The latex-lab-floats package
Tagging of floats

LaTeX Project*

v0.81n 2026-04-24

Abstract

The following code implements a first draft for the tagging of float environments

1 Introduction

The code here handles the tagging of float environments.

Figures (and tables) are in LaTeX typically typeset in float environments. These are boxes which can "float" away to special float areas on the pages, e.g., to the top or the bottom of a page or to special float pages. If the rules allow it they can also be placed in the main text stream ("here"). Floats can also be collected at the end of the document. In either case the order within each type of floats (e.g., figures, tables, algorithms, etc.) is preserved.

A special type, called a H-float, (provided by the float package) is always placed in the main text stream and does not necessarily preserve the order with normal floats of the same type: It is basically a minipage with a caption.

Floats typically contain a figure (or a table, etc.) and a caption, but more complex constructions with subfigures, copyright statements, sources or additional description are possible too.

In the LaTeX source a float is normally more or less at the place of the first call-out, but when preparing a document for print the code is sometimes moved around to place floats in a more visually pleasing way.

2 Tagging

Floats (with the exception of H-floats) do not belong into the text stream, they are "consultation objects": Readers must be able to choose if and when they read the float. Floats can have captions: the PDF rules require that if a `Caption` structure element is used, it is the first or last structure in its parent structure. This poses some challenges on a good tagging.

In PDF 2.0 there is the suitable `Aside` tag which hopefully will be handled correctly regarding the reading order once processor actually support PDF 2.0. But in PDF 1.7 we rolemap it to `Note` and this doesn't lead to a good reading order. The code therefore defers the output of float: it collects the float structures and moves them to a `Sect`

*Initial implementation done by Ulrike Fischer

structure which is flushed out at the end of the document or earlier if the author requests it. (H-floats once they are handled will not be moved).

To fulfill the requirement that a `Caption` should be at the begin or end, we always move it to the begin of the structure. If a float has two captions the author has to insert a command which splits the float in two.

Subfigures and subcaptions are currently not handled, but will be implemented as simple `Part` with their own `Caption`.

Unknown float types are put into a generic structure when they are encountered in the source. To add support for a new float type the `float/new` key described below can be used.

3 Links

hyperref doesn't patch the caption commands anymore if `\DocumentMetadata` is used. The code here has to add a target for links. To improve the linking behavior, it add this target at the begin of the float (and at the begin of split if the float is splitted) and changes `\caption` so that a link to a caption label will go to this target.

4 Keys

The code adds the following keys for the `\tagpdfsetup` command:

float/new
float/flush
float/split
float/container

float/new This sets up the needed code for a new float type. The value is the ⟨*captype*⟩ which should be the same name as the name stored in `\captype`. The code defines symbolic structure names `float/`⟨*captype*⟩`s` (for the container), `float/`⟨*captype*⟩ (for a single float), `float/`⟨*captype*⟩`/caption` and `float/`⟨*captype*⟩`/label` (for the caption and its label) and assigns standard roles to them. The roles can be changed with `\AssignStructureRole`.

float/flush This will flush out all so far deferred floats (currently table and figure) into a container. The value is a sectioning level for which `\toclevel@`⟨*name*⟩ exists, e.g., `section` or `chapter`, or an integer number describing the level, like 1 for the section level. The container will then behave like a sectioning command of this level and create a `Sect` structure element containing the floats. The default value is the current sectioning level. The command should typically be used directly before a sectioning of the same or higher level. E.g. this here is wrong as following text "some text" would no longer be inside subsection A:

```
\DocumentMetadata{tagging=on}
\documentclass{article}

\begin{document}
\section{section A}
\subsection{subsection A}
\begin{figure}
\caption{hallo}
\end{figure}
\tagpdfsetup{float/flush}
some text
\section{section B}
\end{document}
```

Correct would be to put it directly before `\subsection{section B}`. The key issues a `\par` to end the current paragraph.

