The cocotex.dtx Package

A modular package suite for automatic, flexible typesetting

Version 0.4.0

(2024/01/29)

Lupino

lupino@le-tex.de

Table of contents

Introduction

1 Basic concepts

The core concept of the CoCoTEX Framework is the strict separation between document specific information bearing units and publisher specific layout and rendering instructions to a degree that is far more versatile and delicate than LATEX's usual distinction between form and content.

The basic data type in the Framework is the **Container**. On the end-user level, this is virtually always a LATEX environment that contain a specific set of macros used to store the atomic units of information. Those macros and their contents are called **Components**.

The instructions on how those Components are to be processed and ultimately rendered are called **Properties**.

2 Flow of macro definitions and their expansions in modules that use the Property and Component mechanism

WARNING!
The following section is deprecated and will be changed or deleted in future releases.

Modules, that utilize the Property and Component mechanisms, define a *Declare macro*. This Declare macro is basicly a constructor for a new LATEX environment which should share some common *Properties* and *Components* with other environments that are defined with the same Declare macro. Modules, therefore, constitute what in other programming languages may be referred to as *Namespaces*.

The purpose of the Declare macro is

1. to define a LATEX environment to be used in tex documents,
2. to define the Component macros available and allowed within that environment
3. to define the available Properties used to determine the appearance of the environment's content in the final render.
4. to define the processing of the information specific to each instance of the environment.

Within the body of the Declare macro's definition, a Use macro is defined which determines the Namespace-specific processing of an environment's contents. This macro is (usually) expanded at the **\end** of the declared environment. The Use macro is where the actual processing of an environment's contents takes place. Since it is part of the body of the Declare macro, each environment declared with this Declare macro defines it's own Use macro.

The Declare macro usually has at least two arguments: one argument to give a *name* to the soon-to-be-defined environment, and a second one to define the Properties *specific* to that environment *on top of* the Namespace's default Properties. Some environments may also have a Parent which causes Properties cascade across different inter-dependend environments.

Within the tex-document, whenever an environment is used, the flow is as follows:

1. *store* the contents of all Components used within the environment in internal, locally defined, tex macros
2. expand the property lists:

   (a) expand the Default Properties of the Namespace
   (b) If necessary, expand the specific Properties of the parent environment (overwriting the default properties of the same name). This step may occur recursively for each of the parent's own parents.
   (c) expand the Specific Properties of the Environment itself.
3. Expand the Use-Macro
   (a) Process the components, depending on contents, presence, or absence of Components alter other Compo-nentsor trigger property manipulatons, etc.
   (b) Calculate the final states of variable properties (in dependency on the available components, other properties or global parameters)
   (c) Print the overall result of those calculations.

One more driver function

```
%<*driver>
```

If we want to run the splitted development dtx locally, this macro prevents undefined control sequence errors and actually includes the dtx chunks.

```
\def\includeDTX#1{\input src/#1.dtx}
```

End driver function

```
%</driver>
```

Modul 1

cocotex.dtx

This is the main class file for the CoCoTeX LaTeX package.

```
%<*class>
```

File Preamble

```
%%
%% Common document class for \textit{xerif} projects.
%%
%% Maintainer: p.schulz@le-tex.de
%%
%% lualatex - texlive > 2019
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesClass{cocotex}
   [2024/01/29 0.4.0 cocotex]
```

Hard-coded requirements

```
\RequirePackage{kvoptions-patch}
\RequirePackage{xkeyval}
```

Passing options down to the LaTeX standard packages

```
\DeclareOptionX{main}{\PassOptionsToPackage{\CurrentOption}{babel}}
\DeclareOption{es-noindentfirst}{\PassOptionsToPackage{es-noindentfirst}{babel}}
\DeclareOption{es-noshorthands}{\PassOptionsToPackage{es-noshorthands}{babel}}
\PassOptionsToPackage{shorthands=off}{babel}
```

The option `pubtype` (short for “publication type”) has possible four values: `mono`, `collection`, `journal`, and `article` . `mono` (also the default when no `pubtype` is given) and `collection` are used to switch between single and multiple contributor documents; `collection` and `journal` to switch between one-time text collections and periodicals, respectively. All three types implicitly load the LaTeX standard class `book`.

`collection` is used when the document’s components (i. e., chapters) are contributed by different authors like collections or proceedings. `journal` is used for collections where each contribution is accompanied by a myriad of meta data. `mono` stands for monographs, i.e., whole books that are written by the same author(s).

The publicaten type `article` is intended for single articles of a journal. It loads the LaTeX standard class `article`.

```
\newif\ifcollection \collectionfalse
\newif\ifarticle \articlefalse
\newif\ifmonograph \monographfalse
\newif\ifjournal \journalfalse
\define@choicekey{cocotex.cls}{pubtype}[\tp@pubtype\nr]{collection,article,journal,mono}{%
  \ifcase\nr\relax% collection
    \global\collectiontrue
  \or% article
    \global\articletrue
  \or% journal
```

```
    \global\journaltrue
  \else% monograph
    \global\monographtrue
  \fi
}
\DeclareOptionX*{\PassOptionsToClass{\CurrentOption}{article}}
\DeclareOptionX*{\PassOptionsToClass{\CurrentOption}{book}}
```

Passing options down to various CoCoTeX modules:

```
\DeclareOptionX{debug}{\PassOptionsToPackage{\CurrentOption}{coco-kernel}}
\DeclareOptionX{a11y}{\PassOptionsToPackage{\CurrentOption}{coco-common}}
\DeclareOptionX{no-compress}{\let\tp@no@pdf@compression\relax}
\DeclareOptionX{color-enc}{\PassOptionsToPackage{\CurrentOption}{coco-common}}
\DeclareOptionX{usescript}{\PassOptionsToPackage{\CurrentOption}{coco-script}}
\DeclareOptionX{nofigs}{\PassOptionsToPackage{\CurrentOption}{coco-floats}}
\DeclareOptionX{ennotoc}{\PassOptionsToPackage{\CurrentOption}{coco-notes}}
\DeclareOptionX{endnotes}{\PassOptionsToPackage{\CurrentOption}{coco-notes}}
\DeclareOptionX{resetnotesperchapter}{\PassOptionsToPackage{\CurrentOption}{coco-notes}}
\DeclareOptionX{endnotesperchapter}{\PassOptionsToPackage{\CurrentOption}{coco-notes}}
\ProcessOptionsX
```

Disables PDF compression when the `no-compress` document option is set.

```
\ifx\tp@no@pdf@compression\relax
  \AtBeginDocument{%
    \ifx\pdfobjcompresslevel\@undefined
      \edef\pdfobjcompresslevel{\pdfvariable objcompresslevel}%
    \fi
    \pdfcompresslevel=0
    \pdfobjcompresslevel=0
  }%
\fi
```

All publication types supported by CoCoTeX are based on one of LaTeX's default classes `book` or `article`:

```
\RequirePackage{coco-common}
\ifarticle
  \LoadClass[10pt,a4paper]{article}
\else
  \LoadClass[10pt,a4paper]{book}
\fi
```

Offsets are the removed to make all values relative to the upper left corner of the page to ease maintainance.

```
\voffset-1in\relax
\hoffset-1in\relax
```

Typesetting automata need some room to play

```
\emergencystretch=2em
```

and strong restrictions:

```
\frenchspacing
\clubpenalty10000
\widowpenalty10000
```

page style without any headers or footers

```
\def\ps@empty{%
  \let\@oddhead\@empty
  \let\@evenhead\@empty
  \let\@oddfoot\@empty
  \let\@evenfoot\@empty
}
```

vacancy pages need to have page style `empty`:

```
\def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
    \hbox{}\thispagestyle{empty}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
```

re-defined to make front- and backmatter components distinguish-able

```
\ifarticle\else
  \newif\if@frontmatter \@frontmatterfalse
  \renewcommand\frontmatter{%
    \cleardoublepage
    \@mainmatterfalse
    \@frontmattertrue
    \pagenumbering{arabic}}
  \renewcommand\mainmatter{%
    \cleardoublepage
    \@frontmatterfalse
    \@mainmattertrue}
  \renewcommand\backmatter{%
    \cleardoublepage
    \@mainmatterfalse
    \@frontmatterfalse}
\fi
\usepackage{soul}
```

Inclusion of the script module which also loads the babel package

```
\ifLuaTeX
\RequirePackage{coco-script}
\else
\RequirePackage{babel}
\fi
\RequirePackage{coco-headings}
```

Inclusion of the float module

```
\RequirePackage{coco-floats}
```

Inclusion of the title page module

```
\RequirePackage{coco-title}
```

Inclusion of the end-/footnotes module

```
\RequirePackage{coco-notes}
```

Fallback, in case, `coco-headings.sty` is not loaded for some reason.

Some more hard dependencies:

```
\RequirePackage{index}
\makeindex
\RequirePackage{hyperref}
```

Finally, some hyperref settings (TODO: check, which of those are better placed inside the local publisher's styles)

```
  \hypersetup{%
```

first, we want links to be breakable

```
    breaklinks%
```

and the table of contents not to be automatically linked, as this causes problems with the ltpdfa package and we add the links via the coco-common module, anyways.

```
   ,linktoc=none%
```

pdf broders are controlled via the coco-frame module, if necessary

```
   ,pdfborder={0 0 0}%
```

The next option causes hyperref to calculate the encoding of DocumentInfo and other direct-to-PDF data (bookmarks, etc.) automatically

```
   ,pdfencoding=auto%
```

Bookmarks are numbered by default.

```
   ,bookmarksnumbered=true%
}
```

Since ltpdfa messes with a lot of LaTeX Kernel macros (like `\begin` and `\end`) as well as external packages (hyperref), it must be loaded last:

```
\ifx\tp@do@ally\relax
  \RequirePackage{coco-accessibility}
\fi
```

```
%</class>
```

Part I

Core Functions

Modul 2
coco-kernel.dtx

This file provides the object-oriented interfaces for all other CoCoTEX modules.

```
%<*kernel>
```

Preamble and Package Options

```
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-kernel}
    [2024/01/29 0.4.0 cocotex kernel]
```

The debug option triggers the output of additional information messages to the shell.

```
\newif\if@tp@debug \@tp@debugfalse
\DeclareOption{debug}{\global\@tp@debugtrue}%
\ProcessOptions
```

Hard dependencies

```
\RequirePackage{etoolbox}
```

1 Exception handlers

\tpKernelDebugMsg is used to print debug messages iff the debug class option is set.

```
\def\tpKernelDebugMsg#1{\if@tp@debug\message{[tp Kernel Debug]\space\space#1^^J}\fi}
```

\tpPackageError is a macro to create error messages specific to the Framework. #1 is the module, #2 is the type of error, #3 is the immediate error message, #4 is the help string.

```
\def\tpPackageError#1#2#3#4{%
   \GenericError{%
      (#1)\@spaces\@spaces\@spaces\@spaces
   }{%
      [CoCoTeX #1 #2 Error] #3%
   }{}{#4}%
}
```

\tpPackageWarning is a macro to create warnings specific to the Framework. #1 is the module, #2 is the type of error, #3 is the immediate warning message.

```
\def\tpPackageWarning#1#2#3{%
   \GenericWarning{%
      (#1)\@spaces\@spaces\@spaces\@spaces
   }{%
```

```
    [CoCoTeX #1 \if!#2!\else#2 \fi Warning] #3%
  }%
}
```

\tpPackageInfo is a macro to create shell output specific to the Framework. #1 is the module, #2 is the type of message, #3 is the immediate info string.

```
\def\tpPackageInfo#1#2#3{%
  \GenericInfo{%
    (#1)\@spaces\@spaces\@spaces\@spaces
  }{%
    [CoCoTeX #1 \if!#2!\else#2 \fi] #3%
  }%
}
```

2 Containers

Containers are the package's core data structure. They are basicly sets of properties that are processed in the same way.

\tpDeclareContainer is the constructor for new Containers. #1 is the Container's name, #2 its body which conists of Inheritance instructions, Type and Env declarations.

```
\def\tp@warningspaces{\space\space\space\space\space\space\space\space\space\space\space\space\
    space\space\space\space\space\space\space\space\space\space\space\space}%
\long\def\tpDeclareContainer#1#2{%
  \ifcsdef{tp@container@#1}
    {\tpPackageWarning{Kernel}{}{'#1' has already been declared!^^J
\tp@warningspaces Properties from the original Declaration may be^^J
\tp@warningspaces overridden by the global defaults. Use \string\tpAddToType^^J
\tp@warningspaces to alter pre-existing Containers!^^J
}}
    {\csdef{tp@container@#1}{}}%
  \csdef{tp@cur@cont}{#1}%
```

We want the declarator macros to be only allowed inside the \tpDeclareContainer macro.

```
  \begingroup
```

\tpInherit The inherit mechanism is dynamic, i.e., we can load multiple type declarations from multiple containers at once.

```
    \def\tpInherit##1##2{\@tp@inherit{##1}{##2}{#1}}%
```

\tpDeclareType Each Container is defined by the data types it provides. These data types are declared with this macro. The first argument ##1 is the name of the data type. The second argument ##2 is a list of code that is specific to this type, usually something like Component or Property declarations, handlers, and so forth.

```
    \long\def\tpDeclareType##1##2{\csgappto{tp@type@##1@#1}{##2}}%
```

\tpDeclareEnv Each container usually is realised as a LaTeX environment. The \tpDeclareEnv macro is used to set up this environment. Usually, the environment has the same name as the Container. With the optional argument ##1 you can override the environment's name. However, keep in mind that the Container's name is not changed by re-naming the corresponding environment. ##2 is used for the stuff done at the beginning of the environment, ##3 for the stuff done at the end.

In the begin part, the Types declared in the Container declaration's body should be evaluated using the `\tpEvalType` macro, see below.

```
    \def\tpDeclareEnv{\@ifnextchar [{\tp@declare@env}{\tp@declare@env[#1]}}%]
    \def\tp@declare@env[##1]##2##3{%
      \csgdef{##1}{\global\let\reserved@cont\tp@cur@cont\def\tp@cur@cont{#1}##2}%
      \csgdef{end##1}{##3}\global\let\tp@cur@cont\reserved@cont}%
```

```
    \def\x{%
      #2%
    }%
  \expandafter\x\endgroup
}
\@onlypreamble\tpDeclareContainer
```

`\tpAddToType` add additional content (i.e., the next token) to a Type #1 of a previously declared Container #2.

```
\def\tpAddToType#1#2{\csgappto{tp@type@#1@#2}}
```

`\tpEvalType` calls the Declaration list for data Type #2. With optional #1 the Container Class can be overriden.

```
\def\tpEvalType{\tp@opt@curcont\tp@eval@type}
\def\tp@eval@type[#1]#2{%
  \expandafter\ifx\csname tp@type@#2@#1\endcsname\relax
    \tpPackageError{Kernel}{Class}
    {Data Type #2 in Container #1 undefined!}
    {You try to evaluate a data type '#2' from container '#1', but that data type has not been
        declared.}%
  \else
    \tpKernelDebugMsg{Evaluating tp@type@#2@#1:^^J \csmeaning{tp@type@#2@#1}}%
    \csname tp@type@#2@#1\endcsname
  \fi
}
```

`\tpCheckParent` checks if a Container #1 is declared so that another container #2 can inherit.

```
\def\tpCheckParent#1#2{%
  \expandafter\ifx\csname tp@container@#1\endcsname\relax
    \tpPackageError{Kernel}{Class}
    {Parent Container '#1' undeclared}
    {You tried to make a Container named '#2' inherit from a Container named '#1', but a
        Container with that name does not exist.\MessageBreak
     Please make sure that parent Containers are declared before their descendents.}%
  \else
    \csgdef{tp@parent@#2}{#1}%
  \fi
}
```

`\@tp@inherit` is the low-level inherit function. #1 is a comma-separated list of things to be inherited, and #2 is the Container-list that should be inherited from, and #3 is the name of the descending Container.

```
\def\@tp@inherit#1#2#3{\@tp@parse@inherit #1,,\@nil #2,,\@nil #3\@@nil}
```

low-level function to recursively parse the parameters of the `\@tp@inherit` macro, above.

```
\def\@tp@parse@inherit #1,#2,\@nil #3,#4,\@nil #5\@@nil{%
  \let\next\relax
  \if!#1!\else
    \if!#3!\else
      \tp@do@inherit{#1}{#3}{#5}%
      \def\@argii{#2}\def\@argiv{#4}%
      \ifx\@argii\@empty
        \ifx\@argiv\@empty\else
          \def\next{\@tp@parse@inherit #1,,\@nil #4,\@nil #5\@@nil}%
        \fi
      \else
        \ifx\@argiv\@empty
          \def\next{\@tp@parse@inherit #2,\@nil #3,,\@nil #5\@@nil}%
        \else
          \def\next{%
            \@tp@parse@inherit #1,,\@nil #4,\@nil #5\@@nil
            \@tp@parse@inherit #2,\@nil #3,#4,\@nil #5\@@nil
          }%
        \fi\fi\fi\fi
  \next}
```

Ultimately, this function is called for each Type–Container combination invoked by the `\tpInherit` macro.

```
\def\tp@do@inherit#1#2#3{%
  \tpKernelDebugMsg{#3 inherits #1 from #2.}%
  \tpCheckParent{#2}{#3}%
  \expandafter\ifx\csname tp@type@#1@#2\endcsname\relax
    \tpPackageError{Kernel}{Type}{Type ‘#1’ was not declared}{Type ‘#1’ was not declared for
        Container ‘#2’.}%
  \else
    \edef\x{\noexpand\csgappto{tp@type@#1@#3}}%
    \expandafter\x\expandafter{\csname tp@type@#1@#2\endcsname}%
    \tpKernelDebugMsg{value tp@type@#1@#3:^^J \expandafter\meaning\csname tp@type@#1@#3\
        endcsname}%
  \fi
}
```

3 Components

3.1 Simple Components

“Simple Components” are basicly data storages. They are used within Containers to obtain data and store them for further processing at the end of the Container, or even beyond.

`\tpDeclareComp` defines simple component macros.

#1	is the Component’s identifier. The internal macro that is used to store the Component’s value is `\csname tp@<current Container name>@<#1>\endcsname`. If omitted, #1 is the same as #2.
#2	is the Component’s name.
#3	is code that is executed *before* assignment of the user’s value
#4	is code that is executed *after* assignment of the user’s value

```
\def\tpDeclareComp{\tp@opt@second\@tpDeclareComp}
\def\@tpDeclareComp[#1]#2#3#4{%
  \ltx@LocalExpandAfter\global\expandafter\let\csname tp@\tp@cur@cont @#1\endcsname\relax
  \expandafter\long\expandafter\def\csname tp#2\endcsname##1{%
    #3\expandafter\long\expandafter\def\csname tp@\tp@cur@cont @#1\endcsname{##1}\ignorespaces
        #4}%
}
```

\tpDeclareGComp is a shortcut to declare simple, globally available Components with the name #2 and an optional initial value #1. They are usually empty.

```
\def\tpDeclareGComp{\tp@opt@empty\tp@declare@global@comp}%
\def\tp@declare@global@comp[#1]#2{%
  \tpDeclareComp{#2}{\expandafter\global}{}%
  \if!#1!\else\csname tp#2\endcsname{#1}\fi%
}
```

Once declared, a component can be set in two ways: The first way is to use \tp<name> with one argument for its value. The second, preferred, way is to use the \tpComp macro which takes two arguments: #1 is the name of the Component, #2 is the value. This macro checks whether an Component of name #1 has actually been declared and does so, if not.

\tpComp This is the preferred way to fill a Component with content. #1 is the Component's name, #2 is the value.

```
\long\protected\def\tpComp#1#2{%
  \ifx\tp@is@counted\relax
    \ifcsdef{tp@\tp@cur@cont @#1}{}
      {\tp@def@counted@comp{\tp@cnt@grp-#1-\csname \tp@cnt@grp Cnt\endcsname}{#1}{}{}}%
    \csgdef{tp@\tp@cur@cont @\tp@cnt@grp-#1-\csname \tp@cnt@grp Cnt\endcsname}{#2}%
  \else
    \ifcsdef{tp@\tp@cur@cont @#1}{}{\tpDeclareComp{#1}{}{}}%
    \csname tp#1\endcsname{#2}%
  \fi
}
\let\tpSetComp\tpComp
```

\tpUseComp is a high level command to return (or print) the material stored as a Component with the name #1.

```
\def\tpUseComp#1{\csname tp@\tp@cur@cont @#1\endcsname}
```

\tpStoreComp is a high level command to store the value of a Component #2 into a TeX macro #1.

```
\def\tpStoreComp#1#2{%
  \def\@tempa{\protected@edef#1}%
  \expandafter\@tempa\expandafter{\expandafter\expandafter\expandafter\noexpand\csname tp@\
      tp@cur@cont @#2\endcsname}
}
```

\tpGStoreComp is the global variant of \tpStoreComp.

```
\def\tpGStoreComp#1#2{%
  \def\@tempa{\protected@xdef#1}%
  \expandafter\@tempa\expandafter{\expandafter\expandafter\expandafter\noexpand\csname tp@\
      tp@cur@cont @#2\endcsname}
}
```

\tpUseGComp is a high level command to return (or print) the material stored as a global Component from the Container #1 with the name #2.

```
\def\tpUseGComp#1#2{\csname tp@#1@#2\endcsname}
```

\tpGetComp is a high level command to return the contents stored in a Component of name #1 as a paragraph iff the Component is neither empty nor \relax.

```
\def\tpGetComp#1{\tpIfComp{#1}{\tpUseComp{#1}\par}{}}
```

\tpIfComp is a high level macro that executes #2 if the Component macro #1 is used in a Container (empty or non-empty), and #3 if not.

```
\long\def\tpIfComp#1#2#3{\expandafter\ifx\csname tp@\tp@cur@cont @#1\endcsname\relax#3\else#2\fi
    }
```

\tpWhenComp is a high level variant of \tpIfComp that omits the else-branch. #2 is code that is expanded when the Component #1 is used in a container (empty or non-empty).

```
\long\def\tpWhenComp#1#2{\expandafter\ifx\csname tp@\tp@cur@cont @#1\endcsname\relax\else#2\fi}
```

\tpUnlessComp is a high level variant of \tpIfComp that omits the then-branch. #2 is the code that is expanded when a Container #1 is *not* used in a Container (neither empty nor non-empty).

```
\long\def\tpUnlessComp#1#2{\expandafter\ifx\csname tp@\tp@cur@cont @#1\endcsname\relax#2\fi}
```

\tpIfGComp Global variant of \tpIfComp. #1 is the name of the Container, #2 is the name of the Component, #3 is the then-branch, #4 is the else-branch.

```
\long\def\tpIfGComp#1#2#3#4{\expandafter\ifx\csname tp@#1@#2\endcsname\relax#4\else#3\fi}
```

\tpIfCompEmpty is a high level macro that executes #2 if the Component macro #1 is empty (or {}) within its Container, and #3 if it is either not existant or non-empty.

```
\long\def\long@empty{}
\long\def\tpIfCompEmpty#1#2#3{\expandafter\ifx\csname tp@\tp@cur@cont @#1\endcsname\long@empty
    #2\else#3\fi}
```

\tpIfGCompEmpty is a global variant of \tpIfCompEmpty. #1 is the name of the Container, #2 is the name of the Component, #3 is the then-branch, #4 is the else-branch.

```
\long\def\tpIfGCompEmpty#1#2#3#4{\expandafter\ifx\csname tp@#1@#2\endcsname\long@empty#3\else#4\
    fi}
```

\tp@check@empty handles the distinction between empty and un-used components: First, check if #4#3 is set (=anything but \relax). If it is set, check if it is empty. If empty, set #4#3 to \relax, meaning further occurences of \IfComp{#4#3} will execute the else branch. If #4#3 is non-empty, do nothing.

If #4#3 is already \relax, check if the fallback #1#3 is set. If so, make #4#3 an alias of #1#3. If not, do nothing.

Optional #1 is the prefix of the fallback component, #2 is the Container name, #3 is the name of the Component, #4 is the Override's prefix.

```
\def\tp@check@empty{\tp@opt@empty\@tp@check@empty}%]
\def\@tp@check@empty[#1]#2#3#4{%
  \tpIfComp{#4#3}
    {\tpIfCompEmpty{#4#3}
      {\expandafter\global\expandafter\let\csname tp@#2@#4#3\endcsname\relax}
      {}}
    {\tpIfComp{#1#3}
      {\expandafter\expandafter\expandafter\let\expandafter\csname tp@#2@#4#3\expandafter\
          endcsname\csname tp@#2@#1#3\endcsname}
      {}}}
```

3.2 Counted Components

Counted Components are Components that may occur in the same parent Container multiple times. They may be multiple instances of single-macro Components, or recurring collections of multiple Components, called **Component Groups**.

Component Groups

`\tpDeclareComponentGroup` is a user-level macro to declare a new Component Group with the name #1 and the body #2.

```
\def\tpDeclareComponentGroup#1#2{%
  \csnumgdef{#1Cnt}{\z@}%
  \csdef{#1}{\tp@opt@empty{\csname @#1\endcsname}}%
  \csdef{@#1}[##1]{%
    \def\tp@cnt@grp{#1}%
    \csxdef{#1Cnt}{\expandafter\the\expandafter\numexpr\csname #1Cnt\endcsname+\@ne\relax}%
    \if!##1!\else\csgdef{tp@\tp@cur@cont @#1-\csname #1Cnt\endcsname @attrs}{##1}\fi
    #2%
    \csname @#1@hook\endcsname
  }%
  \csdef{end#1}{{\tpToggleCountedCond\csuse{tp@compose@group@#1}}}%
}
```

`\tpGroupHandler` is used to declare a new group handler. A Group Handler is a hook for code #2 that is expanded at the end of a Component Group #1's environment. It is mostly used to process Components within a Group instance and store the result in their own components. For instance, a Group Handler can be used to combine a First Name and a Surname to a combined Component "FullName".

```
\def\tpGroupHandler#1#2{%
  \ifcsdef{@#1}
    {\ifcsdef{tp@compose@group@#1}
      {\csgappto{tp@compose@group@#1}{#2}}
      {\csgdef{tp@compose@group@#1}{#2}}}
    {\tpPackageError{Kernel}{Type}{Component Group '#1' unknown!}{You tried to declare a Group
        Handler for a Component Group that has not been declared, yet! Use \string\
        tpDeclareComponentGroup{#1}{} to declare the Component Group first.}}%
}
```

`\tp@cnt@grp` is a designated group name. Counted Components of the same group use the same counter.

```
\let\tp@cnt@grp\@empty
```

`\tpUseGCompIdx` picks a Component with name #3 and index #2 from a group #1.

```
\def\tpUseGCompIdx#1#2#3{\csname tp@\tp@cur@cont @#1-#3-#2\endcsname}
```

`\tpUseGroupProp` picks a specific Property of a group.

```
\def\tpUseGroupProp#1#2#3{%
  \begingroup
    \@tempcnta\numexpr#2\relax
    \letcs\tpTotalCount{#1Cnt}%
    \def\tp@cnt@grp{#1}%
    \tpToggleCountedCond
    \csnumdef{#1Cnt}{\the\@tempcnta}%
    \tpCurCount=\the\@tempcnta\relax%
    \csname tp@\tp@cur@cont @#3\endcsname%
  \endgroup}
```

Iterating over Component Groups

The following two macros iterate over all instances of a Component Group #1 in the current Container and applies for each instance the Property #2. The result is appended to the the Collector Component #3, if and only if that Component is not yet set for the current Container at the time of the first iteration.

While the first macro only writes the Property *definition* into the Collector Component, the second fully expands the macros inside the Property and stores the result in Component #3.

Use the former to print and the latter to further process the respective results.

`\tpCurCount` stores the number of the current instance of a Counted Component. Use this in the declarations of Properties that are expanded within the Component Group.

```
\newcount\tpCurCount
```

`\tp@assign@res` assignes the result of the Component collection to a control sequence with the name #1 and resets the temporary storage.

```
\def\tp@assign@res#1{%
  \ifx\tp@iterate@res\relax
    \cslet{#1}\relax
  \else
    \expandafter\csname #1\expandafter\endcsname\expandafter{\tp@iterate@res}%
  \fi
  \global\let\tp@iterate@res\relax
}
```

`\tpIfCompOverride` is a switch to apply #2 if the Collection Component #1 has been set manually within a container or #3 if it has been generated from Counted Components.

```
\def\tpIfCompOverride#1#2#3{\expandafter\ifx\csname tp@used@#1@override\endcsname\@empty#2\else
    #3\fi}
```

`\tpComposeCollection` is used to create an unexpanded Collection Component #3 from all instances of Component Group #1 using the instructions given by property #2.

```
\def\tpComposeCollection#1#2#3{%
  \tpIfComp{#3}{\cslet{tp@used@#3@override}\@empty}{%
    \ifcsdef{#1Cnt}{%
      \expandafter\ifnum\csname #1Cnt\endcsname > \z@\relax
        \edef\tp@iterate@res{%
          \noexpand\bgroup
            \noexpand\def\noexpand\tpTotalCount{\csname #1Cnt\endcsname}%
            \noexpand\tpToggleCountedCond
            \noexpand\def\noexpand\tp@cnt@grp{#1}}%
          \expandafter\@tempcntb=\csname #1Cnt\endcsname\relax
          \tp@iterate{\@tempcnta}{\@ne}{\@tempcntb}{%
            \edef\@tempb{%
              %% top-level counter for user interaction
              \noexpand\tpCurCount=\the\@tempcnta
              %% evaluating group attributes
              \ifcsdef{tp@\tp@cur@cont @#1-\the\@tempcnta @attrs}{\noexpand\tpParseAttributes{#1-\
                  the\@tempcnta}{\csname tp@\tp@cur@cont @#1-\the\@tempcnta @attrs\endcsname}}{}
              %% internal counter for macro grabbing
              \noexpand\csnumdef{#1Cnt}{\tpCurCount}%
              \noexpand\tpUseProperty{#2}}%
            \expandafter\expandafter\expandafter\def
            \expandafter\expandafter\expandafter\tp@iterate@res
            \expandafter\expandafter\expandafter{\expandafter\tp@iterate@res\@tempb}%
          }%
          \expandafter\def\expandafter\tp@iterate@res\expandafter{\tp@iterate@res\egroup}%
          \tp@assign@res{tp#3}%
      \fi
    }{}}%
}
```

\tpApplyCollection is an alternative version of \tpComposeCollection and fully expands the Property #2 before it is stored inside the Component #3.

```
\def\tpApplyCollection#1#2#3{%
  \tpIfComp{#3}{\cslet{tp@used@#3@override}\@empty}
    {\tp@apply@collection{#1}{#2}%
     \tp@assign@res{tp#3}%
    }%
}
```

#1 is the group name, #2 is the property to format the collection

```
\def\tp@apply@collection#1#2{%
  \begingroup
    \global\let\tp@iterate@res\relax
    \letcs\tpTotalCount{#1Cnt}%
    \tp@iterate{\@tempcnta}{\@ne}{\tpTotalCount}{%
      \bgroup
        \tpToggleCountedCond
        \def\tp@cnt@grp{#1}%
        \csnumdef{#1Cnt}{\the\@tempcnta}%
        \ifcsdef{tp@\tp@cur@cont @#1-\the\@tempcnta @attrs}{\tpParseAttributes{#1-\the\@tempcnta
            }{\csname tp@\tp@cur@cont @#1-\the\@tempcnta @attrs\endcsname}}{}
        \tpCurCount=\the\@tempcnta
        \protected@xdef\@tempb{\csname tp@\tp@cur@cont @#2\endcsname}%
        \@temptokena \expandafter{\@tempb}%
        \def\@tempc{\csgappto{tp@iterate@res}}%
        \expandafter\@tempc\expandafter{\@tempb}%
      \egroup
    }%
  \endgroup
}
```

\tp@comp@def is used to pass a Counted Component into a TeX macro. #1 is a prefix to the def command, e.g., \global or \protected; #2 is the name of the TeX macro, #3 is the Name of the Counted Component (incl. the tp-prefix), and #4 is the Property that should be applied to all Members of the Counted Component.

```
\def\tp@comp@def{\tp@opt@empty\@tp@comp@def}
\def\@tp@comp@def[#1]#2#3#4{%
  \tp@apply@collection{#3}{#4}%
  \ifx\tp@iterate@res\relax
    #1\let#2\relax%
  \else
    \def\@tempa{#1\def#2}%
    \tp@assign@res{@tempa}%
  \fi
}
```

\tpCompDef is the User-level command for *local* \tp@comp@def.

```
\def\tpCompDef{\tp@comp@def}
```

\tpCompDef is the User-level command for *global* \tp@comp@def.

```
\def\tpCompGDef{\tp@comp@def[\global]}
```

Declaring Counted Component

\tpDeclareCountedComp is a user-level macro to create a new Counted Component. #1 is the user-level name of the Component.

```
\def\tpDeclareCountedComp#1{%
  \tp@def@counted@comp
    {\tp@cnt@grp-#1-\csname \tp@cnt@grp Cnt\endcsname}
    {#1}
    {}
    {\expandafter\global}%
}
```

\tp@def@counted@comp registers counter dependent Components. #1 is the internal name of the Component which is composed out of the group name, the value of the group counter and the user-level macro name #2; #3 is some custom code passed to the second argument of \tpDeclareComp; and #4 is a modifier to the internal macro definition.

```
\def\tp@def@counted@comp#1#2#3#4{%
  \tpDeclareComp[#1]{#2}
    {\bgroup#3\expandafter\global}
    {\def\@tempa{{@tp@reset@components@\tp@cur@cont}}%
     \edef\@tempb{\noexpand\csgundef{tp@\noexpand\tp@cur@cont @#1}}%
     \expandafter\expandafter\expandafter\csgappto\expandafter\@tempa\expandafter{\@tempb}%
     \egroup}%
  #4\expandafter\long\expandafter\def\csname tp@\tp@cur@cont @#2\endcsname{\csname tp@\
      tp@cur@cont @#1\endcsname}%
}
```

Resetting Counted Component

\tp@reset@components is used to reset Counted Components to prevent later Containers of a given type to feed the components from the previous Container of the same type. Usually, this is prevented by keeping Component definitions strictly local.

