Sets the `ActualText` property as presented by the accsupp package for the short form of the acronym.

`long-acc = {⟨text⟩}` if unused then equal to `long`
Sets the `ActualText` property as presented by the accsupp package for the long form of the acronym.

`alt-acc = {⟨text⟩}` if unused then equal to `alt`
Sets the `ActualText` property as presented by the accsupp package for the alternative short form of the acronym.

`foreign-acc = {⟨text⟩}` if unused then equal to `foreign`
Sets the `ActualText` property as presented by the accsupp package for the foreign form of the acronym.

`extra-acc = {⟨text⟩}` if unused then equal to `extra`
Sets the `ActualText` property as presented by the accsupp package for the extra information of the acronym.

`single-acc = {⟨text⟩}` if unused then equal to `long-acc`
Sets the `ActualText` property as presented by the accsupp package for a single appearance of the acronym.

`list-acc = {⟨text⟩}` if unused then equal to `list`
Sets the `ActualText` property as presented by the accsupp package for the appearance in the list of acronyms.

Extra care has to be taken for plural forms as these can not be picked up automatically right now. You have to explicitly set them for the accessibility support, too:

```
\DeclareAcronym{ufo}{
  short = UFO ,
  long = unidentified flying object ,
  foreign = unbekanntes Flugobjekt ,
  foreign-plural-form = unbekannte Flugobjekte ,
  foreign-acc-plural-form = unbekannte Flugobjekte ,
  foreign-babel = ngerman ,
  long-indefinite = an
}
```

21. Localisation

There are places when ACRO uses text strings which depend on the language of the document. In order to recognize the language from babel of polyglossia and print the strings in the correct language ACRO uses the translations [Nie20].

If the language is detected incorrectly or you want ACRO to use another language than it detects you can use the following option:

Key	English	French	German
`list-name`	Acronyms	Acronymes	Abkürzungen
`page`	p.	p.	S.
`pages`	pp.	pp.	S.
`sequens`	f.	sq.	f.
`sequentes`	ff.	sqq.	ff.
`also`	also	aussi	auch
`or`	or	ou	oder
`and`	and	et	und

Table 1: Available translation keywords.

`language = auto|`⟨*language*⟩ Initial: `auto`

The default setting `auto` lets ACRO detect the language setting automatically. Valid choices are all language names known to the package translations. Mostly just type your language and it should work.

ACRO only provides support for a handful of languages. You can easily teach ACRO your language – see section 27 on page 47 – if it isn't supported, yet.[8]

* `\acrotranslate{`⟨*key*⟩`}`

This command fetches the translation of ⟨*key*⟩ for the current language. It is meant for usage in template definitions.

Available keywords and their English, French, and German translations are shown in table 1.

22. Patches

In several situations it can lead to wrong results if ACRO marks an acronym as used too early or at all. This is why it is possible to disable the mechanism which is responsible:

`\acswitchoff`

This disables the mechanism which marks acronyms as used. After this command every acronym command like `\ac` acts like its starred version.

`\acswitchon`

This command enables the mechanism again.

In certain circumstances ACRO uses these commands itself. For example it is often preferable that acronyms are not counted as used in floats, the table of contents or the lists of figures and tables. This is why ACRO turns the mechanism off in these places.

8. If you like you can always open an issue at `https://github.com/cgnieder/acro/issues` and provide your translations so I can add them to ACRO.

Certain table environments typeset their contents twice for measurement purposes. ACRO tries to disable the usage mechanism during these phases. The same is true for single line captions from the caption package.

All these patches can be turned off:

`patch/floats = true|false` Initial: `true`

En-/disable the `floats` patch.

`patch/lists = true|false` Initial: `true`

En-/disable the `lists` patch for the table of contents, the list of figures and the list of tables.

`patch/tabularx = true|false` Initial: `true`

En-/disable the `tabularx` patch.

`patch/ltxtable = true|false` Initial: `true`

En-/disable the `ltxtable` patch.

`patch/tabu = true|false` Initial: `true`

En-/disable the `tabu` patch.

`patch/caption = true|false` Initial: `true`

En-/disable the `caption` patch.

Part III. Extending ACRO

23. Background

23.1. Templates

One of the core ideas of ACRO version 3.0 is the use of *templates* which manage how different how anything is printed, from the output of `\ac` and friends to the list of acronyms. ACRO uses three types of templates:

acronym These templates can be used to define *acronym commands*, see section 29 on page 48.

list These templates are used by the `\printacronyms` command.

heading These templates only make sense if a *list* template uses `\acroheading`. This command makes use of them.

How these templates are defined, which are available from the start and how they are used is explained in section 24 on the following page.

23.2. Objects

ACRO uses certain kinds of objects in some of its commands. It is possible to defines own such objects:

articles Per default only the "indefinite" article is defined. But it is possible to define and add other articles to ACRO. This is explained in section 13.2 on page 23.

endings Per default only the ending "plural" is defined. But it is possible to define and add other endings to ACRO. This is explained in section 12.2 on page 22.

properties You have already learned about properties. It is possible to define and add further acronym properties to ACRO. This is explained in section 28 on page 48.

translations ACRO uses localisation strings at a number of places. It is possible to change these strings and add further strings. This is explained in section 21 on page 34.

24. Templates

24.1. Pre-defined templates

24.1.1. Acronym templates

`alt`
Display the alternative form of an acronym.

`first`
This is a *pseudo* template which always displays what is set through the option `first-style` or the property `first-style`.

`footnote`
A template for the first appearance where the long form is printed in a footnote.

`long`
Display the long form of an acronym.