The floats will then be inserted as a `Sect` of this level (all `Sect` of smaller or equal level are closed). The key then creates a new container for following floats. At the end of the document all remaining floats will be flushed automatically.

float/split This can be used inside a float if there are two captions. It will only work reasonably well if the content of the float parts are in a sensible order and can be separated by this command. More complex setups with tabulars will need more thoughts.

float/defer The value `false` disables the deferring of floats in a container: the float structure is then inserted directly where it appears in the source. The key sets a global boolean. It can be used in the document to switch the collection on and off. The default value is `true`.

float/here This is the inverse of the `defer` key and so a shorter way to disable the deferring. The default value is `true`.

5 Kernel commands

\@current@float@struct This variable holds the number of the current float structure. With tagging this is the structure number, without tagging a unique counter. A float can contain more than one float structure (e.g., if there is more than one caption).

\@makecaption `\@makecaption` is defined by the classes so we overwrite it for now at begin document.

```
⟨@@=tag⟩
⟨*package⟩
```

6 Implementation

```
\ProvidesExplPackage {latex-lab-testphase-float} {\ltlabfloatdate} {\ltlabfloatversion}
  {Code related to the tagging of floats}
```

6.1 Variables

We rolemap floats to Aside, and float sections to Sect.

\g__tag_float_sect_prop
\g__tag_float_types_seq
\@current@float@struct
\g__tag_float_int

These variables will hold the structure number for the float container and the list of float types. Currently only figure and table are supported TODO: interface to declare new float types. To set the target for links we need also a unique counter. With tagging we could use the structure number, but the structure commands now are hidden inside tagging sockets so we use a dedicated counter.

```
\prop_new:N \g__tag_float_sect_prop
\seq_new:N  \g__tag_float_types_seq
\seq_gput_right:Nn \g__tag_float_types_seq {figure}
\seq_gput_right:Nn \g__tag_float_types_seq {table}
\tl_new:N\@current@float@struct
\int_new:N\g__tag_float_int
```

(*End of definition for* `\g__tag_float_sect_prop` *and others. This variable is documented on page* 4.)

\g__tag_float_sect_bool With this boolean float collection is switched on and off. Currently it is always on and set globally. TODO: think if an interface is needed. TODO: would a local variable make more sense?

```
\bool_new:N       \g__tag_float_sect_bool
\bool_gset_true:N  \g__tag_float_sect_bool
```

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

\l__tag_float_tmpa_tl
\l__tag_float_tmpb_tl

```
\tl_new:N\l__tag_float_tmpa_tl
\tl_new:N\l__tag_float_tmpb_tl
```

(*End of definition for* `\l__tag_float_tmpa_tl` *and* `\l__tag_float_tmpb_tl`.)

__tag_float_init: To be able to set unique targets for links, we need a counter outside the tagging sockets. TODO: check if this command should be public or a socket or a hook.

```
\cs_new_protected:Npn \__tag_float_init:
 {
   \int_gincr:N \g__tag_float_int
   \tl_set:Ne \@current@float@struct { \int_use:N \g__tag_float_int}
 }
```

(*End of definition for* `\__tag_float_init:`.)

6.2 Setup for a new float type

A new float type needs

- symbolic names for the float, the caption and the label
- symbolic name for the container
- and it must be recorded in the sequence.

This is all done with the following key. The value is the captype.

```
\keys_define:nn {__tag/setup}
 {
   float/new .code:n =
    {
      \NewStructureName{float/#1s}
      \AssignStructureRole{float/#1s}{Sect}
      \NewStructureName{float/#1}
      \AssignStructureRole{float/#1}{float}
      \NewStructureName{float/#1/caption}
      \AssignStructureRole{float/#1/caption}{Caption}
      \NewStructureName{float/#1/label}
      \AssignStructureRole{float/#1/label}{Lbl}
      \seq_gput_right:Nn \g__tag_float_types_seq {#1}
    }
 }
```

6.3 Moving float structures

Currently it is for all float types or none. Probably we will need some more options here to select some float types.

`float/defer`
`float/here`

```
\keys_define:nn{__tag/setup}
  {
    float/defer .bool_gset:N = \g__tag_float_sect_bool
    ,float/here .bool_gset_inverse:N = \g__tag_float_sect_bool
  }
```

(*End of definition for* `float/defer` *and* `float/here`. *These functions are documented on page* **??**.)