I some cases, however, Components may be declared globally, i.e., they may be re-used after the Container is ended. In this so-called Asynchronuous Processing of Components, the reset should be done at the very beginning of the next instance of the container type to prevent bleeding of one container's components into the next one, specifically if a container occurs more than once in the same document.

#1 is the type of the Component set.

```
\def\tp@reset@components#1{%
  \csname @tp@reset@components@#1\endcsname
  \global\cslet{@tp@reset@components@#1}\relax%
}
```

Toggling Conditionals for Counted Components

\tpToggleCountedCond In order to process Counted Components, we need to re-define the Conditionals in a way such that the Component is expanded twice before the comparison takes place to correctly resolve the Component counter.

Warning! Use this macro only within local groups!

```
\long\def\tpToggleCountedCond{%
  \let\tp@is@counted\relax
```

This re-definitions of \tpIfComp cannot use etoolbox's \cs... macros since the conditional can be embedded inside itself. If an inner csname is undefined, the condition for the outer one would be reset before it can be expanded by \ifx.

```
  \long\def\tpIfComp##1##2##3{%
    \expandafter\expandafter\expandafter\let\expandafter\csname tp@comp@name\expandafter\
        endcsname\csname tp@\tp@cur@cont @##1\endcsname%
    \expandafter\expandafter\expandafter\ifx\tp@comp@name\relax##3\else##2\fi%
  }%
  \long\def\tpIfCompEmpty##1##2##3{%
    \expandafter\expandafter\expandafter\ifx\csname tp@\tp@cur@cont @##1\endcsname\long@empty
        ##2\else ##3\fi}}
```

4 Hooks

Hooks are used to patch code into different parts of a Container's processing chain.

\tpDeclareHook registers a new hook. Optional #1 is the container for which the Hook is declared. If omitted, this defaults to \tp@cur@cont. #2 is the Hook's user-level name. Hooks always default to an empty string.

```
\def\tpDeclareHook{\tp@opt@curcont\tp@declare@hook}
\def\tp@declare@hook[#1]#2{\expandafter\global\expandafter\let\csname tp@hook@#1@#2\endcsname\
    @empty}
```

\tpAddToHook adds new material to a Hook. If the hook has not yet been declared, a tpDeclareHook for that hook is applied first. In that case, use the optional #1 to specify the Container name that hook is intended for. If it is omitted, the current Container is used. #2 is the name of the hook the material in #3 is to be appended to.

```
\def\tpAddToHook{\tp@opt@curcont\tp@add@to@hook}
\def\tp@add@to@hook[#1]#2#3{%
  \expandafter\ifx\csname tp@hook@#1@#2\endcsname\relax
    \tpDeclareHook[#1]{#2}%
  \fi
  \csgappto{tp@hook@#1@#2}{#3}%
}
```

\tpUseHook expands the current state of the hook with the name #2 from Container #1 (current Container if omitted).

```
\def\tpUseHook{\tp@opt@curcont\tp@use@hook}
\def\tp@use@hook[#1]#2{\csuse{tp@hook@#1@#2}}
```

5 Properties

5.1 Setting Properties

\tpSetProperty is a user-level macro that provides the Property–Value interface for Containers. #1 is the name of the Property, #2 is the Value assigned to that Property.

```
\long\def\tpSetProperty#1#2{\long\csdef{tp@\tp@cur@cont @#1}{#2}}
```

\tpPropertyLet can be used to create an alias Property #1 of a given Property #2. Is is equivalent to \tpSetProperty {\#1}{\tpUseProperty{\#2}}.

```
\long\def\tpPropertyLet#1#2{\long\csdef{tp@\tp@cur@cont @#1}{\csuse{tp@\tp@cur@cont @#2}}}
```

\tpPropertyLetX creates a Property #1 with the fully expanded value of another Property #2 Is is equivalent to \tpSetPropertyX{\#1}{\tpUseProperty{\#2}}.

```
\long\def\tpPropertyLetX#1#2{\long\csedef{tp@\tp@cur@cont @#1}{\csuse{tp@\tp@cur@cont @#2}}}
```

\tpSetValProp is a variant of \tpSetProperty that expands the value #2 *once* before assigning it to the Property macro with the name #1. This can be used to assign the current value of a variable macro, dimension, counter or length to a Property.

```
\long\def\tpSetValProp#1#2{\def\@tempa{\tpSetProperty{#1}}\expandafter\@tempa\expandafter{#2}}
```

\tpSetPropertyX is another variant of \tpSetProperty, but it *fully expands* the value (using \edef) defined in #2 before the Property is stored in the Property macro named #1. Use this if you need to use conditionals to determine the actual values of Properties that otherwise expect fixed named or dimensional values.

```
\long\def\tpSetPropertyX#1#2{\long\csedef{tp@\tp@cur@cont @#1}{#2}}
```

\tpAddToDefault adds the material in the next token to a Container of name #1's Property Type.

```
\long\def\tpAddToDefault#1{\tpAddToType{Properties}{#1}}
```

5.2 Using Properties

\tpUseProperty is a user-level command to directly access a previously set Property.

```
\def\tpUseProperty#1{\csuse{tp@\tp@cur@cont @#1}}
```

\tpUsePropEnv is a user-level command to access a previously set Property and make it an environment accessible to Property specific processing instrunctions (see below).

```
\def\tpUsePropEnv#1{\cslet{tp@#1@active}{\relax}\csuse{tp@\tp@cur@cont @#1}\csundef{tp@#1@active
    }}
```

5.3 Processing Instructions

In general, processing instructions are commands that are only visible to a specific process and ignored by others. In CoCoTEX, Processing Instructions (PIs) are commands placed inside a Component that should only take effect when that Component is processed through a specific Property.

`\tpPI` is a Processing Instruction that executes #2 when a Property with the name #1 is currently processed with the `\tpUsePropEnv` macro.

```
\DeclareRobustCommand\tpPI[2]{\ifcsdef{tp@#1@active}{#2}{}}
```

**WARNING!
The following section is deprecated and will be changed or deleted in future releases.**

TODO: Incorporate into the Container inheritance mechanism. Check if inheritance of Container Types is to be distinguished from inheritance of Properties and their Values!

`\tpCascadeProps` recursivly loads a Container's own Properties, the Properties of the Container's parent(s), and the default Properties of the top-level Container. #1 is the current Container's name, #2 is the top-level Container.

```
\def\tpCascadeProps#1#2{%
  \csname tp@#2@default\endcsname
  \expandafter\ifx\csname tp@#2@#1@parent\endcsname\relax\else
    \expandafter\tp@inherit@props\expandafter{\csname tp@#2@#1@parent\endcsname}{#2}%
  \fi
  \csname tp@#2@#1@properties\endcsname
}
```

This low-level macro recursivly loads properties from parent namespaces, if they exist. #1 is the parent (may be empty), #2 is the macro family.

```
\def\tp@inherit@props#1#2{%
  \expandafter\ifx\csname tp@#2@#1@parent\endcsname\relax\else
    \edef\@tempa{\csname tp@#2@#1@parent\endcsname}%
    \expandafter\tp@inherit@props\expandafter{\@tempa}{#2}%
  \fi
  \csname tp@#2@#1@properties\endcsname
}
```

5.4 Property Conditionals

`\tpIfProp` checks if a Property with the name #1 is defined and non-empty. If so, do #2, otherwise do #3.

```
\long\def\tpIfProp#1#2#3{%
  \expandafter\ifx\csname tp@\tp@cur@cont @#1\endcsname\relax#3\else
    \expandafter\ifx\csname tp@\tp@cur@cont @#1\endcsname\long@empty #3\else#2\fi
  \fi
\ignorespaces}
```

`\tpIfPropVal` checks if a Property #1 expands to #2. If so, do #3, otherwise do #4.

Warning: Do not use this conditional in Properties that are used in `\tpApplyCollection`!

```
\long\def\tpIfPropVal#1#2#3#4{\long\def\@tempa{#2}%
  \expandafter\ifx\csname tp@\tp@cur@cont @#1\endcsname\@tempa\relax#3\else#4\fi\ignorespaces}
```

6 Helper macros

6.1 Handling of Optional Arguments

Two simple internal macros to ease up the handling of optional arguments.

\tp@opt@curcont overrides Container Names with the optional argument.

```
\long\def\tp@opt@curcont#1{\@ifnextchar[{#1}{#1[\tp@cur@cont]}}%]
```

\tp@opt@empty passes an empty string if the optional argument is missing.

```
\long\def\tp@opt@empty#1{\@ifnextchar[{#1}{#1[]}}%]
```

\tp@opt@second passes the first mandatory argument to the optional argument if the latter is missing.

```
\let\tp@opt@second\@dblarg
```

6.2 Iterators

\tp@iterate traverses in #1-th steps (optional, defaults to +1) through counter #2 start at number #3 until and including number #4 and do at every loop #5 (from forloop.sty):

```
\long\def\tp@iterate{\@ifnextchar[{\@tp@iterate}{\@tp@iterate[\@ne]}}%]
\long\def\@tp@iterate[#1]#2#3#4#5{%
  \advance#2 by #1\relax
  #2=#3\relax%
  \expandafter\ifnum#2>#4\relax%
  \else
    #5%
    \tp@iterate[#1]{#2}{\the#2}{#4}{#5}%
  \fi}%
```

6.3 Attributes

Many macros and environments deal with optional arguments that are used to alter the behaviour of that macro or environment. The combination of a parameter and its set of possible values are calles **Attributes**. In this section, we define the parsers for those paramters.

In order to catch the babel package's messing with the quote symbol, we make sure it has the correct cat-code.

```
\begingroup
\catcode`"=12
```

\tpParseAttributes High level wrapper for the attribute parser; #1 is the parent node of the attribute, #2 is the attribute chain

```
\gdef\tpParseAttributes#1#2{%
  \if!#1!\else
    \if!#2!\else
      \def\tp@cur@node{#1}%
      \@tp@parse@attributes #2,,\@nil
    \fi\fi}
```

The actual, recursively applying, parser comes in two parts:

`\@tp@parse@attributes` parses the single attributes in an optional argument,

```
\gdef\@tp@parse@attributes #1,#2,\@nil{%
  \if!#1!\else
    \tp@parse@kv#1==\@nil
    \if!#2!\else
      \@tp@parse@attributes#2,\@nil
    \fi\fi}
\endgroup
```

and

`\tp@parse@kv` distinguishes between the parameter name and its value(s).

```
\gdef\tp@parse@kv#1=#2=#3\@nil{%
  \edef\@argii{#2}%
  \ifx\@argii\@empty
    \expandafter\let\csname tp@\tp@cur@node @attr@#1\endcsname\@empty%
  \else
    \ifx #2 =\else
      \expandafter\def\csname tp@\tp@cur@node @attr@#1\endcsname{#2}%
    \fi
  \fi}
```

`\tpGetAttr` returns the value of an attribute.

#1 is the attribute node, #2 is the attribute name.

```
\def\tpGetAttr#1#2{\csuse{tp@#1@attr@#2}}
```

`\tpIfAttr` can be used to call macros depending on whether an attribute is set.

#1 is the attribute node, #2 is the attribute name, #3 and #4 are the true and false branch, respectively.

```
\def\tpIfAttr#1#2#3#4{\ifcsdef{tp@#1@attr@#2}{#3}{#4}}
```

`\tpIfAttrStr` can be used to call macros depending if an attribute is set to the current (sub)container or group and what value it has.

#1 is the attribute node, #2 is the attribute name, #3 is the comparision value (a string!), #4 and #5 are the true and false branch, respectively.

```
\def\tpIfAttrStr#1#2#3#4#5{\tpIfAttr{#1}{#2}{\ifcsstring{tp@#1@attr@#2}{#3}{#4}{#5}}{#5}}
```

`\tpIfAttrIsset` can be used to check of a value-less attribute has been set (i.e., it expands to `\@empty`).

#1 is the attribute node, #2 is the attribute name, #3 and #4 are the true and false branch, respectively.

```
\def\tpIfAttrIsset#1#2#3#4{\tpIfAttr{#1}{#2}{\expandafter\ifx\csname tp@#1@attr@#2\endcsname\
    @empty#3\else#4\fi}{#4}}
```

6.4 Style Classes

Style Classes are locally usable sub-Containers.

`\tpDeclareClass` The top-level macro `\tpDeclareClass[#1]{#2}[#3]{#4}` has four arguments, two of which are optional. #2 is the name of the class. If this argument is empty, the special class name `default` is used.

#4 is the declaration block of the class. This argument usually containsa set of property assignments using the `\tpSetProperty{<prop>}{<val>}` macro, see Sect. 5. The first optional argument #1 is the Style Class' parent Container. Using parent Containers, you can have Style Classes of the same name for different (sub-)Containers, e.g., a `default` class for each float and heading Container. The second optional argument #3 is the parent Style Class. Properties from that Style Class are loaded automatically prior to the loading of the current Style Class's Properties. This applies recursively allowing for a cascading of property values, as in CSS.

```
\long\def\tpDeclareClass{\@ifnextchar [{\@tp@set@class}{\@tp@set@class[default]}}%]
\long\def\@tp@set@class[#1]#2{\tp@opt@empty{\tp@set@class[#1]{#2}}}%
\long\gdef\tp@default@class@default{}
\long\def\tp@set@class[#1]#2[#3]#4{%
  \def\@argii{#2}\ifx\@argii\@empty\let\@argii\tp@str@default\fi%
  \if!#3!\else
    \expandafter\long\expandafter\def\csname tp@#1@class@\@argii @parent\endcsname{#3}%
  \fi
  \expandafter\long\expandafter\def\csname tp@#1@class@\@argii\endcsname{#4}%
}
```

`\tpUseClass` is a user-level macro to expand and âĂIJactivateâĂİ a Style Class' Properties, those of its recursive ancestor Style Classes, and the default Style Class respecting the current Container. #1 is the Style Class name, #2 is the Container.

```
\def\tpUseClass#1#2{%
  \expandafter\ifx\csname tp@#2@class@#1\endcsname\relax
    \expandafter\ifx\csname tp@default@class@#1\endcsname\relax
      \PackageError{cocotex.cls}{Class ‘#1’ with scope ‘#2’ not defined!}{Please declare the
           class ‘#1’!}%
    \fi
  \fi
  \csname tp@default@class@#1\endcsname%
  \expandafter\ifx\csname tp@#2@class@#1@parent\endcsname\relax\else
    \expandafter\tpUseClass\expandafter{\csname tp@#2@class@#1@parent\endcsname}{#2}%
  \fi
  \csname tp@#2@class@#1\endcsname}
```

`\CoCoTeX` the CoCoTeX Logo.

```
\def\CoCoTeX{{\ttfamily CoCo\TeX}}
```

7 Legacy Functions

WARNING!
The following section is deprecated and will be changed or deleted in future releases.

```
 \def\tpNamespace#1{\def\tp@cur@cont{#1}}
```

```
%</kernel>
```

Modul 3

coco-common.dtx

This file provides some macros that are used in more than one CoCoTEX module.

```
%<*common>
```

```
%%
%% module for CoCoTeX that provides some commonly used base macros.
%%
%% Maintainer: p.schulz@le-tex.de
%%
%% lualatex - texlive > 2019
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-common}
   [2024/01/29 0.4.0 CoCoTeX common module]
\RequirePackage{iftex}
```

1 Package options

1.1 Accessibility Features

The option `a11y` triggers loading of the CoCoTEX Accessibility Module and its features.

```
\DeclareOptionX{a11y}{\let\tp@do@ally\relax}
```

Default color encoding passed as option to the `xcolor` package.

```
\def\tp@color@enc{cmyk}
\define@choicekey{coco-common.sty}{color-enc}[\@tp@color@enc\nr]{srgb,rgb,gray,cmy,cmyk,natural
    }[cmyk]{%
  \let\tp@color@enc\@tp@color@enc
  \ifcase\nr\relax% srgb
    \def\tp@color@enc{rgb}%
  \or% rgb
  \or% gray
  \or% cmy
    \def\tp@color@enc{cmyk}%
  \or% cmyk
  \else% natural, i.e. no conversion of color spaces takes place
  \fi
}
\ProcessOptionsX
\PassOptionsToPackage{\tp@color@enc}{xcolor}%
```

`\tpIfAlly` is a switch to distinct between compilation with (implicit #1) or without (implicit #2) activated accessibility features.

```
\def\tp@if@ally{\ifx\tp@do@ally\relax\expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi}
\let\tpIfAlly\tp@if@ally
\def\tp@if@preamble{\ifx\@nodocument\relax\expandafter\@secondoftwo\else\expandafter\@firstoftwo
    \fi}
\let\tpIfPreamble\tp@if@preamble
```

2 Commonly Used Low-Level Macros and Registers

Contains common macros used in the CoCoTeX modules and that are intended for macro and stylesheet programming.

```
\RequirePackage{coco-kernel}
```

2.1 Hard Dependencies

Hard requirements for all CoCoTeX modules:

```
\RequirePackage{xcolor}
```

Including the graphicx package and catching case-insensitive graphics file's endings from Word:

```
\RequirePackage{graphicx}
\DeclareGraphicsRule{.EPS}{eps}{.EPS}{}
```

2.2 Common Variables

String Variables for Value Comparisions

\tp@str@default

```
\def\tp@str@default{default}
```

\tp@str@table

```
\def\tp@str@table{table}
```

\tp@str@figure

```
\def\tp@str@figure{figure}
```

Box Registers

Some temporary boxes that won't interfere with LaTeX's temporary boxes.

\tp@tempboxa

```
\newbox\tp@tempboxa
```

\tp@tempboxb

```
\newbox\tp@tempboxb
```

Length and Skip Registers

\tp@tempskipa

```
\newskip\tp@tempskipa
```

2.3 Helper macros

\afterfi used to execute code after the next \fi:

```
\def\afterfi#1\fi{\fi#1}
```

\tp@topstrut is a \strut that has the height of \topskip and the depth of the difference between the \baselineskip and \topskip.

```
\def\tp@topstrut{\vrule\@width\z@\@height\topskip\@depth\dimexpr\baselineskip-\topskip\relax}
```

\afterbox prevents indentation and additional spacing after environments. Intended to be used in combination with \aftergroup.

```
\def\@afterbox{%
  \everypar{%
    \if@nobreak
      \@nobreakfalse
      \clubpenalty \@M
      \if@afterindent \else
        {\setbox\z@\lastbox}%
        \everypar{}%
      \fi
    \else
      \clubpenalty \@clubpenalty
      {\setbox\z@\lastbox}%
      \everypar{}%
    \fi}}
```

2.4 Masks

These macros are intended to mask non-content markup, like page- or line breaking commands in order to find and remove or alter them easier.

\hack intended to mask line breaking macros.

```
\let\hack\@firstofone
```

\hackfor intended to hide line breaking macros.

```
\let\hackfor\@gobble
```

\Hack intended to mask page breaking macros.

```
\let\Hack\@firstofone
```

\Hackfor intended to hide page breaking macros.

```
\let\Hackfor\@gobble
```

`\@gobbleopt` intended to nullify a macro's argument with a possible optional argument interfering.

Use it like this: `\let\yourMacroWithOptArg\@gobbleopt`

```
\long\def\@gobbleopt{\@ifnextchar[\@@gobbleopt{\@@gobbleopt[]}}%]
\long\def\@@gobbleopt[#1]#2{}%
```

`\tpGobble` is used to de-activate certain macros to prevent them from being called multiple times while processing contents. An example is a footnote inside a caption while calculating the height of the caption. In this case, we need the space the footnote symbol requires without the actual footnote being written into the footnote insert, since that should happen when we actually print the caption.

```
\def\tpGobble{%
  \renewcommand\footnote[2][\the\c@footnote]{\def\@thefnmark{##1}\@makefnmark}%
  \renewcommand\index[2][]{}%
  \renewcommand\marginpar[2][]{}%
  \renewcommand\glossary[2][]{}%
  \let\label\@gobble
}%
```

2.5 Arithmetics

`\CalcRatio` is used to calculate the ratio between two integers.

```
\def\CalcRatio#1#2{\strip@pt\dimexpr\number\numexpr\number\dimexpr#1\relax*65536/\number\dimexpr
    #2\relax\relax sp}
```

`\CalcModulo` is used to calculate the remainder of integer division of #1 by #2. This needs a different approach than the common modulo definition, which would return negative results in some cases, as TeX rounds up the quotient of #1 and #2 if the first decimal place is equal to or greater 5.

```
\def\CalcModulo#1#2{\number\numexpr#1+#2-((#1+#2/2)/#2)*#2\relax}
```

`\minusvspace` Counterpart to LaTeX's `\addvspace`: if the value of `\minusvspace` is larger than `\lastskip`, `\lastskip` is used. Otherwise, the value of `\minusvspace` is used.

```
\def\@xminusvskip{%
  \ifdim\lastskip<\@tempskipb
  \else
    \ifdim\lastskip<\z@
    \else
      \ifdim\@tempskipb<\z@
        \advance\@tempskipb\lastskip
      \fi
      \vskip-\lastskip
      \vskip \@tempskipb
    \fi
  \fi}
\def\minusvspace#1{%
  \ifvmode
     \if@minipage\else
       \ifdim \lastskip =\z@
```

Compatibility to texlive pre 2020:

```
         \ifx\@vspace@calcify\@undefined
           \vskip #1\relax
```

```
        \else
          \@vspace@calcify{#1}%
        \fi
      \else
      \setlength\@tempskipb{#1}%
        \@xminusvskip
      \fi
    \fi
  \else
    \@noitemerr
  \fi}
```

2.6 Determine actual page number

We need to determine the real page a floating object is printed. This mechanism is largely an adaption of the mechanism used in the `marginnote` package.

Counting absolute page numbers, however, may be misleading when the `coco-title` module is loaded and the cover page is not followed by an empty page. Therefore, we save the default page counter from LaTeX to evaluate it independently from the actual manner of counting.

`\the@tp@thispage`

```
\def\the@tp@thispage{}%
```

`\tp@abspage`

```
\newcount\tp@abspage \tp@abspage\z@
```

`\thetp@abspage`

```
\def\thetp@abspage{\the\tp@abspage}
```

`\if@tp@odd`

```
\newif\if@tp@odd \@tp@oddtrue
```

```
\AtBeginDocument{%
  \global\tp@abspage=\c@page\relax%
  \g@addto@macro\@outputpage{\global\tp@abspage\c@page}%
}
```

`\tp@test@page` We split this into two parts. The first one is run before the floating object is placed. It will store the page according to the placement in the tex source code.

```
\def\tp@test@page{%
  \expandafter\ifx\csname the@tp@thispage\endcsname\@empty
    \gdef\the@tp@atthispage{1}%
  \else
    \expandafter\ifnum \the@tp@thispage=\tp@abspage%
      \begingroup
        \@tempcnta\the@tp@atthispage\relax
        \advance\@tempcnta\@ne\relax
        \xdef\the@tp@atthispage{\the\@tempcnta}%
      \endgroup
    \else
```

```
      \gdef\@tp@atthispage{1}%
    \fi
  \fi
  \xdef\the@tp@thispage{\the\tp@abspage}%
  \let\@tp@currpage\relax
  \expandafter\ifx\csname \tp@cur@cont-\the@tp@thispage-\the@tp@atthispage\endcsname\relax
    \ifodd\tp@abspage\relax\@tp@oddtrue\else\@tp@oddfalse\fi
  \else
    \edef\@tp@currpage{\expandafter\expandafter\expandafter\@firstofone\csname \tp@cur@cont-\
        the@tp@thispage-\the@tp@atthispage\endcsname}%
    \ifodd\@tp@currpage\relax\@tp@oddtrue\else\@tp@oddfalse\fi
  \fi
}
```

`\tp@save@page` the second macro writes the actual position of the floating object into the aux files. This macro has to be placed inside the float environment/macro.

```
\def\tp@save@page{%
  \protected@write\@auxout{\def\the@tp@cur@cont{\tp@cur@cont}\let\thetp@abspage\relax}{%
    \string\expandafter\string\gdef\string\csname\space \tp@cur@cont-\the@tp@thispage-\
        the@tp@atthispage\string\endcsname{\thetp@abspage}}%
}
```

3 Re-Thinking LaTeX Core Functions

3.1 Keeping .aux-Files Up-to-Date

`\tpBreak` is a general line break macro intended to be re-defined if necessary without touching LaTeX's kernel page and line breaking macros.

```
\DeclareRobustCommand{\tpBreak}{\hfill\break}
```

3.2 Content lists

This part contains macros to "simplify" the generation of content lists like the table of contents or list of figures/tables, etc.

Entries in the list-files (e. g., `\jobname.toc`, `\jobname.lof`, etc.) usually contain `\contentsline` macros that expand to `l@<level>`. Whenever a level of Components that are to be written into content lists is declared, the package automatically generates a `\tp@l@<level>` macro for this level of entries. The content-baring argument of `\tpContentsline` (or `\tp@l@<level>`, resp.) contains Components.

Once a list file is read, those `\tp@l@<level>` macros are expanded in two steps. Each entry constitutes a Container in its own right. It therefore can have multiple Components. The first step is the extraction phase, where the entry's Container is dynamically declared, the corresponding properties are initialised, and its Components are extracted

`\tp@init@l@` is a low-level macro used to dynamically define `\tp@l@<level>` macros. Optional #1 is an override for counters that have to be restored, #2 is the list file ending (raw entries being stored in a file `\jobname.#2`), #3 is a number that indicated the nesting depth, #4 is the nested level's unique name.

```
\def\tp@init@l@{\tp@opt@empty\@tp@init@l@}%
\def\@tp@init@l@[#1]#2#3#4{%
  \expandafter\ifx\csname c@#2depth\endcsname\relax
    \expandafter\global\expandafter\newcount\csname c@#2depth\endcsname
    \expandafter\global\csname c@#2depth\endcsname=0\relax
  \fi
  \expandafter\ifx\csname tp@#2@extract@data\endcsname\relax
    \expandafter\let\csname tp@#2@extract@data\endcsname\tp@extract@generic
  \fi
  \expandafter\ifx\csname tp@#2@print@entry\endcsname\relax
    \expandafter\let\csname tp@#2@print@entry\endcsname\tp@print@generic
  \fi
  \expandafter\long\expandafter\gdef\csname tp@l@#4\endcsname##1##2{%
    \ifLuaTeX\suppresslongerror=1\fi
    \expandafter\ifnum \csname c@#2depth\endcsname<#3\relax
    \else
      \bgroup
        \long\def\tpTocLink####1{\hyper@linkstart{link}{\Hy@tocdestname}{####1}\hyper@linkend}%
        \csname tp@#2@extract@data\endcsname{#3}{#4}{##1}{##2}%
        \csname tp@#2@print@entry\endcsname{#4}%
      \egroup
    \fi
    \ifLuaTeX\suppresslongerror=0\fi
  }}
```

`\tpContentsline` has two purposes: It re-directs `l@<level>` macros to our own version, and it ensures that LaTeX won't break if Components in the content lists contain `\par`. In order for the latter to work correctly, however, we need to patch `\contentsline` to make it `\long`, first.

```
\AtBeginDocument{%
  \begingroup\toks0=\expandafter{\contentsline{#1}{#2}{#3}{#4}}
  \edef\x{\endgroup\long\def\noexpand\contentsline##1##2##3##4{\the\toks0 }}\x
}
\long\def\tpContentsline#1#2#3#4{\bgroup\csletcs{l@#1}{tp@l@#1}\contentsline{#1}{#2}{#3}{#4}\
    egroup}
```

`\tp@extract@generic`

```
\def\tp@extract@generic#1#2#3#4{}
```

`\tp@print@generic`

```
\def\tp@print@generic#1{}
```

`\tp@expand@l@contents` expands the content of the `tp@l@<level>` macro and contains some code to catch and handle standard LaTeX headings. #1 is the content of the `tp@l@`-macro, #2 is the namespace, #3 is the Component prefix and #4 is the name of the Content component.

```
\def\tp@expand@l@contents#1#2#3#4{%
  \global\let\tp@tempa\relax
  \sbox\z@{\def\numberline##1{\xdef\tp@tempa{\noexpand\csdef{tp@#2@#3Number}{##1}}}#1}%
  \ifdim\wd\z@>\z@
    \let\numberline\@gobble%
    \protected@csedef{tp@#2@#3#4}{#1}%
    \tp@tempa
  \else
    #1%
  \fi
  \global\let\tp@tempa\relax
}
```

3.3 Indentation and Left Margins of Potentially Numbered Items

The **left margin** means the space between the left border of the page area and the imaginary line that multi-line text aligns to. The **indent** is the offset of the very first line of that block of text relative to that value.

If the `indent` is a negative value you'll get a hanging indent; if it is positive, you get a paragraph style indent, and if it is set to `0pt`, you get a clean alignment of the whole item.

CoCoTeX provides a feature that allows the indention of counted elements to be just as wide as the widest Number of the same level (if `indent` is set to `auto`), as well as a feature that allows the indent to be as wide as all Numbers of the same cotainer type (if `indent` is set to `auto-global`).

The approach to set the `indent`, `margin-left` and the position of the Number Component in numbered items such as Headings, entries in ToC and listof-X, captions, etc. is to store the maximum width for each level and the maximum width across all Numbers of a Container Type in the .aux file at the very end of the compilation after it has been constantly updated during the entire LaTeX runtime. That way, for the next LaTeX run, the maximum values are available immediately and can be used to fortify those parameters.

`\tp@store@latest` low-level macro that stores the maximum value of a dimension Property #1. An internal Property `\#1-local` is constantly updated whenever the macro is called and the previously stored value is lower than the one given in `#2`.

The first call of the macro for a given Property triggers an addendum to the `\@enddocumenthook` which causes the last value for that dimension to be stored in the .aux file. If the Property hasn't been set from a previous LaTeX run or a previous call to the `\tp@store@latest` macro for the same Property and the same level, it is set to #2.

#1 is the internal name of the property, #2 is the check value.

```latex
\def\tp@store@latest#1#2{%
  \expandafter\ifx\csname tp-\tp@cur@cont-#1\endcsname\relax
    \csxdef{tp-\tp@cur@cont-#1}{#2}%
  \else
    \expandafter\ifdim\csname tp-\tp@cur@cont-#1\endcsname<#2\relax
      \csxdef{tp-\tp@cur@cont-#1}{#2}%
    \fi
  \fi
  \expandafter\ifx\csname tp-\tp@cur@cont-#1-local\endcsname\relax
    \csxdef{tp-\tp@cur@cont-#1-local}{#2}%
  \else
    \expandafter\ifdim\csname tp-\tp@cur@cont-#1-local\endcsname<#2\relax
      \csxdef{tp-\tp@cur@cont-#1-local}{#2}%
    \fi
  \fi
```

The second step is to store the highest values in the .aux file for later LaTeX runs. A `\write\@auxout` command for the storage macro is therefore added to the `\@enddocumenthook` and a flag is set that indicates that the write command has already been added to the hook, since that needs to be done only once for each to-be-stored dimension.

Note that the value that is eventually stored, is the updated *local* maximum, not the value that is retrieved at the beginning of the run. This allows the values to be down-graded if the LaTeX source changed during two consecutive runs. However, if values change, you still need to do at least two more LaTeX runs before the values stabilize.

```latex
  \ifcsdef{tp-\tp@cur@cont-#1-stored-trigger}{}
    {\edef\@tempa{%
       \noexpand\immediate\noexpand\write\noexpand\@auxout{%
         \noexpand\string\noexpand\csgdef{tp-\tp@cur@cont-#1}{%
           \noexpand\csname tp-\tp@cur@cont-#1-local\noexpand\endcsname}}}%
     \expandafter\AtEndDocument\expandafter{\@tempa}%
     \csgdef{tp-\tp@cur@cont-#1-stored-trigger}{\@empty}}}
```

`\tp@format@number` calculates number widths and prepares macros to be used by the user. #1 is the internal Property prefix, #2 is the user-level Component prefix, #3 is the numerical list level.

```
\def\tp@format@number#1#2#3{%
  \tpSetValProp{#1curr-number-level}{#3}%
```

First step: measuring the natural width of the Number if it exists for the current item.

```
  \tpIfComp{#2Number}
    {\sbox\z@{\tpUseProperty{#1number-format}}}
    {\sbox\z@{}}%
```

Second step: we store the width of `\box0` if it is wider than the previously stored width for that level. The end value will be written into the .aux file during expansion of the `\@enddocumenthook`. We do the same for the maximum across *all* levels of the same Container Type.

```
  \tp@store@latest{#1number-#3-maxwd}{\the\wd\z@}%
  \tp@store@latest{#1number-maxwd}{\the\wd\z@}%
```

We provide the maximum level as a user-level Property `#1number-width-level-max`, the global maximum across all levels as `#1number-width-max`, and the width of the current number as `#1number-width`.

```
  \tpSetValProp{#1number-width-level-max}{\csname tp-\tp@cur@cont-#1number-#3-maxwd\endcsname}%
  \tpSetValProp{#1number-width-max}{\csname tp-\tp@cur@cont-#1number-maxwd\endcsname}%
  \tpSetValProp{#1number-width}{\the\wd\z@}%
```

Third step: we calculate and fortify the actual `#1margin-left` (i.e., the overall left indent of the whole item) and `#1indent` (offset of the first line) of the entry.

```
  \tp@get@indent{#1}{#3}%
  \tp@set@hang{#1}%
}
```

`\tp@set@hang` determines and sets the hanging indent of a counter. #1 is the internal Property prefix.

```
\def\tp@set@hang#1{%
```

First, we set the `#1hang-number` to be an alias of `#1number-format` as fallback.

```
  \tpPropertyLet{#1hang-number}{#1number-format}%
```

Then, we check for `#1indent`.

```
  \tpIfProp{#1indent}
    {\ifdim\tpUseProperty{#1indent}<\z@
```

If it is set and negative, we alter the `#1hang-number` Property in such a way that it is shifted to the left by `#1indent` amount and put into a hbox of -`#1indent` width (remember that the value is negative).

```
       \tpSetProperty{#1hang-number}{%
         \hskip\tpUseProperty{#1indent}%
         \hbox to -\tpUseProperty{#1indent}{%
           \tpIfPropVal{#1number-align}{left}{}{\hss}%
           \tpUseProperty{#1number-format}%
           \tpIfPropVal{#1number-align}{right}{}{\hss}}}%
     \fi}{}}
```

In all other cases, we stick to the default (`#1number-format`) we set in the first step.