`long-short`
A template for the first appearance where the long form is printed and the short form follows in parentheses.

`short`
Display the short form of an acronym.

`short-long`
A template for the first appearance where the short form is printed and the long form follows in parentheses.

`single`
A template which is used when the property `single` has been set *and* the option `single` has been set *and* if the acronym is only used a single time.

`show`
A template which writes all properties of an acronym into the log file.

24.1.2. List templates

`description`
The default list style which places the short form in the item of a `description` environment and adds the all the rest as description of the item.

`lof`
A style which mimicks the list of figures. This style does not support page ranges.

`longtable`
A style that uses a `longtable` environment for building the list. This needs the longtable package [Car19] loaded.

`longtabu`
A style that uses a `longtabu` environment for building the list. This needs the longtable package and the tabu package [Che19] loaded.[9]

`supertabular`
Introduced in version v3.2
A style that uses a `supertabular` environment for building the list. This needs the supertabular package [BJ20] loaded.

`tabular`
A style that uses a `tabular` environment for building the list. Since a `tabular` cannot break across pages this is only suited for short lists.

`toc`
A style which mimicks the table of contents. This style does not support page ranges.

24.1.3. Heading templates

`addchap`
Only defined in a KOMA-Script class and if `\chapter` is defined. Uses `\addchap` for the heading.

`addsec`
Only defined in a KOMA-Script class. Uses `\addsec` for the heading.

`chapter`
Only defined if `\chapter` is defined. Uses `\chapter` for the heading.

`chapter*`
Uses `\chapter*` for the heading.

9. Please note that this package currently is un-maintained and has a number of open bugs. For further information refer to `https://github.com/tabu-issues-for-future-maintainer/tabu`

none
Displays nothing.

section
Uses \section for the heading.

*section**
Uses \section* for the heading.

24.2. Defining new templates

For the definition of templates these commends are available:

\NewAcroTemplate[⟨*type*⟩]{⟨*name*⟩}{⟨*code*⟩}
This defines a template of type ⟨*type*⟩ with the name ⟨*name*⟩ which inserts ⟨*code*⟩ when used. A template of type ⟨*type*⟩ with name ⟨*name*⟩ must not exist. The default type is `acronym`.

\RenewAcroTemplate[⟨*type*⟩]{⟨*name*⟩}{⟨*code*⟩}
This re-defines a template of type ⟨*type*⟩ woth the name ⟨*name*⟩ which inserts ⟨*code*⟩ when used. A template of type ⟨*type*⟩ with name ⟨*name*⟩ must exist. The default type is `acronym`.

\SetupAcroTemplate[⟨*type*⟩]{⟨*name*⟩}{⟨*code*⟩}
Introduced in version v3.2
Adds ⟨*code*⟩ to the beginning of the template ⟨*name*⟩ of type ⟨*type*⟩. The default type is `acronym`.

\SetupNextAcroTemplate[⟨*type*⟩]{⟨*name*⟩}{⟨*code*⟩}
Introduced in version v3.2
Adds ⟨*code*⟩ to the beginning of *the next use* of the template ⟨*name*⟩ of type ⟨*type*⟩. The default type is `acronym`.

⋆ \AcroTemplateType
Within a template this expands to the ⟨*type*⟩ of the current template.

⋆ \AcroTemplateName
Within a template this expands to the ⟨*name*⟩ of the current template.

How to use these commands is best explained by examples of how the existing templates have been defined. The following sections will show several examples for their usage.

24.3. Commands to be used in template definitions

ACRO provides and uses a large number of commands that are meant to be used in template definitions and that often are useless or will raise errors if used outside. Depending on their purpose the commands can be used in different types of templates or only in certain types of templates.

In the descriptions below a ⋆ indicates a fully expandable command when used in an \edef, \write or in \expanded.

A *TF* always refers to a ⟨*true*⟩ and ⟨*false*⟩ branch and indicates that *three* commands exist: one exactly as described, one with only the `T` and the ⟨*true*⟩ branch, and one with only the `F` and the ⟨*false*⟩ branch. So \acroif*TF* means there is \acroifTF, \acroifT, and \acroifF, where \acroifT and \acroifF each have an argument less than \acroifTF.

24.3.1. Commands for common uses

* `\acrolistname`
Expands to whatever is currently set with `list/name`.

`\acrowrite{⟨property⟩}`
Prints the property ⟨*property*⟩ of the current acronym. Depending on the circumstances this prints the property together with an article or an ending either in uppercase or lowercase form. Default is the lowercase form without ending or article. The actual outcome is determined by switches which are explained in section 29 on page 48.

`\acroformat{⟨type⟩}{⟨text⟩}`
This formats ⟨*text*⟩ according to ⟨*type*⟩ where ⟨*type*⟩ has either been set as property or as option from the `format` module. Valid values are `short`, `long`, `alt`, `extra`, `foreign`, `list`, and `first-long`.

`\acroshow{⟨property⟩}`
For debugging puposes: writes the property ⟨*property*⟩ of the current acronym to the log file.

`\acroifTF{⟨property⟩}{⟨true⟩}{⟨false⟩}`
Checks if the property ⟨*property*⟩ has been set for the current acronym and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

`\acroifbooleanTF{⟨property⟩}{⟨true⟩}{⟨false⟩}`
Introduced in version v3.1 (2020/05/03)
Returns ⟨*true*⟩ if the boolean property ⟨*property*⟩ has been set to `true` and ⟨*false*⟩ otherwise.

