The shipout routine: hooks and interfaces*

Frank Mittelbach, LaTeX Project Team

October 30, 2026

Contents

1 Introduction **2**
1.1 Overloading the \shipout primitive . . . 2
1.2 Provided hooks . . . 3
1.3 Legacy LaTeX commands . . . 5
1.4 Special commands for use inside the hooks . . . 5
1.5 Provided LuaTeX callbacks . . . 6
1.6 Information counters . . . 6
1.7 Debugging shipout code . . . 7

2 Emulating commands from other packages **7**
2.1 Emulating atbegshi . . . 7
2.2 Emulating everyshi . . . 8
2.3 Emulating atenddvi . . . 8
2.4 Emulating everypage . . . 9

3 The Implementation **9**
3.1 Debugging . . . 11
3.2 Handling the end of job hook . . . 23

4 Legacy LaTeX 2ε interfaces **26**

5 Internal commands needed elsewhere **26**

6 Package emulation for compatibility **28**
6.1 Package atenddvi emulation . . . 28
6.2 Package atbegshi emulation . . . 29
6.3 Package everyshi emulation . . . 30

Index **31**

*This file has version v1.0r dated 2026-10-13, © LaTeX Project.

1 Introduction

The code provides an interface to the `\shipout` primitive of TEX which is called when a finished pages is finally "shipped out" to the target output file, e.g., the `.dvi` or `.pdf` file. A good portion of the code is based on ideas by Heiko Oberdiek implemented in his packages atbegshi and atenddvi even though the interfaces are somewhat different.[1]

1.1 Overloading the `\shipout` primitive

`\shipout`

With this implementation TEX's shipout primitive is no longer available for direct use. Instead `\shipout` is running some (complicated) code that picks up the box to be shipped out regardless of how that is done, i.e., as a constructed `\vbox` or `\hbox` or as a box register.

It then stores it in a named box register. This box can then be manipulated through a set of hooks after which it is shipped out for real.

Each shipout that actually happens (i.e., where the material is not discarded for one or the other reason) is recorded and the total number is available in a readonly variable and in a LATEX counter.

`\RawShipout`

This command implements a simplified shipout that bypasses the foreground and background hooks, e.g., only `shipout/firstpage` and `shipout/lastpage` are executed and the total shipout counters are incremented.

The command doesn't use `\ShipoutBox` but its own private box register so that it can be used inside of shipout hooks to do some additional shipouts while already in the output routine with the current page being stored in `\ShipoutBox`. It does have access to `\ShipoutBox` if it is used in `shipout/before` (or `shipout/after`) and can use its content.

It is safe to use it in `shipout/before` or `shipout/after` but not necessarily in the other `shipout/...` hooks as they are intended for special processing.

`\ShipoutBox`
`\l_shipout_box`

This box register is called `\ShipoutBox` (alternatively available via the L3 name `\l_shipout_box`).

This box is a "local" box and assignments to it should be done only locally. Global assignments (as done by some packages with older code where this is box is known as 255) may work but they are conceptually wrong and may result in errors under certain circumstances.

During the execution of `shipout/before` this box contains the accumulated material for the page, but not yet any material added by other shipout hooks. During execution of `shipout/after`, i.e., after the shipout has happened, the box also contains any background or foreground material.

Material from the hooks `shipout/firstpage` or `shipout/lastpage` is not included (but only used during the actual shipout) to facilitate reuse of the box data (e.g., `shipout/firstpage` material should never be added to a later page of the output).

[1] Heiko's interfaces are emulated by the kernel code, if a document requests his packages, so older documents will continue to work.

\l_shipout_box_ht_dim
\l_shipout_box_dp_dim
\l_shipout_box_wd_dim
\l_shipout_box_ht_plus_dp_dim

The shipout box dimensions are available in the L3 registers `\l_shipout_box_ht_dim`, etc. (there are no LaTeX 2_ε names).[2] These variables can be used inside the hook code for `shipout/before`, `shipout/foreground` and `shipout/background` if needed.

1.2 Provided hooks

shipout/before
shipout/after
shipout/foreground
shipout/background
shipout/firstpage
shipout/lastpage

The code for `\shipout` offers a number of hooks into which packages (or the user) can add code to support different use cases. These are:

`shipout/before` This hook is executed after the finished page has been stored in `\ShipoutBox` / `\l_shipout_box`). It can be used to alter that box content or to discard it completely (see `\DiscardShipoutBox` below).

You can use `\RawShipout` inside this hook for special use cases. It can make use of `\ShipoutBox` (which doesn't yet include the background and foreground material).

Note: It is not possible (or say advisable) to try and use this hook to typeset material with the intention to return it to main vertical list, it will go wrong and give unexpected results in many cases—for starters it will appear after the current page not before or it will vanish or the vertical spacing will be wrong!

`shipout/background` This hook adds a picture environment into the background of the page with the `(0,0)` coordinate in the top-left corner using a `\unitlength` of `1pt`.

It should therefore only receive `\put` commands or other commands suitable in a `picture` environment and the vertical coordinate values would normally be negative.

Technically this is implemented by adding a zero-sized `\hbox` as the very first item into the `\ShipoutBox` containing that `picture` environment. Thus the rest of the box content will overprint what ever is typeset by that hook.

`shipout/foreground` This hook adds a picture environment into the foreground of the page with the `(0,0)` coordinate in the top-left corner using a `\unitlength` of `1pt`.

Technically this is implemented by adding a zero-sized `\hbox` as the very last item into the `\ShipoutBox` and raising it up so that it still has its `(0,0)` point in the top-left corner. But being placed after the main box content it will be typeset later and thus overprints it (i.e., is in the foreground).

`shipout` This hook is executed after foreground and/or background material has been added, i.e., just in front of the actual shipout operation. Its purpose is to allow manipulation of the finalized box (stored in `\ShipoutBox`) with the extra material also in place (which is not yet the case in `shipout/before`).

It cannot be used to cancel the shipout operation via `\DiscardShipoutBox` (that has to happen in `shipout/before`, if desired!

[2] Might need changing, but HO's version as strings is not really helpful I think).

shipout/firstpage The material from this hook is executed only once at the very beginning of the first output page that is shipped out (i.e., not discarded at the last minute). It should only contain `\special` or similar commands needed to direct post processors handling the `.dvi` or `.pdf` output.[3]

This hook is added to the very first page regardless of how it is shipped out (i.e., with `\shipout` or `\RawShipout`).

shipout/lastpage The corresponding hook to add `\special`s at the very end of the output file. It is only executed on the very last page of the output file — or rather on the page that LaTeX believes is the last one. Again it is executed regardless of the shipout method.

It may not be possible for LaTeX to correctly determine which page is the last one without several reruns. If this happens and the hook is non-empty then LaTeX will add an extra page to place the material and also request a rerun to get the correct placement sorted out.

shipout/after This hook is executed after a shipout has happened. If the shipout box is discarded this hook is not looked at.

You can use `\RawShipout` inside this hook for special use cases and the main `\ShipoutBox` is still available at this point (but in contrast to `shipout/before` it now includes the background and foreground material).

Note: Just like `shipout/before` this hook is not meant to be used for adding typeset material back to the main vertical list—it might vanish or the vertical spacing will be wrong!

As mentioned above the hook `shipout/before` is executed first and can manipulate the prepared shipout box stored in `\ShipoutBox` or set things up for use in `\write` during the actual shipout. It is even run if there was a `\DiscardShipoutBox` request in the document.

The other hooks (except `shipout` and `shipout/after`) are added inside hboxes to the box being shipped out in the following order:

`shipout/firstpage`	only on the first page
`shipout/background`	
⟨*boxed content of* `\ShipoutBox`⟩	
`shipout/foreground`	
`shipout/lastpage`	only on the last page

If any of the hooks has no code then the corresponding box is added at that point.

Once the (page) box has got the above extra content it can again be manipulated using the `shipout` hook and then is shipped out for real.

Once the (page) box has been shipped out the `shipout/after` hook is called (while you are still inside the output routine). It is not called if the shipout box was discarded.

In a document that doesn't produce pages, e.g., only makes `\typeout`s, none of the hooks are ever executed (as there is no `\shipout`) not even the `shipout/lastpage` hook.

If `\RawShipout` is used instead of `\shipout` then only the hooks `shipout/firstpage` and `shipout/lastpage` are executed (on the first or last page), all others are bypassed.

[3] In LaTeX 2_ε that was already existing, but implemented using a box register with the name `\@begindvibox`.

1.3 Legacy LaTeX commands

\AtBeginDvi
\AtEndDvi

\AtBeginDvi {⟨*code*⟩}

\AtBeginDvi is the existing LaTeX 2_ε interface to fill the shipout/firstpage hook. This is not really a good name as it is not just supporting .dvi but also .pdf output or .xdv.

\AtEndDvi is the counterpart that was not available in the kernel but only through the package atenddvi. It fills the shipout/lastpage hook.

Neither interface can set a code label but uses the current default label.

As these two wrappers have been available for a long time we continue offering them (but not enhancing them, e.g., by providing support for code labels).

For new code we strongly suggest using the high-level hook management commands directly instead of "randomly-named" wrappers. This will lead to code that is easier to understand and to maintain and it also allows you to set code labels if needed.

For this reason we do not provide any other "new" wrapper commands for the above hooks in the kernel, but only keep the existing ones for backward compatibility.

1.4 Special commands for use inside the hooks

\DiscardShipoutBox
\shipout_discard:

\AddToHookNext {shipout/before} {...\DiscardShipoutBox...}

The \DiscardShipoutBox declaration (L3 name \shipout_discard:) requests that on the next shipout the page box is thrown away instead of being shipped to the .dvi or .pdf file.

Typical applications wouldn't do this unconditionally, but have some processing logic that decides to use or not to use the page.