`\tp@calc@margin@left` determines the left margin of the current level by subtracting the current level's indent from the left margin of the next-higher level. "Next-higher" meaning "hierarchically", i.e., the level counter is *lower*. Remember that for hang indent, the indent is negative, so `margin-left` grows larger.

#1 is the Property prefix, #2 is the current numerical list level.

```
\def\tp@calc@margin@left#1#2{%
  \@tempcnta\numexpr#2-1\relax
  \expandafter\ifx\csname tp-\tp@cur@cont-#1\the\@tempcnta-margin-left\endcsname\relax
    \@tempdima=-\tpUseProperty{#1indent}\relax%
  \else
    \@tempdima=\dimexpr\csname tp-\tp@cur@cont-#1\the\@tempcnta-margin-left\endcsname-\
         tpUseProperty{#1indent}\relax
  \fi
  \tp@store@latest{#1#2-margin-left}{\the\@tempdima}%
  \tpSetProperty{#1margin-left}{\the\@tempdima}}
```

`\tp@get@indent` Eventually, write the actually used values for margin-left and indent into the current container's Property list. #1 is the internal property prefix, #2 is the numerical list level.

```
\def\tp@get@indent#1#2{%
```

First, we need to store the initial values for both `#1margin-left` and `#1indent` since, first their values might be non-dimensional, and second, they will be altered during macro expansion to ultimatly being passed to `\hskip`.

```
  \tpPropertyLetX{int-#1margin-left}{#1margin-left}%
  \tpPropertyLetX{int-#1indent}{#1indent}%
  \tpIfPropVal{#1indent}{auto-global}
```

If `#1indent` is set to `auto-global`, the item gets an `indent` that is set to the negative value of the maximum width of all numbers across all Levels of the same Container Type. The same maximum is added to the user-set value of `margin-left`.

```
    {\tpSetPropertyX{#1indent}{-\tpUseProperty{#1number-width-max}}%
```

If the user has not set `margin-left`, we set it to `\z@`.

```
     \tpIfPropVal{#1margin-left}{}
       {\tpSetProperty{int-#1margin-left}{\z@}}
       {\tpPropertyLetX{int-#1margin-left}{#1margin-left}}%
     \tpSetPropertyX{#1margin-left}{\dimexpr\tpUseProperty{#1number-width-max}+\tpUseProperty{int
         -#1margin-left}\relax}}
```

Next, we check if `#1margin-left` is set to `auto`.

```
    {\tpIfPropVal{int-#1margin-left}{auto}
```

If `#1margin-left` is set to `auto`, all items of the same level get the same left margin that is determined by the sums of the indents of all higher levels.

```
      {\tpIfPropVal{int-#1indent}{auto}
```

if `#1indent` is also set to `auto`, the `indent` of the current item is set to the wides Number of the same level.

```
        {\tpSetPropertyX{#1indent}{-\tpUseProperty{#1number-width-level-max}}}
```

otherwise it is set to the value of indent, or `0pt` if it was not set at all.

```
        {\tpIfProp{int-#1indent}
           {\tpSetPropertyX{#1indent}{\tpUseProperty{int-#1indent}}}
           {\tpSetProperty{#1indent}{\z@}}}%
```

the final value for `margin-left` is calculated by the `\tp@calc@margin@left` macro, above. It will be set to the sum of indent and

```
      \tp@calc@margin@left{#1}{#2}}
```

This branch is reached when the left margin is not set to `auto`.

```
     {\tpIfProp{int-#1margin-left}
       {\tpIfPropVal{int-#1indent}{auto}
```

If `margin-left` is set to a specific value and `indent` is set to `auto`, set the actual indent to the width of the level's widest Number.

```
         {\tpSetPropertyX{#1indent}{-\tpUseProperty{#1number-width-level-max}}}
         {\tpIfProp{int-#1indent}
```

Otherwise, if `indent` is set to a specific width, apply that value, or else set the inden to `0pt`.

```
            {\tpSetPropertyX{#1indent}{\tpUseProperty{int-#1indent}}}
            {\tpSetProperty{#1indent}{\z@}}}}
```

If `margin-left` is not set,

```
       {\tpIfPropVal{int-#1indent}{auto}
```

and `indent` is set to `auto`, set `margin-left` to the width of the level's widest Number and the actual `indent` to the negative of that.

```
         {\tpPropertyLetX{#1margin-left}{#1number-width-level-max}%
          \tpSetPropertyX{#1indent}{-\tpUseProperty{#1number-width-level-max}}}
         {\tpIfProp{int-#1indent}
```

If `margin-left` is not set, and `indent` is set to a specific value, apply that value for `indent` and set `margin-left` to `0pt`. In this branch, `indent` should have a positive value, otherwise the content would probably lap over the left edge of the type area.

```
            {\tpSetPropertyX{#1indent}{\tpUseProperty{int-#1indent}}%
             \tpSetProperty{#1margin-left}{\z@}}
```

otherwise set both `indent` nad `margin-left` to `0pt`.

```
            {\tpSetProperty{#1indent}{\z@}%
             \tpSetProperty{#1margin-left}{\z@}}}}}}
```

3.4 Label generation and selection

`\tpSetBabelLabel` defined a language-dependent string macro for German and English varieties. #1 is the language, #2 is the internal reference name, and #3 is the language specific label.

```
\def\tpSetBabelLabel#1#2#3{%
  \def\@lang{#1}%
  \expandafter\def\expandafter\@tempa\expandafter{\expandafter\def\csname #2name\endcsname{#3}}%
  \ifdefstring\@lang{german}{%
    \expandafter\addto\expandafter\captionsgerman\expandafter{\@tempa}%
    \expandafter\addto\expandafter\captionsngerman\expandafter{\@tempa}%
  }{%
    \ifdefstring\@lang{english}{%
      \expandafter\addto\expandafter\captionsbritish\expandafter{\@tempa}%
    \expandafter\addto\expandafter\captionsUKenglish\expandafter{\@tempa}%
    \expandafter\addto\expandafter\captionsenglish\expandafter{\@tempa}%
    \expandafter\addto\expandafter\captionsamerican\expandafter{\@tempa}%
    \expandafter\addto\expandafter\captionsUSenglish\expandafter{\@tempa}%
  }{}}}
```

3.5 Link Generation

\tpCompLink creates a hyperlink with the target taken from Component with the name #1 and the label #2.

```
\def\tpCompLink#1#2{%
  \protected@edef\@argi{\expandonce{\tpUseComp{#1}}}%
  \expandafter\href\expandafter{\@argi}{#2}%
}
```

\tpPageLabel enables referencing pages via **??**y using to create a hyperref anchor for label #1.

```
\def\tpPageLabel#1{\phantomsection\label{#1}}
```

```
%</common>
```

Modul 4

coco-accessibility.dtx

This file provides code for the interaction between the CoCoTEX framwork and the ltpfdfa package.

Please consider this module as highly experimental!

There are two files created from this dtx: one coco-accessibility.sty and one coco-accrssibility.lua.

1 LaTeX code

```
%<*a11y-sty>
```

1.1 General Processing

The coco-accessibility.sty starts with some general package information like name, current version and date of last changes.

```
%%
%% Accessibility features for \textit{xerif} projects.
%%
%% Maintainer: p.schulz@le-tex.de
%%
%% lualatex - texlive > 2018
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-accessibility}
   [2024/01/29 0.4.0 CoCoTeX accessibility module]
```

\tp@if@ally If the coco-ally package is loaded, the conditional from the coco-common module is re-defined to always expand the true branch and discard the false branch:

```
\def\tp@if@ally{\expandafter\@firstoftwo}
```

The ltpdfa package is a hard requirement for the accessibility features of CoCoTEX:

```
\RequirePackage[pdftex,pdflang=De]{ltpdfa}%,nodetree,dospaces,doparas,,debug
```

The local preferences for CoCoTEX's accessibility features is done via the tpMeta environment. Therefore, we hook the neccessary Components and Properties right into the titlepage container. Therefore, coco-title.sty is a hard requirement for CoCoTEX's accessibility module:

```
\RequirePackage{coco-title}%
```

1.2 Lua injection

Some features are realized by Lua code, so we tell LuaLaTeX to include the code that is generated from material later in this source file:

```
\directlua{ally = require('coco-accessibility')}
```

1.3 XMP Integration

The first feature of `coco-ally` is the integration of XMP meta data into the output PDF. Note that XMP integration is also a built-in feature of the `coco-title` module. The following code provides a superior alternative to that via the `ltpdfa` package.

`\tp@title@insert@xmp` is an override of the same macro in `coco-title.sty` (see. Sect. 2.4). If the `ally` document option is set, XMP inclusion is done via the `ltpdfa` package.

First we check if the specified xmp file exists. If it exists, the `DocumentInfo` is extracted from the XMP file. Otherwise, we set the `DocumentInfo` from the contents of the `titlepage` Container and let `ltpdfa` generate the `xmp` file.

```
\def\tp@title@insert@xmp{%
  \edef\tp@xmp@file@name{\tpUseGComp{titlepage}{XmpFile}.xmp}%
  \IfFileExists{\tp@xmp@file@name}
    {\addToConfig{metadata}{xmpfile=\tp@xmp@file@name}%
     \directlua{ally.meta.extract()}}
    {\tpPackageWarning{A11y}{File}{%
\tp@xmp@file@name\space not found.^^J
Note that the ltpdfa package will create one^^J
from the Components given in the tpMeta Container.}}}
```

1.4 Output Intent and ICC Profiles

First, we declare some Components that represent the three necessary parameters for the output intent:

```
\tpAddToType{Components}{titlepage}{%
```

Compoent titlepage::IccProfileFile holds the path (relative to the main tex file) and name of the .icc file.

```
  \tpDeclareGComp{IccProfileFile}
```

Compoent titlepage::IccComponents holds the number of components in the color profile

```
  \tpDeclareGComp{IccComponents}
```

Compoent titlepage::IccIdentifier holds the identifier of the color profile

```
  \tpDeclareGComp{IccIdentifier}}
```

The Components are composed via a new Property `output-intent` which we add to `coco-title`'s Properties list:

```
\tpAddToType{Properties}{titlepage}{%
```

`Property titlepage::output-intent` sends the output intent information to the ltpdfa package. It must contain of three data fields:

`profile` with the name of the to-be-embedded `.icc` file,
`componetns` with an integer telling the pdfwriter how many values are coded by each color (e.g., `4` for cmyk, `3` for rgb)
`identifier` with the identifying name of the profile (e.g., `Coated FOGRA39` for the included cmyk profile, etc.)

```
  \tpSetProperty{output-intent}{%
    profile=\tpIfComp{IccProfileFile}{\tpUseComp{IccProfileFile}}{suppl/\tp@color@enc.icc};%
    components=\tpIfComp{IccComponents}{\tpUseComp{IccComponents}}{\ifdefstring\tp@color@enc{
        cmyk}{4}{3}};%
    identifier=\tpIfComp{IccIdentifier}{\tpUseComp{IccIdentifier}}{\ifdefstring\tp@color@enc{
        cmyk}{Coated FOGRA39}{sRGB IEC61966-2.1}}%
  }}
```

The Component Handler which links the new Components to that Property is added to titlepage's *`document`*`-meta-hook`:

```
\tpAddToHook[titlepage]{document-meta-hook}{\edef\x{\noexpand\addToConfig{intent}{\tpUseProperty
    {output-intent}}}\x}
```

1.5 Transformation of Typographic Unicode characters

In order for screen readers to work correctly, some unicode characters that mask purely typographic glyphs (e.g., ligatures) need to be mapped to their underlaying orthographic characters. This is done via pdftex's `glyphtounicode` tables:

```
\protected\def\pdfglyphtounicode{\pdfextension glyphtounicode}
\input glyphtounicode
\edef\pdfgentounicode{\pdfvariable gentounicode}
\pdfgentounicode = 1
```

1.6 Encoding of the PDF-A Conformance

As before, the parameters for the PDF conformity level are encoded via specific Components in the titlepage Container:

```
\tpAddToType{Components}{titlepage}{%
```

`Compoent PDFAID::d` efines the PDF/A ID (Default: 2, meaning: PDF/A-2)

```
  \tpDeclareGComp[2]{PDFAID}%
```

`Compoent PDFALevel::d` efines the PDF/A Level (Default: A, meaning PDF/A-2A)

```
  \tpDeclareGComp[A]{PDFALevel}%
```

`Compoent PDFUAID::d` efines the PDF standard (Default: 1, meaning: PDF/UA-1). Use `\tpPDFUAID{}` (i.e. set it to nothing) to make the document conform to the PDF/A standard, but **not** to the PDF/UA standard.

```
  \tpDeclareGComp[1]{PDFUAID}}%
```

The checking if the values are valid, and the separation of the various parts of the standard is done via a lua script in the *document*-meta-hook. The conformance DocumentInfo nodes are only written, if *neither* PDFAID, *nor* PDFALevel is empty.

```
\tpAddToHook[titlepage]{document-meta-hook}{%
  \tpIfCompEmpty{PDFAID}{}{\tpIfCompEmpty{PDFALevel}{}{%
      \edef\x{\noexpand\setDocInfo{conformance}{%
          pdfaid=\tpUseComp{PDFAID};%
          level=\tpUseComp{PDFALevel}%
          \tpIfCompEmpty{PDFUAID}{}{;pdfuaid=\tpUseComp{PDFUAID}}}}%
      \x}}}
```

1.7 Automatic PDF Tagging

Document Root Node

The following code causes the ltpdfa package to tag the *document* environmant as the structural representation's root node:

```
\AtBeginShipout{\directlua{ltpdfa.pageprocessor(tex.box["AtBeginShipoutBox"])}}%
```

Some environments must *not* be auto-tagged by ltpdfa!

```
\tpIfAlly{%
  \ltOmitEnv{tpMeta}
  \ltOmitEnv{tpAuthor}
  \ltOmitEnv{tpEditor}
  \ltOmitEnv{tpSeriesEditor}
  \ltOmitEnv{tpAffil}
  \ltOmitEnv{tpFunding}
  \ltOmitEnv{heading}
}{}
```

End of TEX source code.

```
%</a11y-sty>
```

```
%<*a11y-lua>
```

2 Lua code

2.1 Local Variables and Tables

ltpdfa is an instance of the ltpdfa Lua table.

```
local ltpdfa = require('ltpdfa')
```

2.2 Meta Data Extraction

meta is a table that holds the metadata that are extracted from the \jobname.xmp file via its extract member.

```
local meta = {
  Author = '',
  Title = '',
  Creator = '',
  Producer = '',
  Keywords = '',
```

The method meta.extract() reads the meta data from the \jobname.xmp and stores certain values to be accessed by LaTeX. This is used to fill the DocumentInfo when a xmp file is available during the expansion of \tp@write@pdf@meta from the coco-title module (see Sect. 2).

```
  extract = function ()
    local xmpfile = ltpdfa.metadata.xmphandler.fromFile(ltpdfa.config.metadata.xmpfile)
    local f = io.open(xmpfile, "r")
    local content = f:read("*all")
    f:close()
    if (content:find('<dc:title>')) then
      Title = content:gsub('.*<dc:title>[^<]*<rdf:Alt>[^<]*<rdf:li[^>]*>(.*)</rdf:li>[^<]*</rdf:
          Alt>[^<]*</dc:title>.*', "%1")
      -- log(">>>" .. meta.Title)
    end
    local authors
    local author = {}
    if (content:find('<dc:creator>')) then
      authors = content:gsub('.*<dc:creator>[^<]*<rdf:Seq>(.*)</rdf:Seq>[^<]*</dc:creator>.*', "%
          1")
      for k in string.gmatch(authors, "<rdf:li>([^>]+)</rdf:li>") do
        table.insert(author , k)
      end
      Author = table.concat(author, ', ')
    end
  end
}
```

2.3 Public Methods

cocotex is the base table that contains all public methods and sub-tables available in the CoCoTEX framework. Here, it is defined unless it is already defined elsewhere.

```
if type(cocotex) ~= 'table' then
  cocotex = {}
end
```

cocotex.ally is a globally available namespace for coco-accessibility specific lua tables.

```
cocotex.ally = {
  meta = meta
}
```

After loading coco-accessibility.lua via the *require*() method, a cocotex.ally table is returned.

```
return cocotex.ally
```

no more lua code.

```
%</a11y-lua>
```

Modul 5

coco-meta.dtx

This file provides some macros that are used to process meta data, both for the whole document, as well as parts of a document.

```
%<*meta>
```

File preamble

```
%%
%% module for CoCoTeX that provides handling of a document's meta data.
%%
%% Maintainer: p.schulz@le-tex.de
%%
%% lualatex - texlive > 2019
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-meta}
   [2024/01/29 0.4.0 CoCoTeX meta module]
\RequirePackage{coco-common}
```

`Container CommonMeta` is an abstract Container for commonly used meta data, both for whole documents as well as parts of documents.

```
\tpDeclareContainer{CommonMeta}{%
  \tpDeclareType{Components}{%
    \tpDeclareRole[author]{Author}%
    \tp@declare@common@meta@comp
    \tp@extended@common@meta@macros
    \tp@declare@meta@affils
  }%
  \tpDeclareType{Properties}{}%
}
```

1 Counted Container Handlers

1.1 Generic Blocks

`\tp@meta@generic@comp` is used to define a generic meta data block.It provides two Components for each instance, one for the block's Heading and one for its Content.

```
\def\tp@meta@generic@comp{%
  \tpDeclareComp{GenericMetaBlock}{\expandafter\global}{}%
  \tpDeclareComponentGroup{tpGenericMeta}{%
    \tpDeclareCountedComp{Heading}%
    \tpDeclareCountedComp{Content}%
  }}
```

\tp@meta@generic@eval evaluates the Components and tells the Framework how the generic counted Sub-Containers should be rendered.

```
\def\tp@meta@generic@eval{{%
  \def\tp@cur@cont{titlepage}%
  \tpComposeCollection{tpGenericMeta}{generic-meta-format}{GenericMetaBlock}
}}
```

1.2 Contributor Roles

Contributors are counted sub-containers that represent the meta-data of people that share a role in contributing content to a document. Examples for such roles are an article/chapter/book's authors, or a collection/series' editors.

\tpDeclareRole is used to declare the Components that belong to each member of a contributor role. #2 is the name of the role, optional #1 is the internal name of the Role's formatting Property. If omitted, it is the same as #2.

The output of all members of a role is controlled by a Component called "<role>NameList" that is formatted according to the <role>-format Property. For reasons of naming conventions, the role names for a Component and its respective Property do not necessarily need to be identical.

```
\def\tpDeclareRole{\tp@opt@second\tp@declare@role}%
\def\tp@declare@role[#1]#2{%
  \tpDeclareComponentGroup{tp#2}{%
    \tpDeclareCountedComp{FullName}%
    \tpDeclareCountedComp{CiteName}%
    \tpDeclareCountedComp{ShortCiteName}%
    \tpDeclareCountedComp{PDFInfoName}%
    \tpDeclareCountedComp{Initial}%
    \tpDeclareCountedComp{LastName}%
    \tpDeclareCountedComp{FirstName}%
    \tpDeclareCountedComp{MidName}%
    \tpDeclareCountedComp{Honorific}%
    \tpDeclareCountedComp{Lineage}%
    \tpDeclareCountedComp{ORCID}%
    \tpDeclareCountedComp{AffilRef}% for references to the tpAffil Group
    \tpDeclareCountedComp{Affiliation}% for affiliations as direct tpAuthor meta data
    \tpDeclareCountedComp{Email}%
    \tpDeclareCountedComp{CorrespondenceAs}%
  }%
  \tpGroupHandler{tp#2}{%
    \tpIfComp{FullName}{}{\tpFullName{\tpUseProperty{#1-full-name-format}}}%
    \tpIfComp{Initial}{}{\tpInitial{\tpUseProperty{initials-format}}}%
    \tpIfComp{CiteName}{}{\tpCiteName{\tpUseProperty{#1-cite-name-format}}}%
    \tpIfComp{ShortCiteName}{}{\tpShortCiteName{\tpUseProperty{#1-short-cite-name-format}}}%
    \tpIfComp{PDFInfoName}{}{\tpPDFInfoName{\tpUseProperty{#1-pdfinfo-name-format}}}%
    \tpIfComp{CorrespondenceAs}{}{\tpCorrespondenceAs{\tpUseProperty{#1-correspondence-as-format
        }}}%
    \tpIfComp{AffilRef}{\tpIfComp{Affiliation}{%
        \tpPackageError{Meta}{Ambiguity}
          {You cannot use both Containers \string\tpAffilRef\space and \string\tpAffiliation\
              space in the same 'tp#2' Sub-Container}
          {At least one 'tp#2' Sub-Container contains both \string\tpAffilRef\space and \string\
              tpAffiliation. This is not allowed. Please decide for one affiliation strategy:
              Either two lists with cross-references, or affiliations directly as an author's
              meta-data.}}{}}{}%
  }%
  \tpDeclareRoleBlock{#2}{NameList}{#1-list-print-format}%
  \tpDeclareRoleBlock{#2}{CitationList}{#1-list-cite-format}%
  \tpDeclareRoleBlock{#2}{ShortCitationList}{#1-list-short-cite-format}%
  \tpDeclareRoleBlock[apply]{#2}{PDFInfo}{#1-list-pdfinfo-format}%
  \tpDeclareRoleBlock{#2}{Correspondence}{#1-list-correspondence-format}%
}
```

\tpAddToRole appends another Component declaration block #2 to a pre-defined Role #1.

```
\def\tpAddToRole#1#2{%
  \csgappto{@tp#1@hook}{#2}%
}
```

\tpDeclareRoleBlock is used to create a new output container (named \tp#2#3) for a given Role #2. A Role Block is a Component of the parent Container which contains certain Components of all members of the Role within its parent Container. Format and selection of the utilised Components are specified via the Property given in #4. The optional argument #1 tells the evaluator in the Container's end macro how the collector is to be composed. Valid values are compose (default) or apply.

```
\def\tpDeclareRoleBlock{\@ifnextchar[\tp@declare@role@block{\tp@declare@role@block[compose]}}%]
\def\tp@declare@role@block[#1]#2#3#4{%
  \ifcsdef{tp@meta@role@#1}
    {\tpDeclareComp{#2#3}{\expandafter\global}{}%
     \csgdef{tp@meta@role@\tp@cur@cont @#2@#3}{#4}%
     \csappto{@tp@meta@role@eval@\tp@cur@cont @#2}
        {\csname tp@meta@role@#1\endcsname{#2}{#3}}}
    {\tpPackageError{Meta}{Argument}
      {Invalid optional argument in \string\tpDeclareRoleBlock!}
      {Only ‘apply’ or ‘compose’ are allowed as values^^Jin the optional argument of \string\
           tpDeclareRoleBlock!}}}%
```

\tp@meta@role@eval creates the name lists for the role. #1 is the name of the role.

```
\def\tp@meta@role@eval#1{\csname @tp@meta@role@eval@\tp@cur@cont @#1\endcsname}
```

\@tp@meta@role@eval #1 is the name of the macro used to compose the Collection (either \tpComposeCollection, or \tpApplyCollection), #2 is the name of the role and #3 is the name of the list. The access Component is #2#3, i.e., both argumets together.

```
\def\@tp@meta@role@eval#1#2#3{%
```

First, we check if the Collection Component has already been set in the input. If so, we set an internal flag to indicate that the Collection Component has been filled manually.

```
  \tpIfComp{#2#3}{\cslet{tp@used@#2#3@override}\@empty}{%
```

Second, we check if the counter for the Role is defined and greater than 0. If neither is the case, this means that the Group does not occur in the input, at all, so we don't need to do anything.

```
    \ifcsdef{tp#2Cnt}
      {\expandafter\ifnum\csname tp#2Cnt\endcsname>\z@
```

otherwise, we call the Property that is stored in \tp@meta@role@\tp@cur@cont @#2@#3 and store the result in the Component #2#3.

```
        #1{tp#2}{\csname tp@meta@role@\tp@cur@cont @#2@#3\endcsname}{#2#3}%
      \fi
    }{}}}
```

`\tp@meta@role@apply` #1 is the name of the role and #2 is the name of the composition. This macro applies (i.e. *fully expands*) the `\tp@meta@role@\tp@cur@cont @#1@#2` Property and stores the result in the `#1#2` Component.

```
\def\tp@meta@role@apply#1#2{\@tp@meta@role@eval\tpApplyCollection{#1}{#2}}
```

`\tp@meta@role@compose` #1 is the name of the role and #2 is the name of the composition. This stores the *unexpaded* contents of the `\tp@meta@role@\tp@cur@cont @#1@#2` Property in the `#1#2` Component.

```
\def\tp@meta@role@compose#1#2{\@tp@meta@role@eval\tpComposeCollection{#1}{#2}}
```

2 Labeled Components

`\tpDeclareLabeledComp` declares two Components: one named `\csname tp#2\endcsname` for the value and another one named `\csname tp#2Label\endcsname` for its corresponding label. #3 is used for property overrides. The optional Argument #1 allows to set a default value for the Label.

```
\def\tpDeclareLabeledComp{\tp@opt@empty\tp@declare@labeled@comp}
\def\tp@declare@labeled@comp[#1]#2#3{%
  \tpDeclareComp{#2}{\expandafter\global}{}%
  \tpDeclareComp{#2Label}{\expandafter\global}{}%
  \csxdef{labeled-meta-property-infix-\tp@cur@cont-#2}{#3}%
  \if!#1!\else
    \long\csgdef{tp@\tp@cur@cont @#2Label}{#1}%
  \fi
}
```

`\tpUseLabeledComp` declares two Components: one named `\csname tp#1\endcsname` for the value and another one named `\csname tp#1Label\endcsname` for its corresponding label. An optional Argument allows to set a default value for the Label.

```
\def\tpUseLabeledComp#1{%
  \tpIfComp{#1}{%
```

`\tpCurInfix` stores the currently active property infix for the Labeled Component

```
    \letcs\tpCurInfix{labeled-meta-property-infix-\tp@cur@cont-#1}%
```

`\tpCurComp` stores the currently active Component name

```
    \def\tpCurComp{#1}%
```

```
    \tpIfProp{labeled-meta-\tpCurInfix-format}
      {\tpUseProperty{labeled-meta-\tpCurInfix-format}}
      {\tpUseProperty{labeled-meta-format}}%
  }{}}
```

3 common meta data

`\tp@declare@common@meta@comp` defines some commonly used meta Components

```
\def\tp@declare@common@meta@comp{%
   \tpDeclareComp{Copyright}{\expandafter\global}{}% Copyright text
   \tpDeclareComp{DOI}{\expandafter\global}{}% DOI
}%
```

`\tp@extended@common@meta@macros` provides some extended markup. Some headings use these Components for compilations of contributions by different authors. They are also loaded by article title pages.

```
\def\tp@extended@common@meta@macros{%
  \tpDeclareLabeledComp[Abstract]{Abstract}{abstract}%
  \tpDeclareLabeledComp[Keywords]{Keywords}{keyword}%
  \tpDeclareLabeledComp{DOI}{doi}%
  \tpDeclareLabeledComp{TitleEn}{title-en}%
}
```

3.1 Affiliations

`\tp@meta@affils` is a wrapper that creates the user-level macros for the affiliations.

```
\def\tp@declare@meta@affils{%
  \tpDeclareComp{AffilBlock}{\expandafter\global}{}%
  \tpDeclareComponentGroup{tpAffil}{%
    \tpDeclareCountedComp{Affiliation}%
    \tpDeclareCountedComp{Address}%
    \tpDeclareCountedComp{Institute}%
    \tpDeclareCountedComp{Country}%
    \tpDeclareCountedComp{Department}%
    \tpDeclareCountedComp{AffilID}%
  }%
  \tpGroupHandler{tpAffil}{%
    \tpIfComp{AffilID}{}{\expandafter\tpAffilID\expandafter{\tpAffilCnt}}%
    \tpIfComp{Affiliation}{}{\tpAffiliation{\tpUseProperty{affiliation-format}}}%
  }%
}
```

Defaut Property settings for the Meta Container.

```
\tpAddToDefault{CommonMeta}{%
  \tpSetProperty{initials-format}{%
    \expandafter\ifx\csname tp@\tp@cur@cont @\tp@cnt@grp-FirstName-\the\tpCurCount\endcsname\
        long@empty\else
      \expandafter\ifx\csname tp@\tp@cur@cont @\tp@cnt@grp-FirstName-\the\tpCurCount\endcsname\
          relax\else
        \expandafter\expandafter\expandafter\@car\csname tp@\tp@cur@cont @\tp@cnt@grp-FirstName-\
            the\tpCurCount\endcsname\relax\@nil\tpUseProperty{initials-period}%
      \expandafter\ifx\csname tp@\tp@cur@cont @\tp@cnt@grp-MidName-\the\tpCurCount\endcsname\
          long@empty\else
        \expandafter\ifx\csname tp@\tp@cur@cont @\tp@cnt@grp-MidName-\the\tpCurCount\endcsname\
            relax\else
          \tpUseProperty{initials-sep}%
          \expandafter\expandafter\expandafter\@car\csname tp@\tp@cur@cont @\tp@cnt@grp-MidName-\
              the\tpCurCount\endcsname\relax\@nil\tpUseProperty{initials-period}%
        \fi\fi
    \fi\fi
  }
  \tpSetProperty{initials-sep}{~}
  \tpSetProperty{initials-period}{.}
```

```
%
%% Properties that control how the composed compoents WITHIN each item in a Role are formatted:
%
\tpSetProperty{role-full-name-format}{%
  \if\tpUseComp{Honorific}\relax
  \else
    \tpUseComp{Honorific}\space
  \fi
  \tpUseComp{FirstName}\space
  \if\tpUseComp{MidName}\relax
  \else
    \tpUseComp{MidName}\space
  \fi
  \tpUseComp{LastName}%
  \if\tpUseComp{Lineage}\relax
  \else
    \space\tpUseComp{Lineage}%
  \fi%
}% How FullName for each name is built
\tpSetProperty{role-cite-name-format}{\tpIfComp{LastName}{\tpUseComp{LastName},~\tpUseComp{Initial}}{\tpUseComp{FullName}}}% How CiteName for each name is built
\tpSetProperty{role-short-cite-name-format}{\tpUseComp{LastName}}% how ShortCiteName for each name is built
\tpPropertyLet{role-pdfinfo-name-format}{role-cite-name-format}% How PDFInfoName for each item is built
\tpSetProperty{role-correspondence-as-format}{\tpUseComp{Email}}% How PDFInfoName for each item is built
%% Properties that control how the single items in a compoent list are formatted:
\tpSetProperty{role-block-print-format}{\tpUseComp{FullName}\ifnum\tpCurCount<\tpTotalCount\tpUseProperty{counted-name-sep}\fi}% How <Role>NameList for each name is build
\tpSetProperty{role-block-cite-format}{\tpUseComp{CiteName}\ifnum\tpCurCount<\tpTotalCount\tpUseProperty{counted-name-sep}\fi}% How each item in Component <Role>CitationList is formatted
\tpSetProperty{role-block-short-cite-format}{\tpUseComp{ShortCiteName}\ifnum\tpCurCount<\tpTotalCount\tpUseProperty{counted-name-sep}\fi}% How each item in the Component <Role>ShortCitationList is formatted
\tpSetProperty{role-block-pdfinfo-format}{\tpUseComp{PDFInfoName}\ifnum\tpCurCount<\tpTotalCount\tpUseProperty{counted-name-sep}\fi}% How each item in the Component <Role>PDFInfo is formatted
\tpSetProperty{role-block-correspondence-format}{%
  \tpIfAttrIsset{\tp@cnt@grp-\the\tpCurCount}{corresp}
    {\ifx\is@first@corresp\relax
       \tpUseProperty{corresp-sep}%
     \else
       \global\let\is@first@corresp\relax
     \fi
     \tpUseComp{CorrespondenceAs}%
   }{}}% How each item in the Component <Role>Correspondence is formatted
% Aliasses
% for Role "Author":
\tpPropertyLet{author-cite-name-format} {role-cite-name-format}%
\tpPropertyLet{author-short-cite-name-format} {role-short-cite-name-format}%
\tpPropertyLet{author-full-name-format} {role-full-name-format}%
\tpPropertyLet{author-pdfinfo-name-format} {role-pdfinfo-name-format}%
\tpPropertyLet{author-correspondence-as-format} {role-correspondence-as-format}%
%
\tpPropertyLet{author-list-print-format} {role-block-print-format}%
\tpPropertyLet{author-list-cite-format} {role-block-cite-format}%
\tpPropertyLet{author-list-short-cite-format} {role-block-short-cite-format}%
\tpPropertyLet{author-list-pdfinfo-format} {role-block-pdfinfo-format}%
\tpPropertyLet{author-list-correspondence-format} {role-block-correspondence-format}%
%
```

```
\tpSetProperty{counted-name-sep}{,\space}%
\tpSetProperty{name-and}{\space and\space}%
\tpSetProperty{name-etal}{\space et~al.}%
\tpSetProperty{name-sep}{,\space}%
\tpSetProperty{corresp-mark}{*}%
\tpSetProperty{corresp-sep}{,\space}%
%
% Affiliation Properties
%
\tpSetProperty{affiliation-format}{% Format of the affiliation block
  \tpIfComp{Institute}{\tpUseComp{Institute}}{}%
  \tpIfComp{Department}{, \tpUseComp{Department}}{}%
  \tpIfComp{Address}{, \tpUseComp{Address}}{}%
}%
\tpSetProperty{affil-sep}{\par}
\tpSetProperty{affil-block-item-face}{}% Font of a single item in the affiliation list
\tpSetProperty{affil-block-item-format}{% Format of a single item in the affiliation list
  \textsuperscript{\tpUseComp{AffilID}}%
  \bgroup
    \tpUseProperty{affil-block-item-face}%
    \tpUseComp{Affiliation}
  \egroup%
  \ifnum\tpCurCount<\tpTotalCount\relax\tpUseProperty{affil-sep}\fi%
}
\tpSetProperty{affil-block-face}{\small\normalfont}%
\tpSetProperty{affil-block-format}{%
  \tpIfComp{AffilBlock}
    {\bgroup
       \tpUseProperty{affil-block-face}%
       \tpUseComp{AffilBlock}%
     \egroup
     \par
   }{}}
%
% Labeled Meta Properties
%
\tpSetProperty{labeled-meta-format}{%
  \tpIfProp{labeled-meta-before-\tpCurInfix}
    {\tpUseProperty{labeled-meta-before-\tpCurInfix}}
    {\tpUseProperty{labeled-meta-before}}%
  \bgroup
    \tpIfProp{labeled-meta-\tpCurInfix-face}
      {\tpUseProperty{labeled-meta-\tpCurInfix-face}}
      {\tpUseProperty{labeled-meta-face}}%
    \tpIfProp{labeled-meta-\tpCurInfix-label-format}
      {\tpUseProperty{labeled-meta-\tpCurInfix-label-format}}
      {\tpUseProperty{labeled-meta-label-format}}%
    \tpUseComp{\tpCurComp}%
  \egroup
  \tpIfProp{labeled-meta-after-\tpCurInfix}
    {\tpUseProperty{labeled-meta-after-\tpCurInfix}}
    {\tpUseProperty{labeled-meta-after}}%
}
\tpSetProperty{labeled-meta-label-format}{%
  \tpIfComp{\tpCurComp Label}{%
    \bgroup
      \tpUseProperty{labeled-meta-before-\tpCurInfix-label}%
      \tpIfProp{labeled-meta-\tpCurInfix-label-face}
        {\tpUseProperty{labeled-meta-\tpCurInfix-label-face}}
        {\tpUseProperty{labeled-meta-label-face}}%
      \tpUseComp{\tpCurComp Label}%
      \tpIfProp{labeled-meta-\tpCurInfix-label-sep}
        {\tpUseProperty{labeled-meta-\tpCurInfix-label-sep}}
        {\tpUseProperty{labeled-meta-label-sep}}%
    \egroup
  }{}}
  \tpSetProperty{labeled-meta-label-face}{\bfseries}
  \tpSetProperty{labeled-meta-label-sep}{:\enskip}
  \tpSetProperty{labeled-meta-face}{}
  \tpSetProperty{labeled-meta-before}{}
  \tpSetProperty{labeled-meta-after}{\par}
}
```

```
%</meta>
```

Part II

Document Level Structures

Modul 6

coco-headings.dtx

This module provides handlers for headings like parts, chapters, sections, or inline headings common to all CoCoTeX projects.

```
%<*headings>
```

```
%%
%% module for CoCoTeX that extends heading objects.
%%
%% Maintainer: p.schulz@le-tex.de
%%
%% lualatex - texlive >= 2019
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-headings}
    [2024/01/29 0.4.0 CoCoTeX headings module]
\RequirePackage{coco-meta}
```

Headings are handled differently with `cocotex.cls` compared to standard LaTeX, since cocotex manuscripts tend to have a whole collection of additional information that are pressed into the headings, like subtitles or section authors down to subsection level, etc. Therefore, the `\@startsection` and `\@make[s]chapterhead` facilities from LaTeX are no longer sufficient. At the same time, the package does not redefine those macros and keeps them available for backwards compatibility.