`\acroifallTF{⟨properties⟩}{⟨true⟩}{⟨false⟩}`
Checks if all properties in the csv list ⟨*properties*⟩ have been set for the current acronym and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

`\acroifanyTF{⟨properties⟩}{⟨true⟩}{⟨false⟩}`
Checks if any of the properties in the csv list ⟨*properties*⟩ has been set for the current acronym and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

`\acroiftagTF{⟨tag⟩}{⟨true⟩}{⟨false⟩}`
Checks if the current acronym has been given the tag ⟨*tag*⟩ and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

`\acroifstarredTF{⟨true⟩}{⟨false⟩}`
☆ New
Checks if the current call of the acronym is a starred command or not and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

`\AcroPropertiesMap{⟨code⟩}`
Maps over all defined acronym properties. Within ⟨*code*⟩ you can refer to the current property with #1.

`\AcroAcronymsMap{⟨code⟩}`
☆ New
Maps over all defined acronyms. Within ⟨*code*⟩ you can refer to the current property with #1 or with `\AcronymID`.

`\AcroMapBreak`

☆ New
Stops the map `\AcroAcronymsMap` and is usually used in combination with a boolean check.

`\AcroPropertiesSet{⟨id⟩}{⟨csv list of properties⟩}`

☆ New
Allows the setting of properties of acronym ⟨*id*⟩ outside of `\declareAcronym`.

24.3.2. Commands for usage in acronym templates

* `\acroifusedTF{⟨true⟩}{⟨false⟩}`
Checks if the current acronym has been used before and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

`\acroiffirstTF{⟨true⟩}{⟨false⟩}`
Checks if the current usage of the current acronym is the first time and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

`\acroifsingleTF{⟨true⟩}{⟨false⟩}`
Checks if the current acronym is used a single time and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

`\acrogroupcite`

24.3.3. Commands for usage in list templates

* `\acroifchapterTF{⟨true⟩}{⟨false⟩}`
This just check if `\chapter` is defined. Used in the *toc* template.

* `\acroifpagesTF{⟨true⟩}{⟨false⟩}`
This is ⟨*true*⟩ if the option `pages/display` is set, *and* the current acronym is not single, *and* has at least one page number. ⟨*false*⟩ otherwise.

`\acropages{⟨first⟩}{⟨range⟩}`
If `\acroifpagesTF` would be ⟨*false*⟩ this would do nothing. Otherwise, if `pages/display` is `first` it prints the first page number, preceded by ⟨*first*⟩ if `pages/name` is true. If `pages/display` is `all` it prints the page range, preceded by ⟨*range*⟩ if `pages/name` is true.

`\acronopagerange`
This disables page ranges. Used in the *toc* and *lof* templates.

`\acroneedpages`
Introduced in version v3.4
This enables the page number displayed. Used in the *toc* and *lof* templates.

`\acropagefill`
If `\acroifpagesTF` would be ⟨*false*⟩ this would do nothing. Otherwise it prints whatever is set by `pages/fill`.

`\acronymsmap{⟨code⟩}`
Maps over the acronyms in order of appearance in the list. Which acronyms these are depends on settings. They might only have certain tags, be ones local to barriers, …

Within ⟨*code*⟩ `#1` refers to the current ID of the acronym. Also `\AcronymID` expands to the current ID. The latter is important for all the commands that check or print properties of acronyms.

`\acronymsmapTF{⟨code⟩}{⟨true⟩}{⟨false⟩}`
This does the same as `\acronymsmap` and also leaves ⟨*true*⟩ in the input stream if the list is not empty and ⟨*false*⟩ otherwise. This is useful to trigger a rerun warning.

`\AcronymTable`
This is an empty token list at the beginning of a list template.

`\AcroAddRow{⟨code⟩}`
Adds ⟨*code*⟩ to the right of `\AcronymTable` and ensures that `\AcronymID` has the correct global definition for this code. With this the code for the *`tabular`* template and other table templates can be built in a comfortable way.

`\AcroNeedPackage{⟨package⟩}`
Checks if the package ⟨*package*⟩ is loaded and throws an error otherwise.

`\AcroRerun`
Triggers ACRO to throw an "empty list" rerun warning.

24.4. New acronym templates

Some templates are quite short and self-explaining:

```
\NewAcroTemplate{short}{\acrowrite{short}}
```

Some are a little bit more elaborate:

```
\NewAcroTemplate{alt}{%
  \acroifTF{alt}
    {\acrowrite{alt}}
    {\acrowrite{short}}%
}
```

And some templates need to do a lot more:

```
\NewAcroTemplate{long-short}{%
  \acroiffirstTF{%
    \acrowrite{long}%
    \acspace(%
      \acroifT{foreign}{\acrowrite{foreign}, }%
      \acrowrite{short}%
      \acroifT{alt}{ \acrotranslate{or} \acrowrite{alt}}%
      \acrogroupcite
    )%
  }%
  {\acrowrite{short}}%
}
```

24.5. New list templates

This section shows the definition of three templates: *description*, *tabular*, and *toc*.