__tag_float_container_new:n This creates a new container and then stashes it.

```
\cs_new_protected:Npn \__tag_float_container_new:nN #1 #2 % #1 captype #2 tl-var to return nu
  {
    \tag_struct_begin:n{tag=\UseStructureName{float/#1s},stash}
    \tl_set:Ne #2 {\int_use:N\c@g__tag_struct_abs_int}
    \prop_gput:Nne\g__tag_float_sect_prop {#1-struct}{\int_use:N\c@g__tag_struct_abs_int}
    \tag_struct_end:
  }
\cs_generate_variant:Nn \__tag_float_container_new:nN {eN}
```

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

__tag_float_init_collect: This initializes a container structure for every float type. It can be used more than once in a document, this allows to have e.g. chapter wise containers.

```
\cs_new_protected:Npn\__tag_float_init_collect:
  {
    \bool_if:NT\g__tag_float_sect_bool
      {
        \seq_map_inline:Nn\g__tag_float_types_seq
          {
            \__tag_float_container_new:nN {##1}\l__tag_tmpa_tl
          }
      }
  }
```

(*End of definition for* `\__tag_float_init_collect:`.)

__tag_float_stop_sect: This pushes out the floats. For every type is checks if there is actually a float of this type and then writes out the container structure.

```
\cs_new_protected:Npn \__tag_float_stop_sect:
  {
    \seq_map_inline:Nn\g__tag_float_types_seq
      {
        \prop_get:NnNT\g__tag_float_sect_prop{##1-used}\l__tag_float_tmpa_tl
          {
            \prop_get:NnNF\g__tag_float_sect_prop{##1-struct}\l__tag_float_tmpb_tl
             {
               \__tag_float_container_new:eN {##1}\l__tag_float_tmpb_tl
             }
            \exp_args:Ne
            \tag_struct_use_num:n{\l__tag_float_tmpb_tl}
            \prop_gremove:Nn \g__tag_float_sect_prop{##1-used}
          }
      }
  }
```

(*End of definition for* `\__tag_float_stop_sect:`.)

flush/float This is a key for `\tagpdfsetup` to flush out the collected floats. The value allows to set to which level the create Sect belongs. So `section` will close all previous Sect until the section level and create a new section structure.

```
\keys_define:nn { __tag / setup }
 {
   float/flush .code:n =
```

```
    {
      \par
      \cs_if_exist:cTF{toclevel@#1}
       { \UseTaggingSocket{sec/end}{\int_eval:n{\use:c{toclevel@#1}}}}
       { \UseTaggingSocket{sec/end}{\int_eval:n{#1}}}
      \__tag_float_stop_sect:
      \__tag_float_init_collect:
    },
  float/flush .default:n = \@currentseclevel
}
```

(*End of definition for* `flush/float`. *This function is documented on page* ??.)

`flush-floats (deprecated)` This is the same key for the deprecated command `\tagtool` to flush out the collected floats. Kept for compatibility for now but will be removed at some time. The value allows to set to which level the create Sect contains. So `section` will close all previous Sect until the section level and create a new section.

```
\keys_define:nn { tag / tool}
 {
   flush-floats .meta:nn = {__tag/setup}{float/flush=#1},
   flush-floats .default:n = \@currentseclevel
 }
```

(*End of definition for* `flush-floats (deprecated)`. *This function is documented on page* ??.)

We need at least one pair

```
\AddToHook{begindocument/end}[latex-lab/float]
  {\__tag_float_init_collect:}
\AddToHook{tagpdf/finish/before}[latex-lab/float]
  {\par\__tag_sec_end:n{-10}\__tag_float_stop_sect:}
\DeclareHookRule{tagpdf/finish/before}{latex-lab/float}{before}{tagpdf}
```

6.4 Splitting floats

`float/split` TODO: check if the target affect spacing!!

```
\keys_define:nn { __tag / setup}
  {
   float/split .code:n =
    {
      \UseTaggingSocket{float/end}
      \__tag_float_init:
      \UseTaggingSocket{float/begin}
      \MakeLinkTarget*{floatstructure.\@current@float@struct}
    }
 }
```

(*End of definition for* `float/split`. *This function is documented on page* 3.)