Note that if this declaration is used directly in the document it may depend on the placement to which page it applies, given that LaTeX output routine is called in an asynchronous manner! Thus normally one would use this only as part of the shipout/before code.

> *Todo: Once we have a new mark mechanism available we can improve on that and make sure that the declaration applies to the page that contains it — not done (yet)*

\DiscardShipoutBox cannot be used in any of the shipout/... hooks other than shipout/before.

In the atbegshi package there are a number of additional commands for use inside the shipout/before hook. They should normally not be needed any more as one can instead simply add code to the hooks shipout/before, shipout, shipout/background or shipout/foreground.[4] If atbegshi gets loaded then those commands become available as public functions with their original names as given below.

[4] If that assumption turns out to be wrong it would be trivial to change them to public functions (right now they are private).

1.5 Provided LuaTeX callbacks

`pre_shipout_filter`

Under LuaTeX the `pre_shipout_filter` Lua callback is provided which gets called directly after the `shipout` hook, immediately before the shipout primitive gets invoked. The signature is

```
function(<node> head)
  return true
end
```

The `head` is the list node corresponding to the box to be shipped out. The return value should always be `true`.

1.6 Information counters

`\ReadonlyShipoutCounter` `\ifnum\ReadonlyShipoutCounter=...`
`\g_shipout_readonly_int` `\int_use:N \g_shipout_readonly_int % expl3 usage`

This integer holds the number of pages shipped out up to now (including the one to be shipped out when inside the output routine). More precisely, it is incremented only after it is clear that a page will be shipped out, i.e., after the `shipout/before` hook (because that might discard the page)! In contrast `shipout/after` sees the incremented value.

Just like with the `page` counter its value is only accurate within the output routine. In the body of the document it may be off by one as the output routine is called asynchronously!

Also important: it *must not* be set, only read. There are no provisions to prevent that restriction, but if you manipulate it, chaos will be the result. To emphasize this fact it is not provided as a LaTeX counter but as a TeX counter (i.e., a command), so `\Alph{\ReadonlyShipoutCounter}` etc, would not work.

`totalpages` `\arabic{totalpages}`
`\g_shipout_totalpages_int` `\int_use:N \g_shipout_totalpage_int % expl3 usage`

In contrast to `\ReadonlyShipoutCounter`, the `totalpages` counter is a LaTeX counter and incremented for each shipout attempt including those pages that are discarded for one or the other reason. Again `shipout/before` sees the counter before it is incremented. In contrast `shipout/after` sees the incremented value.

Furthermore, while it is incremented for each page, its value is never used by LaTeX. It can therefore be freely reset or changed by user code, for example, to additionally count a number of pages that are not build by LaTeX but are added in a later part of the process, e.g., cover pages or picture pages made externally.

Important: as this is a page-related counter its value is only reliable inside the output routine!

`\PreviousTotalPages` `\PreviousTotalPages`

Command that expands to the number of total pages from the previous run. If there was no previous run or if used in the preamble it expands to `0`. Note that this is a command and not a counter, so in order to display the number in, say, Roman numerals you have to assign its value to a counter and then use `\Roman` on that counter.

1.7 Debugging shipout code

\DebugShipoutsOn
\DebugShipoutsOff
\shipout_debug_on:
\shipout_debug_off:

\DebugShipoutsOn

Turn the debugging of shipout code on or off. This displays changes made to the shipout data structures.

Todo: This needs some rationalizing and may not stay this way.

2 Emulating commands from other packages

The packages in this section are no longer necessary, but as they are used by other packages, they are emulated when they are explicitly loaded with `\usepackage` or `\RequirePackage`.

Please note that the emulation only happens if the package is explicitly requested, i.e., the commands documented below are not automatically available in the LaTeX kernel! If you write a new package we suggest to use the appropriate kernel hooks directly instead of loading the emulation.

2.1 Emulating atbegshi

\AtBeginShipoutUpperLeft
\AtBeginShipoutUpperLeftForeground

\AddToHook {shipout/before} {...\AtBeginShipoutUpperLeft{⟨*code*⟩}...}

This adds a `picture` environment into the background of the shipout box expecting ⟨*code*⟩ to contain `picture` commands. The same effect can be obtained by simply using kernel features as follows:

```
\AddToHook{shipout/background}{⟨code⟩}
```

There is one technical difference: if `\AtBeginShipoutUpperLeft` is used several times each invocation is put into its own box inside the shipout box whereas all ⟨*code*⟩ going into `shipout/background` ends up all in the same box in the order it is added or sorted based on the rules for the hook chunks.

`\AtBeginShipoutUpperLeftForeground` is similar with the difference that the `picture` environment is placed in the foreground. To model it with the kernel functions use the hook `shipout/foreground` instead.

\AtBeginShipoutAddToBox
\AtBeginShipoutAddToBoxForeground

\AddToHook {shipout/before} {...\AtBeginShipoutAddToBox{⟨*code*⟩}...}

These work like `\AtBeginShipoutUpperLeft` and `\AtBeginShipoutUpperLeftForeground` with the difference that ⟨*code*⟩ is directly placed into an `\hbox` inside the shipout box and not surrounded by a `picture` environment.

To emulate them using `shipout/background` or `shipout/foreground` you may have to wrap ⟨*code*⟩ into a `\put` statement but if the code is not doing any typesetting just adding it to the hook should be sufficient.

\AtBeginShipoutBox

This is the name of the shipout box as atbegshi knows it.

\AtBeginShipoutOriginalShipout

This is the name of the \shipout primitive as atbegshi knows it. This bypasses all the mechanisms set up by the LaTeX kernel and there are various scenarios in which it can therefore fail. It should only be used to run existing legacy atbegshi code but not in newly developed applications.

The kernel alternative is \RawShipout which is integrated with the LaTeX mechanisms and updates, for example, the \ReadonlyShipoutCounter counter. Please use \RawShipout for new code if you want to bypass the before, foreground and background hooks.

\AtBeginShipoutInit

By default atbegshi delayed its action until \begin{document}. This command was forcing it in an earlier place. With the new concept it does nothing.

\AtBeginShipout
\AtBeginShipoutNext

\AtBeginShipout{⟨*code*⟩} ≡ \AddToHook{shipout/before}{⟨*code*⟩}
\AtBeginShipoutNext{⟨*code*⟩} ≡ \AddToHookNext{shipout/before}{⟨*code*⟩}

This is equivalent to filling the shipout/before hook by either using \AddToHook or \AddToHookNext, respectively.

\AtBeginShipoutFirst
\AtBeginShipoutDiscard

The atbegshi names for \AtBeginDvi and \DiscardShipoutBox.

2.2 Emulating everyshi

The everyshi package is providing commands to run arbitrary code just before the shipout starts. One point of difference: in the new shipout hooks the page is available as \ShipoutBox for inspection of change, one should not manipulate box 255 directly inside shipout/before, so old code doing this would change to use \ShipoutBox instead of 255 or \@cclv.

\EveryShipout

\EveryShipout{⟨*code*⟩} ≡ \AddToHook{shipout/before}{⟨*code*⟩}

\AtNextShipout

\AtNextShipout{⟨*code*⟩} ≡ \AddToHookNext{shipout/before}{⟨*code*⟩}

However, most use cases for everyshi are attempts to put some picture or text into the background or foreground of the page and that can be done today simply by using the shipout/background and shipout/foreground hooks without any need to coding.

2.3 Emulating atenddvi

The atenddvi package implemented only a single command: \AtEndDvi and that is now available out of the box so the emulation makes the package a no-op.

2.4 Emulating everypage

This package patched the original `\@begindvi` hook and replaced it with its own version. Its functionality is now covered by the hooks offered by the kernel so that there is no need for such patching any longer.

`\AddEverypageHook`

`\AddEverypageHook{`⟨*code*⟩`}` ≡
    `\AddToHook{shipout/background}{\put(1in,-1in){`⟨*code*⟩`}}`

`\AddEverypageHook` is adding something into the background of every page at a position of 1in to the right and 1in down from the top left corner of the page. By using the kernel hook directly you can put your material directly to the right place, i.e., use other coordinates in the `\put` statement above.

`\AddThispageHook`

`\AddThispageHook{`⟨*code*⟩`}` ≡
    `\AddToHookNext{shipout/background}{\put(1in,-1in){`⟨*code*⟩`}}`

The `\AddThispageHook` wrapper is similar but uses `\AddToHookNext`.

3 The Implementation

```
1 ⟨@@=shipout⟩
```

At the moment most the whole module rolls back in one go, but if we make any modifications in later releases this will then need splitting.

```
2 ⟨*2ekernel | latexrelease⟩
3 ⟨latexrelease⟩\IncludeInRelease{2026/11/01}%
4 ⟨latexrelease⟩                    {\__shipout_picture_overlay:n}{Link handling in overlay pictures}
5 \ExplSyntaxOn
```

`\__shipout_picture_overlay:n`

Put the argument into a `picture` environment that doesn't take up any size and uses `1pt` for `\unitlength`.