First the *description* template:

```
\NewAcroTemplate[list]{description}{%
  \acroheading
  \acropreamble
  \begin{description}
    \acronymsmapF{%
      \item[\acrowrite{short}\acroifT{alt}{/\acrowrite{alt}}]
        \acrowrite{list}%
        \acroifanyT{foreign,extra}{ (}%
        \acroifT{foreign}{\acrowrite{foreign}\acroifT{extra}{, }}%
        \acroifT{extra}{\acrowrite{extra}}%
        \acroifanyT{foreign,extra}{)}%
        \acropagefill
        \acropages
          {\acrotranslate{page}\nobreakspace}
          {\acrotranslate{pages}\nobreakspace}%
      }
      {\item\AcroRerun}
  \end{description}
}
```

The following shows how to define templates using some kind of table environment. Special care is necessary due to the way LaTeX tables work: first the table body is built and only then the table itself is printed:

```
\NewAcroTemplate[list]{tabular}{%
  \AcroNeedPackage{array}%
  \acronymsmapF{%
    \AcroAddRow{
      \acrowrite{short}%
      \acroifT{alt}{/\acrowrite{alt}}
      &
      \acrowrite{list}%
      \acroifanyT{foreign,extra}{ (}%
      \acroifT{foreign}{\acrowrite{foreign}\acroifT{extra}{, }}%
      \acroifT{extra}{\acrowrite{extra}}%
      \acroifanyT{foreign,extra}{)}%
      \acropagefill
      \acropages
        {\acrotranslate{page}\nobreakspace}
        {\acrotranslate{pages}\nobreakspace}%
      \tabularnewline
    }%
  }
  {\AcroRerun}%
  \acroheading
  \acropreamble
  \par\noindent
  \begin{tabular}{>{\bfseries}lp{.7\linewidth}}
    \AcronymTable
  \end{tabular}
}
```

```
\NewAcroTemplate[list]{toc}{%
  \acroheading
  \acropreamble
  \acronopagerange
  \acronymsmapF{%
    \contentsline{\acroifchapterTF{chapter}{section}}
      {\acrowrite{short}\acroifT{alt}{/\acrowrite{alt}}}
      {}{}%
    \contentsline{\acroifchapterF{sub}section}
      {
        \acrowrite{list}%
        \acroifT{foreign}{\acrowrite{foreign}\acroifT{extra}{, }}%
        \acroifT{extra}{\acrowrite{extra}}%
        \acroifanyT{foreign,extra}{)}%
      }
      {\acropages{}{}}
      {}%
  }
  {\AcroRerun}
}
```

24.6. New heading templates

Let's take a look at the two templates *section* and *section** which should give you enough information to build your own:

```
\NewAcroTemplate[heading]{section} {\section {\acrolistname}}
\NewAcroTemplate[heading]{section*}{\section*{\acrolistname}}
```

25. Endings

Referring to section 12.2 on page 22 this section explains how to define and use additional endings.

`\DeclareAcroEnding{⟨name⟩}{⟨short default⟩}{⟨long default⟩}`

This command can be used to define properties and options analoguous to the plural endings which have been defined this way:

```
\DeclareAcroEnding{plural}{s}{s}
```

In general `\DeclareAcroEnding{⟨foo⟩}{⟨x⟩}{⟨y⟩}` defines these options

`short-⟨foo⟩-ending = {⟨value⟩}`	Initial: ⟨x⟩
`long-⟨foo⟩-ending = {⟨value⟩}`	Initial: ⟨y⟩

and these properties

`short-⟨foo⟩ = {⟨value⟩}`	Initial: ⟨x⟩
`short-⟨foo⟩-form = {⟨value⟩}`	(initially empty)
`alt-⟨foo⟩ = {⟨value⟩}`	Initial: ⟨x⟩
`alt-⟨foo⟩-form = {⟨value⟩}`	(initially empty)
`long-⟨foo⟩ = {⟨value⟩}`	Initial: ⟨y⟩
`long-⟨foo⟩-form = {⟨value⟩}`	(initially empty)
`foreign-⟨foo⟩ = {⟨value⟩}`	Initial: ⟨y⟩
`foreign-⟨foo⟩-form = {⟨value⟩}`	(initially empty)
`single-⟨foo⟩ = {⟨value⟩}`	Initial: ⟨y⟩
`single-⟨foo⟩-form = {⟨value⟩}`	(initially empty)

`extra-`⟨*foo*⟩ `= {`⟨*value*⟩`}` Initial: ⟨*y*⟩

`extra-`⟨*foo*⟩`-form = {`⟨*value*⟩`}` (initially empty)

In addition another command is defined which is meant to be used in template definitions.

`\acro`⟨*foo*⟩

This command tells the template that the ending ⟨*foo*⟩ should be used.

Section 29 on page 48 has an example of how this can be used to define a possessive ending and commands that make use of them like this:

```
\acfg{MP}
```

Member's of Parliament (MP's)

26. Articles

Referring to section 13.2 on page 23 this section explains how to define and use additional articles.

`\DeclareAcroArticle{`⟨*name*⟩`}{`⟨*default*⟩`}`

This command can be used to define properties and options analoguous to the indefinite article which have been defined this way:

```
\DeclareAcroArticle{indefinite}{a}
```

In general `\DeclareAcroArticle{`⟨*foo*⟩`}{`⟨*x*⟩`}` defines the option

⟨*foo*⟩ `= {`⟨*value*⟩`}` Initial: ⟨*x*⟩

and these properties

`short-`⟨*foo*⟩ `= {`⟨*value*⟩`}` Initial: ⟨*x*⟩

`alt-`⟨*foo*⟩ `= {`⟨*value*⟩`}` Initial: ⟨*x*⟩

`long-`⟨*foo*⟩ `= {`⟨*value*⟩`}` Initial: ⟨*x*⟩

`foreign-`⟨*foo*⟩ `= {`⟨*value*⟩`}` Initial: ⟨*x*⟩

`single-`⟨*foo*⟩ `= {`⟨*value*⟩`}` Initial: ⟨*x*⟩

`extra-`⟨*foo*⟩ `= {`⟨*value*⟩`}` Initial: ⟨*x*⟩

In addition another command is defined which is meant to be used in template definitions.

`\acro`⟨*foo*⟩

This command tells the template that the article ⟨*foo*⟩ should be used.