First, we load the `bookmark` package:

```
\RequirePackage{bookmark}%
```

Since we use our own heading levels, we disable all automatically generated bookmarks.

```
\hypersetup{bookmarksdepth=-999}%
```

1 Facility for declaring heading levels and their layouts

Container heading

```
\tpDeclareContainer{heading}{%
  \tpInherit{Components,Properties}{CommonMeta}%
  \tpDeclareType{Components}{%
```

We already have the Author Component inherited from the `CommonMeta` Container. We therefore just need to declare the overrides.

```
    \tp@heading@authors%
```

The remaining Components are built as usual.

```
    \tp@provide@hd@macros{Title}%
    \tp@provide@hd@macros{Subtitle}%
    \tp@provide@hd@macros{Number}%
    \tp@provide@hd@macros{LicenceLogo}%
    \tp@provide@hd@macros{LicenceName}%
    \tp@heading@quotes
  }%
  \tpDeclareType{Properties}{}%
  \tpDeclareEnv{\heading}{\endheading}%
}
```

`\tpDeclareHeading` is the user-level macro to declare new headings.

#1 (optional) inherit-from: load all properties from that heading level, first.
#2 level: used for toc entries. -1 for part, 0 for chapter, 1 for section, etc.
#3 name: part, chapter, section, etc, to be used in toc, head lines, bookmarks, etc.
#4 Property definitions and switches

```
\long\def\tpDeclareHeading{\tp@opt@empty\@tpDeclareHeading}
\long\def\@tpDeclareHeading[#1]#2#3#4{%
```

First, we check if the heading has already been declared.

```
  \ifcsdef{tp@container@#3}{%
```

If yes, then we check if the new declaration's parameters match with the pre-existing one. We start with the heading level.

```
    \tpPackageInfo{Headings}{}{Appending to ‘#3’}%
    \ifcsstring{tp@hdg@#3@level}{#2}{}{%
      \tpPackageError{Headings}
        {Level Mismatch}
        {Level of heading ‘#3’ cannot be altered!}
        {The already existing heading ‘#3’ has toc level ‘\csname tp@hdg@#3@level\endcsname’,
            but your^^J%
         re-declaration states ‘#2’.^^J%
         ^^J%
         Consider declaring a new heading alltogether with ‘#3’ as parent,^^J%
         or add Properties to ‘#3’ using \string\tpAddToType\string{Properties\string}\string
             {#3\string}.}%
     }%
```

we also check the parent.

```
    \if!#1!\else
      \ifcsstring{tp@parent@#3}{#1}{}{%
        \tpPackageError{Headings}
          {Parent Mismatch}
          {Parent of heading ‘#3’^^J cannot be altered!}
          {The already existing heading ‘#3’ inherits from ‘\csname tp@parent@#3\endcsname’,^^J%
           but your re-declaration sets Parent to ‘#1’.^^J%
           ^^J%
           Consider declaring a new heading alltogether with ‘#1’ as parent.}%
      }%
    \fi
```

and finally pass the new Properties to the existing heading.

```
    \tpAddToType{Properties}{#3}{#4}%
```

Finally, we need to re-define the \tpUseHeading macro so that changes to the heading's Property list will be taken into account for all dependend constructions like list-ofs and toc-entries.

```
    \tp@declare@heading{#2}{#3}%
  }{%
```

If the heading does not already exist, we build a new one.

Each new heading constitutes its own Sub-Container of the heading Container. The name of this Sub-Container is the headings name.

```
    \tpDeclareContainer{#3}{%
      \csgdef{tp@hdg@#3@level}{#2}%
      \tpPackageInfo{Headings}{}{Declaring heading ‘#3’}%
      \edef\@argi{#1}%
      \tpDeclareType{Parent}{\tp@heading@create@parent{#1}{#3}}
```

We inherit everything from the heading levels parent, or from the default heading if no parent is present.

```
      \ifx\@argi\@empty
        \tpInherit{Components,Properties}{heading}%
      \else
        \tpInherit{Components,Properties,Parent}{#1}%
      \fi
```

The main body of the heading Declaration is a list of Property definitions which we append to the Sub-Container's "Property" Type.

```
      \tpDeclareType{Properties}{%
        #4%
      }%
```

For each heading we declare some common macros like the ToC entry handlers, the heading's counters and its hooks.

```
      \tpDeclareType{Init}{%
        \tp@init@hooks{#3}%
        \let\@tp@cur@cont\tp@cur@cont
        \def\tp@cur@cont{heading}%
        \tp@init@l@{toc}{#2}{#3}%
        \let\tp@cur@cont\@tp@cur@cont
        \tp@init@cnt{#3}%
      }%
```

Unlike other Sub-Containers, headings form no own LaTeX environment. Instead, headings are specifications of one common heading environment. Is is outsourced into the internal \tp@declare@heading macro, which is defined below.

The reason for that is that we don't want to define versions of the same property macros for each and every single heading level. Instead, we locally re-define the general low-level macros that represent the heading's properties for each instance of the generalised heading container.

```
      \tp@declare@heading{#2}{#3}%
    }%
  }%
}
```

\tp@heading@create@parent stores the heading level's name and its parent, if it exists.

```
\def\tp@heading@create@parent#1#2{%
```

```
  \def\tp@heading@name{#2}%
  \if!#1!\else
    \tpCheckParent{#1}{#2}%
  \fi%
}
```

`\tp@declare@heading` consists of two parts: In the first part, the inheritance mechanism and the initializers for each new heading level are triggered.

```
\def\tp@declare@heading#1#2{%
  \tpEvalType{Parent}%
  \tpEvalType{Init}%
```

`\tpUseHeading` is defined as second step. It is called at the end of each `heading` environment to process the Components within the Container instance. Each heading level has its own “version” of this macro.

```
  \csgdef{tpUseHeading#2}{%
```

Since heading levels don’t define their own environments, we make sure that `heading` is the namespace we are working in.

```
    \tpNamespace{heading}%
    \@setpar{\@@par}%
```

Properties are stored in macros specific to the current heading Sub-Container, therefore we evaluate the level’s Properties, not those of the `heading` Container. However, since we made use of the inheritance mechanism earlier, each Sub-Container’s Property list also contains the general `heading` Property list.

```
    \def\tpHeadingLevel{#1}%
    \tpEvalType[#2]{Properties}%
```

Processing the author name list (from coco-meta.sty).

```
    \tp@meta@role@eval{Author}%
    \tpComposeCollection{tpAuthor}{author-contact-block-format}{AuthorContactBlock}%
    \tpComposeCollection{tpAffil}{affil-block-format}{AffilBlock}%
```

Processing the `tpQuote` environments, if any.

```
    \tpComposeCollection{tpQuote}{quote-block-format}{QuoteBlock}%
```

Hyperref related stuff.

```
    \def\Hy@toclevel{#1}%
```

Call the mechanism to calculate the heading’s counter.

```
    \tp@auto@number{#1}{#2}%
```

Here, the actual construction of the heading begins.

```
    \tpUseProperty{heading-par}%
    \tp@hd@use@hook{before-hook}{#2}%
    \tpUseProperty{before-heading}%
```

Add vertical space before the heading

```
    \tp@do@before@skip
```

The counters we calculated earlier and the space needed to render them are evaluated

```
  \tp@format@number{}{}{#1}%
```

The value of after-skip is essential to determine whether the heading is to be displayed as block or inline element. In case, some heading definition omits setting a proper value, we build a fallback.

```
  \tpIfProp{after-skip}{\expandafter\global\expandafter\@tempskipa\expandafter=\tpUseProperty{
      after-skip}\relax}{\global\@tempskipa=1sp\relax}%
  \tp@hd@use@hook{before-print-hook}{#2}%
  \def\@svsec{%
```

The `heading block` is the composition of all of the heading's Components that are to be printed where the `heading` environment is in the source.

```
    \tpUseProperty{before-heading-block}%
```

Labels to be used with LaTeX's cross reference mechanism are defined

```
    \tp@heading@create@labels{#2}% label facility
    \leftskip\tpUseProperty{margin-left}%
    \rightskip\tpUseProperty{margin-right}%
    \bgroup
      \tpUseProperty{heading-block}%
```

Generate entries for ToC, bookmarks and page headers. This has to be here because in rare cases, abstracts could cause the whole heading to spread over more than one page and that results in the ToC entry pointing to the last page.

Style progammers need to make sure that no page breaks are allowed within the heading-block!

```
      \tpIfPropVal{no-toc}{true}{}{\tp@make@toc}% ToC entries
      \tpIfPropVal{no-BM}{true}{}{\tp@make@bookmarks}% Bookmarks
      \tpUseProperty{toc-hook}%
      \tpIfProp{extended}{\tpUseProperty{extended-heading}}{}%
    \egroup%
    \tp@make@run% Running headers
    \tpUseProperty{after-heading-block}%
  }%
```

Finally, we decide whether the printable material we stored in `\@svsec` is to be rendered as a block or inline. This is adopted from LaTeX's `\@startsection`. The distinction is made by the sign of `after-skip`: a positive value yields a block heading, a negative value yields an inline heading.

```
  \ifdim\@tempskipa <\z@\relax
    \tp@inline@heading
  \else
    \tp@block@heading
  \fi
```

This macro is called at the end of the heading environment. In order to deal with possible vertical spaces after the heading, we wait until the group of the heading environemnt is closed before we actually print the fully composed heading. The definition of `\next` happens in either `\tp@inline@heading` or `\tp@block@heading`.

```
  \aftergroup\next%
 }%
}
```

`\tp@hd@use@hook` recursively includes a hook #1 from the heading #2's parent before expanding its own version.

```
\def\tp@hd@use@hook#1#2{%
  \expandafter\ifx\csname tp@parent@#2\endcsname\relax\else
    \edef\@@parent{#1-\csname tp@parent@#2\endcsname}%
    \expandafter\tpUseHook\expandafter{\@@parent}%
  \fi
  \tpUseHook{#1-\tp@heading@name}%
}
```

`\tp@do@before@skip` is a routine that determins the skip that is inserted before a heading.

```
\def\tp@do@before@skip{%
  \setlength\@tempskipa{\tpUseProperty{before-skip}}%
  \ifdim\@tempskipa<\z@\relax
    \def\do@skip{\minusvspace{-\@tempskipa}}%
  \else
    \def\do@skip{\addvspace{\@tempskipa}}%
  \fi%
  \if@nobreak
    \everypar{}%
    \do@skip
  \else
    \addpenalty\@secpenalty
    \do@skip
  \fi}
```

1.1 Initializers for New Heading Levels

`\tp@init@hooks` initializes the Hooks for heading level #1.

```
\def\tp@init@hooks#1{%
  \tpDeclareHook{toc-before-hook-#1}% Expanded before the toc entry is printed
  \tpDeclareHook{toc-after-hook-#1}% Expanded after the toc entry is printed
  \tpDeclareHook{before-hook-#1}% Expanded before before-heading property is expanded
  \tpDeclareHook{before-print-hook-#1}% Expanded at the very beginning of the local definition of \
      @svsec
}
```

`\tp@init@cnt` initialises a counter with the name #1 for automatic numbering if it doesn't exist, yet.

```
\def\tp@init@cnt#1{\ifcsname c@#1\endcsname\else\@definecounter{#1}\fi}
```

1.2 Initializers for Instances of Heading Levels

`\tp@auto@number` advances the heading counter if the `numbering` Property is set to `auto` and the current heading is not overridden by the `Number` Component. #1 is the numeric level of the heading, #2 is the name of the heading's counter.

```
\def\tp@auto@number#1#2{%
  \tpIfPropVal{numbering}{auto}
    {\expandafter\ifx\csname c@#2\endcsname\relax\tp@init@cnt{#2}\fi
     \tpIfAttrIsset{heading}{nonumber}
      {}
      {\tpIfComp{Number}
        {}
        {\ifnum #1>\c@secnumdepth\relax\else
          \stepcounter{#2}%
          \edef\@tempa{\csname the#2\endcsname}%
          \expandafter\tpNumber\expandafter{\@tempa}%
        \fi}}
    }{}}
```

1.3 Label mechanism

`\@tp@heading@parse@label` separates multiple comma-separated values within the same `label` attribute.

```
\def\@tp@heading@parse@label#1,#2,\@nil{%
  \@tp@heading@create@labels{#1}%
  \if!#2!\else
    \@tp@heading@parse@label#2,\@nil
  \fi}
```

`\tp@heading@create@labels` is the wrapper to handle multiple values in the `label` Attribute.

```
\def\tp@heading@create@labels#1{%
  \ifx\Hy@MakeCurrentHrefAuto\@undefined\else
    \Hy@MakeCurrentHrefAuto{tp.#1}%
    \Hy@raisedlink{\hyper@anchorstart{\@currentHref}\hyper@anchorend}%
  \fi
  \tpIfAttr{heading}{label}
    {\expandafter\@tp@heading@parse@label\tp@heading@attr@label,,\@nil}{}}
```

`\@tp@heading@create@labels` generates the labels to be used with LaTeX's cross reference and `hyperref`'s hyperlink mechanisms, simultanuously. This macro locally redefines LaTeX's `\label` macro and sets both `\@currentlabel` as well as a `\hyperlink` target.

```
\def\@tp@heading@create@labels#1{%
  \if!#1!\else
    \tpIfComp{Number}
      {\edef\@tempa{\expandonce{\tp@heading@Number}}%
       \let\@currentlabel\@tempa\relax
       \let\@currentlabelname\tp@heading@Title}
```

in case, un-numbered headings receive a `label` to be accessed via `\pageref` or something:

```
      {\phantomsection}%
    \expandafter\hypertarget\expandafter{#1}{}%
    \expandafter\tpltx@label\expandafter{#1}%
  \fi
  \global\let\label\tpltx@label}
```

2 Externalisation of Heading Compoents

Components of headings may be used far away from the heading itself. Since, by design, Components are defined strictly local within their containers, those externale usages demand special treatment.

2.1 Common Stuff

`\tp@check@author` checks if the `AuthorNameList` override Component is given in the input for any given output override prefixed by #1. If not, it is built if there are any Author subcontainers, at all.

```
\def\tp@check@author#1{%
  \tpIfComp{#1AuthorNameList}{}{%
    \tpIfComp{AuthorNameList}{%
      \expandafter\csname tp#1AuthorNameList\expandafter\endcsname\expandafter{\
          tp@heading@AuthorNameList}%
    }{\ifnum\tpAuthorCnt>\z@
        \tpCompDef\tp@tempa{tpAuthor}{author-list-format}%
        \ifx\tp@tempa\relax
        \else
          \expandafter\csname tp#1AuthorNameList\expandafter\endcsname\expandafter{\tp@tempa}%
        \fi
      \fi
    }}}%
```

2.2 Table of Contents Entry

`\tp@make@toc` initializes the creation of a `heading` instance's entry in the table of contents.

Each entry is in itself treated as a Container. As such, it consists of Components that are written into the .toc file.

```
\def\tp@make@toc{%
  \tp@check@empty{heading}{Title}{Toc}%
  \tp@check@empty{heading}{Number}{Toc}%
  \tp@check@empty{heading}{Subtitle}{Toc}%
  \tp@check@author{Toc}%
  \tpIfAttrIsset{heading}{notoc}
    {}
    {\protected@edef\tp@heading@toc@entry{%
       \tpIfComp{TocTitle}{\string\tpTocTitle{\string\ignorespaces\space\expandonce{\
           tp@heading@TocTitle}}}{}%
       \tpIfComp{TocNumber}{\string\tpTocNumber{\string\ignorespaces\space\expandonce{\
           tp@heading@TocNumber}}}{}%
       \tpIfComp{TocAuthorNameList}{\string\tpTocAuthorNameList{\string\ignorespaces\space\
           expandonce{\tp@heading@TocAuthorNameList}}}{}%
       \tpIfComp{TocSubtitle}{\string\tpTocSubtitle{\string\ignorespaces\space\expandonce{\
           tp@heading@TocSubtitle}}}{}%
     }%
     \tpIfProp{toc-level}{\edef\tp@heading@name{\tpUseProperty{toc-level}}}{}%
     \protected@write\@auxout
       {\tpGobble}%
       {\string\@writefile{toc}{\protect\tpContentsline{\tp@heading@name}{\tp@heading@toc@entry
           }{\thepage}{\@currentHref}\protected@file@percent}}\relax
    }
}
```

`\tp@toc@extract@data` is called within the `\l@<level>` macro to extract the Components for each entry in the .toc file. #1 is the numerical heading level, #2 is the name of the heading level, #3 is the content of the toc entry (which holds the Components), #4 is the page number.

```
\def\tp@toc@extract@data#1#2#3#4{%
  \tpNamespace{heading}%
  \tpEvalType[#2]{Properties}%
  \tpDeclareComp{TocPage}{}{}%
  \tpTocPage{\tpUseProperty{toc-page-face}#4}%
  \tpDeclareComp{TocTitle}{}{}%
  \tpDeclareComp{TocSubtitle}{}{}%
  \tpDeclareComp{TocNumber}{}{}%
  \tpDeclareComp{TocAuthorNameList}{}{}%
  \tp@expand@l@contents{#3}{heading}{Toc}{Title}%%
  \tp@format@number{toc-}{Toc}{#1}%
}
```

`\tp@toc@print@entry` is also called within the `\l@<level>` macro and eventually prints the entry by expanding a `heading`'s toc-specific Properties.

```
\def\tp@toc@print@entry#1{%
  \bgroup
    \tpUseHook{toc-before-hook-#1}%
    \tpUseProperty{toc-before-entry}%
    \tpUseProperty{toc-format}%
    \tpUseHook{toc-after-hook-#1}%
    \tpUseProperty{toc-after-entry}%
  \egroup}
```

2.3 Facility to create the running title macros

`\tp@make@run` prepares the Components used to compose the running titles. It checks if the user provides page header specific overrides in the `heading` instance. If not, it uses the non-specific Components instead, as long as they are not empty.

After all the header-specific Components are set, the heading level specific property `running-heading` is evaluated and passed to the corresponding `\<level>mark` macros iff they exist.

```
\def\tp@make@run{%
  \tp@check@empty{heading}{Title}{Run}%
  \tp@check@empty{heading}{Number}{Run}%
  \tp@check@author{Run}%
  \tp@check@empty{heading}{Subtitle}{Run}%
  \tpUseProperty{running-extra}%
  \tpIfProp{running-level}
    {\letcs\tp@mark@name{\tpUseProperty{running-level}mark}}
    {\letcs\tp@mark@name{\tp@heading@name mark}}%
    \letcs\tp@heading@parent{tp@parent@\tp@heading@name}%
    \ifx\tp@mark@name\@undefined
      \ifx\tp@heading@parent\relax\else
        \letcs\tp@mark@name{\tp@heading@parent mark}%
      \fi
    \fi
  \ifx\tp@mark@name\@undefined\else
    \begingroup
      \tpGobble
      \protected@edef\@tempa{\csname tp@heading@running-heading\endcsname}%
      \expandafter\tp@mark@name\expandafter{\@tempa}%
    \endgroup
  \fi
}
```

2.4 Facility to create PDF bookmarks

\tp@make@bookmarks generates an entry that is directly written as Bookmark into the PDF file. This is done using the bookmark package.

```
\def\tp@make@bookmarks{%
  \tp@check@empty[Toc]{heading}{Title}{BM}%
  \tp@check@empty[Toc]{heading}{Number}{BM}%
  \tp@check@empty[Toc]{heading}{AuthorNameList}{BM}%
  \tp@check@empty[Toc]{heading}{Subtitle}{BM}%
  \tpIfAttrIsset{heading}{noBM}
    {}
    {\tpIfProp{bookmark-level}{\edef\Hy@toclevel{\tpUseProperty{bookmark-level}}}{}%
     \begingroup
       \tpGobble
       \protected@edef\@tempa{\csname tp@heading@bookmark\endcsname}%
       \bookmark[level=\Hy@toclevel,dest=\@currentHref]{\expandonce{\@tempa}}%
     \endgroup
   }}
```

3 Rendering the Headings

3.1 Inline Headings

\tp@inline@heading Inline headings are stored in a temporary box and expanded after the next (non-heading) paragraph is opened.

```
\newbox\tp@inlinesecbox
\def\tp@inline@heading{%
  \tpIfProp{after-indent}{\global\@afterindenttrue}{\global\@afterindentfalse}%
  \tpIfProp{interline-para}
    {\global\setbox\tp@inlinesecbox\hbox{\ifvoid\tp@inlinesecbox\else\unhbox\tp@inlinesecbox\
        tpUseProperty{interline-para-sep}\fi\@svsec}}%
    {\global\setbox\tp@inlinesecbox\hbox{\@svsec}}
  \@nobreakfalse
  \global\@noskipsectrue
  \gdef\next{%
    \global\everypar{%
      \if@noskipsec
        \global\@noskipsecfalse
        {\setbox\z@\lastbox}%
        \clubpenalty\@M
        \begingroup \unhbox\tp@inlinesecbox \endgroup
        \unskip
        \hskip -\@tempskipa
      \else
        \clubpenalty \@clubpenalty
        \global\setbox\tp@inlinesecbox\box\voidb@x
        \everypar{}%
      \fi}%
    \ignorespaces}}
```

3.2 Block Headings

\tp@block@heading is used to print block headings.

```
\def\tp@block@heading{%
  \@svsec
  \tpUseProperty{after-heading-par}%
  \tpIfProp{after-indent}{\global\@afterindenttrue}{\global\@afterindentfalse}%
  \gdef\next{%
    \ifdim\parskip>\z@\relax\advance\@tempskipa-\parskip\relax\fi
    \vskip \@tempskipa
    \@afterheading
    \ignorespaces}}
```

4 The heading environment

4.1 Environment Macros

`\heading` is the macro called at the begin of the `heading` environment. Optional #1 stores the headings local parameters, #2 is the level of the heading.

```
\def\heading{\@ifnextchar [{\@heading}{\@heading[]}}%]
\def\@heading[#1]#2{%
```

Some LaTeX kernel macros are saved, the namespace is set and counted groups from previous headings are reset.

```
  \tp@heading@reserve
```

Handling of the optional argument

```
  \tpParseAttributes{heading}{#1}%
```

The mandatory argument contains the heading level. This corresponds to LaTeX's way of counting heading levels, where, by default, `part` is `-1`, `chapter` is `0`, `section` is `1`, etc.

```
  \edef\tp@heading@name{#2}%
```

The cascaded Properties of the heading level are expanded. This is excluded into its own macro to simplify re-definition if necessary.

```
  \tpEvalType[#2]{Components}%
}
```

`\endheading` is stuff that happens at the end of the `heading` environment.

```
\def\endheading{%
  \expandafter\ifx\csname tpUseHeading\tp@heading@name\endcsname\relax
    \PackageError{coco-headings.sty}{Heading level \tp@heading@name\space unknown!}{A Heading
        with level \tp@heading@name\space is unknown. Use the \string\tpDeclareHeading\space
        macro to declare heading levels.}%
  \else
    \csname tpUseHeading\tp@heading@name\endcsname%
  \fi
  \tp@heading@reset
}
```

4.2 Content Handlers

`\tp@heading@reserve` re-directs some of LaTeX's kernel macros and makes sure that some other macros have their default values:

```
\def\tp@heading@reserve{%
  \tpNamespace{heading}%
  \let\tpltx@dbl@backslash\\
  \let\\\tpBreak
  \let\tpltx@label\label
  \let\tp@heading@label\relax
  \def\tpAuthorCnt{\z@}%
  \def\tpAffilCnt{\z@}%
  \tp@reset@components{\tp@cur@cont}%
  }
```

`r` estores LaTeX's default definitions (however, this should be unnecessary since `heading` is an environment and therefore constitutes a closed group).

```
\def\tp@heading@reset{%
  \let\tp@cur@cont\relax
  \let\\\tpltx@dbl@backslash
  \let\label\tpltx@label
  \let\tp@heading@name\relax
  \let\tp@heading@label\relax
  }
```

`\tp@heading@quotes` covers multiple quotation blocks assocciated with a heading.

```
\def\tp@heading@quotes{%
  \tpDeclareComp{QuoteBlock}{}{}%
  \tpDeclareComponentGroup{tpQuote}{%
    \tpDeclareCountedComp{QuoteText}%
    \tpDeclareCountedComp{QuoteSource}%
  }%
}
```

`\tp@heading@role@handlers` sets up the additional Components for the Author Role specific to headings.

```
\def\tp@heading@authors{%
  \tpAddToRole{Author}{%
    \tpDeclareCountedComp{AuthorContact}%
  }%
  \tpDeclareRoleBlock{Author}{ContactBlock}{author-contact-block-format}%
  \tpGroupHandler{tpAuthor}{%
    \tpIfComp{AuthorContact}{}{\csname tpAuthorContact\endcsname{\tpUseProperty{author-contact-
        format}}}{}%
  }%
  \tp@provide@hd@overrides{AuthorNameList}%
}
```

`\tp@provide@hd@macros` is a wrapper that creates the user-level macros for the Component itself and its overrides. #1 is the Component name.

```
\def\tp@provide@hd@macros#1{%
  \tpDeclareComp{#1}{}{}%
  \tp@provide@hd@overrides{#1}%
}
```

\tp@provide@hd@overrides declares the Component macros for a heading Component's overrides. #1 is the Component name. The overrides allow a four-way distinction between *i* the data printed in-situ (\tp#1), *ii* data sent to toc (\tpToc#1), (iii) data sent to the page styles (\tpRun#1), and (iv) the data sent to the PDF bookmarks (\tpBM#1).

```
\def\tp@provide@hd@overrides#1{%
  \tpDeclareComp{Toc#1}{}{}% toc overrides
  \tpDeclareComp{Run#1}{}{}% running overrides
  \tpDeclareComp{BM#1}{}{}% bookmark overrides
}
```

5 Defaults

```
\tpAddToDefault{heading}{%
  \tpSetProperty{interline-para}{}%
  \tpSetProperty{interline-para-sep}{\space}
  \tpSetProperty{heading-par}{%
    \tpIfProp{interline-para}{\if@noskipsec \leavevmode \fi}{}%
    \par
    \global\@afterindenttrue
  }%
  \tpSetProperty{after-heading-par}{\par \nobreak}% par commands at the end of non-inline headings
  \tpSetProperty{before-heading}{}%
  \tpSetProperty{title-face}{\bfseries}%
  \tpSetProperty{subtitle-face}{\normalfont}%
  \tpSetProperty{author-face}{\normalfont}%
  \tpSetProperty{quote-face}{\raggedleft}%
  \tpSetProperty{quote-source-face}{}%
  \tpSetProperty{quote-block-format}{%
    \bgroup
      \tpUseProperty{quote-face}%
      \tpUseComp{QuoteText}\par
      \tpIfComp{QuoteSource}{{\tpUseProperty{quote-source-face}--\space\tpUseComp{QuoteSource}}\
          par}{}%
    \egroup}
  \tpSetProperty{heading-block}
    {\tpUseProperty{title-face}%
     \tpIfComp{Number}
       {\tpUseProperty{hang-number}}
       {\leftskip0pt}%
     \tpUseComp{Title}\par%
     \tpIfComp{Subtitle}{{\tpUseProperty{subtitle-face}\tpUseComp{Subtitle}}\par}{}%
     \tpIfComp{AuthorNameList}{{\tpUseProperty{author-face}\tpUseComp{AuthorNameList}}\par}{}%
     \tpIfComp{QuoteBlock}{\tpUseComp{QuoteBlock}}{}%
     \tpIfComp{AffilBlock}{{\tpUseProperty{affil-block-face}\tpUseComp{AffilBlock}}\par}{}%
    }%
  \tpSetProperty{extended-heading}{%
    \tpIfComp{Abstract}
      {\par\vskip\baselineskip
       {\bfseries\tpIfComp{AbstractLabel}{\tpUseComp{AbstractLabel}}{Abstract}}\par
       {\itshape\small\tpUseComp{Abstract}}\par}
      {}%
    \tpIfComp{Keywords}
      {\par\vskip\baselineskip
       {\bfseries\tpIfComp{KeywordsLabel}{\tpUseComp{KeywordsLabel}}{Keywords}}\par
       {\itshape\small\tpUseComp{Keywords}\par}}
     {}%
  }%
 \tpSetProperty{before-skip}{\z@skip}% TODOC: values < 0pt use \minusvspace, else \addvspace. LaTeX's
     default behaviour of @afterindent is relocated to the after-indent property.
 \tpSetProperty{after-heading-block}{}%
 \tpSetProperty{before-heading-block}{\parindent\z@ \parskip\z@}%
 \tpSetProperty{toc-hook}{}% Called, after ToC and BM entries have been written to the .aux file
 \tpSetProperty{after-indent}{}%
 \tpSetProperty{margin-left}{}%
 \tpSetProperty{margin-right}{\@flushglue}%
 \tpSetProperty{after-skip}{1sp}%
 \tpSetProperty{indent}{auto}%
 \tpSetProperty{number-width}{}%
 \tpSetProperty{number-sep}{\space}%
 \tpSetProperty{number-align}{left}%
 \tpSetProperty{number-format}{%
   \bgroup
     \tpUseProperty{title-face}%
     \tpUseProperty{number-face}%
     \tpUseComp{Number}%
     \tpUseProperty{number-sep}%
   \egroup}
 \tpSetProperty{numbering}{auto}%
 %% running header
 \tpSetProperty{running-level}{}% override level for running title, name
 \tpSetProperty{running-heading}{%
   \tpIfComp{RunAuthorNameList}{\tpUseComp{RunAuthorNameList}:\space}{}%
   \tpUseComp{RunTitle}%
 }%
 %% ToC
 \tpSetProperty{no-toc}{false}% toc entries are generally disabled iff true
 \tpSetProperty{no-BM}{false}% bookmark entries are generally disabled, iff true
 \tpSetProperty{toc-margin-top}{\z@}% left indent of the whole entry
 \tpSetProperty{toc-margin-bottom}{\z@}% bottom margin of the whole entry
 \tpSetProperty{toc-margin-left}{auto}% left indent of the whole entry
 \tpSetProperty{toc-margin-right}{\@pnumwidth}% right margin of the whole entry
 \tpSetProperty{toc-title-face}{}% appearance of title
 \tpSetProperty{toc-indent}{auto}% offset of the first line of the entry. auto: hang indent by max-
     number-width for the level
 \tpSetProperty{toc-number-width}{}% current width of the TocNumber
 \tpSetProperty{toc-number-align}{left}% alignment of TocNumber within the hbox when hanging
 \tpPropertyLet{toc-number-face}{toc-title-face}% appearance of the TocNumber
 \tpSetProperty{toc-number-sep}{\enskip}% thing between TocNumber and TocTitle
 \tpSetProperty{toc-number-format}{% Format of the TocNumber
   \bgroup
     \tpUseProperty{toc-number-face}%
     \tpUseComp{TocNumber}%
     \tpUseProperty{toc-number-sep}%
   \egroup}
 \tpSetProperty{toc-page-sep}{\dotfill}% between TocTitle and the page counter
 \tpSetProperty{toc-page-face}{}% appearance of the page value
 \tpSetProperty{toc-page-format}{% format of the page value
   \tpUseProperty{toc-page-sep}%
   \bgroup
     \tpUseProperty{toc-page-face}%
     \tpUseComp{TocPage}%
   \egroup}%
 \tpSetProperty{toc-link}{none}% should toc entries be linked? values: none,title,page,all
 \tpSetProperty{toc-level}{}% override heading level for ToC, name!
 \tpSetProperty{toc-before-entry}{% stuff before anything is output; used to setup margins, alignment,
     line-breaking rules, etc.
   \addvspace{\tpUseProperty{toc-margin-top}}%
```

```
    \parindent \z@
    \let\\\@centercr
    \hyphenpenalty=\@M
    \rightskip \tpUseProperty{toc-margin-right} \@plus 1fil\relax
    \parfillskip -\rightskip
    \leftskip\tpUseProperty{toc-margin-left}%
  }%
  \tpSetProperty{toc-after-entry}{\par\addvspace{\tpUseProperty{toc-margin-bottom}}}% Thing at the
       end of the entry, after the page number
  \tpSetProperty{toc-format}{% Order and formatting of the entry itself
    \tpUseProperty{toc-title-face}%
    \tpIfComp{TocNumber}
      {\tpUseProperty{toc-hang-number}}
      {\leftskip0pt\leavevmode}%
    \tpIfComp{TocAuthorNameList}{\tpUseComp{TocAuthorNameList}:\space}{}%
    \tpUseComp{TocTitle}%
    \tpUseProperty{toc-page-format}%
  }%
  %% PDF-Bookmarks
  \tpSetProperty{bookmark-level}{}% override heading level for PDF bookmarks, numeric!
  \tpSetProperty{bookmark}{%
    \tpIfComp{BMNumber}{\tpUseComp{BMNumber}\space}{}%
    \tpUseComp{BMTitle}%
  }%
  \tpSetProperty{orcid-link}{% how the ORC-ID is rendered
    \tpIfComp{ORCID}{\def\tp@Linkimg{\includegraphics[height=1em]{logos/ORCID.pdf}}\tpCompLink{
        ORCID}{\tp@Linkimg}}{}%
  }%
  %% a single Author's contact infomration block
  \tpSetProperty{author-contact-format}{%Format of a single author's contact information
    \tpUseComp{FullName}\tpIfComp{Affil}{\textsuperscript{\tpUseComp{Affil}}}{}%
    \tpUseProperty{orcid-link}%
    %
  }%
  \tpSetProperty{author-list-format}{% Format of the whole contact information block
    \tpUseComp{FullName}\ifnum\tpCurCount<\tpTotalCount\tpUseProperty{counted-name-sep}\fi
  }%
  \tpSetProperty{author-contact-block-format}{% Format of the whole contact information block
    \tpUseComp{AuthorContact}\ifnum\tpCurCount<\tpTotalCount\tpUseProperty{counted-name-sep}\fi
  }%
}
```