> *Todo: Could perhaps be generalized as it might be useful elsewhere. For now it is not.*

We want to avoid that links in the main galley which break over a page spills over into the picture (gh/2210). The needed code is different for lualatex and other engines and also must be inserted in different places: in vmode for lualatex and in hmode else. The code only works if the PDF management is loaded.

```
6 \cs_new:Npn \__shipout_picture_overlay_linkon:    {}
7 \cs_new:Npn \__shipout_picture_overlay_linkoff:   {}
8 \cs_new:Npn \__shipout_picture_overlay_linkstate: {}
```

We need to wait until we know if the PDF management has been loaded.

```
9  \AddToHookNext{class/before}
10 {
11   \IfPDFManagementActiveT
12    {
13      \sys_if_engine_luatex:TF
14        {
15          \cs_set:Npn \__shipout_picture_overlay_linkstate: {\pdfextension linkstate-3}
16        }
17        {
```

```
          \cs_set_protected:Npn \__shipout_picture_overlay_linkon:  {\pdfannot_link_on:}
          \cs_set_protected:Npn \__shipout_picture_overlay_linkoff: {\pdfannot_link_off:}
        }
     }
  }
\cs_new:Npn \__shipout_picture_overlay:n #1 {
```

The very first time this is executed we have to initialize (and freeze) the origins.

```
    \__shipout_init_page_origins:
    \kern -\c__shipout_horigin_tl \scan_stop:
    \vbox_to_zero:n {
      \kern -\c__shipout_vorigin_tl \scan_stop:
      \unitlength 1pt \scan_stop:
      \__shipout_picture_overlay_linkstate:
```

This mimics a simple zero-sized picture environment. The `\hss` is need in case there is horizontal material (without using `\put` with a positive width.

```
      \hbox_set_to_wd:Nnn \l__shipout_tmp_box \c_zero_dim
        {
          \__shipout_picture_overlay_linkoff:
          \ignorespaces #1 \hss
          \__shipout_picture_overlay_linkon:
        }
      \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
      \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
      \box_use:N \l__shipout_tmp_box
      \tex_vss:D
    }
}
```

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

```
\ExplSyntaxOff
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\__shipout_picture_overlay:n}{Link handling in overlay pictures}
⟨latexrelease⟩\ExplSyntaxOn
⟨latexrelease⟩\cs_new:Npn \__shipout_picture_overlay:n #1 {
⟨latexrelease⟩    \__shipout_init_page_origins:
⟨latexrelease⟩    \kern -\c__shipout_horigin_tl \scan_stop:
⟨latexrelease⟩    \vbox_to_zero:n {
⟨latexrelease⟩      \kern -\c__shipout_vorigin_tl \scan_stop:
⟨latexrelease⟩      \unitlength 1pt \scan_stop:
⟨latexrelease⟩      \hbox_set_to_wd:Nnn \l__shipout_tmp_box \c_zero_dim
⟨latexrelease⟩                          { \ignorespaces #1 \hss }
⟨latexrelease⟩      \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
⟨latexrelease⟩      \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
⟨latexrelease⟩      \box_use:N \l__shipout_tmp_box
⟨latexrelease⟩      \tex_vss:D
⟨latexrelease⟩    }
⟨latexrelease⟩}
⟨latexrelease⟩\ExplSyntaxOff
⟨latexrelease⟩\EndIncludeInRelease
```

```
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                    {\shipout}{Hook management (shipout)}%
\ExplSyntaxOn
```

3.1 Debugging

\g__shipout_debug_bool

Holds the current debugging state.

```
\bool_new:N \g__shipout_debug_bool
```

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

\shipout_debug_on:
\shipout_debug_off:
__shipout_debug:n
__shipout_debug_gset:

Turns debugging on and off by redefining `\__shipout_debug:n`.

```
\cs_new_eq:NN \__shipout_debug:n  \use_none:n
\cs_new_protected:Npn \shipout_debug_on:
  {
    \bool_gset_true:N \g__shipout_debug_bool
    \__shipout_debug_gset:
  }
\cs_new_protected:Npn \shipout_debug_off:
  {
    \bool_gset_false:N \g__shipout_debug_bool
    \__shipout_debug_gset:
  }
\cs_new_protected:Npn \__shipout_debug_gset:
  {
    \cs_gset_protected:Npe \__shipout_debug:n ##1
      { \bool_if:NT \g__shipout_debug_bool {##1} }
  }
```

(*End of definition for* `\shipout_debug_on:` *and others. These functions are documented on page 7.*)

\ShipoutBox
\l_shipout_box

The box filled with the page to be shipped out (both L3 and LaTeX 2_ε name).

```
\box_new:N  \l_shipout_box
\cs_set_eq:NN \ShipoutBox \l_shipout_box
```

(*End of definition for* `\ShipoutBox` *and* `\l_shipout_box`. *These functions are documented on page 2.*)

\l__shipout_raw_box

The `\RawShipout` gets its own box but it is internal as there is no hook manipulation for it.

```
\box_new:N  \l__shipout_raw_box
```

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

__shipout_finalize_box:

For LuaTeX invoke the `pre_shipout_filter` callback.

```
\sys_if_engine_luatex:TF
  {
    \newprotectedluacmd \__shipout_finalize_box:
    \exp_args:Ne \everyjob {
      \exp_not:V \everyjob
      \exp_not:N \lua_now:n {
        luatexbase.create_callback('pre_shipout_filter', 'list')
        local~call, getbox, setbox = luatexbase.call_callback, tex.getbox, tex.setbox~
        lua.get_functions_table()[\the \allocationnumber] = function()
          local~head = getbox(\the \l_shipout_box)
```

```
          local~result = call('pre_shipout_filter', head)
          if~not (result == head) then~
            setbox(\the \l_shipout_box, result~or~nil)
          end~
        end
      }
    }
  } {
    \cs_set_eq:NN \__shipout_finalize_box: \scan_stop:
  }
```

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

`\__shipout_execute:` This is going to the be the code run by `\shipout`. The code follows closely the ideas from atbegshi, so not documenting that here for now.

```
\cs_set_protected:Npn \__shipout_execute: {
  \tl_set:Ne \l__shipout_group_level_tl
     { \int_value:w \tex_currentgrouplevel:D }
  \tex_afterassignment:D \__shipout_execute_test_level:
  \tex_setbox:D \l_shipout_box
}
```

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

`\shipout` Overloading the `\shipout` primitive:

```
\cs_gset_eq:NN \shipout \__shipout_execute:
```

(*End of definition for* `\shipout`. *This function is documented on page* 2.)

`\l__shipout_group_level_tl` Helper token list to record the group level at which `\__shipout_execute:` is encountered.

```
\tl_new:N \l__shipout_group_level_tl
```

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

`\__shipout_execute_test_level:` If the group level has changed then we are still constructing `\l_shipout_box` and to continue we need to wait until the current group has finished, hence the `\tex_aftergroup:D`.

```
\cs_new:Npn \__shipout_execute_test_level: {
  \int_compare:nNnT
     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
     \tex_aftergroup:D \__shipout_execute_cont:
}
```

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

`\__shipout_execute_cont:` This does the actual shipout running several hooks as part of it. The code for them is passed as argument `#2` to `#4` to `\__shipout_execute_main_cont:Nnnn`; the first argument is the box to be shipped out.

```
\cs_new:Npn \__shipout_execute_cont: {
  \__shipout_execute_main_cont:Nnnn
     \l_shipout_box
     { \hook_use:n {shipout/before} }
     { \hook_if_empty:nF {shipout/foreground}
         { \__shipout_add_foreground_picture:n
             { \hook_use:n {shipout/foreground} } }
```

If the user hook for the background (`shipout/background`) has no code, there might still code in the kernel hook so we need to test for this too. We only test for the `\@kernel@before@shipout@background` though. If the `\@kernel@after@shipout@background` needs executing even if the user hook is empty then we can add another test (or the kernel could put something into the before hook).

```
      \bool_lazy_and:nnF
        { \hook_if_empty_p:n {shipout/background} }
        { \tl_if_empty_p:N \@kernel@before@shipout@background }
        { \__shipout_add_background_picture:n
          { \@kernel@before@shipout@background
            \hook_use:n {shipout/background}
            \@kernel@after@shipout@background }
        }
    }
    { \hook_use:n {shipout/after} }
}
```

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

`\__shipout_execute_main_cont:Nnnn`

When we have reached this point the shipout box has been processed and is available in `\l_shipout_box` and ready for real ship out (unless it gets discarded during the process).

The three arguments hold hook code that is executed just before the actual shipout (`#1`), within the shipout adding background and foreground material (`#2`) and after the shipout has happened (`#3`). These are passed as arguments because the same code without those hooks is also used when doing a "raw" shipout implemented by `\RawShipout`. The only hook that is always executed is that for the very last page, i.e., `shipout/lastpage`.

First we quickly check if it is void (can't happen in the standard LaTeX output routine but `\shipout` might be called from a package that has some special processing logic). If it is void we aren't shipping anything out and processing ends.[5]

```
\cs_new:Npn \__shipout_execute_main_cont:Nnnn #1#2#3#4 {
  \box_if_empty:NTF #1
    { \@latex@warning@no@line{ Ignoring~ void~ shipout~ box } }
    {
```

Otherwise we assume that we will ship something and prepare for final adjustments (in particular setting the state of `\protect` while we are running the hook code). We also save the current `\protect` state to restore it later.

```
%       \bool_gset_false:N \g__shipout_discard_bool  % setting this would disable
                                                     % \DiscardShipoutBox on doc-level
      \cs_set_eq:NN \__shipout_saved_protect: \protect
      \set@typeset@protect
```

We also store the current shipout box dimension in registers, so that they can be used in the hook code.[6]

```
      \__shipout_get_box_size:N #1
```

Then we execute the `shipout/before` hook (or nothing in case of `\RawShipout`).

```
      #2
```

[5] In that case we don't reset the deadcycles, that would be up to the OR processing logic to do.

[6] This is not really necessary as the code could access them via `\box_ht:N`, etc., but it is perhaps convenient.

In `\g_shipout_totalpages_int` we count all shipout attempts so we increment that counter already here (the other one is incremented later when we know for sure that we do a `\shipout`.