Section 29 on page 48 has examples of how this can be used to define definite articles and commands that make use of them like this:

```latex
\dacs{hadopi} \par
\dacl{hadopi}
```

l'HADOPI
la Haute Autorité pour la diffusion des œuvres et la protection des droits sur l'Internet

27. Translations

For adding additional keywords, or for adding translations to existing keywords, or for changing existing translations ACRO uses this command:

`\DeclareAcroTranslation{⟨key⟩}{⟨language=translation list⟩}`
With this command new translations keywords can be added and translations for existing keywords can be changed.

`\AddAcroTranslations{⟨key⟩}{⟨language=translation list⟩}`
Basically the same but this time per language rather than per keyword.

As an example this is how ACRO declares translations for the `pages` keyword:

```latex
\DeclareAcroTranslation{pages}{
  Fallback   = pp\abbrdot ,
  English    = pp\abbrdot ,
  French     = pp\abbrdot ,
  German     = S\abbrdot ,
  Portuguese = pp\abbrdot
}
```

Translations for a language could be added this way[10]:

```latex
\AddAcroTranslations{Italian}{
  list-name = Acronimi ,
  page      = p\abbrdot ,
  pages     = pp\abbrdot ,
  sequens   = s\abbrdot ,
  sequentes = ss\abbrdot ,
  also      = anche ,
  and       = e ,
  or        = o
}
```

The existing keywords had been shown in table 1 on page 35.

10. ACRO already has the translations for Italian.

28. Properties

As you know from section 5 ACRO comes with quite a number of predefined properties which control various aspects of acronyms. However, there are cases when additional properties would be nice to have and to use. It can be done with the following command:

`\DeclareAcroProperty*?!|>{⟨name⟩}`

This defines the new property ⟨*name*⟩. The command has five optional arguments most of which you probably never need.

The optional star `*` ensures that each acronym gets a *unique* value for the property.

The optional question mark `?` creates a *boolean* property. That is a property that only can get the values `true` or `false` and when it is used without value (not an empty value!) then `true` is assumed.

The optional exclamation mark `!` creates a *mandatory* property. An error is raised if an acronym does not set it.

The optional pipe `|` creates a *static* property which means its value is written to an auxiliary file and read in again at begin document. Once set the value is the same throughout the document.

Introduced in version v3.2

The optional greater as symbol `>` creates a *display* property. This additionally defines the two boolean options ⟨*name*⟩`/display` and `list/`⟨*name*⟩`/display`, both initially set to true. If these options are set to false the acronym commands or the list act as if the property ⟨*name*⟩ has not been set. The `foreign` property is an example.

`\DeclareAcroPropertyAlias*?!|>{⟨name1⟩}{⟨name2⟩}`

This newly declares property ⟨*name1*⟩ and makes it an alias of property ⟨*name2*⟩. This means that ⟨*name1*⟩ gets the same value that ⟨*name2*⟩ has unless it is set explicitly. Property ⟨*name2*⟩ must exist.

`\MakeAcroPropertyAlias{⟨name1⟩}{⟨name2⟩}`

This makes property ⟨*name1*⟩ and makes it an alias of property ⟨*name2*⟩. Both properties must exist.

Exmaples for defining and using new properties are shown in section A, for example, examples 8 or 9.

29. Own acronym commands

29.1. Background

You can define own acronym commands or redefine the existing ones with commands similar to `\NewDocumentCommand` from the xparse package [L3P].

`\NewAcroCommand{⟨command⟩}{⟨arg. spec.⟩}{⟨code⟩}`

This creates the new command ⟨*command*⟩ with the argument specification[11] `so`⟨*arg. spec.*⟩ and

11. in the sense of an xparse command.

replacement text ⟨*code*⟩. There are significant differences to `\NewDocumentCommand`: the new command always has two additional arguments: an optional star and an optional argument for options. You can ignore this fact in your definition, though. However, the command *must* at least have one argument *and* the first argument *must* refer to the ID. Everything else is up to you.

The new command has the suiting framework to recognize trailing tokens, count usage, index, and add a citation if necessary.

`\RenewAcroCommand{`⟨*command*⟩`}{`⟨*arg. spec.*⟩`}{`⟨*code*⟩`}`
Like `\NewAcroCommand` but redefines an existing command.

`\UseAcroTemplate[`⟨*type*⟩`]{`⟨*name*⟩`}[`⟨*argument number*⟩`]`⟨*arguments*⟩
The argument ⟨*type*⟩ defaults to `acronym` and ⟨*argument number*⟩ defaults to `1`. The command must be followed by as many mandatory arguments as you specify with ⟨*argument number*⟩. All predefined acronym templates use the first argument as ID so they must use one argument.

Let's see an example. This is the definition of `\ac`:

```
\NewAcroCommand\ac{m}{\UseAcroTemplate{first}{#1}}
```

Equivalent definitions would be:

```
\NewAcroCommand\ac{m}{\UseAcroTemplate[acroynm]{first}{#1}}
\NewAcroCommand\ac{m}{\UseAcroTemplate[acroynm]{first}[1]{#1}}
\NewAcroCommand\ac{m}{\UseAcroTemplate{first}[1]{#1}}
\NewAcroCommand\ac{m}{\UseAcroTemplate{first}[2]{#1}{}}
```

There are a number of switch commands which determine a certain behavior. They tell the following template how to interpret certain conditionals and how to use `\acrowrite`.

`\acrocite`
Tells ACRO to output the citation.

`\acrodonotuse`
Tells ACRO to not count this as usage.

`\acroplural`
Use plural form.