`split-float (deprecated)` This is the same key for the deprecated command `\tagtool` to split a float. Kept for compatibility for now but will be removed at some time.

```
\keys_define:nn { tag / tool}
  {
   split-float .meta:nn = {__tag/setup}{float/split}
  }
```

(*End of definition for* `split-float (deprecated)`. *This function is documented on page* ??.)

6.5 Tagging sockets

The tagging sockets are declared in lttagging. Currently we have the tagging sockets `float/hmode/begin`, `float/hmode/end`, `float/begin` and `float/end`. All sockets have no argument.

pport/float/hmode/begin) (*plug*) This plug should be used if a float is called in hmode. It then closes the MC-chunks and starts the structure.

```
\NewTaggingSocketPlug{float/hmode/begin}{kernel}
  {
    \__tag_float_stop_par:
  }
\AssignTaggingSocketPlug{float/hmode/begin}{kernel}
```

support/float/hmode/end) (*plug*) This plug should be used if a float is called in hmode; at the end of the float it then restarts the MC.

```
\NewTaggingSocketPlug{float/hmode/end}{kernel}
  {
    \__tag_float_start_par:
  }
\AssignTaggingSocketPlug{float/hmode/end}{kernel}
```

(tagsupport/float/begin) (*plug*)

```
\NewTaggingSocketPlug{float/begin}{kernel}
  {
    \__tag_float_begin:
  }
\AssignTaggingSocketPlug{float/begin}{kernel}
```

t (tagsupport/float/end) (*plug*)

```
\NewTaggingSocketPlug{float/end}{kernel}
  {
    \__tag_float_end:
  }
\AssignTaggingSocketPlug{float/end}{kernel}
```

__tag_float_stop_par:
__tag_float_start_par:

if a float is in a par, we need commands to stop and restart the P-mc

```
\cs_new_protected:Npn \__tag_float_stop_par:
  {
    \tag_mc_end:
    \bool_if:NF \g__tag_float_sect_bool
     {
      \tag_struct_end:
     }
  }
\cs_new_protected:Npn \__tag_float_start_par:
  {
    \bool_if:NF \g__tag_float_sect_bool
     {
      \tag_struct_begin:n{tag=\UseStructureName{para/textblock}}%
     }
   \tag_mc_begin:n{tag=P}
  }
```

(*End of definition for* `\__tag_float_stop_par:` *and* `\__tag_float_start_par:`.)

These commands are the main commands to start and end the float tagging.

```
\cs_new_protected:Npn \__tag_float_begin:
 {%
```

We test if the float structure should be included directly or move to a dedicated section.

```
  \seq_if_in:NoTF\g__tag_float_types_seq{\@captype}
    {
     \bool_if:NTF\g__tag_float_sect_bool
      {
        \prop_get:NoNF \g__tag_float_sect_prop{\@captype-struct}\l__tag_float_tmpa_tl
         {
           \__tag_float_container_new:eN {\@captype}\l__tag_float_tmpa_tl
         }
       \tag_struct_begin:e
          {
            tag=\UseStructureName{float/\@captype},
            parent=\l__tag_float_tmpa_tl
          }%
       \prop_gput:Nee \g__tag_float_sect_prop {\@captype-used}{true}
      }
      {
        \tag_struct_begin:n{tag=\UseStructureName{float/\@captype}}
      }
    }
    {
      \tag_struct_begin:n{tag=\UseStructureName{float/generic}}
    }
     \tl_set:Ne\@current@float@struct{\tag_get:n{struct_num}}%
     \typeout{Float structure: \@current@float@struct}
 }

\cs_new_protected:Npn\__tag_float_end:{\tag_struct_end:} %end Aside

```

6.6 Patching

This patches the main command `\@xfloat`. There is a : in the code, so we disable expl3 syntax

```
\ExplSyntaxOff
\def\@xfloat #1[#2]{%
  \@nodocument
  \def \@captype {#1}%
   \def \@fps {#2}%
   \@onelevel@sanitize \@fps
   \def \reserved@b {!}%
   \ifx \reserved@b \@fps
     \@fpsadddefault
   \else
     \ifx \@fps \@empty
       \@fpsadddefault
     \fi
```

```
  \fi
  \ifhmode
    \@bsphack
```