We increment it after running the above hook so that the values for `\g_shipout_totalpages_int` and `\g_shipout_readonly_int` are in sync while the hook is executed (in the case that totalpages isn't manually altered or through discarding pages that is).

```
      \int_gincr:N \g_shipout_totalpages_int
```

The above hook might contain code that requests the page to be discarded so we now test for it.

```
      \bool_if:NTF \g__shipout_discard_bool
        { \@latex@info@no@line{Completed~ page~ discarded}
          \bool_gset_false:N \g__shipout_discard_bool
```

As we are discarding the page box and not shipping anything out, we need to do some house cleaning and reset TEX's deadcycles so that it doesn't complain about too many calls to the OR without any shipout.

```
          \tex_deadcycles:D \c_zero_int
```

Todo: In atbegshi the box was dropped but is that actually needed? Or the resetting of `\protect` *to its kernel value?*

```
%           \group_begin:
%             \box_set_eq_drop:NN #1 #1
%           \group_end:
%           \cs_set_eq:NN \protect \exp_not:N
        }
```

Even if there was no explicit request to discard the box it is possible that the code for the hook `shipout/before` has voided the box (by mistake or deliberately). We therefore test once more but this time make it a warning, because the best practice way is to use the request mechanism.

```
        { \box_if_empty:NTF #1
            { \@latex@warning@no@line { Ignoring~ void~ shipout~ box.
                 \MessageBreak The~ shipout~ box~ was~ voided~ by~ hook~ code }
            }
```

Finally, if the box is still non-empty we are nearly ready to ship it out. First we increment the total page counter so that we can later test if we have reached the final page according to our available information.[7]

```
            {
              \int_gincr:N \g_shipout_readonly_int
              \__shipout_debug:n {
                \typeout{Absolute~ page~ =~ \int_use:N \g_shipout_readonly_int
                         \space (target:~ \@abspage@last)}
              }
```

Then we store the box sizes again (as they may have changed) and then look at the hooks `shipout/foreground` and `shipout/background`. If either or both are non-empty we add a `picture` environment to the box (in the foreground and/or in the background) and execute the hook code inside that environment.

```
              \__shipout_get_box_size:N #1
```

[7] Doing that earlier would be wrong because we might end up with the last page counted but discard and then we have no place to add the final objects into the output file.

The first hook we run is the `shipout/firstpage` hook. This is only done once, then the `\__shipout_run_firstpage_hook:` command redefines itself to do nothing. If the hook contains `\special`s for integration at the top of the page they will be temporarily stored in a safe place and added later with `\__shipout_add_firstpage_specials:`.

```
\__shipout_run_firstpage_hook:
```

Run the hooks for background and foreground or, if this is called by `\RawShipout`, copy the box `\l__shipout_raw_box` to `\l_shipout_box` so that firstpage and lastpage material gets added if necessary (that is always done to `\l_shipout_box`.

```
#3
```

We then run `\__shipout_add_firstpage_specials:` that adds the content of the hook `shipout/firstpage` to the start of the first page (if non-empty). It is then redefined to do nothing on later pages.

```
\__shipout_add_firstpage_specials:
```

Then we check if we have to add the `shipout/lastpage` hook or the corresponding kernel hook because we have reached the last page. This test will be false for all but one (and hopefully the correct) page.

```
\int_compare:nNnT \@abspage@last = \g_shipout_readonly_int
  { \bool_lazy_and:nnF
      { \hook_if_empty_p:n {shipout/lastpage} }
      { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
      { \__shipout_debug:n { \typeout{Executing~ lastpage~ hook~
            on~ page~ \int_use:N \g_shipout_readonly_int } }
        \__shipout_add_foreground_box:n
          { \UseHook{shipout/lastpage}
            \@kernel@after@shipout@lastpage }
```

We record that we have handled the `shipout/lastpage` hook but only if we really did.

```
        \bool_gset_true:N \g__shipout_lastpage_handled_bool
      }
  }

\hook_use:n {shipout}
\__shipout_finalize_box:
```

Finally we run the actual TEX primitive for shipout. As that will expand delayed `\write` statements inside the page in which protected commands should not expand we first change `\protect` to the appropriate definition for that case.

```
\cs_set_eq:NN \protect \exp_not:N
\tex_shipout:D \box_use:N \l_shipout_box
```

The `\l_shipout_box` may contain the firstpage material if this was the very first shipout. That makes it unsuitable for reuse in another shipout, so as a safety measure the next command resets `\l_shipout_box` to its earlier state if that is necessary. On later pages this is then a no-op.

```
\__shipout_drop_firstpage_specials:
```

The `shipout/after` hook (if in `#4`) needs to run with `\protect`ed commands again being executed, because that hook will "typeset" material added at the top of the next page.

```
\set@typeset@protect
#4
}
}
```

Restore the value of \protect in case \shipout is called outside of the output routine (where it is automatically restored because of the implicit group).

```
      \cs_set_eq:NN \protect \__shipout_saved_protect:
    }
}
```

(*End of definition for* __shipout_execute_main_cont:Nnnn.)

__shipout_execute_raw:
__shipout_execute_test_level_raw:

This implements the "raw" shipout which bypasses the before, foreground, background and after hooks. It follows the same pattern than __shipout_execute_raw: except that it finally calls __shipout_execute_main_cont:Nnnn with three empty arguments. instead of the hook code.

```
\cs_set_protected:Npn \__shipout_execute_raw: {
  \tl_set:Ne \l__shipout_group_level_tl
     { \int_value:w \tex_currentgrouplevel:D }
  \tex_afterassignment:D \__shipout_execute_test_level_raw:
  \tex_setbox:D \l__shipout_raw_box
}
\cs_new:Npn \__shipout_execute_test_level_raw: {
  \int_compare:nNnT
     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
     \tex_aftergroup:D \__shipout_execute_nohooks_cont:
}
```

Well, not totally empty arguments, we add some debugging if we are actually doing a shipout.

```
\cs_new:Npn \__shipout_execute_nohooks_cont: {
  \__shipout_execute_main_cont:Nnnn \l__shipout_raw_box
     {} { \__shipout_debug:n{ \typeout{Doing~ raw~ shipout~ ...} }
          \box_set_eq:NN \l_shipout_box \l__shipout_raw_box } {}
}
```

(*End of definition for* __shipout_execute_raw: *and* __shipout_execute_test_level_raw:.)

\RawShipout

The interface name for raw shipout.

```
\cs_gset_eq:NN \RawShipout \__shipout_execute_raw:
```

(*End of definition for* \RawShipout. *This function is documented on page* 2.)

__shipout_saved_protect:

Remember the current \protect state.

```
\cs_new_eq:NN  \__shipout_saved_protect: \protect
```

(*End of definition for* __shipout_saved_protect:.)

shipout/before
shipout
shipout/after
shipout/foreground
shipout/background
shipout/firstpage
shipout/lastpage

Declaring all hooks for the shipout code.

```
\hook_new:n{shipout/before}
\hook_new:n{shipout}
\hook_new:n{shipout/after}
\hook_new:n{shipout/foreground}
\hook_new:n{shipout/background}
\hook_new:n{shipout/firstpage}
\hook_new:n{shipout/lastpage}
```

(*End of definition for* shipout/before *and others. These functions are documented on page* 3.)

\@kernel@after@shipout@lastpage
\@kernel@before@shipout@background
\@kernel@after@shipout@background

And here are the internal kernel hooks going before or after the public ones where needed.

```
\let\@kernel@after@shipout@lastpage\@empty
\let\@kernel@before@shipout@background\@empty
\let\@kernel@after@shipout@background\@empty
```

(*End of definition for* `\@kernel@after@shipout@lastpage`, `\@kernel@before@shipout@background`, *and* `\@kernel@after@shipout@background`.)

__shipout_run_firstpage_hook:

There are three commands to handle the `shipout/firstpage` hook: `\__shipout_run_firstpage_hook:`, `\__shipout_add_firstpage_specials:` and `\__shipout_drop_firstpage_specials:`.

That hook is supposed to contain `\special`s and similar material to be placed at the very beginning of the output page and so it needs careful placing to avoid that anything else gets in front of it. And this means we have to wait with this until other hooks such as `shipout/background` have added their bits. It is also important that such `\special`s show up only on the very first page, so if this page gets saved before `\shipout` for later reuse, we have to make sure that they aren't in the saved version.

In addition the hook may also contain code to be executed "first", e.g., visible from code in `shipout/background` and this conflicts with adding the `\special`s late.

Therefore the processing is split into different parts: `\__shipout_run_firstpage_hook:` is done early and checks if there is any material in the hook.

```
\cs_new:Npn \__shipout_run_firstpage_hook: {
  \hook_if_empty:nTF {shipout/firstpage}
```

If not then we define the other two commands to do nothing.

```
      {
        \cs_gset_eq:NN \__shipout_add_firstpage_specials:  \prg_do_nothing:
        \cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:
      }
```

If there is material we execute inside a box, which means any `\special` will end up in that box and any other code is executed and can have side effects (as long as they are global).

```
      {
        \hbox_set:Nn \l__shipout_firstpage_box { \UseHook{shipout/firstpage} }
      }
```

Once we are here we change the definition to do nothing next time and we also change the command used to implement `\AtBeginDvi` to become a warning and not add further material to a hook that is never used again.

```
  \cs_gset_eq:NN \__shipout_run_firstpage_hook: \prg_do_nothing:
  \cs_gset:Npn \__shipout_add_firstpage_material:Nn ##1 ##2 {
    \@latex@warning{ First~ page~ is~ already~ shipped~ out,~ ignoring
                     \MessageBreak \string##1 }
  }
}
```

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

__shipout_add_firstpage_specials:
__shipout_drop_firstpage_specials:

The `\__shipout_add_firstpage_specials:` then adds the `\special`s stored in `\l__shipout_firstpage_box` to the page to be shipped out when the time is ready. Note that if there was no material in the `shipout/firstpage` hook then this command gets redefined to do nothing. But for most documents there is something, e.g., some PostScript

header, or some metadata declaration, etc. so by default we assume there is something to do.

```
\cs_new:Npn \__shipout_add_firstpage_specials: {
```

First we make a copy of the `\l_shipout_box` that we can restore it later on.

```
  \box_set_eq:NN \l__shipout_raw_box \l_shipout_box
```

Adding something to the beginning means adding it to the background as that layer is done first in the output.

```
  \__shipout_add_background_box:n { \hbox_unpack_drop:N \l__shipout_firstpage_box }
```

After the actual shipout `\__shipout_drop_firstpage_specials:` is run to restore the earlier content of `\l_shipout_box` and then redefines itself again to do nothing.