`\acroindefinite`
Use indefinite article

`\acroupper`
Use uppercase form.

`\acrofull`
Use first or full form.

Here is an example that makes use of them:

```
\NewAcroCommand\Iacs{m}{%
  \acroupper\acroindefinite\UseAcroTemplate{short}{#1}%
}
```

29.2. Create commands for possessive endings

Let's say you want to add an ending for the genitive case. First you define the appropriate ending:

```
\DeclareAcroEnding{possessive}{'s}{'s}
```

Then you define commands which make use of this ending:

```
\NewAcroCommand\acg{m}{\acropossessive\UseAcroTemplate{first}{#1}}
\NewAcroCommand\acsg{m}{\acropossessive\UseAcroTemplate{short}{#1}}
\NewAcroCommand\aclg{m}{\acropossessive\UseAcroTemplate{long}{#1}}
\NewAcroCommand\acfg{m}{%
  \acrofull
  \acropossessive
  \UseAcroTemplate{first}{#1}%
}
\NewAcroCommand\iacsg{m}{%
  \acroindefinite
  \acropossessive
  \UseAcroTemplate{short}{#1}%
}
```

You maybe also define acronyms with corresponding properties[12]:

```
\DeclareAcronym{MP}{
  short = MP ,
  long  = Member of Parliament ,
  plural-form = Members of Parliament ,
  long-possessive-form = Member's of Parliament
}
```

12. Bear with me if this is incorrect: English is not my native language.

Now you can use it like this:

```
This is the \acg{MP} first day at work after \dots
```

This is the MP's first day at work after …

30. Own ACRO style files

When you want to use your definitions regarding ACRO repeatedly then it makes sense to put them in a file which you put somewhere in your local LaTeX tree. There are three options:

1. Put them in a simple `.tex` file in `\input` it.
2. Put in in a `.sty` file and include it with `\usepackage` *after* ACRO.
3. Create a style file following this pattern decribed below.

`acro.style.`⟨*name*⟩`.code.tex`

This file should start with

```
\AcroStyle{name}
```

and input the file with `\acsetup` using the option

`load-style = {`⟨*name*⟩`}`
This is more or less the same as if you'd use the package variant but naturally ensures that you load it after ACRO and in the future might provide other bells and whistles, too.

The command

`\AcroStyle*{`⟨*style*⟩`}[`⟨*details*⟩`]`
has an optional star which switches to expl3 syntax. It also has an optional argument ⟨*details*⟩ with the same purpose and usage as the one from `\ProvidesPackage`. A typical usage would look like

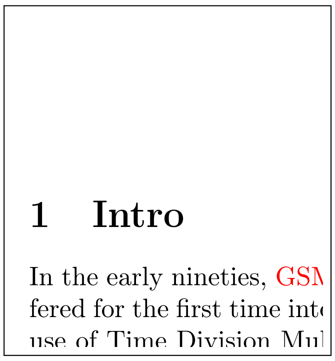
```
\AcroStyle{abbrev}[2020/04/21 abbreviations with acro (CN)]
\NewDocumentCommand\newabbreviation{mmm}{%
  \DeclareAcronym{#1}{ short = #2 , #3 , class = abbrev , no-index }%
}
\NewDocumentCommand\printabbreviations{O{}}{%
  \printacronyms[#1,include=abbrev]%
}
```

Part IV.
Appendix

A. Examples

Example 1: Basic usage
Links: [TEX] [PDF] File: acro.example.basic.tex

```

\acsetup{
  make-links ,
  pages / display = first ,
  pages / fill    = {, }
}

\DeclareAcronym{CDMA}{
```

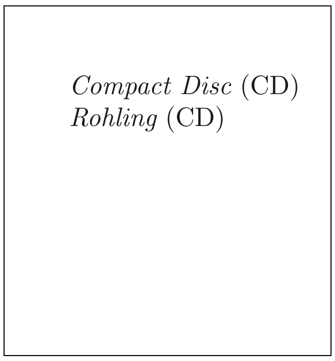


1 Intro

In the early nineties, GSM
fered for the first time int
use of Time Division Mul

Example 2: Re-implement \acflike
Links: [TEX] [PDF] File: acro.example.acflike.tex

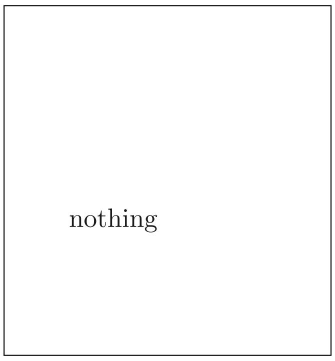
```
\RenewAcroCommand\acflike{mm}{%
  \acroformat{long}{#2} (\UseAcroTemplate{short
 }{#1})%
}

\begin{document}

\ac{cd} \par
\acflike{cd}{Rohling}
```



Compact Disc (CD)
Rohling (CD)