6 Miscellaneous

6.1 Alternative paragraph separation

`\tpNewPar` is a user-level macro to have a vertical skip between two local paragraphs and no indent in the second one. The amount of vertical space between the paragraphs can be adjusted with the optional argument. If #1 is omitted, `\tpnewparskip` is inserted, which defaults to `1\baselineskip` if the dimension isn't set to something other than 0pt in the preamble. This macro is intended to be used at the end of the first of the paragraphs.

```
\newdimen\tpnewparskip \AtBeginDocument{\ifdim\tpnewparskip=\z@\relax \tpnewparskip=1\
    baselineskip\relax\fi}
\def\tpNewPar{\@ifnextchar[{\@tpnewpar}{\@tpnewpar[\the\tpnewparskip]}}%]
\def\@tpnewpar[#1]{%
  \ifhmode\par\fi
  \vskip#1\relax
  \@afterheading
}
```

**WARNING!
The following section is deprecated and will be changed or deleted in future releases.**

\TitleBreak

```
\let\TitleBreak\tpBreak
```

```
%</headings>
```

Modul 7

coco-notes.dtx

This file contains the code for foot- and endnote handling. It provides a switch between endnotes and footnotes as well as options to handle the resetting of footnote/endnote counters.

```
%<*endnotes>
```

```
%%
%% module for CoCoTeX that handles footnote/endnote switching.
%%
%% Maintainer: p.schulz@le-tex.de
%%
%% lualatex - texlive > 2019
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-notes}
    [2024/01/29 0.4.0 le-tex coco notes module]
```

internal switch for endnotes (`\endnotestrue`) or footnotes (`\endnotesfalse`, default).

```
\newif\ifendnotes \endnotesfalse
\newif\ifendnotelinks \endnotelinksfalse
```

package options:

- `endnotes` activates endnotes.
- `ennotoc` prevents chapter headings in the Notes section from creating toc entries.
- `resetnotesperchapter` resets foot- and endnotes at the start of each chapter level heading. If omitted (default) foot- or endnotes are numbered throughout the whole document
- `endnotesperchapter` implies `endnotes` and allows the output of all collected endnotes at the end of each chapter. It also sets the note's heading to section level (otherwise it is chapter level).

```
\DeclareOption{endnotes}{\global\endnotestrue}
\DeclareOption{ennotoc}{\global\let\tp@ennotoc\relax}
\DeclareOption{resetnotesperchapter}{\global\let\reset@notes@per@chapter\relax}
\DeclareOption{endnoteswithchapters}{\global\endnotestrue\global\let\endnotes@with@chapters\
    relax}
\DeclareOption{endnotelinks}{\global\endnotelinkstrue}
\ProcessOptions
```

footnote package is mandatory since it provides the `\savenotes` and `\spewnotes` macros:

```
\RequirePackage{footnote}
```

Handling of endnotes:

```
\newif\if@enotesopen
\AtBeginDocument{\edef\tpfn@parindent{\the\parindent}}
\ifendnotes
  \RequirePackage{endnotes}
  \@ifpackageloaded{coco-headings}{\let\tp@useTeXHeading\relax}{}
 % Allow linking endnotes to their respective occurrence in the document.
 \ifendnotelinks
   \global\newcount\endnoteLinkCnt \global\endnoteLinkCnt\z@
   \def\@endnotemark{%
     \leavevmode
     \ifhmode\edef\@x@sf{\the\spacefactor}\nobreak\fi
     \phantomsection%
     \label{endnote-\the\endnoteLinkCnt}%
     \hyperref[endnotetext-\the\endnoteLinkCnt]{\makeenmark}%
     \ifhmode\spacefactor\@x@sf\fi%
     \relax%
   }
 \fi
 \let\footnote=\endnote
 \def\enotesize{\normalsize}%
 \def\enoteformat{%
   % Create the label right at the start of the endnote text to prevent erroneous pointing to the next page
       .
   \ifendnotelinks%
     \phantomsection%
     \label{endnotetext-\currentEndnote}%
   \fi
   \noindent
   \leavevmode
   \hskip-2em\hb@xt@2em{%
     \ifendnotelinks
       \hyperref[endnote-\currentEndnote]{\@theenmark}\hss%
     \else
       \@theenmark\hss%
     \fi%
   }%
   \expandafter\parindent\tpfn@parindent\relax\expandafter%
 }%
 \gdef\enoteheading{%
   \leftskip2em
 }%
 \def\printnotes{%
   \ifx\endnotes@with@chapters\relax
     \ifnum\c@endnote>\z@
       \expandafter\global\expandafter\let\csname enotes@in@\the\realchap\endcsname\@empty
     \fi
   \fi
   \if@enotesopen
     \global\c@endnote\z@%
     \bgroup
     %\parindent\z@
     \parskip\z@
     \theendnotes
     \egroup
   \fi}
\else
 \newcount\c@endnote \c@endnote\z@
 \let\printnotes\relax
\fi
\newcount\realchap \realchap\z@
\ifx\endnotes@with@chapters\relax
 \AtBeginDocument{%
   \tpAddToHook[heading]{before-hook-chapter}{%
     \ifnum\c@endnote>\z@\relax
       \expandafter\global\expandafter\let\csname enotes@in@\the\realchap\endcsname\@empty
     \fi
     \global\advance\realchap\@ne
     \global\c@endnote\z@
     \def\tp@par@title{\tpIfComp{TocTitle}{\tpUseComp{TocTitle}}{\tpUseComp{Title}}}%
     \def\tp@par@runtitle{\tpIfComp{RunTitle}{\tpUseComp{RunTitle}}{\tpUseComp{Title}}}%
     \addtoendnotes{%
       \noexpand\expandafter\noexpand\ifx\noexpand\csname enotes@in@\the\realchap\noexpand\
           endcsname\noexpand\@empty
         \bgroup
           \noexpand\leftskip\noexpand\z@
           \noexpand\begin{heading}\ifx\tp@ennotoc\relax[notoc]\fi{section}%
             \noexpand\tpTitle{\tp@par@title}%
             \noexpand\tpRunTitle{\tp@par@runtitle}%
           \noexpand\end{heading}%
         \egroup
       \noexpand\fi}%
   }%
 }
\fi
\ifx\reset@notes@per@chapter\relax
 \AtBeginDocument{%
   \tpAddToHook[heading]{before-hook-chapter}{%
     \global\c@footnote\z@
     \global\c@endnote\z@
   }%
 }%
\fi
```

Here we make a small adjustment to the `\fn@fntext` macro from the `footnote` package by making it `\long` and therefore allowing `\par` inside it’s argument.

```
\long\def\fn@fntext#1{%
  \ifx\ifmeasuring@\@@undefined%
    \expandafter\@secondoftwo\else\expandafter\@iden%
  \fi%
  {\ifmeasuring@\expandafter\@gobble\else\expandafter\@iden\fi}%
  {%
    \global\setbox\fn@notes\vbox{%
      \unvbox\fn@notes%
      \fn@startnote%
      \@makefntext{%
        \rule\z@\footnotesep%
        \ignorespaces%
        #1%
        \@finalstrut\strutbox%
      }%
      \fn@endnote%
    }%
  }%
}
```

Re-definition of `footnote` package’s footnote mark retriever to allow non-numeric values in the optional argument of `\footnote`.

```
\def\fn@getmark@i#1[#2]{%
  \sbox\z@{\@tempcnta0#2\relax}%
  \ifdim\wd\z@>0\p@\relax
    \def\thempfn{#2}%
    \fn@getmark@iii%
  \else
    \csname c@\@mpfn\endcsname#2%
    \fn@getmark@ii%
  \fi
}
\def\fn@getmark@iii#1{%
  \unrestored@protected@xdef\@thefnmark{\thempfn}%
  \endgroup%
  #1%
}
```

And the same for plain LaTeX:

```
\def\@xfootnote[#1]{%
   \begingroup
     \sbox\z@{\@tempcnta0#1\relax}%
     \ifdim\wd\z@>0\p@\relax
       \unrestored@protected@xdef\@thefnmark{#1}%
     \else
       \csname c@\@mpfn\endcsname #1\relax
       \unrestored@protected@xdef\@thefnmark{\thempfn}%
     \fi
   \endgroup
   \@footnotemark\@footnotetext%
}
```

Linking endnotes requires overwriting the endnotetext macro to save a global counter to the *.ent file.

```
\global\newif\if@haveenotes
\long\def\@endnotetext#1{%
  \global\@haveenotestrue
  \if@enotesopen \else \@openenotes \fi
  \immediate\write\@enotes{%
    \ifendnotelinks
      \string\def\string\currentEndnote{\the\endnoteLinkCnt}%
    \fi%
    \@doanenote{\@theenmark}%
  }%
  \begingroup
     \def\next{#1}%
     \newlinechar='40
     \immediate\write\@enotes{\meaning\next}%
  \endgroup
  \immediate\write\@enotes{\@endanenote}%
  \ifendnotelinks
    \global\advance\endnoteLinkCnt\@ne%
  \fi%
}
```

```
%</endnotes>
```

Modul 8

coco-script.dtx

This package is used to handle non-latin based script systems like Japanese, Chinese, Armenian and the like.

```
%<*script>
```

```
%% module for CoCoTeX that handles script switching.
%%
%% Maintainer: p.schulz@le-tex.de
%%
%% lualatex - texlive > 2019
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-script}
    [2024/01/29 0.4.0 CoCoTeX script module]
```

The argument of the `usescript` option is a list of script systems that are used in the document. It is used to determine the additional fonts that are to be loaded via the babel package.

```
\let\usescript\relax
\define@key{coco-script.sty}{usescript}{\def\usescript{#1}}
\ProcessOptionsX
\RequirePackage[quiet]{fontspec}
\RequirePackage[bidi=basic,silent]{babel}
\def\parse@script#1,#2,\relax{%
  \tp@script@callback{#1}%
  \edef\@argii{#2}%
  \let\next\relax
  \ifx\@argii\@empty\else
    \def\next{\parse@script#2,\relax}%
  \fi\next}
\ifx\usescript\relax\else
  \def\tp@script@callback#1{\expandafter\global\expandafter\let\csname use@script@#1\endcsname\
      @empty}%
  \expandafter\parse@script\usescript,,\relax
\fi
\message{^^J [coco-script Fonts loaded: \meaning\usescript]^^J}
```

1 Default fallback font

The default fall backfont is the NotoSans Font Family

```
\newfontfamily\fallbackfont{NotoSerif-Regular.ttf}%
[BoldFont = NotoSerif-Bold.ttf,%
 ItalicFont = NotoSerif-Italic.ttf,%
 BoldItalicFont = NotoSerif-BoldItalic.ttf,%
 Path = ./fonts/Noto/Serif/,%
 WordSpace = 1.25]
```

```
\newfontfamily\sffallbackfont{NotoSans-Regular.ttf}%
[BoldFont = NotoSans-Bold.ttf,%
 ItalicFont = NotoSans-Italic.ttf,%
 BoldItalicFont = NotoSans-BoldItalic.ttf,%
 Path = ./fonts/Noto/Sans/,%
 WordSpace = 1.25]
\DeclareTextFontCommand\textfallback{\fallbackfont}
\DeclareTextFontCommand\textsffallback{\sffallbackfont}
```

2 Generic Fonts Declaration Mechanism

#1	Options passed to `\babelprovide`
#2	language
#3	argument(s) passed to `\babelfont{rm}`
#4	argument(s) passed to `\babelfont{sf}`

```
\def\tpDeclareBabelFont{\@ifnextchar[\tp@declare@babel@font{\tp@declare@babel@font[]}}%]
\def\tp@declare@babel@font[#1]#2#3#4{%
  \expandafter\ifx\csname use@script@#2\endcsname\@empty
    \babelprovide[#1]{#2}%
    \message{^^J [coco-script Loaded Script: #2]^^J}%
    %%
    \expandafter\gdef\csname tp@babel@rm@font@#2\endcsname{#3}%
    \expandafter\gdef\csname tp@babel@sf@font@#2\endcsname{#4}%
    \if!#2!\else
      \def\tp@tempa{\babelfont[#2]{rm}}%
      \expandafter\expandafter\expandafter\tp@tempa\csname tp@babel@rm@font@#2\endcsname
    \fi
    \if!#3!\else
      \def\tp@tempa{\babelfont[#2]{sf}}%
      \expandafter\expandafter\expandafter\tp@tempa\csname tp@babel@sf@font@#2\endcsname
    \fi
  \fi
}
```

Top level macro to declare a font alias.

#1	font family alias
#2	font family fallback

```
\def\tpBabelAlias#1#2{%
  \ifx\usescript\relax\else
    \def\tp@script@callback##1{%
      \expandafter\ifx\csname tp@no@fallback@##1\endcsname\relax
        \expandafter\ifx\csname tp@babel@#2@font@##1\endcsname\relax
          \PackageError
            {coco-script.sty}
            {\expandafter\string\csname #2family\endcsname\space for Language ‘##1’ was not
                declared!}
            {You attempted to declare an alias towards a font family that has not been declared
                for the language ‘##1’, yet.}%
        \else
          \def\tp@tempa{\babelfont[##1]{#1}}%
          \expandafter\expandafter\expandafter\tp@tempa\csname tp@babel@#2@font@##1\endcsname
        \fi
      \else
        \PackageInfo{coco-script.sty}{^^J\space\space\space\space No fallback for ‘##1’;^^J\space
            \space\space\space Skipping font family ‘#1’->‘#2’}%
      \fi}%
    \expandafter\parse@script\usescript,,\relax
  \fi}
```

3 Predefined script systems

3.1 Support for Armenian script

```
\ifx\use@script@armenian\@empty
 \message{^^J [coco-script Loaded Script: Armenian]^^J}
 \def\NotoArmenianPath{./fonts/Noto/Armenian/}
 \newfontfamily\fallbackfont@armenian{NotoSansArmenian-Regular.ttf}%
   [BoldFont = NotoSansArmenian-Bold.ttf,%
    Path = \NotoArmenianPath,%
    WordSpace = 1.25]
 \DeclareTextFontCommand\armenian{\fallbackfont@armenian}
 \let\tp@no@fallback@armenian\@empty%
\fi
```

3.2 Support for Chinese script

```
\tpDeclareBabelFont{chinese}{[%
   Path=./fonts/Noto/Chinese/,
   BoldFont = NotoSerifSC-Bold.otf,%
   WordSpace = 1.25]{NotoSerifSC-Regular.otf}}
 {[%
   Path=./fonts/Noto/Chinese/,
   BoldFont = NotoSansSC-Bold.otf,%
   WordSpace = 1.25]{NotoSansSC-Regular.otf}%
 }
```

3.3 Support for Japanese script

```
\tpDeclareBabelFont{japanese}{[%
   Path=./fonts/Noto/Japanese/,
   BoldFont = NotoSerifJP-Bold.otf,%
   WordSpace = 1.25]{NotoSerifJP-Regular.otf}
 }{[%
   Path=./fonts/Noto/Japanese/,
   BoldFont = NotoSansJP-Bold.otf,%
   WordSpace = 1.25]{NotoSansJP-Regular.otf}
 }
```

3.4 Support for Hebrew script

```
\tpDeclareBabelFont{hebrew}{[%
   Scale=MatchUppercase,%
```

```
    Path=./fonts/Noto/Hebrew/,%
    Ligatures=TeX,%
    BoldFont = NotoSerifHebrew-Bold.ttf]{NotoSerifHebrew-Regular.ttf}%
}{[%
    Scale=MatchUppercase,%
    Path=./fonts/Noto/Hebrew/,%
    Ligatures=TeX,%
    BoldFont = NotoSansHebrew-Bold.ttf]{NotoSansHebrew-Regular.ttf}%
}
```

3.5 Support for Arabic script

```
\tpDeclareBabelFont{arabic}{[%
    BoldFont = NotoNaskhArabic-Bold.ttf,%
    Path = ./fonts/Noto/Arabic/%
    ]{NotoNaskhArabic-Regular.ttf}}
  {[%
    BoldFont = NotoSansArabic-Bold.ttf,%
    Path = ./fonts/Noto/Arabic/%
    ]{NotoSansArabic-Regular.ttf}%
  }
```

3.6 Support for Greek script

```
\tpDeclareBabelFont{greek}{[%
    BoldFont = NotoSerif-Bold.ttf,%
    ItalicFont = NotoSerif-Italic.ttf,%
    BoldItalicFont = NotoSerif-BoldItalic.ttf,%
    Path = ./fonts/Noto/Serif/,%
    WordSpace = 1.25
    ]{NotoSerif-Regular.ttf}}
  {[BoldFont = NotoSans-Bold.ttf,%
    ItalicFont = NotoSans-Italic.ttf,%
    BoldItalicFont = NotoSans-BoldItalic.ttf,%
    Path = ./fonts/Noto/Sans/,%
    WordSpace = 1.25%
    ]{NotoSans-Regular.ttf}%
  }
```

3.7 Support for Syrian script

Since Babel does not support the Syrian script natively, we create a `babel-syriac.ini` file and include it, if it is needed. If we don’t, the kerning and ligatures of Syriac text will be off.

Please note that due to the restrictions of the `listings`-Package, some Unicode characters cannot be displayed correctly in the documentation of the following code. Therefore, Syriac letters appear as “x” in the following source code listing.

```
\expandafter\ifx\csname use@script@syriac\endcsname\@empty%
\RequirePackage{filecontents}
\begin{filecontents*}{babel-syriac.ini}
[identification]
charset = utf8
version = 0.1
```

```
date = 2019-08-25
name.local = xxxxxxxxxx
name.english = Classical Syriac
name.babel = classicalsyriac
tag.bcp47 = syc
tag.opentype = SYR
script.name = Syriac
script.tag.bcp47 = Syrc
script.tag.opentype = syrc
level = 1
encodings =
derivate = no
[captions]
[date.gregorian]
[date.islamic]
[time.gregorian]
[typography]
[characters]
[numbers]
[counters]
\end{filecontents*}
\fi
```

Now, we can create the fallback font and import the newly created ini file:

```
\tpDeclareBabelFont[import=syriac]{syriac}{[%
   BoldFont = NotoSansSyriac-Black.ttf,%
   ItalicFont = NotoSansSyriac-Regular.ttf,%
   BoldItalicFont = NotoSansSyriac-Black.ttf,%
   Path = ./fonts/Noto/Syriac/,%
   WordSpace = 1.25
   ]{NotoSansSyriac-Regular.ttf}}
 {[BoldFont = NotoSansSyriac-Black.ttf,%
   ItalicFont = NotoSansSyriac-Regular.ttf,%
   BoldItalicFont = NotoSansSyriac-Black.ttf,%
   Path = ./fonts/Noto/Syriac/,%
   WordSpace = 1.25%
   ]{NotoSansSyriac-Regular.ttf}%
 }
```

3.8 Support for medieval scripts and special characters

only `rm`!

```
\babelfont{mdv}[%
Path=fonts/Junicode/,%
ItalicFont = Junicode-Italic.ttf,%
BoldFont = Junicode-Bold.ttf,%
BoldItalicFont = Junicode-BoldItalic.ttf,%
]{Junicode.ttf}
\def\mdvfont#1{{\mdvfamily#1}}
```

```
%</script>
```

Modul 9

coco-title.dtx

This file provides macros and facilities for title pages.

```
%<*title>
```

```
%%
%% module for CoCoTeX for maketitle.
%%
%% Maintainer: p.schulz@le-tex.de
%%
%% lualatex - texlive > 2019
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-title}
    [2024/01/29 0.4.0 CoCoTeX title module]
\RequirePackage{coco-meta}
```

1 Top-Level Interface

Container titlepage is the main Container for the document's locally defined meta data.

```
\tpDeclareContainer{titlepage}{%
  \tpInherit {Components,Properties}{CommonMeta}%
  \tpDeclareType{Components}{%
    \tp@title@simple@comps
    \tp@meta@generic@comp
```

The following macro provides some meta data Components defined in the `coco-meta` module. They are:

- `Abstract` and `AbstractTitle`,
- `Keywords` and `KeywordsTitle`,
- `DOI` and `DOITitle`, and
- `TitleEn` and `TitleEnTitle`, intended for foreign language publications where the title is translated into English.

```
    \tp@title@fundings@comp
    \tp@title@role@handlers{author}{Author}%
    \tp@titlepage@role{editor}{Editor}%
    \tp@titlepage@role{series-editor}{SeriesEditor}%
  }%
  \tpDeclareType{Properties}{}%
  \tpDeclareEnv[tpMeta]{\tp@Meta}{\tp@endMeta}%
}
```

\tp@titlepage@role declares the roles for editors and series editors and initializes the biography meta block for both.

```
\def\tp@titlepage@role#1#2{%
  \tpDeclareRole[#1]{#2}%
  \tp@title@role@handlers{#1}{#2}%
}
```

`\tp@title@role@handlers` adds title page specific Components and Handlers to the Author, Editor and Series-Editor Roles.

```
\def\tp@title@role@handlers#1#2{%
  \tpAddToRole{#2}{%
    \tpDeclareCountedComp{Bio}%
    \tpDeclareCountedComp{Biography}}%
  \tpGroupHandler{tp#2}{%
    \tpIfComp{Biography}{}{\tpIfComp{Bio}{\tpBiography{\tpUseProperty{#1-biography-format}}}{}}%
  }%
  \tpDeclareRoleBlock[apply]{#2}{BioBlock}{#1-bio-block-format}%
}
```

`\tpDeclareTitlepage` is the default titlepage declarator with the next token being added the titlepage's Property list.

```
\def\tpDeclareTitlepage{\tpAddToType{Properties}{titlepage}}
```

`\tp@Meta` is the code executed at the beginning of the `tpMeta` Container

```
\def\tp@Meta{%
  \tpEvalType{Components}%
}
```

`\tpAddTitleRole` is a user-level macro to add both a new Role with the name #2 and a controlling Property #1 to the `titlepage` container.

```
\def\tpAddTitleRole#1#2{%
  \tpAddToType{Components}{titlepage}{\tp@titlepage@role{#1}{#2}}%
  \tpAddTitleEval{\tp@title@eds@eval{#2}}%
}
```

`\tpAddTitleEval` is a User-level macro to add additional Material titlepage evaluators (the next token).

```
\def\tpAddTitleEval{\csgappto{tp@title@add@eval}}
```

`\tp@title@add@eval` is a hook for additional titlepage evaluators

```
\def\tp@title@add@eval{}
```

`\tp@endMeta` is the code executed at the end of the `tpMeta` Container

```
\def\tp@endMeta{%
  \tpNamespace{titlepage}%
  \tpEvalType{Properties}%
  \tp@maketitle
  \tp@meta@role@eval{Author}%
  \tpApplyCollection{tpAffil}{affil-block-item-format}{AffilBlock}%
  \tp@title@eds@eval{Editor}%
  \tp@title@eds@eval{SeriesEditor}%
  \tp@meta@generic@eval
  \tp@title@fundings@eval
  \tp@title@add@eval
  \tp@if@preamble\tp@title@set@pdfmeta\relax
  \tpUseHook{document-meta-hook}%
  \let\tp@cur@cont\@empty
}
```

2 Procesing of PDF Meta Data

The next few macros handle the content that is written directly into the pdf as meta data.

`\tp@title@set@pdfmeta` is the wrapper for the whole meta data handling.

```
\def\tp@title@set@pdfmeta{%
```

`\tp@write@pdf@meta` is used to transfer the DocumentInfo meta date to the pdf writer.

First we check, whether `coco-accessibility.sty` is used. If so, we check if the User has provided an `xmp` file by reading the required meta data field given in ##2 from that xmp file. If the temporary storage `\@tempa` is empty, this means that either that `coco-accessibility.sty` isn't loaded, that the user has not provided an `xmp` file, or that the specific field was empty or missing.

In this case, we take the value given in `\#\#3` and store it in `\@tempa`. If the storage is still empty (i.e. the field is also missing in the `tpMeta` environment), we do nothing.

If the user has provided the meta datum in the `tpMeta` environment, we pass it either to `hyperref`'s hypersetup variable given in `\#\#1` (when `coco-accessibility.sty` is *not* used), or we pass it to `\setDocInfo` from the `ltpdfa` package using the data field given in `\#\#2`. In this case, the ltpdfa automatically creates a `\jobname.xmp` with the given meta data will be generated for the next LaTeX run.

```
  \def\tp@write@pdf@meta##1##2##3{%
    \let\@tempa\@empty
    \@ifpackageloaded{coco-accessibility}
      {\edef\@tempa{\expandonce{\directlua{tex.print(cocotex.ally.meta.##2)}}}}{}%
    \ifx\@tempa\@empty
      \protected\def\@tempa{##3}%
      \ifx\@tempa\@empty\else
        \@ifpackageloaded{coco-accessibility}
          {\pdfstringdef\x{##3}\setDocInfo{##2}{\x}}
          {\edef\x{\noexpand\hypersetup{##1={\expandonce\@tempa}}}\x}%
      \fi
    \fi
  }%
```

```
  \tp@title@insert@xmp
  \tp@title@process@bkt
  \tp@title@process@bka
  \tp@title@process@bkc
}
```

2.1 Processing of the Document's Title

`\tp@title@process@bkt` processes the document's main title

```
\def\tp@title@process@bkt{%
  \let\tpBreak\space
  \protected@xdef\@title{\tpUseComp{Title}}%
  \tp@write@pdf@meta{pdftitle}{Title}{\tpUseComp{Title}}%
  \protected@edef\tp@run@book@title{\tpUseProperty{run-book-title}}%
  \expandafter\gdef\expandafter\tpRunBookTitle\expandafter{\tp@run@book@title}%
}
```

2.2 Processing of the Document's Author

`\tp@title@process@bka` processes the document's main author or, if that doesn't exist, the main editor, or throws a warning if neither exist.

```
\def\tp@title@process@bka{%
  \@tempswatrue
  \begingroup
    \tpGobble
    \renewcommand\foreignlanguage[2]{{##2}}%
    \ifnum\tpAuthorCnt>\z@
      \@setpar{\@@par}%
      \tpCompGDef\tpRunBookName{tpAuthor}{author-list-pdfinfo-format}%
    \else
      \ifnum\tpEditorCnt>\z@
        \tpCompGDef\tpRunBookName{tpEditor}{editor-list-pdfinfo-format}%
      \else
        \tpPackageWarning{transcript-title}{Meta Data}{No author or editor given!}%
        \@tempswafalse
      \fi
    \fi
    \if@tempswa
      \expandafter\author\expandafter{\tpRunBookName}%
      \tp@write@pdf@meta{pdfauthor}{Author}{\tpRunBookName}%
    \fi
  \endgroup
}
```

2.3 Processing of the PDF's Creator, Producer, and Keywords Meta Data

`\tp@title@process@bkc` processes the metadata for the pdf creator

```
\def\tp@title@process@bkc{%
  \tp@write@pdf@meta{pdfcreator}{Creator}{\tpIfComp{PDFCreator}{\tpUseComp{PDFCreator}}{\
      tpUseComp{Publisher}\tpIfComp{PubPlace}{, \tpUseComp{PubPlace}}{}}}%
  \tp@write@pdf@meta{pdfproducer}{Producer}{\tpUseComp{PDFProducer}}%
  \tp@write@pdf@meta{pdfkeywords}{Keywords}{\tpUseComp{Keywords}}%
}
```

2.4 Including the XMP Meta Data

`\tp@title@insert@xmp` inserts the contents of the XMP meta data file into the pdf, if it exists.

```
\def\tp@title@insert@xmp{%
  \edef\include@xmp{\noexpand\@include@xmp{\tpUseComp{XmpFile}.xmp}}
  \def\@include@xmp##1{\IfFileExists{##1}{\@@include@xmp{##1}}{}}
  \def\@@include@xmp##1{%
    \begingroup
      \immediate\pdfobj stream attr {/Type /Metadata /Subtype /XML}
      file{##1}
      \pdfcatalog{/Metadata \the\pdflastobj\space 0 R}
    \endgroup}%
  \include@xmp
}
```

3 Intermediate Level Interfaces

before-maketitle-hook Hook that is expanded right before the titlepage is printed.

```
\tpDeclareHook[titlepage]{before-maketitle-hook}
\tpDeclareHook[titlepage]{document-meta-hook}
```

\tp@maketitle collects the meta information and constructs the tpMaketitle macro

```
\def\tp@maketitle{%
  \ifarticle
    \gdef\tpMaketitle{%
      \let\tp@cnt@grp\@empty
      \tpUseHook[titlepage]{before-maketitle-hook}%
      \bgroup
        \tpNamespace{titlepage}%
        \tpEvalType{Properties}%
        \tpUseProperty{article-title}%
      \egroup
      \tpUseHook[titlepage]{after-maketitle-hook}%
    }%
  \else
    \gdef\tpMaketitle{%
      \let\tp@cnt@grp\@empty
      \tpUseHook[titlepage]{before-maketitle-hook}%
      \bgroup
        \tpNamespace{titlepage}%
        \tpEvalType{Properties}%
        \tpUseProperty{before-titlepage}%
        \tpIfComp{Cover}{%
          \tpUseProperty{coverpage}%
        }{}%
        \tpUseProperty{before-titlepage-roman}%
        \tpUseProperty{titlepage-roman}%
        \tpUseProperty{after-titlepage}%
      \egroup
      }%
  \fi
}
```

3.1 Funds, Grants, and Supporters

This is a Subcontainer within tpMeta which allows to set up multiple funding, grant, or supporter callouts.