As a final act we change the definition to do nothing next time.

```
  \cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:
}
```

The `\__shipout_drop_firstpage_specials:` is run after the shipout has occurred but before the `shipout/afterpage` hook is executed. That is the point where we have to restore the `\ShipoutBox` to its state without the `shipout/firstpage` material.

```
\cs_new:Npn \__shipout_drop_firstpage_specials: {
    \box_set_eq:NN \l_shipout_box \l__shipout_raw_box
```

If there was no such material then `\__shipout_run_firstpage_hook:` will have changed the definition to a no-op already. Otherwise this is what we do here.

```
    \cs_gset_eq:NN \__shipout_drop_firstpage_specials:  \prg_do_nothing:
  }
```

(*End of definition for* `\__shipout_add_firstpage_specials:` *and* `\__shipout_drop_firstpage_specials:`.)

`\l__shipout_firstpage_box` The box to hold any firstpage `\specials`.

```
\box_new:N \l__shipout_firstpage_box
```

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

`\g__shipout_lastpage_handled_bool` A boolean to signal if we have already handled the `shipout/lastpage` hook.

```
\bool_new:N \g__shipout_lastpage_handled_bool
```

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

`\__shipout_add_firstpage_material:Nn` This command adds material to the `shipout/firstpage` hook. It is used in `\AtBeginDvi`, etc. The first argument is the command through which is it called. Initially this is ignored but once we are passed the first page it can be used to generate a warning message mentioning the right user command.

```
\cs_new:Npn \__shipout_add_firstpage_material:Nn #1#2 {
   \AddToHook{shipout/firstpage}{#2}
}
```

(*End of definition for* `\__shipout_add_firstpage_material:Nn`.)

`\__shipout_get_box_size:N` Store the box dimensions in dimen registers.

Todo: This could/should perhaps be generalized to set height depth and width given an arbitrary box.

```
\cs_new:Npn \__shipout_get_box_size:N #1 {
  \dim_set:Nn \l_shipout_box_ht_dim { \box_ht:N #1 }
  \dim_set:Nn \l_shipout_box_dp_dim { \box_dp:N #1 }
  \dim_set:Nn \l_shipout_box_wd_dim { \box_wd:N #1 }
  \dim_set:Nn \l_shipout_box_ht_plus_dp_dim
      { \l_shipout_box_ht_dim + \l_shipout_box_dp_dim }
}
```

(*End of definition for* `\__shipout_get_box_size:N`.)

`\l_shipout_box_ht_dim` `\l_shipout_box_dp_dim` `\l_shipout_box_wd_dim` `\l_shipout_box_ht_plus_dp_dim`

And here are the variables set by `\__shipout_get_box_size:N`.

```
\dim_new:N \l_shipout_box_ht_dim
\dim_new:N \l_shipout_box_dp_dim
\dim_new:N \l_shipout_box_wd_dim
\dim_new:N \l_shipout_box_ht_plus_dp_dim
```

(*End of definition for* `\l_shipout_box_ht_dim` *and others. These functions are documented on page* 3.)

`\g__shipout_discard_bool`

Indicate whether or not the current page box should be discarded

```
\bool_new:N \g__shipout_discard_bool
```

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

`\l__shipout_tmp_box` `\l__shipout_saved_badness_tl`

We need a box for the background and foreground material and a token register to remember badness settings as we disable them during the buildup below.

```
\box_new:N \l__shipout_tmp_box
\tl_new:N  \l__shipout_saved_badness_tl
```

(*End of definition for* `\l__shipout_tmp_box` *and* `\l__shipout_saved_badness_tl`.)

`\__shipout_add_background_box:n`

In standard LaTeX the shipout box is always a `\vbox` but here we are allow for other usage as well, in case some package has its own output routine.

```
\cs_new:Npn \__shipout_add_background_box:n #1
{ \__shipout_get_box_size:N \l_shipout_box
```

But we start testing for a vertical box as that should be the normal case.

```
  \box_if_vertical:NTF \l_shipout_box
      {
```

Save current values of `\vfuzz` and `\vbadness` then change them to allow box manipulations without warnings.

```
        \tl_set:Ne \l__shipout_saved_badness_tl
           { \vfuzz=\the\vfuzz\relax
             \vbadness=\the\vbadness\relax }
        \vfuzz=\c_max_dim
        \vbadness=\c_max_int
```

Then we reconstruct `\l_shipout_box` ...

```
        \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
            {
```

... the material in `#1` is placed into a horizontal box with zero dimensions.

```
              \hbox_set:Nn \l__shipout_tmp_box
                  { \l__shipout_saved_badness_tl #1 }
              \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
              \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
              \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
```

The we typeset that box followed by whatever was in `\l_shipout_box` before (unpacked).

```
                \skip_zero:N \baselineskip
                \skip_zero:N \lineskip
                \skip_zero:N \lineskiplimit
                \box_use:N \l__shipout_tmp_box
                \vbox_unpack:N \l_shipout_box
```

The `\kern` ensures that the box has no depth which is afterwards explicitly corrected.

```
                \kern \c_zero_dim
              }
        \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
        \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
```

Todo: The whole boxing maneuver looks a bit like overkill to me, but for the moment I leave.

```
        \l__shipout_saved_badness_tl
      }
      {
```

A horizontal box is handled in a similar way. The last case would be a void box in which case we do nothing hence the missing `F` branch.

```
        \box_if_horizontal:NT \l_shipout_box
            {
              \tl_set:Ne \l__shipout_saved_badness_tl
                 { \hfuzz=\the\hfuzz\relax
                   \hbadness=\the\hbadness\relax }
              \hfuzz=\c_max_dim
              \hbadness=\c_max_int
              \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
                   {
                     \hbox_set:Nn \l__shipout_tmp_box
                          { \l__shipout_saved_badness_tl #1 }
                     \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
                     \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
                     \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
                     \box_move_up:nn
                         \l_shipout_box_ht_dim
                         { \box_use:N \l__shipout_tmp_box }
                     \hbox_unpack:N \l_shipout_box
                   }
              \l__shipout_saved_badness_tl
            }
      }
}
```

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

`\__shipout_add_foreground_box:n` Foreground boxes are done in the same way, only the order and placement of boxes has to be done differently.

```
\cs_new:Npn \__shipout_add_foreground_box:n #1
{
  \box_if_vertical:NTF \l_shipout_box
    {
      \tl_set:Ne \l__shipout_saved_badness_tl
```

```
          { \vfuzz=\the\vfuzz\relax
            \vbadness=\the\vbadness\relax }
        \vfuzz=\c_max_dim
        \vbadness=\c_max_int
        \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
          {
            \hbox_set:Nn \l__shipout_tmp_box
                 { \l__shipout_saved_badness_tl #1 }
            \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
            \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
            \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
            \skip_zero:N \baselineskip
            \skip_zero:N \lineskip
            \skip_zero:N \lineskiplimit
            \vbox_unpack:N \l_shipout_box
            \kern -\l_shipout_box_ht_plus_dp_dim
            \box_use:N \l__shipout_tmp_box
            \kern  \l_shipout_box_ht_plus_dp_dim
          }
        \l__shipout_saved_badness_tl
        \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
        \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
      }
      {
        \box_if_horizontal:NT \l_shipout_box
          {
            \tl_set:Ne \l__shipout_saved_badness_tl
              { \hfuzz=\the\hfuzz\relax
                \hbadness=\the\hbadness\relax }
            \hfuzz=\c_max_dim
            \hbadness=\c_max_int
            \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
                 {
                   \hbox_unpack:N \l_shipout_box
                   \kern -\box_wd:N \l_shipout_box
                   \hbox_set:Nn \l__shipout_tmp_box
                       { \l__shipout_saved_badness_tl #1 }
                   \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
                   \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
                   \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
                   \box_move_up:nn { \box_ht:N \l_shipout_box }
                                   { \box_use:N \l__shipout_tmp_box }
                   \kern \box_wd:N \l_shipout_box
                 }%
                 \l__shipout_saved_badness_tl
          }
      }
}
```

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

`\__shipout_init_page_origins:` Two constants holding the offset of the top-left with respect to the media box.
`\c__shipout_horigin_tl` Setting the constants this way is courtesy of Bruno.
`\c__shipout_vorigin_tl`

We delay setting the constants to the last possible place as there might be updates in the preamble or even in the `begindocument` hook that affects their setup.

```
\cs_new:Npn \__shipout_init_page_origins: {
  \tl_const:Ne \c__shipout_horigin_tl
     {
       \cs_if_exist_use:NTF \pdfvariable { horigin }
          { \cs_if_exist_use:NF \pdfhorigin { 1in } }
     }
  \tl_const:Ne \c__shipout_vorigin_tl
     {
       \cs_if_exist_use:NTF \pdfvariable { vorigin }
          { \cs_if_exist_use:NF \pdfvorigin { 1in } }
     }
```

After the constants have been set there is no need to execute this command again, in fact it would raise an error, so we redefine it to do nothing.

```
  \cs_gset_eq:NN \__shipout_init_page_origins: \prg_do_nothing:
}
```

(*End of definition for* `\__shipout_init_page_origins:`, `\c__shipout_horigin_tl`, *and* `\c__shipout_vorigin_tl`.)