If the float is in hmode we have to interrupt the P

```
    \UseTaggingSocket{float/hmode/begin}%
    \@floatpenalty -\@Mii
  \else
    \@floatpenalty-\@Miii
  \fi
\ifinner
   \@parmoderr\@floatpenalty\z@
\else
  \@next\@currbox\@freelist
    {%
     \@tempcnta \sixt@@n
     \expandafter \@tfor \expandafter \reserved@a
       \expandafter :\expandafter =\@fps
       \do
        {%
         \if \reserved@a h%
           \ifodd \@tempcnta
           \else
             \advance \@tempcnta \@ne
           \fi
         \else\if \reserved@a t%
           \@setfpsbit \tw@
         \else\if \reserved@a b%
           \@setfpsbit 4%
         \else\if \reserved@a p%
           \@setfpsbit 8%
         \else\if \reserved@a !%
           \ifnum \@tempcnta>15
             \advance\@tempcnta -\sixt@@n\relax
           \fi
         \else
           \@latex@error{Unknown float option `\reserved@a'}%
           {Option `\reserved@a' ignored and `p' used.}%
           \@setfpsbit 8%
         \fi\fi\fi\fi\fi
         }%
     \@tempcntb \csname ftype@\@captype \endcsname
     \multiply \@tempcntb \@xxxii
     \advance \@tempcnta \@tempcntb
     \global \count\@currbox \@tempcnta
     }%
  \@fltovf
\fi
```

This starts the structure for the float.

```
\csname __tag_float_init:\endcsname
\UseTaggingSocket{float/begin}%
\global \setbox\@currbox
  \color@vbox
```

```
      \normalcolor
      \vbox \bgroup
        \hsize\columnwidth
        \@parboxrestore
        \@floatboxreset
```

We add a target for links. TODO: check that it doesn't affect spacing!!

```
        \MakeLinkTarget*{\@captype.struct.\@current@float@struct}%
}%
```

The end code of the float ...

```
\def\end@float{%
  \@endfloatbox
  \UseTaggingSocket{float/end}%
  \ifnum\@floatpenalty <\z@
    \@largefloatcheck
    \@cons\@currlist\@currbox
    \ifnum\@floatpenalty <-\@Mii
      \penalty -\@Miv
      \@tempdima\prevdepth
      \vbox{}%
      \prevdepth\@tempdima
      \penalty\@floatpenalty
    \else
      \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
      \UseTaggingSocket{float/hmode/end}%
    \fi
  \fi
}
```

and similar for double floats:

```
\def\end@dblfloat{%
  \if@twocolumn
    \@endfloatbox
    \UseTaggingSocket{float/end}%
    \ifnum\@floatpenalty <\z@
      \@largefloatcheck
      \global\dp\@currbox1sp %
      \@cons\@currlist\@currbox
      \ifnum\@floatpenalty <-\@Mii
        \penalty -\@Miv
        \@tempdima\prevdepth
        \vbox{}%
        \prevdepth\@tempdima
        \penalty\@floatpenalty
      \else
        \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
        \UseTaggingSocket{float/hmode/end}%
      \fi
    \fi
  \else
    \end@float
  \fi
}%
\ExplSyntaxOn
```

6.7 Handling captions

6.7.1 (Tagging) sockets

These sockets are in lttagging: `caption/begin` (1 argument), `caption/end`, `caption/label/begin`, `caption/label/end`.