\tp@title@fundings@comp wrapper to set up the Subcontainer

```
\def\tp@title@fundings@comp{%
  \tpDeclareComp{FundingBlock}{\expandafter\global}{}%
  \tpDeclareComponentGroup{tpFunding}{%
    \tpDeclareCountedComp{FundName}%
    \tpDeclareCountedComp{FundLogo}%
    \tpDeclareCountedComp{FundID}%
  }{}%
}
```

`\tp@title@fundings@eval` Evaluator for the funding

```
\def\tp@title@fundings@eval{{%
    \def\tp@cur@cont{titlepage}%
    \tpComposeCollection{tpFunding}{fund-format}{FundingBlock}%
}}
```

`\tp@title@eds@eval` evaluator for the editors

```
\def\tp@title@eds@eval#1{%
  \tp@meta@role@eval{#1}%
  \tp@create@editor@string{#1}}
```

`\tp@create@editor@string` evaluates the editor string and adds a suffix.

```
\def\tp@create@editor@string#1{%
  \expandafter\ifx\csname tp@\tp@cur@cont @#1NameList\endcsname\relax\else
    \csgappto{tp@\tp@cur@cont @#1NameList}{{\letcs\tpTotalCount{tp#1Cnt}\tpUseProperty{editor-
        suffix}}}%
  \fi
}%
```

3.2 Simple Component Declarations

`\tp@title@macro` is an alias for `\tpDeclareGComp` for backwards compatibility.

```
\let\tp@title@macro\tpDeclareGComp
```

`\tp@title@simple@comps` wrapper for the Titlepage's simple Components.

```
\def\tp@title@simple@comps{%
  \tpDeclareGComp[\jobname]{XmpFile} % File basename of the XMP file ('.xmp' is added automatically)
  %% Cover
  \tp@title@macro{Cover} % Path to Cover Image(!)
  %% Titles
  \tp@title@macro{Title} % Main Title
  \tp@title@macro{ShortTitle} % Shortened main title
  \tp@title@macro{RunTitle} % Shortened main title override for headers
  \tp@title@macro{AltTitle} % Alternative main title (e.g. for bastard title page)
  \tp@title@macro{Subtitle} % Sub Title
  \tp@title@macro{RunNames} % Shortened list of names (authors and/or publishers)
  \tp@title@macro{AltNames} % Alternative list of names (e.g. for bastard title page)
  %% Series
  \tp@title@macro{Series} % Series Title
  \tp@title@macro{SubSeries} % Series Subtitle
  \tp@title@macro{SeriesNote} % Series Notes
  \tp@title@macro{Volume} % Series Volume
```

```
\tp@title@macro{Number} % Series Number
\tp@title@macro{EditorNameList} % Editor Text Line
\tp@title@macro{SeriesEditorNameList} % Series Editor Text Line
%% Publisher
\tp@title@macro{Publisher} % Publisher Name
\tp@title@macro{PubDivision} % Publishing Division
\tp@title@macro{PubDivInfo} % Publishing Division Info
\tp@title@macro{PubPlace} % Publisher Location
\tp@title@macro{PubLogo} % Publisher Logo
\tp@title@macro{PubNote} % Additional publisher notes
\tp@title@macro{PubWeb} % Publisher URL
%% Pubication Meta
\tp@title@macro{PDFCreator} % Creator for pdf metadata
\tp@title@macro[le-tex xerif]{PDFProducer} % PDF producer for pdf metadata
\tp@title@macro{Dedication} % Dedication
\tp@title@macro{Acknowledgements} % Acknowledgements
\tp@title@macro{Statement} % Acknowledgements
\tp@title@macro{EditionNote} % Edition Note
\tp@title@macro{Editorial} % Editorial
\tp@title@macro{Edition} % Edition
\tp@title@macro{Year} % Publication Year
\tp@title@macro{ISBNPreText} % Text before ISBN block
\tp@title@macro{ISBN} % ISBN
\tp@title@macro{ISSN} % ISSN
\tp@title@macro{EISSN} % Ebook-ISSN
\tp@title@macro{EpubPreText} % Text between ISBN and eISBN
\tp@title@macro{EISBN} % Ebook-ISBN
\tp@title@macro{EpubISBN} % Epub-ISBN
\tp@title@macro{ElibPDF} % ???
\tp@title@macro{BiblISSN} % Bibl-ISBN
\tp@title@macro{BibleISSN} % Bible-ISBN
%% Funding
\tp@title@macro{FundingPreText} % Text before the Funding list
\tp@title@macro{FundingPostText} % Text after the Funding list
%% Imprint Meta
\tp@title@macro{Biblio} % Bibliographical Information
\tp@title@macro{BiblioTitle} % Heading Bibliographical Information
\tp@title@macro{Print} % Printer
\tp@title@macro{PrintNote} % Print Note
\tp@title@macro{Lectorate} % Lector
\tp@title@macro{Translator} % Translator
\tp@title@macro{CoverConcept} % Cover Concept
\tp@title@macro{CoverDesign} % Cover Designer
\tp@title@macro{CoverImage} % Cover Image Creator
\tp@title@macro{Typesetter} % Typesetting company
\tp@title@macro{QA} % Quality Assurance
\tp@title@macro{UsedFont} % Used Font(s)
\tp@title@macro{Conversion} % Data Converison
\tp@title@macro{EnvDisclaimer} % Environmental Disclaimer
\tp@title@macro{Advertise} % Advertisements
%% Licencing
\tp@title@macro{LicenceText} % License Description
\tp@title@macro{LicenceLogo} % License Logo
\tp@title@macro{LicenceLink} % License Link
\tp@title@macro{LicenceName} % License Name
\tp@title@macro{CopyrightDisclaimer} % Copyright Disclaimer
%% for journals
\tp@title@macro{JournalName} % Full name of the journal
\tp@title@macro{JournalAbbrev} % Short name of the journal
\tp@title@macro{Issue} % Issue of the journal
\tp@title@macro{PubCycle} % Publication cycle
```

```
  \tp@title@macro{Prices} % Prices of the journal issues or subscription models
  \tp@title@macro{MemberList} % In case of publishing organizations, this macro may hold a list of
      members.
  %% for single articles
  \tp@title@macro{StartPage} % Start page of a single article
  \tp@title@macro{EndPage} % End page of a single article
  \tpDeclareLabeledComp[Cite as]{CiteAs}{cite-as} % As what the article should be cited
  \tpDeclareLabeledComp[Submitted]{Submitted}{sumbitted} % Date the article was submitted
  \tpDeclareLabeledComp[Received]{Received}{received} % Date the article was recieved
  \tpDeclareLabeledComp[Revised]{Revised}{revised} % Date the article was revised
  \tpDeclareLabeledComp[Reviewed]{Reviewed}{reviewed} % Date the article was reviewed
  \tpDeclareLabeledComp[Accepted]{Accepted}{accepted} % Date the article was accepted
  \tpDeclareLabeledComp[Published]{Published}{published} % Date the article was published
  \tpDeclareLabeledComp[Conflict of Interest]{COIStatement}{coi-statement}% Conflict of Interest
      statement
  %% Generic additional information
  \tp@title@macro{AddNoteI} % Additional information, title page I
  \tp@title@macro{AddNoteII} % Additional information, title page II
  \tp@title@macro{AddNoteIII} % Additional information, title page III
  \tp@title@macro{AddNoteIV} % Additional information, title page IV
}
```

4 Default Settings

```
\tpAddToDefault{titlepage}{%
  \tpSetProperty{article-title}{}%
  % Title page hooks
  % Before \tpMaketitle and outside the group
  \tpSetProperty{before-titlepage}{%
    \pagestyle{empty}%
    \parindent\z@
    \parskip\z@
  }%
  \tpSetProperty{after-titlepage}{\pagestyle{headings}}%
  % Pages of title
  %% Cover page
  \tpSetProperty{coverpage}{%
    \bgroup
      \def\thepage{\@alph\c@page}%
      \smash{\rlap{%
          \raise\dimexpr\headheight+\headsep+\topmargin+\topskip-\paperheight\relax
          \vtop{%
            \hskip-\oddsidemargin
            \includegraphics[width=\paperwidth,height=\paperheight]{\tpUseComp{Cover}}%
          }}}%
      \tpUseProperty{after-coverpage}%
    \egroup
  }%
  \tpSetProperty{after-coverpage}{\cleardoublepage}%
  \tpSetProperty{titlepage-roman}{%
    \tpUsePropEnv{titlepage-i}%
    \clearpage
    \tpUsePropEnv{titlepage-ii}%
    \clearpage
    \tpUsePropEnv{titlepage-iii}%
    \clearpage
    \tpUsePropEnv{titlepage-iv}%
    \clearpage
  }%
  %% Generic meta blocks
  \tpSetProperty{generic-meta-heading-face}{\large}% format of the heading of a generic meta block
  \tpSetProperty{generic-meta-format}{% Format of a single generic meta-block
    \tpIfComp{Heading}{{\tpUseProperty{generic-meta-heading-face}\tpUseComp{Heading}\par}\vskip\
        baselineskip}{}%
    \tpUseComp{Content}%
    \par%
  }%
  %% Funding
  \tpSetProperty{funding-columns}{2}
  \tpSetProperty{funding-format}{}%
```

Fallback for the width in case someone sets up a fixed value for a fund's width.

```
  \tpSetProperty{fund-width}{.5\textwidth}
  \tpSetProperty{fund-vertical-sep}{\baselineskip}%
  \tpSetProperty{fund-sep}{%
    \expandafter\@tempcnta\CalcModulo{\tpCurCount}{\tpUseProperty{funding-columns}}%
    \ifnum\@tempcnta=\z@
      \par
      \ifnum\tpCurCount<\tpTotalCount\relax
        \vskip\tpUseProperty{fund-vertical-sep}%
      \fi
    \else
      \hfill
    \fi}
  \tpSetProperty{fund-format}{% Format of a single fund/grant/sponsor
    \strut\vtop{%
      \hsize\tpUseProperty{fund-width}%
      \tpIfComp{FundName}{\tpUseComp{FundName}\\[1ex]}{}%
      \includegraphics[width=\tpUseProperty{fund-width}]{\tpUseComp{FundLogo}}}%
    \tpUseProperty{fund-sep}%
  }%
  \tpSetProperty{funding-sep}{4mm}%
  \tpSetProperty{funding-block}{%
    \bgroup
```

We set `fund-width` here so that the value is calculated only once and only the result is stored in the `fund-width` Property.

```
      \tpSetPropertyX{fund-width}{\dimexpr(\textwidth/\tpUseProperty{funding-columns})-(\
          tpUseProperty{funding-sep}/\tpUseProperty{funding-columns})\relax}
      \tpUseProperty{funding-format}%
      \tpGetComp{FundingPreText}%
      \tpGetComp{FundingBlock}%
      \tpGetComp{FundingPostText}%
      \par
    \egroup
  }
  %% before the roman part of the title pages but after cover page
  \tpSetProperty{before-titlepage-roman}{%
    \setcounter{page}{1}%
    \def\thepage{\roman{page}}%
  }%
  \tpSetProperty{titlepage-i}{%
    \ifmonograph
      \tpUseComp{AuthorNameList}%
    \else
      \tpUseProperty{EditorNameList}%
    \fi%
    \vskip\baselineskip
    \bgroup
      \tpUseProperty{title-face}\tpUseComp{Title}%
    \egroup
    %\expandafter\meaning\csname tp@titlepage@editor-2@FullName\endcsname
  }%
  \tpSetProperty{titlepage-ii}{%
    \tpGetComp{Editorial}%
    \tpGetComp{SeriesNote}%
    \tpGetComp{GenericMetaBlock}%
    \vfill
    \tpUseProperty{bio-output}%
  }%
  \tpSetProperty{titlepage-iii}{%
    \ifmonograph
      \tpUseComp{AuthorNameList}%
    \else
      \tpUseProperty{EditorNameList}%
    \fi%
    \par
    \tpUseProperty{title-format}
    \tpGetComp{Edition}%
    \tpGetComp{EditionNote}%
    \vfill
    \clearpage
  }%
  \tpSetProperty{titlepage-iv}{%
    \tpGetComp{Dedication}% Dedication
    \tpGetComp{Acknowledgements}% Dedication
    \tpUseProperty{imprint-format}%
    \tpUseProperty{funding-block}%
    \vfill
    \bgroup
      \tpUseProperty{imprint-face}%
      \tpIfComp{Biblio}{{\bfseries\tpGetComp{BiblioTitle}}\tpGetComp{Biblio}}{}%
      \tpUseProperty{imprint-sep}%
      \tpUseProperty{imprint}%
    \egroup
    \clearpage
  }%
  %% predefined face and format Properties
  \tpSetProperty{title-face}{\Huge\sffamily\bfseries}%
  \tpSetProperty{title-format}{%
    \bgroup
      \tpUseProperty{title-face}%
      \tpUseComp{Title}\par
    \egroup
    \tpIfComp{Subtitle}{\tpUseProperty{subtitle-format}}{}%
    \tpGetComp{Statement}%
    \vskip\baselineskip
  }%
  \tpSetProperty{subtitle-face}{\Large\sffamily\bfseries}%
  \tpSetProperty{subtitle-format}{%
    \bgroup
      \tpUseProperty{subtitle-face}%
      \tpUseComp{Subtitle}%
    \egroup
    \par
  }%
  %% Imprint
```

```
\tpSetProperty{imprint-face}{\footnotesize}%
\tpSetProperty{imprint-sep}{\ifhmode\par\fi\addvspace{\baselineskip}}%
\tpSetProperty{imprint}{%
  \tpUseProperty{publisher}%
  \tpGetComp{Qualification}%%
  \tpGetComp{Conversion}%%
  \tpGetComp{CoverDesign}%%
  \tpGetComp{CoverImage}%%
  \tpGetComp{Lectorate}%%
  \tpGetComp{QA}%%
  \tpGetComp{Translator}%%
  \tpGetComp{Appraiser}%%
  \tpGetComp{Discussion}%%
  \tpGetComp{Typesetter}%%
  \tpGetComp{Print}%%
  \tpGetComp{UsedFont}%%
  \tpGetComp{DOI}%%
  \tpGetComp{Keywords}%%
  \tpUseProperty{imprint-sep}%
  \tpGetComp{ISBNPreText}%
  \tpGetComp{ISBN}%
  \tpGetComp{EpubPreText}%
  \tpGetComp{EISBN}%
  \tpGetComp{EpubISBN}%
  \tpUseProperty{imprint-sep}%
  \tpGetComp{EnvDisclaimer}%
}%
\tpSetProperty{journal-meta}{%
  \tpUseLabeledComp{Submitted}%
  \tpUseLabeledComp{Received}%
  \tpUseLabeledComp{Revised}%
  \tpUseLabeledComp{Accepted}%
  \tpUseLabeledComp{Published}%
  \tpUseLabeledComp{Copyright}%
  \tpUseLabeledComp{COIStatement}%
  \tpUseLabeledComp{Keywords}
}%
\tpSetProperty{licence}{%
  \tpIfComp{LicenceLogo}{\includegraphics{\tpUseComp{LicenceLogo}}\par}{}%
  \tpGetComp{LicenceText}%
}%
\tpSetProperty{copyright}{%
  \tpIfComp{Copyright}
    {\tpUseComp{Copyright}\par}
    {\textcopyright\space\tpUseComp{Year}\space\tpUseComp{Publisher},\space\tpUseComp{PubPlace
        }\par}%
  }%
\tpSetProperty{publisher}{%
  \tpGetComp{PubDivInfo}%
  \tpUseProperty{copyright}%
  \tpGetComp{PubNote}%
  \tpGetComp{PubWeb}%
}%
% Name Formats
\tpSetProperty{counted-meta-sep}{\ifnum\tpCurCount<\tpTotalCount\relax\vskip\baselineskip\fi}%
     separator between multiple instances of the same meta datum
\tpSetProperty{counted-name-sep}{% Separator between multiple names; titlepage-specific override of
    the same Property in coco-meta!
  \ifnum\tpTotalCount>1\relax
    \ifnum\tpCurCount<\tpTotalCount\relax
      \ifnum\tpCurCount<\numexpr\tpTotalCount-1\relax
```

```
        \tpUseProperty{name-sep}%
      \else
        \tpUseProperty{name-and}%
      \fi
    \fi
  \fi
}%
% Aliasses for different Roles, see coco-meta.sty for the actual Property values:
%% editors:
\tpPropertyLet{editor-cite-name-format} {role-cite-name-format}%
\tpPropertyLet{editor-short-cite-name-format} {role-short-cite-name-format}%
\tpPropertyLet{editor-full-name-format} {role-full-name-format}%
\tpPropertyLet{editor-pdfinfo-name-format} {role-pdfinfo-name-format}%
\tpPropertyLet{editor-correspondence-as-format} {role-correspondence-string-format}%
%
\tpPropertyLet{editor-list-print-format} {role-block-print-format}%
\tpPropertyLet{editor-list-cite-format} {role-block-cite-format}%
\tpPropertyLet{editor-list-short-cite-format} {role-block-short-cite-format}%
\tpPropertyLet{editor-list-pdfinfo-format} {role-block-pdfinfo-format}%
\tpPropertyLet{editor-list-correspondence-format} {role-block-correspondence-format}%
%% series-editors:
\tpPropertyLet{series-editor-cite-name-format} {role-cite-name-format}%
\tpPropertyLet{series-editor-short-cite-name-format} {role-short-cite-name-format}%
\tpPropertyLet{series-editor-full-name-format} {role-full-name-format}%
\tpPropertyLet{series-editor-pdfinfo-name-format} {role-pdfinfo-name-format}%
\tpPropertyLet{series-editor-correspondence-as-format} {role-correspondence-as-format}%
%
\tpPropertyLet{series-editor-list-print-format} {role-block-print-format}%
\tpPropertyLet{series-editor-list-cite-format} {role-block-cite-format}%
\tpPropertyLet{series-editor-list-short-cite-format} {role-block-short-cite-format}%
\tpPropertyLet{series-editor-list-pdfinfo-format} {role-block-pdfinfo-format}%
\tpPropertyLet{series-editor-list-correspondence-format} {role-block-correspondence-format}%
%% name Separators
\tpSetProperty{editor-suffix-sgl}{(Ed.)}%
\tpSetProperty{editor-suffix-pl}{(Eds.)}%
\tpSetProperty{editor-suffix}{%
  \space
  \ifnum\tpTotalCount=\@ne\relax
    \tpUseProperty{editor-suffix-sgl}%
  \else
    \tpUseProperty{editor-suffix-pl}%
  \fi
}%
% Biography
% those Properties control how (Role specific) Biography Blocks are formatted, i.e. the list of all
     Biographies of a specific Role:
\tpSetProperty{role-bio-block-face}{}% face for the entire, role-specific, Biography Block
\tpSetProperty{role-bio-block-format}{{\tpUseProperty{role-bio-block-face}\tpUseComp{Biography
     }}\par}% Format of the whole, Role specific, Biography Block
\tpPropertyLet{author-bio-block-format} {role-bio-block-format}% Override for single author meta
     info
\tpPropertyLet{editor-bio-block-format} {role-bio-block-format}% Override for single editor meta
     info
\tpPropertyLet{series-editor-bio-block-format} {role-bio-block-format}% Override for single
     series editor meta info
% those Properties control how a (Role specific) Biography is formatted:
\tpSetProperty{role-biography-format}{{\bfseries\tpUseComp{FullName}:}\space\tpUseComp{Bio}\
     par}% Format of a single entry in the Role specific Biography
\tpPropertyLet{author-biography-format} {role-biography-format}% Override for single author meta
     info
  \tpPropertyLet{editor-biography-format} {role-biography-format}% Override for single editor meta
      info
  \tpPropertyLet{series-editor-biography-format} {role-biography-format}% Override for single
      series editor meta info
  \tpSetProperty{bio-output-format}{%
    \tpGetComp{AuthorBioBlock}%
    \tpGetComp{EditorBioBlock}%
    \tpGetComp{SeriesEditorBioBlock}%
  }%
  % Running headers
  \tpSetProperty{run-book-title}{%
    \tpIfComp{RunTitle}
      {\tpUseComp{RunTitle}}
      {\tpIfComp{ShortTitle}
        {\tpUseComp{ShortTitle}}
        {\tpIfComp{Title}{\tpUseComp{Title}}{No title given!}}}%
  }%
  \tpSetProperty{run-book-name}{%
    \tpIfComp{RunNames}
      {\tpUseComp{RunNames}}
      {\ifmonograph
         \tpIfComp{AuthorNameList}
           {\tpUseComp{AuthorNameList}}
           {no author defined!}%
       \else
         \tpIfComp{EditorNameList}
           {\tpUseComp{EditorNameList}}
           {no editor defined!}%
       \fi}%
  }%
}
```

```
%</title>
```

Modul 10

coco-floats.dtx

This module provides handlers for floating objects like tables and figures common to all CoCoTEX projects

```
%<*floats>
```

```
%%
%% module for CoCoTeX that extends floating objects.
%%
%% Maintainer: p.schulz@le-tex.de
%%
%% lualatex - texlive > 2019
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-floats}
   [2024/01/29 0.4.0 CoCoTeX floats module]
\DeclareOptionX{nofigs}{\global\let\tp@nofigs\relax}
\ProcessOptionsX
```

1 Package Setup

1.1 Hard requirements

```
\RequirePackage{coco-common}
\RequirePackage{rotating}
\RequirePackage{grffile}
\RequirePackage{footnote}
\RequirePackage[Export]{adjustbox}
\usepackage{stfloats}
\setcounter{dblbotnumber}{5}
```

1.2 Document Class Option overrides

for automatic typesetting and float positioning, we set very high tolerances in macros from LATEX's standard

2 .clo

files:

```
\def\topfraction{0.9}
\def\textfraction{0.1}
\def\bottomfraction{0.8}
```

```
\def\totalnumber{8}
\def\topnumber{8}
\def\bottomnumber{8}
\def\floatpagefraction{0.8}
\@fptop\z@
\@fpbot\@flushglue
```

2.1 Internal registers

Some reserved box registers for measuring, the first one, `\tp@floatbox`, is for the whole float, the second one, `\tp@subfltbox`, is for a single sub-float. The third one, `\tp@calcfltbox`, is used to calculate the overall dimensions of the float.

```
\newbox \tp@floatbox
\newbox \tp@subfltbox
\newbox \tp@calcfltbox
```

Internal counters: `\tpSubFloatCnt` counts the sub-floats within a single float, `\tp@int@flt@cnt` is the internal global counter for all floats.

```
\newcount\tpSubFloatCnt \tpSubFloatCnt=\z@\relax
\newcount\tp@int@flt@cnt \tp@int@flt@cnt\z@
```

Various dimension registers that store dimensions and spaces of floats and sub-floats:

- `\tp@subflt@maxheight` stores and self-updates the height of the largest sub-float inside a float
- `\tp@subflt@sep` is the space between sub-floats
- `\tp@total@flt@width` stores the cumulated overall width of the entire float
- `\tp@calc@flt@width` is an internal dimension used to calculate the ratio between mutiple sub-floats that should be scaled to the same height
- `\tp@total@flt@height` is the overall height of a float
- `\tp@total@flt@depth` is the overall depth of a float

```
\newdimen\tp@subflt@maxheight \tp@subflt@maxheight=\z@\relax
\newdimen\tp@subflt@sep \tp@subflt@sep=\fboxsep\relax
\newdimen\tp@total@flt@width \tp@total@flt@width=\textwidth\relax
\newdimen\tp@calc@flt@width \tp@calc@flt@width=\tp@total@flt@width\relax
\newdimen\tp@total@flt@height \tp@total@flt@height=\textwidth\relax
\newdimen\tp@total@flt@depth \tp@total@flt@depth=\textwidth\relax
```

Those two dimensions are used to pass the `intext-skip` and `float-skip` Properties to the render engine for spacing above and below the float, respectively.

```
\newskip\tp@flt@sep@top \tp@flt@sep@top=\z@\relax
\newskip\tp@flt@sep@bottom \tp@flt@sep@bottom=\z@\relax
```

Internal dimensions for the horizontal margins (right, left, inner and outer, respectively)

```
\newdimen\tp@flt@marg@r \tp@flt@marg@r=\z@\relax
\newdimen\tp@flt@marg@l \tp@flt@marg@l=\z@\relax
\newdimen\tp@flt@marg@i \tp@flt@marg@i=\z@\relax
\newdimen\tp@flt@marg@o \tp@flt@marg@o=\z@\relax
```

Locally adjustable switch to allow captions to break across pages

```
\newif\if@tp@flt@break@capt \@tp@flt@break@captfalse
```

String definitions for Property value comparisons

```
\def\tp@str@figure{figure}
\def\tp@str@table{table}
\def\tp@str@bottom{bottom}
\def\tp@str@top{top}
```

2.2 AtBeginDocument hook

```
\AtBeginDocument{%
```

Storing the final definitions of `\label`

```
  \global\let\tpltx@label\label
```

implementing the `nofigs` option, doing some minor adjustments to the `htmltabs` package and store the final definition of includegraphics.

```
  \ifx\tp@nofigs\relax
    \renewcommand\includegraphics[2][]{}%
  \fi
  \global\let\tpltx@includegraphics\includegraphics
```

Adjustments to the `htmltabs` package, if it is used:

```
  \@ifpackageloaded{htmltabs}
    {\global\let\tp@uses@htmltabs\relax
     \def\ht@adjust@linewidth{%
       \advance\ht@h@offset\leftskip
       \advance\ht@h@offset\@totalleftmargin
       %\advance\linewidth-\leftskip
       \advance\linewidth-\rightskip
     }%
    }{}%
```

In order to catch the actual dimensions of the float box, we need to hook into LaTeX's `\@endfloatbox` macro. This macro is low-level enough so it covers regular, double-column, and rotated floats. Those values will later be written into the `.aux` file for each float. The values, together with the float's overall width, are stored in a macro called `tp-float-\the\tp@int@flt@cnt-dimens`.

```
  \gappto\@endfloatbox{%
    \global\tp@total@flt@height=\ht\@currbox\relax%
    \global\tp@total@flt@depth=\dp\@currbox\relax%
  }%
}%
```

3 Internal macros

3.1 Generic resetter

`\tp@flt@reset@defaults` resets the parameters for sub-floats.

#1	the caption type (e.g., `figure`, `table`)
#2	abbreviation of the caption list (e.g., standard LaTeX uses `lof` for the List of Figures, `lot` for the List of Tables)

```
\def\tp@flt@reset@defaults{%
  \global\tpSubFloatCnt=\z@
  \global\tp@total@flt@width=\z@
  \global\let\tp@has@capt@top\@undefined
  \global\let\tp@has@capt@bottom\@undefined
  \global\let\tp@has@subcapt@top\@undefined
  \global\let\tp@has@subcapt@bottom\@undefined
  \global\let\tp@sub@contentsline@store\@empty
  \global\tp@subflt@maxheight=\z@\relax
  \@tempcnta=\z@\relax
  \tp@reset@components{\tp@cur@cont}%
  \let\tp@prefix\@empty
  \let\ht@cur@element\tp@captype
  \global\let\tp@current@class\relax
}
```

3.2 Internal macros that handle Attributes

`\tp@get@flt@attr` invokes the parser for the optional argument of float environments.

#1	is the content of the optional argument,
#2	is the caption type.

```
\def\tp@get@flt@attr#1#2{%
  \if!#1!\else
    \tpParseAttributes{#2}{#1}%
    \tpIfAttr{#2}{class}
      {\global\letcs\tp@current@class{tp@#2@attr@class}%
       \tpUseClass{default}{\tp@captype}%
       \expandafter\tpUseClass\expandafter{\csname tp@#2@attr@class\endcsname}{\tp@captype}}
      {}%
      \tpIfAttr{#2}{break-caption}{\@tp@flt@break@capttrue}{}%
  \fi
  \tp@get@flt@pos{#2}}
```

`\tp@get@flt@pos` is the handler for determining the floating position. Some float Properties and Attributes restrict and override the explicit float positions, e.g., fully rotated floats must be positioned in `p` mode (i.e., as float page). #1 is the caption type.

```
\def\tp@get@flt@pos#1{%
  \tpIfAttr{#1}{float-pos}
    {\letcs\tp@fps{tp@#1@attr@float-pos}}
    {\let\tp@fps\@empty}%
  \def\@tempa{h!}\ifx\tp@fps\@tempa\let\tp@fps\@empty\fi
  \def\@tempa{h}\ifx\tp@fps\@tempa\def\tp@fps{htbp!}\fi
  \ifx\tp@do@dblfloat\relax
    \ifx\tp@fps\@empty\def\tp@fps{htpb!}\fi% 11514
    \linewidth\dimexpr2\columnwidth+\columnsep\relax
    \hsize\linewidth\relax
  \fi
  \tpIfAttrStr{#1}{orientation}{landscape}
    {\linewidth\textheight
     \hsize\linewidth
     \def\tp@fps{p}}
    {}}
```

`\tp@set@flt@env` determines the low-level LaTeX float environment depending on orientation and document options. If no `float-pos` is given (implicitely or determined), the object is not treated as a float at all.

```
\def\tp@set@flt@env{%
  \ifx\tp@fps\@empty
    \let\tp@b@float\relax
    \let\tp@e@float\relax
    \ifhmode\par\fi
  \else
    \let\tp@b@float\tp@captype%
    \tpIfAttrStr{\tp@captype}{orientation}{landscape}
      {\edef\@tp@b@float{sideways\tp@b@float}%
       \edef\tp@b@float{\noexpand\begin{\@tp@b@float\ifx\tp@do@dblfloat\relax*\fi}}%
       \edef\tp@e@float{\noexpand\end{\@tp@b@float\ifx\tp@do@dblfloat\relax*\fi}}}
      {\edef\tp@flt@env{\ifx\tp@do@dblfloat\relax dbl\fi}%
       \edef\tp@b@float{\expandafter\noexpand\csname @x\tp@flt@env float\endcsname {\tp@captype
           }[\tp@fps]}%
       \edef\tp@e@float{\expandafter\noexpand\csname end@\tp@flt@env float\endcsname}}%
   \fi}
```

`\tp@flt@debug` prints some debug information to `stdout` for a single float that has the Attribute `debug` set.

```
\def\tp@flt@debug#1{%
  \tpIfAttr{#1}{debug}
  {\message{^^J[tp Float Debug]^^J
      Textheight:\space\the\textheight^^J
      Type:\space\space\space\space\space\space\space\tp@cur@cont^^J
\ifx\tp@captype\tp@str@figure
      Path: \space\space\space\space\space\space\@tp@fig@path^^J
\fi
      Class:\space\space\space\space\space\space\tp@current@class^^J
      Floatpos:\space\space\space\tp@fps^^J
      Environ:\space\space\space\space\expandafter\noexpand\tp@b@float...\expandafter\noexpand\
          tp@e@float^^J
      Subfloat:\space\space\space \the\tpSubFloatCnt^^J
\ifnum\tpSubFloatCnt=\z@
      Width:\space\space\space\space\space\space\the\tp@total@flt@width^^J
      Height:\space\space\space\space\space\the\tp@total@flt@height^^J
      Depth:\space\space\space\space\space\space\the\tp@total@flt@depth^^J
\else
      Width \the\tpSubFloatCnt:\space\space\space\space\space\space\expandafter\meaning\csname
          tp@\tp@cur@cont @width-\the\tpSubFloatCnt\endcsname^^J
      Height \the\tpSubFloatCnt:\space\space\space\space\space \expandafter\meaning\csname tp@\
          tp@cur@cont @height-\the\tpSubFloatCnt\endcsname^^J
      Depth \the\tpSubFloatCnt:\space\space\space\space\space\space\expandafter\meaning\csname
          tp@\tp@cur@cont @depth-\the\tpSubFloatCnt\endcsname^^J
\fi}}{}}
```

`\tp@get@seps` determines the top and bottom skips dependent on float position and orientation

```
\def\tp@get@seps{%
  \ifx\tp@fps\@empty
    \expandafter\tp@flt@sep@top\dimexpr\tpUseProperty{intext-skip-top}\relax%
  \else
    \expandafter\tp@flt@sep@top\dimexpr\tpUseProperty{float-skip-top}\relax%
  \fi
    \tpIfAttrStr{\tp@captype}{orientation}{landscape}{}
      {\ifx\tp@fps\@empty
         \expandafter\tp@flt@sep@bottom\dimexpr\tpUseProperty{intext-skip-bottom}\relax%
       \else
         \expandafter\tp@flt@sep@bottom\dimexpr\tpUseProperty{float-skip-bottom}\relax%
       \fi}}
```

`\tp@set@*@sep` Hooks to apply top and bottom skips, respectively.

```
\def\tp@set@top@sep{\addvspace{\tp@flt@sep@top}}
\def\tp@set@bot@sep{\addvspace{\tp@flt@sep@bottom}}
```

4 Float Container and Component Declarations

`\tpMakeFltComp` is a shortcut for float Component declarations. #1 is the generic name of the Component.

```
\def\tpMakeFltComp#1{%
  \tp@def@counted@comp{#1-\the\tpSubFloatCnt}{#1}{\ifx\tp@is@subflt\relax\else\tpSubFloatCnt=\z@
      \relax\fi}{}%
}
```

`\tpMakeFltCompL` is a shortcut to declare Float Components together with their *list-of* overrides. #1 is the generic name of the Component.

```
\def\tpMakeFltCompL#1{%
  \tpMakeFltComp{#1}%
  \tpMakeFltComp{Listof#1}}
```

`\tp@flt@set@hsize` calculates the available maximum width for the float contents and captions according to the values of the `margin-right` and the `margin-left` properties.

```
\def\tp@flt@set@hsize{%
  \expandafter\tp@subflt@sep\tpUseProperty{sub-float-sep}\relax%
  \global\tp@total@flt@width=\hsize\relax
  \expandafter\tp@flt@marg@l\tpUseProperty{margin-left}\relax
  \expandafter\tp@flt@marg@r\tpUseProperty{margin-right}\relax
  \expandafter\tp@flt@marg@i\tpUseProperty{margin-inner}\relax
  \expandafter\tp@flt@marg@o\tpUseProperty{margin-outer}\relax
  \tp@flt@set@margins
  \global\advance\tp@total@flt@width-\tp@flt@marg@r\relax
  }
```

`\tp@flt@set@margins` realises inner and outer margins via the left and right margins.

```
\def\tp@flt@set@margins{%
  \tp@test@page
  \if@tp@odd
    \advance\tp@flt@marg@l\tp@flt@marg@i
    \advance\tp@flt@marg@r\tp@flt@marg@o
  \else
    \advance\tp@flt@marg@l\tp@flt@marg@o
    \advance\tp@flt@marg@r\tp@flt@marg@i
  \fi
}
```

`Container float`

```
\tpDeclareContainer{float}{%
  \tpDeclareType{Components}{%
    \tpMakeFltCompL{Caption}%
    \tpMakeFltCompL{Legend}%
    \tpMakeFltCompL{Source}%
    \tpMakeFltCompL{Number}%
    \tpMakeFltComp{RefLabel}%
    \tpMakeFltComp{AltText}% neu: 2023-06-08; TODO: muss noch implementiert werden
  }%
  \tpDeclareType{Properties}{}%
}
```

`\tpDeclareFloat` is the user-level macro used to (re-)declare a (new) `tpFloat` environment.