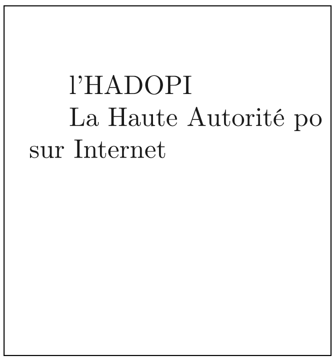
Example 3: Invisible command for backref
Links: [TeX] [PDF] [github] File: acro.example.issue-109.tex

```
  pages/seq=false
}

\DeclareAcronym{ny}{
  short = NY ,
  long = New York ,
}
```


nothing

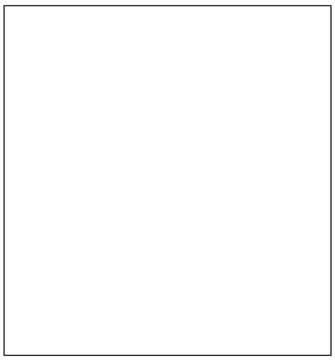
Example 4: Defining a definite article
Links: [TeX] [PDF] [github] File: acro.example.issue-111.tex

```
\NewAcroCommand\dacs{m}{\acrodefinite\UseAcroTemplate{short}{#1}}
\NewAcroCommand\Dacs{m}{\acroupper\acrodefinite\UseAcroTemplate{short}{#1}}
\NewAcroCommand\dacl{m}{\acrodefinite\UseAcroTemplate{long}{#1}}
\NewAcroCommand\Dacl{m}{\acroupper\acrodefinite\UseAcroTemplate{long}{#1}}

\DeclareAcronym{hadopi}{
  short = HADOPI ,
  long = Haute Autorit\'{e} pour la diffusion des \oe uvres et la protection des
```


l’HADOPI
La Haute Autorité po
sur Internet

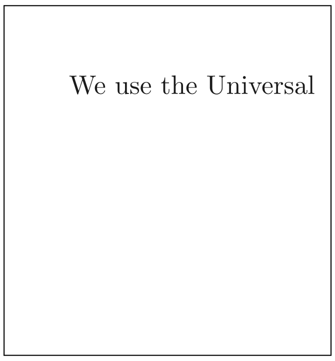
Example 5: Write the list of acronyms to an external file
Links: [TeX] [PDF] [github] File: acro.example.issue-119.tex

```
  \if@filesw
    \newwrite\acro@list
    \immediate\openout\acro@list\jobname.ac\relax
    \immediate\write\acro@list{\string\begin{description}}
    \let\item\relax
    \acronymsmapF{%
      \immediate\write\acro@list{%
        \space\space
```



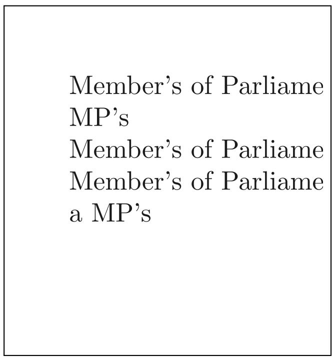
Example 6: Insert word between acronym and citation
Links: [TeX] [PDF] [github] File: `acro.example.issue-154.tex`

```
  short = UC ,
  long = Universal Composability ,
  cite = xxx
}

\begin{document}

We use the \ac{uc}[ model] for\dots
```


We use the Universal

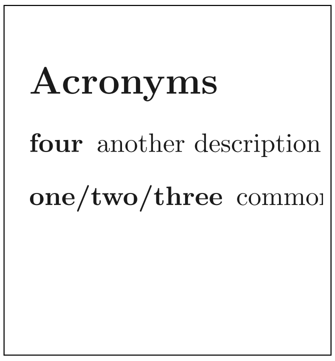
Example 7: How to define a possessive ending
Links: [TeX] [PDF] File: `acro.example.possessive.tex`

```
\NewAcroCommand\acsg{m}{\acropossessive\
  UseAcroTemplate{short}{#1}}
\NewAcroCommand\aclg{m}{\acropossessive\
  UseAcroTemplate{long}{#1}}
\NewAcroCommand\acfg{m}{%
  \acrofull
  \acropossessive
  \UseAcroTemplate{first}{#1}%
}
\NewAcroCommand\iacsg{m}{%
```


Member's of Parliame
MP's
Member's of Parliame
Member's of Parliame
a MP's

Example 8: Additional alternative form
Links: [TeX] [PDF] [forum] File: `acro.example.texsx-505891.tex`

```
\RenewAcroTemplate[list]{description}{%
  \acroheading
  \acropreamble
  \begin{description}
    \acronymsmapF{
      \item [%
          \acrowrite{short}%
          \acroifT{alt}{/}\acrowrite{alt}%
```

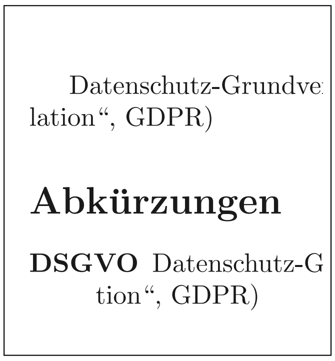

Acronyms

four another description

one/two/three commo

Example 9: Foreign short form
Links: [TEX] [PDF] [forum]   File: `acro.example.texsx-507726.tex`

```

\RenewAcroTemplate{long-short}{%
  \acroiffirstTF{%
    \acrowrite{long}\acspace
    (%
      \acrowrite{short}%
      \acroifT{foreign}{, }%
      \acrowrite{foreign}%
```

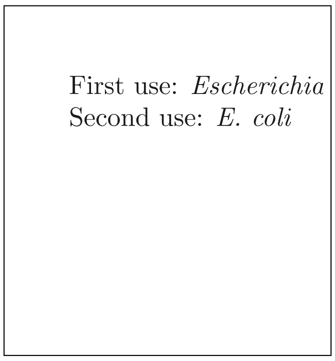

Datenschutz-Grundve
lation“, GDPR)

Abkürzungen

DSGVO Datenschutz-G
tion“, GDPR)