`\__shipout_add_background_picture:n` Put a `picture` env in the background of the shipout box with its reference point in the top-left corner.

```
\cs_new:Npn \__shipout_add_background_picture:n #1 {
   \__shipout_add_background_box:n { \__shipout_picture_overlay:n {#1} }
}
```

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

`\__shipout_add_foreground_picture:n` Put a `picture` env in the foreground of the shipout box with its reference point in the top-left corner.

```
\cs_new:Npn \__shipout_add_foreground_picture:n #1 {
   \__shipout_add_foreground_box:n { \__shipout_picture_overlay:n {#1} }
}
```

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

`\shipout_discard:` Request that the next shipout box should be discarded. At the moment this is just setting a boolean, but we may want to augment this behavior that the position of the call is taken into account (in case LaTeX looks ahead and is not using the position for on the next page).

```
\cs_new_protected:Npn \shipout_discard: {
  \bool_gset_true:N \g__shipout_discard_bool
}
```

(*End of definition for* `\shipout_discard:`. *This function is documented on page* 5.)

3.2 Handling the end of job hook

At the moment this is partly solved by using the existing hooks. But rather than putting the code into these hooks it should be moved to the right place directly as we shouldn't prefill hooks with material unless it needs to interact with other code.

\g_shipout_readonly_int
\ReadonlyShipoutCounter

We count every shipout activity that makes a page (but not those that are discarded) in order to know how many pages got produced.

```
\int_new:N \g_shipout_readonly_int
```

For LaTeX 2_ε it is available as a command (i.e., a TeX counter only.

```
\cs_new_eq:NN \ReadonlyShipoutCounter  \g_shipout_readonly_int
```

(*End of definition for* `\g_shipout_readonly_int` *and* `\ReadonlyShipoutCounter`*. These functions are documented on page 6.*)

\g_shipout_totalpages_int
\c@totalpages

We count every shipout attempt (even those that are discarded) in this counter. It is not used in the code but may get used in user code.

```
\int_new:N \g_shipout_totalpages_int
```

For LaTeX 2_ε this is offered as a LaTeX counter so can be easily typeset inside the output routine to display things like "`\thepage/\thetotalpages`", etc.

```
\cs_new_eq:NN \c@totalpages \g_shipout_totalpages_int
\cs_new:Npn \thetotalpages { \arabic{totalpages} }
```

(*End of definition for* `\g_shipout_totalpages_int` *and* `\c@totalpages`*. These functions are documented on page 6.*)

\@abspage@last

In `\@abspage@last` record the number of pages from the last run. This is written to the `.aux` and this way made available to the next run. In case there is no `.aux` file or the statement is missing from it we initialize it with the largest possible number in TeX. We use this as the default because then we are inserting the `shipout/lastpage` on the last page (or after the last page) but not on page 1 for a multipage document.

```
\xdef\@abspage@last{\number\maxdimen}
```

(*End of definition for* `\@abspage@last`*.*)

\enddocument

Instead of using the hooks `enddocument` and `enddocument/afterlastpage` we add this code to private kernel hooks to be 100% sure when it is executed and to avoid cluttering the hooks with data that is always there.

Inside `\enddocument` there is a `\clearpage`. Just before that we execute this code here. There is a good chance that we are on the last page. Therefore, if we don't know the value from the last run, we assume that the current page is the right one. So we set `\@abspage@last` and as a result the next shipout will run the `shipout/lastpage` code. Of course, if there are floats that still need a placement this guess will be wrong but then rerunning the document will give us the correct value next time around.

\@kernel@after@enddocument

```
\g@addto@macro \@kernel@after@enddocument {
  \int_compare:nNnT \@abspage@last = \maxdimen
    {
```

We use LaTeX 2_ε coding as `\@abspage@last` is not an L3 name.

```
      \xdef\@abspage@last{ \int_eval:n {\g_shipout_readonly_int + 1} }
    }
}
```

\@kernel@before@enddocument@afterlastpage

After the last page has been shipped out, we force further `\write` calls to be always `\immediate` because we aren't going to ship out any more pages. This goes before the `enddocument/afterlastpage` hook because that may contain such `\write` commands.

```
\g@addto@macro \@kernel@before@enddocument@afterlastpage {
  \__shipout_force_immediate_writes:
  % This is also the point where \__tag_lastpage_label: could be executed
  % instead of going into enddocument/afterlastpage
}
```

\@kernel@after@enddocument@afterlastpage

Once the `\clearpage` has done its work inside `\enddocument` we know for sure how many pages this document has, so we record that in the `.aux` file for the next run.

```
\g@addto@macro \@kernel@after@enddocument@afterlastpage {
```

There is one special case: If no output is produced then there is no point in a) recording the number as 0 will never match the page number of a real page and b) adding an extra page to ran the `shipout/lastpage` is pointless as well (as it would remain forever). So we test for this and run the code only if there have been pages.

```
  \int_compare:nNnF \g_shipout_readonly_int = 0
    {
```

This ends up in the `.aux` so we use LaTeX 2_ε names here.

Todo: This needs an interface for `\nofiles` *in expl3, doesn't at the moment!*

```
      \if@filesw
         \iow_now:Ne \@auxout {
           \gdef\string\@abspage@last {\int_use:N \g_shipout_readonly_int}}
      \fi
```

But we may have guessed wrongly earlier and have run it too early or we still have to run the `shipout/lastpage` even though there is no page to place it into. If that is the case we make a trivial extra page and put it there. This temporary page will then vanish again on the next run but helps to keep pdf viewers happy. In either case we should put out an appropriate "rerun" warning.

```
       \bool_if:NTF \g__shipout_lastpage_handled_bool
          {
```

If the hook was already executed, we have to test if that total shipouts match the shipouts from last run (because that corresponds to the page it was executed). If not we output a warning.

```
            \int_compare:nNnF \@abspage@last = \g_shipout_readonly_int
             {
               \@latex@warning@no@line{Hook~ 'shipout/lastpage'~ executed~
                 on~ wrong~ page~ (\@abspage@last\space not~
                 \int_use:N\g_shipout_readonly_int).\MessageBreak
                 Rerun~ to~ correct~ this}%
             }
          }
          {
```

If the hook was not run, we need to add an extra page and place it there. However, making this extra page in case the hook is actually empty would be forcing a rerun without any reason, so we check that condition and also check if `\@kernel@after@shipout@lastpage` contains any code. If both are empty we omit the page generation.

```
            \bool_lazy_and:nnF
```

```
            { \hook_if_empty_p:n {shipout/lastpage} }
            { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
            {
              \global\advance\c@page\@ne
              \tex_shipout:D\vbox to\textheight
                {
                  \hbox:n { \UseHook{shipout/lastpage}
                            \@kernel@after@shipout@lastpage }
```

This extra page could be totally empty except for the hook content, but to help the user understanding why it is there we put some text into it.

```
                  \__shipout_excuse_extra_page:
                  \null
                }
```

At this point we also signal to LaTeX's endgame that a rerun is necessary so that an appropriate message can be shown on the terminal. We do this by simply defining a command used as a flag and tested in `\enddocument`.

```
              \cs_gset_eq:NN \@extra@page@added \relax
            }
        }
    }
}
```

(*End of definition for* `\enddocument` *and others.*)

`\__shipout_excuse_extra_page:` Say mea culpa ...

```
\cs_new:Npn \__shipout_excuse_extra_page: {
  \vfil
  \begin{center}
    \bfseries Temporary~ page!
  \end{center}
    \LaTeX{}~ was~ unable~ to~ guess~ the~ total~ number~ of~ pages~
    correctly.~ ~ As~ there~ was~ some~ unprocessed~ data~ that~
    should~ have~ been~ added~ to~ the~ final~ page~ this~ extra~
    page~ has~ been~ added~ to~ receive~ it.
    \par
    If~ you~ rerun~ the~ document~ (without~ altering~ it)~ this~
    surplus~ page~ will~ go~ away,~ because~ \LaTeX{}~ now~ knows~
    how~ many~ pages~ to~ expect~ for~ this~ document.
  \vfil
}
```

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

`\PreviousTotalPages` In the preamble before the `aux` file was read `\PreviousTotalPages` is always zero.