`caption/label` (*socket*) This socket is a lightweight start for some interface to format the label or add a font command. The argument is the label text. The default plug `kernel` adds a colon and a space. TODO: revisit after checking float and caption packages to identify which sockets and hooks are needed.

```
\NewSocket{caption/label}{1}
```

`kernel (caption/label)` (*plug*) The standard label formatting from the kernel.

```
\NewSocketPlug{caption/label}{kernel}
 {
   #1:~
 }
\AssignSocketPlug{caption/label}{kernel}
```

`default` (*plug*) The caption begin socket takes an argument: the structure number of the parent float. If the argument is empty, the current structure is used. TODO: a tagpdf key that moves a structure to the begin of the parent. The caption is moved to the first position with the firstkid option.

```
\NewTaggingSocketPlug{caption/begin}{default}
  {
     \tl_if_empty:eTF {#1}
      {
        \tag_struct_begin:n{tag=\UseStructureName{float/\@captype/caption},firstkid}
      }
      {
        \tag_struct_begin:n{tag=\UseStructureName{float/\@captype/caption},parent=#1,firstkid
      }
     \bool_set_true:N \l__tag_para_flattened_bool
  }
\AssignTaggingSocketPlug{caption/begin}{default}
```

`default` (*plug*)

```
\NewTaggingSocketPlug{caption/end}{default}
  {
    \tag_struct_end:
  }
\AssignTaggingSocketPlug{caption/end}{default}
```

`ort/caption/label/begin)` (*plug*)

```
\NewTaggingSocketPlug{caption/label/begin}{default}
  {
```

suppress para tagging at the begin.

```
     \tagpdfparaOff
     \tag_struct_begin:n{tag=\UseStructureName{float/\@captype/label}}
     \tag_mc_begin:n{}
  }
\AssignTaggingSocketPlug{caption/label/begin}{default}
```

pport/caption/label/end) (*plug*)

```
\NewTaggingSocketPlug{caption/label/end}{default}
  {
    \tag_mc_end:
    \tag_struct_end:
    \tagpdfparaOn
  }
\AssignTaggingSocketPlug{caption/label/end}{default}
```

6.7.2 Redefinitions

With hyperref that means that the `\refstepcounter` now can affect spacing so we change that to the kernel refstepcounter:

```
\def\caption{%
   \ifx\@captype\@undefined
     \@latex@error{\noexpand\caption\c_space_tl outside~float}\@ehd
     \expandafter\@gobble
   \else
```

if a caption is used outside a float no target has been set and `\@current@float@struct` is empty

```
      \tl_if_empty:NTF\@current@float@struct
       {
         \refstepcounter\@captype
       }
       {
         \@kernel@refstepcounter\@captype
```

we need to reset the target for `\addcontentsline`. We use `\@captype` to support autoref.

```
         \xdef\@currentHref{\@captype.struct.\@current@float@struct}%
       }
      \expandafter\@firstofone
   \fi
   {\@dblarg{\@caption\@captype}}%
}
```

`\@makecaption` `\@makecaption` is defined by the classes so we overwrite it for now at begin document.

```
\NewHookWithArguments{cmd/@makecaption/before}{2}
\AddToHook{begindocument}
  {
    \long\def\@makecaption#1#2{%
      \UseHookWithArguments{cmd/@makecaption/before}{2}{#1}{#2}%
      \vskip\abovecaptionskip
```

We don't want tagging when storing the caption for the singleline check.

```
      \SuspendTagging{\@makecaption}
      \sbox\@tempboxa{\UseSocket{caption/label}{#1}#2}%
      \ResumeTagging{\@makecaption}
```

We pass `\@current@float@struct` as parent structure number. If that is empty the socket will use the parent structure and hope ...

```
      \UseTaggingSocket{caption/begin}{\@current@float@struct}
      \ifdim \wd\@tempboxa >\hsize
```

```
      \UseTaggingSocket{caption/label/begin}
      \UseSocket{caption/label}{#1}
      \UseTaggingSocket{caption/label/end}
      \UseTaggingSocket{para/begin}
          #2
      \par
      \else
```

we don't reuse the box as it doesn't contain tagging, but set the text explicitly.

```
          \global \@minipagefalse
        \hb@xt@\hsize{\hfil
         \UseTaggingSocket{caption/label/begin}
         \UseSocket{caption/label}{#1}
         \UseTaggingSocket{caption/label/end}
         \UseTaggingSocket{para/begin}
          #2
         \UseTaggingSocket{para/end}
         \hfil}%
       \fi
      \UseTaggingSocket{caption/end}
      \vskip\belowcaptionskip}
}
```

(*End of definition for* `\@makecaption`. *This function is documented on page* 4.)

⟨/package⟩