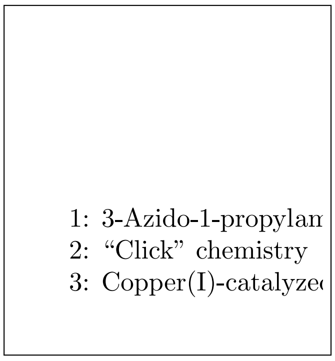
Example 10: Species
Links: [TEX] [PDF] [forum]   File: `acro.example.texsx-513623.tex`

```
    #1 ,
    tag = species ,
    first-style= long ,
    format = \itshape
  }%
}
\newspecies{ecoli}{E.~coli}{Escherichia coli}
```


First use: *Escherichia*
Second use: *E. coli*

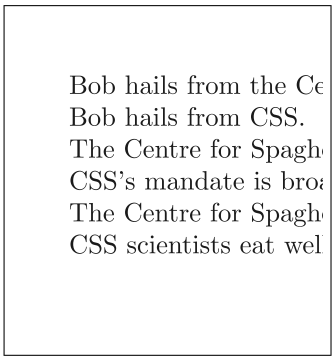
Example 11: Capitalization
Links: [TEX] [PDF] [forum]   File: `acro.example.texsx-515295.tex`

```
  short = 3-APA ,
  long = \iupac{3-azido-1-propyl|amine}
}
\DeclareAcronym{CuAAC}{
  short = \enquote{click} chemistry ,
  long = copper(I)-catalyzed azide-alkyne
  cycloaddition
}
```


1: 3-Azido-1-propylan
2: “Click” chemistry
3: Copper(I)-catalyzed

Example 12: Articles and possessive forms
Links: [TEX] [PDF] [forum] File: `acro.example.texsx-542461.tex`

```
\NewAcroCommand\Dac{m}{\acroupper\acrodefinite\
  UseAcroTemplate{first}{#1}}

\NewAcroCommand\Dacg{m}{%
  \acroupper
  \acrodefinite
  \acropossessive
  \UseAcroTemplate{first}{#1}%
}
```


Bob hails from the Ce
Bob hails from CSS.
The Centre for Spagh
CSS's mandate is broa
The Centre for Spagh
CSS scientists eat wel

B. Acronyms

Below all abbreviations are listed which have been defined for the manual.

CD compact disc 4, 13

CTAN Comprehensive TEX Archive Network

e.g. for example (*Latin*: *exempli gratia*)

ECU Steuergerät (*Englisch*: Electronic Control Unit) 24

etc. *et cetera* 14, 29

HADOPI Haute Autorité pour la diffusion des œuvres et la protection des droits sur l'Internet 46

ID identification string . . . . 6, 42, 48 f.

JPEG/JPG Joint Photographic Experts Group 12

LA Los Angeles

LPPL LATEX Project Public License

MP Member of Parliament 22, 46, 50

NATO Organisation des Nordatlantikvertrags (*Englisch*: North Atlantic Treaty Organization)

NY New York 26

PDF Portable Document Format . . . . 9, 20, 32 f.

PNG Portable Network Graphics

PU Polyurethan

TEX.sx TEX StackExchange

UFO unidentified flying object (*German*: unbekanntes Flugobjekt) 12, 22, 25

C. References

[BJ20] Johannes Braams and Theo Jurriens.
supertabular. version 4.1g, Feb. 2, 2020 (or newer).
URL: `https://www.ctan.org/pkg/supertabular/`.

[Bra19] Johannes Braams, current maintainer: Javier Bezos.
babel. version 3.33, July 19, 2019 (or newer).
URL: `https://www.ctan.org/pkg/babel/`.

[Car19] David Carlisle. longtable. version 4.12, Feb. 6, 2019 (or newer).
URL: `https://www.ctan.org/pkg/longtable/`.

[Cha19] François Charette, current maintainer: Arthur Reutenauer.
polyglossia. version 1.44, Apr. 4, 2019 (or newer).
URL: `https://www.ctan.org/pkg/polyglossia/`.

[Che19] Florent Chervet. tabu. version 2.8+, Jan. 12, 2019 (or newer).
URL: `https://www.ctan.org/pkg/tabu/`.

[Kle18] Josef Kleber. pdfcomment. version 2.4a, Nov. 1, 2018 (or newer).
URL: `https://www.ctan.org/pkg/pdfcomment/`.

[L3P] The LaTeX3 Project Team. xparse. Mar. 6, 2020 (or newer).
URL: `https://www.ctan.org/pkg/xparse/`.

[LKW19] Philipp Lehman, Philip Kime, and Moritz Wemheuer.
biblatex. version 3.14, Dec. 1, 2019 (or newer).
URL: `https://www.ctan.org/pkg/biblatex/`.

[Nie20] Clemens Niederberger. translations. version 1.8, Feb. 28, 2020 (or newer).
URL: `https://www.ctan.org/pkg/translations/`.

[Obe18] Heiko Oberdiek. accsupp. version 0.5, Mar. 28, 2018 (or newer).
URL: `https://www.ctan.org/pkg/oberdiek/`.

[Obe19] Heiko Oberdiek. bookmark. version 1.28, Dec. 3, 2019 (or newer).
URL: `https://www.ctan.org/pkg/bookmark/`.

[ORT20] Heiko Oberdiek, Sebastian Rahtz, and The LaTeX3 Project Team.
hyperref. version 7.00d, Jan. 14, 2020 (or newer).
URL: `https://www.ctan.org/pkg/hyperref/`.

[Tal17] Nicola L.C. Talbot. mfirstuc. version 2.06, Nov. 14, 2017 (or newer).
URL: `https://www.ctan.org/pkg/glossaries/`.

[Wik20] Wikipedia. *New York City*. 2020.
URL: `http://en.wikipedia.org/wiki/New_York_City` (visited on 04/11/2020).

D. Index