`\@kernel@before@begindocument`

```
\def\PreviousTotalPages{0}
```

In the `aux` file there should be an update for `\@abspage@last` recording the number of pages from the previous run. If not that macro holds the value of `\maxdimen`. So we test for it and update `\PreviousTotalPages` if there was a real value. This should happen just before the `begindocument` hook is executed so that the value can be used inside that hook.

```
\g@addto@macro\@kernel@before@begindocument
  {\ifnum\@abspage@last<\maxdimen
     \xdef\PreviousTotalPages{\@abspage@last}\fi}
```

(*End of definition for* `\PreviousTotalPages` *and* `\@kernel@before@begindocument`*. These functions are documented on page 6.*)

4 Legacy LaTeX 2ε interfaces

`\DiscardShipoutBox` Request that the next shipout box is to be discarded.

```
\cs_new_eq:NN \DiscardShipoutBox \shipout_discard:
```

(*End of definition for* `\DiscardShipoutBox`*. This function is documented on page 5.*)

`\AtBeginDvi` If we roll forward from an earlier kernel `\AtBeginDvi` is defined so we better not use `\cs_new_protected:Npn` here.

```
\cs_set_protected:Npn \AtBeginDvi
                      {\__shipout_add_firstpage_material:Nn \AtBeginDvi}
```

(*End of definition for* `\AtBeginDvi`*. This function is documented on page 5.*)

`\DebugShipoutsOn`
`\DebugShipoutsOff`

```
\cs_new_eq:NN \DebugShipoutsOn  \shipout_debug_on:
\cs_new_eq:NN \DebugShipoutsOff \shipout_debug_off:
```

(*End of definition for* `\DebugShipoutsOn` *and* `\DebugShipoutsOff`*. These functions are documented on page 7.*)

5 Internal commands needed elsewhere

These internal commands use double and triple `@` signs so we need to stop getting them translated to the module name.

```
⟨@@=⟩
```

`\@expl@@@shipout@add@firstpage@material@@Nn`
`\@expl@@@shipout@add@background@box@@n`
`\@expl@@@shipout@add@foreground@box@@n`
`\@expl@@@shipout@add@background@picture@@n`
`\@expl@@@shipout@add@foreground@picture@@n`

Some internals needed elsewhere.

```
\cs_set_eq:NN \@expl@@@shipout@add@firstpage@material@@Nn
              \__shipout_add_firstpage_material:Nn
\cs_set_eq:NN \@expl@@@shipout@add@background@box@@n
              \__shipout_add_background_box:n
\cs_set_eq:NN \@expl@@@shipout@add@foreground@box@@n
              \__shipout_add_foreground_box:n
\cs_set_eq:NN \@expl@@@shipout@add@background@picture@@n
              \__shipout_add_background_picture:n
\cs_set_eq:NN \@expl@@@shipout@add@foreground@picture@@n
              \__shipout_add_foreground_picture:n
```

(*End of definition for* `\@expl@@@shipout@add@firstpage@material@@Nn` *and others.*)

```
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
```

Rolling back here doesn't undefine the interface commands as they may be used in packages without rollback functionality. So we just make them do nothing which may or may not work depending on the code usage.

```
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\shipout}{Hook management (shipout)}%
⟨latexrelease⟩
```

If we roll forward then `\tex_shipout:D` may not be defined in which case `\shipout` does have it original definition and so we must not `\let` it to something else which is `\relax`!

```
⟨latexrelease⟩\ifcsname tex_shipout:D\endcsname
⟨latexrelease⟩\expandafter\let\expandafter\shipout
⟨latexrelease⟩                 \csname tex_shipout:D\endcsname
⟨latexrelease⟩\fi
⟨latexrelease⟩
⟨latexrelease⟩\let \RawShipout\@undefined
⟨latexrelease⟩\let \ShipoutBox\@undefined
⟨latexrelease⟩\let \ReadonlyShipoutCounter \@undefined
⟨latexrelease⟩\let \c@totalpages \@undefined
⟨latexrelease⟩\let \thetotalpages \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let \DiscardShipoutBox \@undefined
⟨latexrelease⟩\let \DebugShipoutsOn \@undefined
⟨latexrelease⟩\let \DebugShipoutsOff \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\DeclareRobustCommand \AtBeginDvi [1]{%
⟨latexrelease⟩  \global \setbox \@begindvibox
⟨latexrelease⟩    \vbox{\unvbox \@begindvibox #1}%
⟨latexrelease⟩}
⟨latexrelease⟩
⟨latexrelease⟩\let \AtBeginShipout \@undefined
⟨latexrelease⟩\let \AtBeginShipoutNext \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let \AtBeginShipoutFirst \@undefined
⟨latexrelease⟩
⟨latexrelease⟩\let \ShipoutBoxHeight \@undefined
⟨latexrelease⟩\let \ShipoutBoxDepth \@undefined
⟨latexrelease⟩\let \ShipoutBoxWidth \@undefined
⟨latexrelease⟩
```

We do not undo a substitution when rolling back. As the file support gets undone the underlying data is no longer used (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\undeclare@...` and its support macros available in all earlier kernel releases which is pointless (and actually worse).

```
⟨latexrelease⟩
⟨latexrelease⟩\let  \AtEndDvi \@undefined
```

We do not reenable a disabled package load when rolling back. As the file support gets undone the underlying data is no longer checked (and sufficiently obscure that it should not interfere with existing commands) and properly removing it would mean we need to make the `\reenable@package@load` command available in all earlier kernel releases which is pointless (and actually worse).

```
%\reenable@package@load{atenddvi}
```

```
⟨latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

__shipout_force_immediate_writes: Change all writes to be immediate. To be used after the final page has been shipped out.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2025/06/01}%
⟨latexrelease⟩   {\__shipout_force_immediate_writes:}{Force immediate writes}%
\ExplSyntaxOn
\cs_new:Npn \__shipout_force_immediate_writes: {
```

Need to check if any variants are manually defined and if so adjust them too — not done!

```
  \cs_gset_eq:NN \iow_shipout:Nn \iow_now:Nn
  \cs_gset_eq:NN \lua_shipout:n \lua_now:n
  \cs_gset:Npn \write {\tex_immediate:D \tex_write:D}
```

As the writes will happen without a page break, reset the page number so they reference the last page.

```
  \global\advance\c@page\m@ne
}
\ExplSyntaxOff
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩   {\__shipout_force_immediate_writes:}{Force immediate writes}%
```

We want a definition for this even when doing rollback so that it can stay in the kernel hook without forcing a rollback there as well.

```
⟨latexrelease⟩\ExplSyntaxOn
⟨latexrelease⟩ \cs_new_eq:NN \__shipout_force_immediate_writes: \prg_do_nothing:
⟨latexrelease⟩\ExplSyntaxOff
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

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

6 Package emulation for compatibility

6.1 Package **atenddvi** emulation

\AtEndDvi This package has only one public command, so simulating it is easy and actually sensible to provide as part of the kernel.

```
⟨/2ekernel⟩
⟨*2ekernel | latexrelease⟩
⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
⟨latexrelease⟩                  {\AtEndDvi}{atenddvi emulation}%
\ExplSyntaxOn
\cs_new_protected:Npn \AtEndDvi #1 {\AddToHook{shipout/lastpage}{#1}}
\ExplSyntaxOff
```

As the package is integrate we prevent loading (no need to roll that back):

```
\disable@package@load{atenddvi}
   {\PackageWarning{atenddvi}
     {Functionality of this package is already\MessageBreak
      provided by LaTeX.\MessageBreak\MessageBreak
      It is there no longer necessary to load it\MessageBreak
      and you can safely remove it.\MessageBreak
      Found on}}
⟨/2ekernel | latexrelease⟩
⟨latexrelease⟩\EndIncludeInRelease
⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
⟨latexrelease⟩                  {\AtEndDvi}{atenddvi emulation}%
⟨latexrelease⟩\let \AtEndDvi \@undefined
⟨latexrelease⟩\EndIncludeInRelease
⟨*2ekernel⟩
```

(*End of definition for* `\AtEndDvi`*. This function is documented on page 5.*)

```
⟨/2ekernel⟩
```

6.2 Package **atbegshi** emulation

```
⟨*atbegshi-ltx⟩
\ProvidesPackage{atbegshi-ltx}
   [2021/01/10 v1.0c
     Emulation of the original atbegshi^^Jpackage with kernel methods]
```

\AtBeginShipoutBox

```
\let \AtBeginShipoutBox \ShipoutBox
```

(*End of definition for* `\AtBeginShipoutBox`*. This function is documented on page 7.*)

\AtBeginShipoutInit Compatibility only, we aren't delaying ...

```
\let \AtBeginShipoutInit \@empty
```

(*End of definition for* `\AtBeginShipoutInit`*. This function is documented on page 8.*)

\AtBeginShipout
\AtBeginShipoutNext Filling hooks

```
\protected\long\def\AtBeginShipout     #1{\AddToHook{shipout/before}{#1}}
\protected\long\def\AtBeginShipoutNext #1{\AddToHookNext{shipout/before}{#1}}
```

(*End of definition for* `\AtBeginShipout` *and* `\AtBeginShipoutNext`*. These functions are documented on page 8.*)

\AtBeginShipoutFirst Slightly more complex as we need to know the name of the command under which the `shipout/firstpage` hook is filled.

```
\protected \def \AtBeginShipoutFirst
   {\@expl@@@shipout@add@firstpage@material@@Nn \AtBeginShipoutFirst}
```

(*End of definition for* `\AtBeginShipoutFirst`*. This function is documented on page 8.*)

\AtBeginShipoutDiscard Just a different name.

```
\let \AtBeginShipoutDiscard \DiscardShipoutBox
```

(*End of definition for* `\AtBeginShipoutDiscard`*. This function is documented on page 8.*)

\AtBeginShipoutAddToBox
\AtBeginShipoutAddToBoxForeground
\AtBeginShipoutUpperLeft
\AtBeginShipoutUpperLeftForeground

We don't expose them.

```
\let \AtBeginShipoutAddToBox
               \@expl@@@shipout@add@background@box@@n
\let \AtBeginShipoutAddToBoxForeground
               \@expl@@@shipout@add@foreground@box@@n

\let \AtBeginShipoutUpperLeft
               \@expl@@@shipout@add@background@picture@@n
\let \AtBeginShipoutUpperLeftForeground
               \@expl@@@shipout@add@foreground@picture@@n
```

(*End of definition for* `\AtBeginShipoutAddToBox` *and others. These functions are documented on page* 7.)

\AtBeginShipoutOriginalShipout

This offers the raw `\shipout` primitive of the engine. A page shipped out with this is not counted by `\ReadonlyShipoutCounter` counter and thus the mechanism to place `\specials` at the very end of the output might fail, etc. It should therefore not be used in new applications but is only provided to allow running legacy code. For new code use the commands provided by the kernel instead.

```
\ExplSyntaxOn
\cs_new_eq:NN \AtBeginShipoutOriginalShipout \tex_shipout:D
```

(*End of definition for* `\AtBeginShipoutOriginalShipout`. *This function is documented on page* 8.)

\ShipoutBoxHeight
\ShipoutBoxWidth
\ShipoutBoxDepth

This is somewhat different from the original in atbegshi where `\ShipoutBoxHeight` etc. only holds the `\the\ht<box>` value. This may has some implications in some use cases and if that is a problem then it might need changing.