#1	Name of the float Container from which the declared Container should inherit Properties (*optional*)
#2	top-level name of the float environment (e.g., `tpTable`, `tpFigure`)
#3	caption type (e.g., `table`, `figure`)
#4	list (e.g., `lot`, `lof`)
#5	Property list

```
\def\tpDeclareFloat{\tp@opt@empty\@tpDeclareFloat}
\long\def\@tpDeclareFloat[#1]#2#3#4#5{%
  \def\tp@float@parent{#1}%
```

If the float Container has already been declared, we only load its parent's Properties and Containers (if any), and add the override Properties to the Container's Property List. Otherwise, we would re-load the system's defaults and override the Properties of the earlier Declaration.

```
  \ifcsdef{tp@container@#2}{%
    \tpPackageInfo{Floats}{}{Appending to '#2'}%
    \ifx\tp@float@parent\@empty\else
      \tpPackageError{Float}{Type}
        {Attempt to change parent of pre-existing float^^JContainer '#2'}
        {You cannot use the optional argument of \string\tpDeclareFloat\space for pre-existing^^J
            %
float containers!^^J^^J%
Use \string\tpAddToType{<Type>}{#2}{<code>}\space to alter the #2 container!}
    \fi
    \tpAddToType{Properties}{#2}{#5}%
```

Other than Properties, the Float's default caption type or list-of handler may also be overridden by a re-definition.

```
    \tpAddToType{FloatEnvInfo}{#2}{%
      \def\tp@captype{#3}%
      \def\tp@caplisttype{#4}%
    }%
  }{%
```

Otherwise, we declare a new Container and invoke all the Initializers.

```
    \tpDeclareContainer{#2}{%
      \tpPackageInfo{Floats}{}{Declaring float '#2'}%
      \ifx\tp@float@parent\@empty
        \tpInherit{Properties,Components}{float}
      \else
        \tpInherit{Properties,Components}{\tp@float@parent}
      \fi
      \tpDeclareType{FloatEnvInfo}{%
        \tpNamespace{#2}%
      \def\tp@captype{#3}%
      \def\tp@caplisttype{#4}%
    }% /FloatEnvInfo
```

The macro actually defines two LaTeX environments; a normal one for one-column floats, and a starred one for page-wide floats in two-column mode.

```
    \tpDeclareEnv[#2]{\tp@float}{\endtp@float}%
    \tpDeclareEnv[#2*]{\if@twocolumn\let\tp@do@dblfloat\relax\else\fi\tp@float}{\if@twocolumn\
        let\tp@do@dblfloat\relax\fi\endtp@float}%
    \tpDeclareType{Components}{%
      \tpUseProperty{float-handler}%
    }%
```

Generating the Handlers for the list-of entries and define the corresponding `l@` macros

```
    \tp@flt@generate@listof@handlers{#4}{#3}{#2}%
    \bgroup
      \def\tp@cur@cont{#2}%
      \tp@init@l@[list-of]{#4}{0}{#3}% Generate listof-Entries for first level floats
      \tp@init@l@[list-of]{#4}{1}{sub#3}% Generate listof-Entries for sub-floats
    \egroup
    \tpDeclareType{Properties}{#5}%
  }% /container
 }% /ifcsdef{tp@app@container@#2}
}
```

`\tp@flt@generate@listof@handlers` generates handlers for listof-entries.

`#1`	is the file ending
`#2`	is the caption type
`#3`	is the Container name

```
\def\tp@flt@generate@listof@handlers#1#2#3{%
```

`tp@<list>@extract@data` The first macro that is dynamicly defined, is the Component collector.

`##1`	is a numeric level that represents the order of the listof-entries
`##2`	is the caption type
`##3`	is the content of the `l@<level>` macro
`##4`	is the page number associated with that entry.

```
 \expandafter\gdef\csname tp@#1@extract@data\endcsname##1##2##3##4{%
   \tpNamespace{#3}%
   \tpEvalType[#3]{Properties}%
   \tpDeclareComp{ListofCaption}{}{}%
   \tpDeclareComp{ListofLegend}{}{}%
   \tpDeclareComp{ListofSource}{}{}%
   \tpDeclareComp{ListofNumber}{}{}%
   \tpDeclareComp{ListofPage}{}{}%
   \tpListofPage{\tpUseProperty{list-of-page-face}##4}%
   \tp@expand@l@contents{##3}{#3}{Listof}{Caption}%%
   \tp@format@number{list-of-}{Listof}{##1}%
 }%
```

`\csname tp@<list>@print@entry\endcsname` The second dynamically defined macro is the entry renderer. It applies the Listof properties and selects the components to be printed. ##1 is the caption type of the float.

```
  \expandafter\gdef\csname tp@#1@print@entry\endcsname##1{%
    \bgroup
      \tpUseHook{list-of-before-hook-##1}%
      \tpUseProperty{list-of-before-entry}%
      \tpUseProperty{list-of-block}%
      \tpUseHook{list-of-after-hook-##1}%
      \tpUseProperty{list-of-after-entry}%
    \egroup}%
```

`\csname tp@make@listof@<type>\endcsname` The last macro to be defined here is the list-of writer. This macro is responsible to write the entry into TeX's auxiliary file system. ##1 is the name of the "level" for the entry.

```
  \expandafter\gdef\csname tp@make@listof@#2\endcsname##1{%
    \tpGobble
    \tp@flt@check@empty{Number}{number}%
    \tp@flt@check@empty{Caption}{caption}%
    \tp@flt@check@empty{Legend}{legend}%
    \tp@flt@check@empty{Source}{source}%
    \tpIfAttrIsset{#2}{nolist}{}
      {\let\@tp@listof@entry\relax
      \tpIfComp{ListofCaption}{\csgappto{@tp@listof@entry}{\string\tpListofCaption{\tpUseComp{
          ListofCaption}}}}{}%
      \tpIfComp{ListofNumber}{\csgappto{@tp@listof@entry}{\string\tpListofNumber{\tpUseComp{
          ListofNumber}}}}{}%
      \tpIfComp{ListofLegend}{\csgappto{@tp@listof@entry}{\string\tpListofLegend{\tpUseComp{
          ListofLegend}}}}{}%
      \tpIfComp{ListofSource}{\csgappto{@tp@listof@entry}{\string\tpListofSource{\tpUseComp{
          ListofSource}}}}{}%
      \ifx\@tp@listof@entry\relax
        \ifx\tp@is@subflt\relax\else
          \tp@restore@contentsline
        \fi
      \else
        \protected@edef\tp@listof@entry{\@tp@listof@entry}%
        \ifx\tp@is@subflt\relax
          \tp@store@sub@contentsline{#1}{\tp@captype}{\expandonce{\tp@listof@entry}}%
        \else
          \tp@flt@addcontentsline{#1}{\tp@captype}{\expandonce{\tp@listof@entry}}%
          \tp@restore@contentsline
        \fi
      \fi
    }%
  }%
}
```

`\tp@store@sub@contentsline` saves the contentsline macros for prematurely expanded captions.

If we immediatetly write the list-of entries for sub-floats into the list-of files, they will be printed before their respective parent entry. This is because sub-floats are processed before their parent floats. To avoid the wrong order in the list-of, we progressively store the sub-float's addcontentsline commands in the `\tp@sub@contentsline@store` macro and expand it after the list-of for the parent float has been processed.

```
\def\tp@store@sub@contentsline#1#2#3{%
  \protected@xdef\tp@sub@contentsline@store{\expandonce{\tp@sub@contentsline@store}\noexpand\
      tp@flt@addcontentsline{#1}{#2}{#3}\relax}}
```

`\tp@restore@contetnsline` restores and expands the list of sub-float addcontentsline commands, if there are any.

```
\def\tp@restore@contentsline{%
  \ifx\tp@sub@contentsline@store\@empty\else
    \tp@sub@contentsline@store
    \global\let\tp@sub@contentsline@store\@empty
  \fi
}
```

`\tp@flt@addcontentsline` fork of LaTeX's `\addtocontents` macro

#1	extension of the list file
#2	caption type; passed to the first argument of LaTeX's `\contentsline`
#3	the entry itself; passed to the second argument of LaTeX's `\contentsline`

```
\def\tp@flt@addcontentsline#1#2#3{%
  \protected@write\@auxout
    {\tpGobble}%
    {\string\@writefile{#1}{\protect\tpContentsline{#2}{#3}{\thepage}{\@currentHref}\
        protected@file@percent}}\relax
}
```

`\tp@flt@check@empty` fork of CoCoTeX kernel's `\tp@check@empty`, probably DEPRECATED(?)

```
\def\tp@flt@check@empty#1#2{%
  \ifx\tp@is@subflt\relax\else\tpSubFloatCnt\z@\fi
  \tpIfComp{Listof#1}
    {}
    {\tpIfComp{#1}
      {\csletcs{tp@\tp@cur@cont @Listof#1-\the\tpSubFloatCnt}{tp@\tp@cur@cont @#1-\the\
          tpSubFloatCnt}}
      {\csname Listof#1\endcsname{}}}}
```

5 Label and Referencing mechanisms

`\tp@flt@create@counters` creates auto-numbered counters. We advance the caption type only locally since they are automatically and globally updated when `\tp@make@anchors` is called.

```
\def\tp@flt@create@counters{%
  \tpIfAttrIsset{\tp@captype}{nonumber}{}
    {\tpIfPropVal{numbering}{auto}
      {\tpIfComp{number-0}
         {}%
         {\expandafter\advance\csname c@\tp@captype\endcsname\@ne\relax
          \tp@set@label{0}%
          \expandafter\advance\csname c@\tp@captype\endcsname\m@ne\relax
         }%
       \ifnum\tpSubFloatCnt=\z@\relax\else
         \@tempcnta\z@
         \tp@iterate{\@tempcnta}{\@ne}{\tpSubFloatCnt}{%
           \tpIfComp{number-\the\@tempcnta}
             {}%
             {\tpIfAttr{\tp@captype}{subfloat}
                {\tp@set@sublabel{\the\@tempcnta}}
                {\expandafter\advance\csname c@\tp@captype\endcsname\@ne\relax
                 \tp@set@label{\the\@tempcnta}%
                 \expandafter\advance\csname c@\tp@captype\endcsname\m@ne\relax}}}%
      \fi}
     {}%
  }}
```

`\tp@set@label` generates the first level float counter. #1 is the sub-float counter.

```
\def\tp@set@label#1{%
  \expandafter\expandafter\expandafter\edef\expandafter\csname tp@\tp@cur@cont @number-#1\
      expandafter\endcsname\expandafter{\csname the\tp@captype\endcsname}%
}
```

`\tp@set@sublabel` generates second level counters for numbered sub-floats. #1 is the sub-float counter

TODO: float-number und sub-number sollten beides Components sein, nicht Properties!

```
\def\tp@set@sublabel#1{%
  \tpSetValProp{float-number}{\csname tp@\tp@cur@cont @number-0\endcsname}%
  \tpSetValProp{sub-number}{%
    \begingroup
      \expandonce{\tpUseProperty{sub-number-face}}%
      \relax\tpUseProperty{sub-number-before}%
      \csname @\tpUseProperty{sub-number-style}\endcsname{#1}%
      \tpUseProperty{sub-number-after}%
    \endgroup}%
  \expandafter\expandafter\expandafter\edef\expandafter\csname tp@\tp@cur@cont @number-#1\
      expandafter\endcsname\expandafter{\tpUseProperty{sub-number-format}}%
}
```

The next two macros are a re-implementation of `hyperref`'s anchor mechanism to make labels work. If no explicit label is given, the mechanism generates one, unique to each (sub)float.

`\tp@make@anchors` iterates through the (sub-)floats of a `float` Container instance and generates the anchor (and hidden label, if necessary) for each of them

```
\def\tp@make@anchors{\@tempcnta\z@\tp@iterate{\@tempcnta}{\z@}{\tpSubFloatCnt}{\tp@make@anchor{\
    the\@tempcnta}}}
```

`\tp@make@anchors` generates the anchor and label of a single (sub-)float. #1 is the value of the internal sub-float counter.

```
\def\tp@make@anchor#1{%
  \bgroup
    \tpSubFloatCnt#1\relax
    \tpIfComp{RefLabel}
      {\expandafter\let\expandafter\@currentlabel\csname tp@\tp@cur@cont @number-\the\
          tpSubFloatCnt\endcsname}
      {\edef\@currentlabel{tp-\tp@cur@cont-number-\the\tp@int@flt@cnt}}%
    \expandafter\H@refstepcounter\expandafter{\tp@captype}%
    \expandafter\hyper@makecurrent\expandafter{\tp@captype}%
    \global\let\Hy@tempa\Hy@float@caption
    \expandafter\hyper@@anchor\expandafter{\@currentHref}{\relax}%
    \tpIfComp{RefLabel}
      {\expandafter\let\expandafter\@currentlabel\csname tp@\tp@cur@cont @number-\the\
          tpSubFloatCnt\endcsname
       \edef\@tempa{\tpUseComp{RefLabel}}%
       \expandafter\tpltx@label\expandafter{\@tempa}}{\relax}%
  \egroup}
```

6 Processing the Float

6.1 Common Float and Sub-Float Environments

`\tp@float` is a mid-level Macro that provides the common floating LaTeX environment. #1 is the float environment's kv-attribute list.

#1 float position (optional)

```
\def\tp@float{\tp@opt@empty\@tp@float}
\def\@tp@float[#1]{%
  \par
  \begingroup
    \global\advance\tp@int@flt@cnt\@ne
    \tpEvalType{FloatEnvInfo}%
    \tp@flt@reset@defaults
    \tpToggleCountedCond
    \tpEvalType{Properties}%
    \tp@get@flt@attr{#1}{\tp@captype}%
    \tp@flt@set@hsize
    \tp@get@seps
    \tpEvalType{Components}%
    \tpUseProperty{before-float}%
    \tp@set@flt@env
    \ifx\tp@fps\@empty\else\savenotes\fi
}
```

`\endtp@float` is the end of the common float environment.

```
\def\endtp@float{%
    \tp@b@float
    \tp@set@top@sep
    \tp@test@caption{0}{capt}{top}%
    \tp@test@caption{0}{capt}{bottom}%
    \tp@flt@create@counters%
    \tp@flt@compose
    \tp@save@page
    \tp@set@bot@sep
    \tp@e@float
    \tp@flt@debug{\tp@captype}%
    \ifx\tp@fps\@empty\else\spewnotes\fi
  \endgroup
  \immediate\write\@auxout
    {\string\expandafter\string\gdef\string\csname\space tp-float-\the\tp@int@flt@cnt-dimens\
        string\endcsname{%
      {\the\tp@total@flt@width}%
      {\the\tp@total@flt@height}%
      {\the\tp@total@flt@depth}%
    }}%
  \global\let\tp@current@class\relax
}
```

`\tpSubFloat` is the user-level environment for sub-floats

TODO: transform into a Component Group

```
\def\tpSubFloat{%
  \ifx\tp@is@subflt\relax
```

```
    \PackageError{coco-floats.sty}{Nested tpSubFloats detected!}{You cannot (yet) nest a `
        tpSubFloat' environment into another `tpSubFloat' environment!}%
  \else
    \let\tp@is@subflt\relax
    \global\advance\tpSubFloatCnt\@ne
    \ignorespaces
  \fi}
```

`\endtpSubFloat` is the end of the sub-float environment

```
\def\endtpSubFloat{%
  \tpUseProperty{subfloat-handler}%
  \expandafter\xdef\csname tp@\tp@cur@cont @width-\the\tpSubFloatCnt\endcsname{\the\wd\
      tp@subfltbox}%
  \expandafter\xdef\csname tp@\tp@cur@cont @height-\the\tpSubFloatCnt\endcsname{\the\ht\
      tp@subfltbox}%
  \expandafter\xdef\csname tp@\tp@cur@cont @depth-\the\tpSubFloatCnt\endcsname{\the\dp\
      tp@subfltbox}%
  \@tempdima=\dimexpr\the\ht\tp@subfltbox+\the\dp\tp@subfltbox\relax
  \@tempdimb=\dimexpr\the\wd\tp@subfltbox\relax
  \ifdim\@tempdima>\tp@subflt@maxheight\relax
    \global\tp@subflt@maxheight=\@tempdima\relax
  \fi
  \ignorespaces
  \tpIfAttr{\tp@captype}{subfloat}
    {\csname tp@make@listof@\tp@captype\endcsname{sub\tp@captype}}% real subfloats
    {\csname tp@make@listof@\tp@captype\endcsname{\tp@captype}}% subfloats are counted separately
  \setbox\tp@subfltbox\box\voidb@x
  \let\tp@is@subflt\@undefined
}
```

6.2 Processing the Contents of the Float Environment

`\tp@flt@process` prints the contents of a float environment.

```
\def\tp@flt@process{%
  \tp@test@subcapt
  \ifx\tp@has@capt@top\@empty\leavevmode\fi
  \tp@make@outer@caption{top}%
  \ifnum\tpSubFloatCnt=\z@\relax
    \bgroup\advance\hsize-\tp@flt@marg@l
      \tpUseProperty{float-render}%
    \egroup
  \else
    \let\tp@is@subflt\relax
    \tp@flt@calc@sameheight
    \ifx\tp@has@subcapt@top\@empty\tp@flt@calc@row@ht{top}\fi%
    \ifx\tp@has@subcapt@bottom\@empty\tp@flt@calc@row@ht{bottom}\fi%
    \def\tp@prefix{sub}%
    \tpUseProperty{subfloat-render}%
    \let\tp@prefix\@empty
    \let\tp@is@subflt\@undefined
  \fi
  \tp@make@outer@caption{bottom}%
}
```

`\tp@flt@compose` This macro prints the entire float object.

```
\def\tp@flt@compose{%
  \bgroup
    \hsize\tp@total@flt@width
    \tp@flt@process
    \tp@make@anchors%
    \csname tp@make@listof@\tp@captype\endcsname{\tp@captype}% single float
    \par
  \egroup}
```

6.3 Caption mechanism

\tp@test@caption tests if the current sub-float has any top or bottom caption that needs to be printed.

#1	is the value of the sub-float counter
#1	indicates if the caption belongs to the whole float (capt) or a sub-float (subcapt)
#1	top or bottom

We compare the caption of the current \SubFloatCnt level with a caption of a non-existing Float level in case there is non-expandable material hard-coded into the caption-#3 Property. If we were to compare the width of the \hbox with \z@, this scenario would give us false positives.

Warning: Long captions can cause the hbox's width to exceed \maxdimen. To avoid LaTeX errors in this case, we compare sp instead of pt. This, however, means that if the difference is less than 1pt, the test fails and no caption is printed!

```
\def\tp@test@caption#1#2#3{%
  \setbox\tp@tempboxa\hbox{\tpGobble\tpSubFloatCnt0#1\relax\tpUseProperty{caption-#3}\relax}%
  \setbox\tp@tempboxb\hbox{\tpGobble\tpSubFloatCnt\m@ne\relax\tpUseProperty{caption-#3}\relax}%
  \edef\my@wda{\expandafter\strip@pt\wd\tp@tempboxa sp}%
  \edef\my@wdb{\expandafter\strip@pt\wd\tp@tempboxb sp}%
  \ifdim\my@wda>\my@wdb\relax
    \expandafter\global\expandafter\let\csname tp@has@#2@#3\endcsname\@empty
  \fi
}
```

\tp@test@subcapt tests if the current float has any top or bottom captions that need to be printed

```
\def\tp@test@subcapt{%
  \tp@iterate{\@tempcnta}{\@ne}{\tpSubFloatCnt}{%
    \tp@test@caption{\the\@tempcnta}{subcapt}{top}%
    \tp@test@caption{\the\@tempcnta}{subcapt}{bottom}%
  }%
}
```

\tp@capt@top@offset determines the spacing inserted **above both captions**.

```
\def\tp@capt@top@offset{%
  \ifx\@argi\tp@str@top
  \else
    \par\if@tp@flt@break@capt\else\nopagebreak\fi%
    \expandafter\@tempskipa\tpUseProperty{\tp@prefix caption-sep-bottom}\relax%
    \advance\@tempskipa\dimexpr-\topskip+\dp\strutbox\relax
    \if@tp@flt@break@capt\advance\@tempskipa\dimexpr-\baselineskip-\ht\strutbox+\topskip\relax\
        fi
    \ifx\tp@has@subcapt@bottom\@empty
      \ifnum\tpSubFloatCnt=\z@
        %% subcapt-bot exists and capt-bot is rendered
      \advance\@tempskipa\dimexpr\dp\strutbox\relax
      \expandafter\advance\expandafter\@tempskipa\tpUseProperty{subcaption-add-sep-bottom}\
          relax%
    \fi
  \fi
  \vskip\@tempskipa
  \leavevmode
\fi}
```

d etermines the spacing inserted **below the captions**.

```
\def\tp@capt@bottom@offset{%
  \ifx\@argi\tp@str@top
    \@tempskipa\z@
    \expandafter\advance\expandafter\@tempskipa\tpUseProperty{\tp@prefix caption-sep-top}%
    %
    \ifnum\tpSubFloatCnt=\z@
      \ifx\tp@has@subcapt@top\@empty
        %% subcapt-top exists and capt-top is rendered
        \advance\@tempskipa\dimexpr\ht\strutbox-\topskip-\p@\relax
        \expandafter\advance\expandafter\@tempskipa\tpUseProperty{subcaption-add-sep-top}\relax%
      \else
        \advance\@tempskipa\dimexpr-\dp\strutbox\relax
      \fi
    \fi
    \vskip\@tempskipa
    \par\if@tp@flt@break@capt\else\nopagebreak\fi
  \else
    \ifnum\tpSubFloatCnt>\z@
      \vskip\dp\strutbox
    \fi
  \fi}
```

`\tp@make@caption` prints the caption.

`#1` is the placement (`top`, `bottom`)
`#2` is the vertical alignment (`top`, `middle`, `bottom`)
`#3` is the left margin.

```
\long\def\tp@make@caption#1#2{%
  \edef\@argi{#1}\edef\@argii{#2}%
  \tp@capt@top@offset
  \ifnum\tpSubFloatCnt=\z@
    \def\next{%
      \tpIfAttrStr{\tp@captype}{orientation}{landscape}
        {\setbox\@tempboxa\vbox\bgroup\hsize\textheight}
        {\hskip\tp@flt@marg@l%
         \setbox\@tempboxa\vbox\bgroup\advance\hsize-\tp@flt@marg@l}%
      }%
  \else
    \expandafter\tp@tempskipa\csname tp@flt@capt@row@height@#1\endcsname\relax
    \expandafter\advance\expandafter\tp@tempskipa\dimexpr-\baselineskip+\topskip\relax
    \def\next{\setbox\@tempboxa\vbox to \tp@tempskipa\bgroup}%
  \fi
  \next%
    \ifx\@argii\tp@str@top\else\if@tp@flt@break@capt\else\vss\fi\fi
    \tpUseProperty{\tp@prefix caption-face}%
    \tpUseProperty{\tp@prefix caption-face-#1}%
    \tp@topstrut\tpUseProperty{caption-#1}\strut%
    \ifx\@argii\tp@str@bottom\else\if@tp@flt@break@capt\else\vss\fi\fi%
  \egroup%
  \if@tp@flt@break@capt\unvbox\@tempboxa\else\box\@tempboxa\fi%
  \tp@capt@bottom@offset
}
```

`\tp@make@outer@caption` is a shell for the outer captions. #1 is the placement (`top`, `bottom`)

```
\def\tp@make@outer@caption#1{%
  \def\@argi{#1}%
  \expandafter\ifx\csname tp@has@capt@#1\endcsname\@empty
    \setbox\z@\vbox{%
      \tpGobble
      \tpSubFloatCnt\z@
      \tp@make@caption{#1}{top}%
    }%
    \immediate\write\@auxout{\string\expandafter\string\gdef\string\csname\space tpFloat\the\
        tp@int@flt@cnt Cap#1\string\endcsname{\the\dimexpr \ht\z@+\dp\z@\relax}}%
    \bgroup
      \savenotes
      \if@tp@flt@break@capt\else\nopagebreak\fi
      \tpSubFloatCnt\z@
      \tp@make@caption{#1}{top}%
      \spewnotes
    \egroup
    \ifx\@argi\tp@str@top\if@tp@flt@break@capt\else\nopagebreak\fi\fi
  \fi
}
```

`\tpRenderSubFloats` iterates through the single sub-floats and renders them in a nice row.

`#1` is the subfloat counter,
`#2` Component name that contains the actual contents of the sub-float, for `tpFigure` it is `Fig`, for `tpTable` it is `Content`.

```
\long\def\tpRenderSubFloats#1#2{%
  \leavevmode
  \savenotes
  \ifnum#1>\@ne\hfill\fi
  \vtop\bgroup
    \expandafter\hsize\csname tp@\tp@cur@cont @res@width-#1\endcsname\relax
    \let\includegraphics\tp@includesubgraphics
    \tp@render@sub@float{#1}{#2}%
  \egroup
  \spewnotes
}
```

`\tp@render@sub@float` renders a single sub-float. For the arguments, see `\tpRenderSubFloats`, above.

```
\long\def\tp@render@sub@float#1#2{%
  \tpSubFloatCnt=#1\relax
  \expandafter\ifx\csname tp@has@\tp@prefix capt@top\endcsname\@empty
    \tp@make@caption{top}{\tpUseProperty{\tp@prefix caption-valign-top}}%
  \fi
  \bgroup\strut\tpUseComp{#2}\strut\par\egroup%
  \expandafter\ifx\csname tp@has@\tp@prefix capt@bottom\endcsname\@empty
    \tp@make@caption{bottom}{\tpUseProperty{\tp@prefix caption-valign-bottom}}%
  \fi
}
```

`\tp@flt@calc@row@ht` calculates the heights of all captions in the same row.

#1 determins if the `top` or `bottom` row is calculated.

```
\def\tp@flt@calc@row@ht#1{%
  \@tempcnta\z@
  \@tempdima\z@
  \tp@iterate{\@tempcnta}{\@ne}{\tpSubFloatCnt}{%
    \setbox\z@\vbox{%
      \tpSubFloatCnt\@tempcnta\relax
      \expandafter\hsize\expandafter\dimexpr\csname tp@\tp@cur@cont @res@width-\the\@tempcnta\
          endcsname\relax
      \tpGobble
      \tpUseProperty{\tp@prefix caption-face}%
      \tpUseProperty{\tp@prefix caption-face-#1}%
      \leavevmode
      \strut\tpUseProperty{caption-#1}\strut%
      }%
    \expandafter\ifdim\dimexpr\ht\z@+\dp\z@\relax>\@tempdima \@tempdima\dimexpr\ht\z@+\dp\z@\
        relax\fi
  }%
  \expandafter\edef\csname tp@flt@capt@row@height@#1\endcsname{\the\@tempdima}%
}
```

`\tp@flt@calc@sameheight` calculates the ratio between each sub-float's height and the height of the largest sub-float

```
\def\tp@flt@calc@sameheight{%
  \@tempdima=\z@\relax
  \@tempcnta=\z@\relax
  \tp@calc@flt@width=\tp@total@flt@width\relax
  \advance\tp@calc@flt@width-\tp@flt@marg@l\relax
  \tp@iterate{\@tempcnta}{\@ne}{\tpSubFloatCnt}{%
    \edef\@tempa{\CalcRatio{\tp@subflt@maxheight}{\csname tp@\tp@cur@cont @height-\the\@tempcnta
        \endcsname}}%
    \ifnum\@tempcnta>\@ne
      \advance\tp@calc@flt@width-\tp@subflt@sep\relax%
    \fi
    \expandafter\@tempdimc\csname tp@\tp@cur@cont @width-\the\@tempcnta\endcsname\relax
    \@tempdimb=\@tempa\@tempdimc\relax
    \expandafter\edef\csname tp@\tp@cur@cont @adj@width-\the\@tempcnta\endcsname{\the\@tempdimb}%
    \advance\@tempdima\@tempdimb
  }%
  \@tempcnta=\z@\relax
  \@tempdimb=\z@\relax
  \@tempdimc=\z@\relax
  \tp@iterate{\@tempcnta}{\@ne}{\tpSubFloatCnt}{%
    \edef\@tempa{\CalcRatio{\csname tp@\tp@cur@cont @adj@width-\the\@tempcnta\endcsname}{\
        @tempdima}}%
    \expandafter\edef\csname tp@\tp@cur@cont @res@width-\the\@tempcnta\endcsname{\dimexpr\@tempa
        \tp@calc@flt@width\relax}%
    \@tempdimc\dimexpr\csname tp@\tp@cur@cont @height-\the\@tempcnta\endcsname\relax
    \@tempdimc\dimexpr\@tempa\@tempdimc\relax
    \ifdim\@tempa\@tempdimb<\@tempdimc\@tempdimb\@tempdimc\relax\fi
  }%
  \expandafter\edef\csname tp@\tp@cur@cont @res@height\endcsname{\the\@tempdimb}%
}
```

7 Handlers for different float types

7.1 Handlers for generic floats

\tpGenericRender is the Component that contains the contents of a generic float.

```
\def\tpGenericRender{\tpUseComp{Content}}
```

\tpGenericHandler is the generic content handler of a float

```
\def\tpGenericHandler{\tpMakeFltComp{Content}}
```

\tpSubGenericHandler is the generic handler of a sub-float.

```
\def\tpSubGenericHandler{}
```

7.2 Handlers for figures

\tpFigureHandler tells the float module the name, main namespace, and main content Container of tpFigure type floats.

```
\def\tpFigureHandler{\tpMakeFltComp{Fig}}
```

\tp@flt@create@natural is the actual handler for sub-figures.

```
\def\tp@flt@create@natural{\tpUseComp{Fig}}
```

\tpSubFigureHandler is the User-level macro that defines the handler for sub-figures. It also contains code for the nofigs package option.

```
\def\tpSubFigureHandler{%
  \ifx\tp@nofigs\relax
    \setbox\tp@subfltbox\hbox{\rule{0pt}{1pt}\rule{1pt}{0pt}}%
  \else
    \setbox\tp@subfltbox\hbox{\tpGobble\tp@flt@create@natural}%
  \fi}
```

\tpFigureRender tells the module how tpFigures are to be rendered.

```
\def\tpFigureRender{%
  \bgroup
    \tpIfAttrStr{\tp@captype}{orientation}{landscape}
      {\hsize\dimexpr\textwidth-\tp@flt@marg@r-\tp@flt@marg@l\relax}%
      {}%
    \let\includegraphics\tp@includesubgraphics
    \hskip\tp@flt@marg@l
    \strut\tpUseComp{Fig}\strut
  \egroup}
```

\tpSubFigureRender tells the module how sub-floats of tpFigure type floats are to be rendered.

```
\def\tpSubFigureRender{%
  \hskip\tp@flt@marg@l
  \tp@iterate{\@tempcnta}{\@ne}{\tpSubFloatCnt}{%
```

```
    \tpRenderSubFloats{\the\@tempcnta}{Fig}%
  }}
```

`\tp@includesubgraphics` is an override of LaTeX's `\includegraphics` patched to adjust for maximum width and height.

```
\def\tp@includesubgraphics{\@ifnextchar [\@tp@includesubgraphics{\@tp@includesubgraphics[]}}%]
\def\@tp@includesubgraphics[#1]#2{%
  \ifx\tp@current@class\relax
    \def\@igopts{max width=\hsize,max height=\vsize}%
  \else
    \def\@igopts{width=\hsize}%
  \fi
  \if!#1!\else
    \def\@igopts{width=\hsize,#1}%
  \fi
  \gdef\@tp@fig@path{#2}%
  \expandafter\tpltx@includegraphics\expandafter[\@igopts]{#2}%
}
```

7.3 Handlers for tables

`\tp@reserve@tabular` is a shell macro that stores the default macro definitions for various tabular mechanisms (currently, only plain *`tabular`*, `tabulary`, `tabularx`, and `htmltabs` are supported as content Component of `tpTable`)

```
\def\tp@reserve@tabular{%
  \@tp@reserve@tab{}%
  \@tp@reserve@tab{x}%
  \@tp@reserve@tab{y}%
  \@tp@reserve@htmltab%
}
```

`\@tp@reserve@tab` stores the default definitions for a specific vanilla-LaTeX tabular environment and re-defines the macros in a way that the tabulars are stored in the `\tp@floatbox` instead of printed onto the page.

```
\def\@tp@reserve@tab#1{%
  \expandafter\expandafter\expandafter\let\expandafter\csname orig@tabular#1\expandafter\
      endcsname\csname tabular#1\endcsname
  \expandafter\expandafter\expandafter\let\expandafter\csname orig@endtabular#1\expandafter\
      endcsname\csname endtabular#1\endcsname
  \expandafter\def\csname tabular#1\endcsname{%
    \global\setbox\tp@floatbox
    \vbox\bgroup
      \if!#1!\else
        \let\tabular\orig@tabular
        \let\endtabular\orig@endtabular
      \fi
      \csname orig@tabular#1\endcsname}%
  \expandafter\def\csname endtabular#1\endcsname{\csname orig@endtabular#1\endcsname\egroup}%
}
```

`\@tp@reserve@htmltab` special handler for tables using the `htmltabs` package:

```
\AtBeginDocument{%
  \@ifpackageloaded{htmltabs}{%
    \def\@tp@reserve@htmltab{%
      \let\tp@addstyle\@empty
    \ifx\tp@fps\@empty
      \expandafter\ifx\csname tpFloat\the\tp@int@flt@cnt Captop\endcsname\relax\else
        \htInitSkip\csname tpFloat\the\tp@int@flt@cnt Captop\endcsname
        \advance\htInitSkip\tp@flt@sep@top%
      \fi
      \expandafter\ifx\csname tpFloat\the\tp@int@flt@cnt Capbottom\endcsname\relax\else
        \htAddToBottom\csname tpFloat\the\tp@int@flt@cnt Capbottom\endcsname
        \advance\htAddToBottom\tp@flt@sep@bottom%
      \fi
    \else
      \def\tp@addstyle{;break-table:false;}%
    \fi
    \edef\tp@tempa{margin-left:\tp@flt@marg@l\tp@addstyle}%
    \expandafter\htAddStyle\expandafter{\tp@tempa}%
    \global\setbox\htTableBox\box\voidb@x
    \let\htOutputTable\relax
  }}{\let\@tp@reserve@htmltab\relax}%
}
```

\tpTableHandler defines the content handler for tpTable.

```
\def\tpTableHandler{%
  \tpMakeFltComp{Content}%
  \tp@reserve@tabular
  }
```

\tpGetTableContent returns the tp@floatbox if it is not un-itialized or void.

```
\def\tpGetTableContent{%
  \ifx\htTableBox\@undefined\else
    \ifvoid\htTableBox\else
      \let\tp@floatbox\htTableBox%
    \fi\fi}
```

\tpSubTableHandler is the handler for sub-tables. So far, coco-floats.sty does not support tables to be sub-floats, so we just generate an Error message.

```
\def\tpSubTableHandler{%
  \PackageError{coco-floats.sty}{tpSubFloat does not support sub-tables (yet)!}{You cannot yet
      use a tables within the ‘tpSubFloat’!}%
}
```

\tpTableRender defines the Renderer for tpTable content Components

```
\def\tpTableRender{%
  \tpGetTableContent
  \tpContent{\unvbox\tp@floatbox}%
  \tpUseComp{Content}%
  \par\if@tp@flt@break@capt\else\nopagebreak\fi
  \vskip\dp\strutbox
}
```

\tpSubTableRender Is the Renderer for table sub-floats (which we don’t allow yet, so this definition is un-used at the moment)

```
\def\tpSubTableRender{%
  \tp@iterate{\@tempcnta}{\@ne}{\tpSubFloatCnt}{%
    \tpGetTableContent
    \tpContent{\unvbox\tp@floatbox}%
    \tpRenderSubFloats{\the\@tempcnta}{Content}%
  }}
```

7.4 Helpers

`\tpFloatBarrier` can be used to force all pending floats to be printed at the next shipout.

```
\def\tpFloatBarrier{\AtBeginShipoutNext{\clearpage}}
```

8 Default Settings

```
\tpAddToDefault{float}{%
  \tpSetProperty{intext-skip-top}{\intextsep}%% non-float sep top
  \tpSetProperty{intext-skip-bottom}{\intextsep}%% non-float sep bottom
  \tpSetProperty{float-skip-top}{\z@}%% float sep top
  \tpSetProperty{float-skip-bottom}{\z@}%% float sep bottom
  \tpSetProperty{sub-float-sep}{\tp@subflt@sep}%% space between sub-floats
  \tpSetProperty{margin-inner}{\z@}%% left margin on odd pages/right margin on even pages
  \tpSetProperty{margin-outer}{\z@}%% right margin on odd pages/left margin on even pages
  \tpSetProperty{margin-left}{\z@}%% left margin
  \tpSetProperty{margin-right}{\z@}%% right margin
  \tpSetProperty{before-float}{\parindent\z@}%% executed before content is evaluated
  \tpSetProperty{float-handler}{\tpGenericHandler}% Alias for the caption type specific content
      handler
  \tpSetProperty{subfloat-handler}{\tpSubGenericHandler}% Alias for the caption type specific content
       handler
  \tpSetProperty{float-render}{\tpGenericRender}% Alias for the caption type specific content printer
  \tpSetProperty{subfloat-render}{\tpGenericRender}% Alias for the caption type specific content
      printer for sub-floats
  \tpSetProperty{subfloat-same-height}{}% if true, the subfloat must/can be adjusted to the same
      heights
  %% captions
  \tpSetProperty{caption-face}{}% style applied to top and bottom captions
  \tpSetProperty{caption-face-top}{}%% style applied to top captions
  \tpSetProperty{caption-face-bottom}{}%% style applied to bottom captions
  \tpSetProperty{source-face}{}% Format of source, additional to caption-format
  \tpSetProperty{legend-face}{}% Format of legend, additional to caption-format
  \tpSetProperty{caption-sep-top}{\z@}%% vertical space between top caption and content
  \tpSetProperty{caption-sep-bottom}{\z@}%% vertical space between content and bottom caption
  \tpSetProperty{caption-top}{%
    \tpIfComp{Number}{{\tpUseProperty{number-face}\tpUseComp{Number}\tpUseProperty{number-sep
        }}}{}%
    \tpUseComp{Caption}%
  }%
  \tpSetProperty{caption-bottom}{%
    \tpIfComp{Legend}{{\tpUseProperty{legend-face}\tpUseComp{Legend}}}{}%
    \tpIfComp{Source}{%
      \tpIfComp{Legend}{\par\nopagebreak}{}%
      {\tpUseProperty{source-face}%
       \tpUseComp{Source}}}{}}%
  \tpPropertyLet{subcaption-face}{caption-face}% style applied to top and bottom captions
  \tpSetProperty{subcaption-face-top}{\tpUseProperty{caption-face-top}}%% style applied to top
      captions
```

```
\tpSetProperty{subcaption-face-bottom}{\tpUseProperty{caption-face-bottom}}%% style applied to
    bottom captions
\tpSetProperty{subcaption-add-sep-top}{\z@}%% additional vertical space between top caption and top
    sub-caption
\tpSetProperty{subcaption-add-sep-bottom}{\z@}%% additional vertical space between bottom sub-
    caption and bottom caption
\tpSetProperty{subcaption-sep-top}{\tpUseProperty{caption-sep-top}}%% vertical space between top
    sub-caption and content
\tpSetProperty{subcaption-sep-bottom}{\tpUseProperty{caption-sep-bottom}}%% vertical space
    between content and bottom sub-caption
\tpSetProperty{subcaption-top}{\tpUseProperty{caption-top}}% in case, sub-float captions diverge
    from main caption
\tpSetProperty{subcaption-bottom}{\tpUseProperty{caption-bottom}}% in case, sub-float captions
    diverge from main caption
\tpSetProperty{subcaption-valign-top}{top}%% vertical alignment of neighboring top-placed sub-
    captions
\tpSetProperty{subcaption-valign-bottom}{top}%% vertical alignment of neighboring bottom-placed sub-
    captions
%% Numbers
\tpSetProperty{numbering}{auto}%% automatic numbering for missing Number component
\tpSetProperty{number-sep}{\enskip}% Separator between label and caption
\tpSetProperty{number-face}{\bfseries}% Format of number, additional to caption-format
\tpSetProperty{sub-number-sep}{\,}%% when sub-captions, this is placed between the float counter and
    the sub-float counter
\tpSetProperty{sub-number-style}{alph}%% counting style of subcaption counters
\tpSetProperty{sub-number-face}{}%% format of subcaption counters
\tpSetProperty{sub-number-before}{(}% stuff that is put immediately before the sub counter
\tpSetProperty{sub-number-after}{)}% stuff that is put immediately after the sub counter
\tpSetProperty{sub-number-format}{% Format of the sub number
  \tpUseProperty{float-number}%
  \tpUseProperty{sub-number-sep}%
  \tpUseProperty{sub-number}}%
%% List-of entries
\tpSetProperty{list-of-page-sep}{\dotfill}%
\tpPropertyLet{list-of-number-face}{list-of-caption-face}%
\tpSetProperty{list-of-number-sep}{\enskip}%
\tpSetProperty{list-of-number-align}{left}%
\tpSetProperty{list-of-number-format}{%
  \bgroup
    \tpUseProperty{list-of-number-face}%
    \tpUseComp{ListofNumber}%
    \tpUseProperty{list-of-number-sep}%
  \egroup}%
\tpSetProperty{list-of-parfillskip}{-\rightskip}%
\tpSetProperty{list-of-margin-right}{\@pnumwidth \@plus 1fil}%
\tpSetProperty{list-of-margin-left}{auto}%
\tpSetProperty{list-of-indent}{auto}% list-of-float appearance
\tpSetProperty{list-of-block}{%
  \tpUseProperty{list-of-caption-face}%
  \tpIfComp{ListofNumber}
    {\tpUseComp{list-of-hang-number}}
    {\leftskip0pt}%
  \tpUseComp{ListofCaption}%
  \tpUseProperty{list-of-page-sep}\tpUseComp{ListofPage}%
}% list-of-float appearance
\tpSetProperty{list-of-before-entry}{%
  \tpGobble
  \leftskip\tpUseProperty{list-of-margin-left}\relax%
  \rightskip \tpUseProperty{list-of-margin-right}\relax%
  \parfillskip \tpUseProperty{list-of-parfillskip}\relax
  \parindent\z@
```

```
    \@afterindenttrue
    \interlinepenalty\@M
    \leavevmode
    \null\nobreak
  }% list-of-float appearance
  \tpSetProperty{list-of-after-entry}{\par}% list-of-float appearance
}
```

Container tpFigure defines the defaults for the `tpFigure` Container.

```
\tpDeclareFloat{tpFigure}{figure}{lof}{%
  \tpSetProperty{subfloat-same-height}{true}% if true, the subfloat must/can be adjusted to the same
      heights
  \tpSetProperty{float-handler}{\tpFigureHandler}%
  \tpSetProperty{subfloat-handler}{\tpSubFigureHandler}%
  \tpSetProperty{float-render}{\tpFigureRender}%
  \tpSetProperty{subfloat-render}{\tpSubFigureRender}%
}
```

Container tpTable defines the default Properties of the `tpTable` Container.

```
\tpDeclareFloat{tpTable}{table}{lot}{%
  \tpSetProperty{sub-caption-valign-top}{bottom}%
  \tpSetProperty{float-handler}{\tpTableHandler}%
  \tpSetProperty{subfloat-handler}{\tpSubTableHandler}%
  \tpSetProperty{float-render}{\tpTableRender}%
  \tpSetProperty{subfloat-render}{\tpSubTableRender}%
}
```

```
%</floats>
```

Modul 11

coco-frame.dtx

This file provides facilities to visualise crop marks and the print area.

```
%<*frame>
```

```
%%
%% module for CoCoTeX for crop marks and print area frames.
%%
%% Maintainer: p.schulz@le-tex.de
%%
%% lualatex - texlive > 2019
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-frame}
    [2024/01/29 0.4.0 coco-frame]\relax
```

1 Top-Level Interface

```
\let\tp@frame n
\define@choicekey{coco-frame.sty}{frame}[\tp@frame\nr]{none,crop,frame}{%
  \ifcase\nr\relax% none
    \let\tp@frame n
  \or% crop
    \let\tp@frame p
  \else% frame
    \let\tp@frame w
  \fi
}%
\ProcessOptionsX\relax
```

2 Cropmark printer

```
\ifx\tp@frame p\relax
  \newdimen\bleed \bleed4mm\relax
  \newdimen\tp@frame@offset \tp@frame@offset4em\relax%
  \voffset\dimexpr\tp@frame@offset-1in\relax
  \hoffset\dimexpr\tp@frame@offset-1in\relax
  \edef\l@offset{\strip@pt\dimexpr\tp@frame@offset*7200/7227\relax}
  \edef\r@offset{\strip@pt\dimexpr(\tp@frame@offset+\paperwidth)*7200/7227\relax}
  \edef\u@offset{\strip@pt\dimexpr(842bp-\tp@frame@offset-\paperheight)*7200/7227\relax}
  \edef\o@offset{\strip@pt\dimexpr(842bp-\tp@frame@offset)*7200/7227\relax}
  \edef\b@l@offset{\strip@pt\dimexpr(\tp@frame@offset-\bleed)*7200/7227\relax}
  \edef\b@r@offset{\strip@pt\dimexpr(\tp@frame@offset+\paperwidth+\bleed)*7200/7227\relax}
  \edef\b@u@offset{\strip@pt\dimexpr(842bp-\tp@frame@offset-\paperheight-\bleed)*7200/7227\relax
      }
  \edef\b@o@offset{\strip@pt\dimexpr(842bp-\tp@frame@offset+\bleed)*7200/7227\relax}
  \edef\@tempa{%
    /TrimBox [\l@offset\space\u@offset\space\r@offset\space\o@offset]
    /BleedBox[\b@l@offset\space\b@u@offset\space\b@r@offset\space\b@o@offset]
    %/CropBox[\b@l@offset\space\b@u@offset\space\b@r@offset\space\b@o@offset]
    %/MediaBox[\b@l@offset\space\b@u@offset\space\b@r@offset\space\b@o@offset]
  }
  \expandafter\pdfpageattr\expandafter{\@tempa}
\fi
```

Setting PDF boundaries

```
\ifx\tp@frame n\relax
  \RequirePackage{luatex85}
  \pdfpagewidth\paperwidth
  \pdfpageheight\paperheight
\else
  \ifx\tp@frame p\relax
    \edef\stockwidth{\the\dimexpr\paperwidth+\tp@frame@offset+\tp@frame@offset\relax}
    \edef\stockheight{\the\dimexpr\paperheight+\tp@frame@offset+\tp@frame@offset\relax}
  \fi
```

Cropmarks and page area frames both are painted via the `crop` package.

```
  \RequirePackage{crop}
  \renewcommand*\CROP@marks{%
    \CROP@setmarkcolor
    \CROP@user@b
    \vskip1in\hskip1in\relax
    \CROP@ulc\null\hfill\CROP@@@info\CROP@upedge\hfill\null\CROP@urc\hskip-1in\null
    \vfill
    \CROP@ledge\hfill\CROP@redge
    \vfill
    \hskip1in\relax
    \CROP@llc\null\hfill\CROP@loedge\hfill\null\CROP@lrc\hskip-1in\null
    \vskip-1in}%
  \ifx\tp@frame p\relax
    \def\camcross{%
      \smash{\rlap{%
          \kern-0.15\p@
          \vrule\@width0.3\p@\@height1.7mm\@depth1.7mm\relax
          \kern-0.15\p@
          \kern-1.7mm\relax
          \vrule\@width0.3\p@\@height1.7mm\@depth1.7mm\relax
          \kern-0.3\p@
          \raise1.7mm\rlap{\vrule\@width3.4mm\@height\z@\@depth0.3\p@}%
          \lower1.7mm\rlap{\vrule\@width3.4mm\@height0.3\p@\@depth\z@}%
          \hbox{\vrule\@width3.4mm\@height0.15\p@\@depth0.15\p@}%
          \kern-0.3\p@
          \vrule\@width0.3\p@\@height1.7mm\@depth1.7mm\relax}}}
    \def\cammcrossleft{%
      \llap{\camcross\vrule\@width6mm\@height0.15\p@\@depth0.15\p@\kern4mm}}
    \def\cammcrossright{%
      \rlap{\kern4mm\vrule\@width6mm\@height0.15\p@\@depth0.15\p@\camcross}}
    \def\cammcrossup{%
      \rlap{\smash{\raise10mm\hbox{\camcross}%
          \kern-0.15\p@\vrule\@width0.3\p@\@height10mm\@depth-4mm}}}%
    \def\cammcrossdown{%
```

```
      \rlap{\smash{\lower10mm\hbox{\camcross}%
        \kern-0.15\p@\vrule\@width0.3\p@\@height-4mm\@depth10mm}}}%
    \def\CROP@@ulc{\cammcrossup\cammcrossleft}
    \def\CROP@@urc{\cammcrossup\cammcrossright}
    \def\CROP@@llc{\cammcrossdown\cammcrossleft}
    \def\CROP@@lrc{\cammcrossdown\cammcrossright}
    \renewcommand*\CROP@@info{{%
      \global\advance\CROP@index\@ne
      \def\x{\discretionary{}{}{\hbox{\kern.5em---\kern.5em}}}%
      \ifx\CROP@pagecolor\@empty
      \else
        \advance\dimen@\CROP@overlap
      \fi
      \hb@xt@\z@{%
        \hss
        \lower1em\vbox to\z@{\vss
          \centering
          \hsize\dimexpr\paperwidth-20\p@\relax
          \normalfont
          \large
          \vskip5mm\relax
          \addvspace{\bleed}}%
        \hss}}%
    }%
    \crop[cam]
```

the code for the page area frame

```
  \else% w
    \@tempdima\dimexpr\textheight\relax
    \divide\@tempdima by\baselineskip
    \multiply\@tempdima by65536\relax
    \edef\cnt@baselines{\strip@pt\@tempdima}%
    \def\tp@frame@lines{%
      \@tempcnta\z@
      \loop\advance\@tempcnta\@ne
        \hsize1em\relax
        \ifodd\count\z@
          \vrule\@width1em\@height0.2\p@\@depth0.02\p@
          \llap{\smash{\the\@tempcnta\,}}%
        \fi%
        \rlap{%
          \ifodd\count\z@\else\fi
          \vrule\@width\columnwidth\@height0.00005\p@\@depth0\p@
          \if@twocolumn
            \kern\columnsep\vrule\@width\columnwidth\@height0.00005\p@\@depth0\p@
          \fi
          \ifodd\count\z@\else
            \vrule\@width1em\@height0.00005\p@\@depth0\p@%
            \llap{\smash{\the\@tempcnta\,}}%
          \fi
        }%
        \break
      \ifnum\@tempcnta<\cnt@baselines
      \repeat}
    \def\tp@margin@frame{%
      \vrule height\textheight%
      \hskip-\marginparwidth\relax
      \vbox to\textheight{\hsize\marginparwidth\relax
        \rlap{\vbox to\z@{\hrule width\marginparwidth}}%
        \null\vss
      \rlap{\vbox to\z@{\hrule width\marginparwidth}}%
    }%
    \vrule height\textheight%
  }
  \renewcommand*\CROP@@frame{%
    \vskip0in%
    \color[cmyk]{0.4,0,0,0}%
    \ifodd\count\z@\let\@themargin\oddsidemargin\else\let\@themargin\evensidemargin\fi
    \advance\@themargin1in
    \moveright\@themargin
    \vbox to\z@{\baselineskip\z@skip\lineskip\z@skip\lineskiplimit\z@
      \vskip\topmargin\vbox to\z@{\vss\hrule width\textwidth}%
      \vskip\headheight\vbox to\z@{\vss\hrule width\textwidth}%
      \vskip\headsep\vbox to\z@{\vss\hrule width\textwidth}%
      \hbox to\textwidth{%
        \ifodd\count\z@
          \rlap{\hskip\dimexpr\textwidth+\marginparsep+\marginparwidth\relax\tp@margin@frame}%
        \else
          \rlap{\hskip-\marginparsep\relax\tp@margin@frame}%
        \fi
        \llap{\vbox to\textheight{\tiny\let\@tempa\f@size\normalsize\let\f@size\@tempa\
            selectfont
          \vskip\topskip\tp@frame@lines\null\vss}}%
        \llap{\vrule height\textheight}%
        \if@twocolumn
          \hskip\columnwidth\rlap{\vrule height\textheight}%
          \hskip\columnsep\rlap{\vrule height\textheight}%
        \fi
        \hfil\vrule height\textheight
      }%
      \vbox to\z@{\vss\hrule width\textwidth}%
      \vskip\footskip\vbox to\z@{\vss\hrule width\textwidth}%
      \vss}%
    \vbox to\z@{\baselineskip\z@skip\lineskip\z@skip\lineskiplimit\z@%
      \vskip-0in\rlap{\hskip1in%
        \vbox to\z@{\vbox to\z@{\vss\hrule width\paperwidth}%
          \hbox to \paperwidth{\llap{\vrule height\paperheight}\hfil%
            \vrule height\paperheight}%
          \vbox to\z@{\vss\hrule width\paperwidth}%
          \vss}}\vss}}
  \crop[frame,noinfo]%
 \fi
\fi
```

```
%</frame>
```

Modul 12

coco-lists.dtx

This module provides handlers for lists like glossaries and descriptions.

```
%<*lists>
```

```
%%
%% module for CoCoTeX that handles lists.
%%
%% Maintainer: marcus.hottenroth@le-tex.de
%%
%% lualatex -texlive ≥2019
%%
\NeedsTeXFormat{LaTeX2e}[2018/12/01]
\ProvidesPackage{coco-lists}
   [2024/01/29 0.4.0 CoCoTeX lists module]
\RequirePackage{coco-common}
\usepackage{enumerate}
\ifx\labelitemfont\@undefined\let\labelitemfont\relax\fi
\renewcommand\labelitemi {\labelitemfont \textendash}
\setlength\leftmargini{\parindent}%
\def\@listi{%
  \leftmargin\leftmargini
  \parsep \z@
  \listparindent\parindent
  \topsep .5\baselineskip % Hier Properties nutzen!
  \itemsep\z@}
\let\@listI\@listi
\def\@listii {\leftmargin\leftmarginii
            \labelwidth\leftmarginii
            \advance\labelwidth-\labelsep
            \topsep \z@
            \parsep \z@
            \itemsep \parsep}

\def\@listiii{\leftmargin\leftmarginiii
            \labelwidth\leftmarginiii
            \advance\labelwidth-\labelsep
            \topsep \z@
            \parsep \z@
            \partopsep \z@
            \itemsep \topsep}

\def\@@enum@[#1]{%
  \@enLab{}\let\@enThe\@enQmark
  \@enloop#1\@enum@
  \ifx\@enThe\@enQmark\@warning{The counter will not be printed.%
   ^^J\space\@spaces\@spaces\@spaces The label is: \the\@enLab}\fi
  \expandafter\edef\csname label\@enumctr\endcsname{\the\@enLab}%
  \expandafter\let\csname the\@enumctr\endcsname\@enThe
  \csname c@\@enumctr\endcsname7
  \@enum@}
```

```

\def\@enum@{%
  \list{\csname label\@enumctr\endcsname}%
  {%
    \usecounter{\@enumctr}%
    \labelsep\z@
    \labelwidth\leftmargin
    \def\makelabel##1{\hb@xt@\leftmargin{##1\hss}}}}
\def\itemize{%
  \ifnum \@itemdepth >\thr@@\@toodeep\else
    \advance\@itemdepth\@ne
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
    \expandafter
    \list
      \csname\@itemitem\endcsname
      {\labelsep\z@
       \itemindent\z@
       \labelwidth\leftmargin
       \def\makelabel##1{\hb@xt@\leftmargin{##1\hss}}}%
  \fi}
\let\orig@doendpe\@doendpe
\def\endenumerate{\endlist
  \gdef\@doendpe{%
    \@endpetrue
    \everypar{{\setbox\z@\lastbox}\everypar{}\@endpefalse}%
    \global\let\@doendpe\orig@doendpe}}
\def\enditemize{\endlist
  \gdef\@doendpe{%
    \@endpetrue
    \everypar{{\setbox\z@\lastbox}\everypar{}\@endpefalse}%
    \global\let\@doendpe\orig@doendpe}}
% Counter for the description lists.
\newcount\tp@descriptionlist
% Macro for saving the maximum label widths associated with the respective list;
% 0pt as fallback value, if there is no *.aux file yet.
\global\newdimen\tp@maxLabelWidth%
\def\tp@getMaxLabelWidth{%
  \global\tp@maxLabelWidth=0pt%
}
\renewenvironment{description}[1][]{%
  \small
  % Read maximum label width for this list from the *.aux file and save as \tp@maxLabelWidth.
  \tp@getMaxLabelWidth
  \list{}%
   {\labelwidth\tp@maxLabelWidth
    \leftmargin\dimexpr\tp@maxLabelWidth+\labelsep\relax
    \topsep .5\baselineskip
    \itemsep\z@
    \partopsep\z@
    \parsep\z@
    \itemindent\z@
    \def\makelabel##1{%
      \sbox\z@{##1}%
      \ifdim\tp@maxLabelWidth<\wd\z@\relax
        \global\tp@maxLabelWidth=\wd\z@\relax
      \fi
      \hb@xt@\labelwidth{\unhbox\z@\hss}%
    }%
   }%
}{\endlist
```

```
\immediate\write\@auxout{\string\g@addto@macro\string\tp@getMaxLabelWidth{\string\ifnum\string\
    the\tp@descriptionlist=\the\tp@descriptionlist\relax\string\global\string\tp@maxLabelWidth=\
    the\tp@maxLabelWidth\string\fi}}%
\global\advance\tp@descriptionlist by \@ne
\gdef\@doendpe{%
   \@endpetrue
   \everypar{{\setbox\z@\lastbox}\everypar{}\@endpefalse}%
   \global\let\@doendpe\orig@doendpe}}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Environment declarations, CoCoTeX style.
% Supposed to eventually replace all the definitions above.
% Inheritance mechanism known from headings also applies here.
\def\tp@ifstring#1#2{%
  \edef\@tempa{#1}%
  \edef\@tempb{#2}%
  \ifx\@tempa\@tempb\relax%
}
% Convert a number to a lowercase letter.
\def\tp@numToLCLetter#1{%
  \count255=\the\lccode`a%
  \advance\count255 by -\@ne%
  \advance\count255 by #1%
  \char\count255%
}
% Convert a number to an uppercase letter.
\def\tp@numToUCLetter#1{%
  \count255=\uccode`A%
  \advance\count255 by -\@ne%
  \advance\count255 by #1%
  \char\count255%
}
\tpAddToDefault{list}{%
  \tpSetProperty{after-skip}{\z@}% Vertical space after the list.
  \tpSetProperty{before-skip}{\z@}% Vertical space before the list.
  \tpSetProperty{item-indent}{0\p@}% Vertical difference from property left-margin.
  \tpSetProperty{label-char}{} % Only applies with label-type «char» (or empty).
  \tpSetProperty{label-prefix-delimiter}{} % The character/string between the prefix (inherited from
      list one level above) and the actual item's label. Used for numbered lists.
  \tpSetProperty{label-sep}{5mm}
  \tpSetProperty{label-suffix}{}
  \tpSetProperty{label-type}{char} % Label types: char (use label-char; default), number, Alpha, alpha,
      Roman, roman.
  \tpSetProperty{label-width}{0\p@} % Label width is internally increased to width of label character.
  \tpSetProperty{left-margin}{0\p@}
}
\long\def\tpDeclareList{\@ifnextchar[{\@tpDeclareList}{\@tpDeclareList[]}}%]
\long\def\@tpDeclareList[#1]#2#3{%
  \tpNamespace{list}%
  \expandafter\def\csname tp@list@name\endcsname{#2}%
  %
  \if!#1!\else\expandafter\protect\expandafter\def\csname tp@list@#3@parent\endcsname{#1}\fi%
  \expandafter\protect\expandafter\def\csname tp@list@#2@properties\endcsname{#3}%

  % Define the macro for list with name/class #2.
  \expandafter\def\csname tpUseList#2\endcsname{%
    \if!#1!\else\edef\tp@list@parent{#1}\fi%
    \tpNamespace{list}%
    \tpCascadeProps{#2}{list} % Load the namespace defaults defined in \tpAddToDefault, the parent
        properties (if any), and the specific list properties.
  }
```

```
}
\tpDeclareContainer{tpList}{%
  \tpDeclareType{Properties}{\tp@list@default}%
}
\def\tpList{\@ifnextchar [{\tp@list}{\tp@list[]}}%]
\def\endtpList{%
  \endlist%
  \global\advance\tp@currListDepth by -\@ne%
  \expandafter\ifx\csname tpUseList\tp@list@name\endcsname\relax
    \PackageError{coco-lists.sty}{List \tp@list@name\space unknown!}{A list with name \
        tp@list@name\space is unknown. Use the \string\tpDeclareList\space macro to declare list
         types.}%
  \else
    % If the parent list ends, gather the sublists and write their label widths to the aux file.
    \ifnum\tp@currListDepth=-\@ne\relax%
      \count255=\z@
      \loop
        \immediate\write\@auxout{\string\expandafter\string\gdef\string\csname\space\string
            tp@maxLabelWidth@\the\tp@listNumber @\the\count255\endcsname{\csname
            tp@maxLabelWidth@\the\tp@listNumber @\the\count255\endcsname}}
        \advance\count255 by \@ne
      \expandafter\ifx\csname tp@maxLabelWidth@\the\tp@listNumber @\the\count255\endcsname\relax\
          else\repeat
    \fi
    \csname tpUseList\tp@list@name\endcsname%
    \vskip\tpUseProperty{after-skip}
  \fi%
  \gdef\@doendpe{%
    \@endpetrue
    \everypar{{\setbox\z@\lastbox}\everypar{}\@endpefalse}%
    \global\let\@doendpe\orig@doendpe%
  }
}
\global\newcount\tp@currListDepth \global\tp@currListDepth=-\@ne
\expandafter\gdef\csname tp@inheritablePrefix\the\tp@currListDepth\endcsname{}
\global\newcount\tp@listNumber \global\tp@listNumber=-\@ne
\def\tp@list[#1]#2{%
  % Increment the list depth and, in case the depth is zero, i.e. a completely new list and no sublist
      starts, the list number.
  \global\advance\tp@currListDepth by \@ne%
  \ifnum\tp@currListDepth = \z@
    \global\advance\tp@listNumber by \@ne%
  \fi

  % Assign a new counter for the item numbers as well as an inheritable prefix for sublists, depending on
      the list depth.
  \global\expandafter\newcount\csname tp@itemNumber\the\tp@currListDepth\endcsname%
  \expandafter\gdef\csname tp@inheritablePrefix\the\tp@currListDepth\endcsname{}%
  \gdef\tp@inheritedPrefixAbove{}%

  \newbox\tp@labelbox%
  \edef\tp@list@name{#2} % Needed for afterskips to apply.
  \tpCascadeProps{#2}{list} % Load the properties.
  % If the list has the keyword «inherit» and is enumerated, set its prefix according to the latest item
      label in the parent list.
  \tp@ifstring{#1}{inherit}%
    \tpIfPropVal{label-type}{char}{}{\gdef\tp@inheritedPrefixAbove{\csname tp@inheritablePrefix\
        \the\numexpr\the\tp@currListDepth-1\relax \endcsname}}%
  \fi

  \vskip\tpUseProperty{before-skip}
```

```

\tpIfPropVal{label-type}{char}{%
  \tpSetProperty{label-prefix-delimiter}{}%
  \tpSetProperty{label-suffix}{}%
}{%
  \tpSetProperty{label-char}{}%
}%
\tpIfPropVal{label-type}{number}{\edef\tp@convertNumber##1{##1}}{}%
\tpIfPropVal{label-type}{Alpha}{\edef\tp@convertNumber##1{\tp@numToUCLetter{##1}}}{}%
\tpIfPropVal{label-type}{alpha}{\edef\tp@convertNumber##1{\tp@numToLCLetter{##1}}}{}%
\tpIfPropVal{label-type}{Roman}{\def\tp@convertNumber##1{\uppercase\expandafter{\romannumeral
    ##1}}}{}%
\tpIfPropVal{label-type}{roman}{\def\tp@convertNumber##1{\romannumeral##1}}{}%

% Use the label prefix delimiter only if there actually is a label prefix.
\ifx\empty\tp@inheritedPrefixAbove\empty
  \tpSetProperty{label-prefix-delimiter}{}%
\fi
% Set the label width based on the potentially longest label string.
\setbox\tp@labelbox = \hbox{\tp@inheritedPrefixAbove\tpUseProperty{label-prefix-delimiter}\
    tpUseProperty{label-char}\tpUseProperty{label-suffix}}%
\ifdim\wd\tp@labelbox > \tpUseProperty{label-width}\relax%
  \tpSetProperty{label-width}{\the\wd\tp@labelbox}%
\fi%

% If the macro already exists (loaded from the aux file), ...
\expandafter\ifx\csname tp@maxLabelWidth@\the\tp@listNumber @\the\tp@currListDepth\endcsname\
    relax%
\else%
  % ...set the «label-width» property accordingly.
  \tpSetProperty{label-width}{\csname tp@maxLabelWidth@\the\tp@listNumber @\the\
      tp@currListDepth\endcsname}%
\fi

\list{%
  % Label. Uses [] in description items. Empty otherwise.%
}{%
  \labelwidth\tpUseProperty{label-width}%
  \labelsep\dimexpr\tpUseProperty{label-sep}+\tpUseProperty{item-indent}\relax%
  \leftmargin\dimexpr\tpUseProperty{left-margin}+\tpUseProperty{label-width}+\tpUseProperty{
      label-sep}\relax%
  \topsep0mm%
  \partopsep0mm%
  \itemindent\tpUseProperty{item-indent}%
  \def\makelabel##1{%
    % If the list is an enumerated one, increment the item counter and set the label accordingly.
    \tpIfPropVal{label-type}{char}{}{%
      \global\expandafter\advance\csname tp@itemNumber\the\tp@currListDepth\endcsname by \@ne%
      \tpSetProperty{label-char}{\tp@convertNumber{\the\csname tp@itemNumber\the\
          tp@currListDepth\endcsname}}%
    }
    \ifx\empty##1\empty%
      % Checking this condition is not necessary by all means, but prevents inheriting and accumulating
          characters if «inherit» option is set in the TeX document.
      \tpIfPropVal{label-type}{char}{}{%
        \global\expandafter\edef\csname tp@inheritablePrefix\the\tp@currListDepth\endcsname{\
            tp@inheritedPrefixAbove\tpUseProperty{label-prefix-delimiter}\tpUseProperty{label-
            char}}%
      }
      % Measure the actual full label width.
```

```
        \hbox to \tpUseProperty{label-width}{\tp@inheritedPrefixAbove\tpUseProperty{label-prefix-
            delimiter}\tpUseProperty{label-char}\tpUseProperty{label-suffix}\hss}%
        \setbox\tp@labelbox = \hbox{\tp@inheritedPrefixAbove\tpUseProperty{label-prefix-delimiter
            }\tpUseProperty{label-char}\tpUseProperty{label-suffix}}%
      \else
        \hbox to \tpUseProperty{label-width}{##1\hss}%
        \setbox\tp@labelbox = \hbox{##1}%
      \fi
      % If the macro for the list and the according depth is not set yet, ...
      \expandafter\ifx\csname tp@maxLabelWidth@\the\tp@listNumber @\the\tp@currListDepth\
          endcsname\relax%
        % ...define it based on the calculated full label width.
        % (Needs \xdef! Fully expands the macro definition. Otherwise, the saved macro would change its
            value with \tp@labelbox's content.)
        \expandafter\xdef\csname tp@maxLabelWidth@\the\tp@listNumber @\the\tp@currListDepth\
            endcsname{\the\wd\tp@labelbox}%
      \else%
        % If the currently defined macro holds a smaller label width than the actual label box, update the
            macro.
        \expandafter\ifdim\csname tp@maxLabelWidth@\the\tp@listNumber @\the\tp@currListDepth\
            endcsname < \the\wd\tp@labelbox\relax%
          \expandafter\xdef\csname tp@maxLabelWidth@\the\tp@listNumber @\the\tp@currListDepth\
              endcsname{\the\wd\tp@labelbox}%
        \fi
      \fi
    }%
  }%
}
\tpDeclareList{default}{}
\def\tp@list@load@props{\csname tp@list@\tp@list@name @properties\endcsname}
```

```
%</lists>
```