```
\cs_new:Npn \ShipoutBoxHeight { \dim_use:N \l_shipout_box_ht_dim }
\cs_new:Npn \ShipoutBoxDepth  { \dim_use:N \l_shipout_box_dp_dim }
\cs_new:Npn \ShipoutBoxWidth  { \dim_use:N \l_shipout_box_wd_dim }
\ExplSyntaxOff
```

(*End of definition for* `\ShipoutBoxHeight`, `\ShipoutBoxWidth`, *and* `\ShipoutBoxDepth`.)

⟨/atbegshi-ltx⟩

If the package is requested we substitute the one above:

⟨*2ekernel⟩
`\declare@file@substitution{atbegshi.sty}{atbegshi-ltx.sty}`
⟨/2ekernel⟩

6.3 Package everyshi emulation

This is now directly handled in that package so emulation is not necessary any more. Rather important :-)

⟨@@=⟩

Index

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

A

\AddEverypageHook *9*
\AddThispageHook *9*
\AddToHook 251, 547, 570, *7*
\AddToHookNext 9, 571, *9*
\advance 432, 530
\allocationnumber 95
\Alph *6*
\arabic 397, *6*
\AtBeginDvi 465, 501, *5*
\AtBeginShipout 506, 570, *8*
\AtBeginShipoutAddToBox 575, *7*
\AtBeginShipoutAddToBoxForeground 575, *7*
\AtBeginShipoutBox 568, *7*
\AtBeginShipoutDiscard 574, *8*
\AtBeginShipoutFirst 509, 572, *8*
\AtBeginShipoutInit 569, *8*
\AtBeginShipoutNext 507, 570, *8*
\AtBeginShipoutOriginalShipout . 583, *8*
\AtBeginShipoutUpperLeft 575, *7*
\AtBeginShipoutUpperLeftForeground 575, *7*
\AtEndDvi 516, 542, *5*
\AtNextShipout *8*

B

\baselineskip 283, 334
\begin 447
begindocument (hook) *22*, *25*
\bfseries 448
bool commands:
  \bool_gset_false:N 76, 142, 151
  \bool_gset_true:N 71, 181, 391
  \bool_if:NTF 82, 149, 417
  \bool_lazy_and:nnTF . . . 127, 173, 428
  \bool_new:N 67, 249, 264
box commands:
  \box_dp:N 255
  \box_ht:N 254, 363, *13*
  \box_if_empty:NTF 139, 158
  \box_if_horizontal:NTF 295, 347
  \box_if_vertical:NTF 269, 320
  \box_move_up:nn 309, 363
  \box_new:N 84, 86, 248, 265
  \box_set_dp:Nn 37, 57, 282, 291, 308, 333, 344, 362
  \box_set_eq:NN 210, 240, 245
  \box_set_eq_drop:NN 154
  \box_set_ht:Nn 36, 56, 281, 290, 307, 332, 343, 361
  \box_set_wd:Nn 280, 306, 331, 360
  \box_use:N 38, 58, 187, 286, 311, 339, 364
  \box_wd:N 256, 357, 365

C

\clearpage *24*
cs commands:
  \cs_gset:Npn 234, 529
  \cs_gset_eq:NN 113, 212, 227, 228, 233, 242, 246, 382, 440, 527, 528
  \cs_gset_protected:Npe 81
  \cs_if_exist_use:NTF 374, 375, 379, 380
  \cs_new:Npn 6, 7, 8, 23, 48, 115, 120, 138, 202, 207, 224, 239, 244, 250, 253, 267, 318, 371, 384, 387, 397, 445, 526, 585, 586, 587
  \cs_new_eq:NN 68, 213, 394, 396, 464, 467, 468, 538, 584
  \cs_new_protected:Npn 69, 74, 79, 390, 547, *26*
  \cs_set:Npn 15
  \cs_set_eq:NN 85, 105, 144, 156, 186, 193, 470, 472, 474, 476, 478
  \cs_set_protected:Npn 18, 19, 107, 196, 465
\csname 488

D

\DebugShipoutsOff 467, 499, *7*
\DebugShipoutsOn 467, 498, *7*
\DeclareRobustCommand 501
\def 460, 570, 571, 572
dim commands:
  \dim_new:N 260, 261, 262, 263
  \dim_set:Nn 254, 255, 256, 257
  \dim_use:N 585, 586, 587
  \c_max_dim 274, 300, 325, 352
  \c_zero_dim 30, 36, 37, 54, 56, 57, 280, 281, 282, 288, 306, 307, 308, 331, 332, 333, 360, 361, 362
\DiscardShipoutBox . . 143, 464, 497, 574, *5*

E

\end 449
\endcsname 486, 488
enddocument (hook) *23*

\enddocument 399, *23*
enddocument/afterlastpage (hook) . . . *23*
\EndIncludeInRelease . . . . . .
. . 44, 63, 482, 519, 534, 540, 557, 561
\everyjob 90, 91
\EveryShipout *8*
exp commands:
\exp_args:Ne 90
\exp_not:N 92, 156, 186
\exp_not:n 91
\expandafter 487
\ExplSyntaxOff 42, 62, 480, 532, 539, 548, 588
\ExplSyntaxOn 5, 47, 66, 525, 537, 546, 583

F

\fi 416, 463, 489

G

\gdef 415
\global 432, 502, 530
group commands:
\group_begin: 153
\group_end: 155

H

\hbadness 299, 301, 351, 353
\hbox *3*
hbox commands:
\hbox:n 435
\hbox_set:Nn . . 231, 278, 304, 329, 358
\hbox_set_to_wd:Nnn . 30, 54, 302, 354
\hbox_unpack:N 312, 356
\hbox_unpack_drop:N 241
\hfuzz 298, 300, 350, 352
hook commands:
\hook_if_empty:nTF 124, 225
\hook_if_empty_p:n 128, 174, 429
\hook_new:n . . . . . .
. . . . 214, 215, 216, 217, 218, 219, 220
\hook_use:n . . . 123, 126, 132, 136, 184
Hooks:
begindocument *22*, *25*
enddocument *23*
enddocument/afterlastpage *23*
shipout *5*, *6*, *3*, *4*, *4*
shipout/... . . . . *2*, *5*
shipout/after . *6*, *6*, *2*, *2*, *2*, *15*, *4*, *4*, *4*
shipout/afterpage *18*
shipout/background . . . . . .
. . . . *5*, *7*, *7*, *8*, *3*, *13*, *3*, *14*, *17*, *17*, *4*
shipout/before *5*, *5*, *6*, *6*, *2*,
8, *2*, *8*, *2*, *3*, *3*, *13*, *14*, *3*, *3*, *4*, *4*, *4*, *5*, *5*
shipout/firstpage *2*, *2*,
15, *15*, *17*, *17*, *18*, *4*, *18*, *4*, *4*, *5*, *2*, *29*
shipout/foreground . . . . . .
. . . . . *5*, *7*, *7*, *8*, *3*, *14*, *3*, *4*
shipout/lastpage *2*, *13*, *15*,
15, *18*, *4*, *23*, *23*, *24*, *24*, *4*, *4*, *4*, *5*, *2*
\hss 33, 55, *10*

I

\ifcsname 486
\ifnum 462, *6*
\IfPDFManagementActiveT 11
\ignorespaces 33, 55
\immediate *24*
\IncludeInRelease . . . . . .
. . . . 3, 45, 64, 483, 523, 535, 544, 558
int commands:
\int_compare:nNnTF . . . . . .
. . . . . 116, 172, 203, 400, 411, 419
\int_eval:n 402
\int_gincr:N 148, 163
\int_new:N 393, 395
\int_use:N 165, 177, 415, 423, *6*
\int_value:w 109, 198
\c_max_int 275, 301, 326, 353
\c_zero_int 152
iow commands:
\iow_now:Nn 414, 527
\iow_shipout:Nn 527

K

\kern 25,
27, 50, 52, 288, 338, 340, 357, 365, *20*

L

\LaTeX 450, 456
\let 221, 222, 223, 487, 491, 492,
493, 494, 495, 497, 498, 499, 506,
507, 509, 511, 512, 513, 516, 560,
568, 569, 574, 575, 577, 579, 581, *27*
\lineskip 284, 335
\lineskiplimit 285, 336
\long 570, 571
lua commands:
\lua_now:n 92, 528
\lua_shipout:n 528

M

\maxdimen 398, 400, 462, *25*
\MessageBreak . . . . . .
. . . . 160, 236, 423, 551, 552, 553, 554

N

\newprotectedluacmd 89
\nofiles *24*
\null 438
\number 398

P

\PackageWarning 550
\par 454
pdfannot commands:
\pdfannot_link_off: 19
\pdfannot_link_on: 18
\pdfextension 15
\pdfhorigin 375
\pdfvariable 374, 379
\pdfvorigin 380
pre commands:
pre_shipout_filter *6*
\PreviousTotalPages 460, *6*
prg commands:
\prg_do_nothing: 227, 228, 233, 242, 246, 382, 538
\protect 144, 156, 186, 193, 213, *16*
\protected 570, 571, 572
\ProvidesPackage 565
\put *9*

R

\RawShipout 212, 491, *4*
\ReadonlyShipoutCounter 393, 493, *6*
\relax 272, 273, 298, 299, 323, 324, 350, 351, 440, *27*
\RequirePackage *7*
\Roman *6*

S

scan commands:
\scan_stop: . 25, 27, 28, 50, 52, 53, 105
\setbox 502
shipout (hook) *5*, *6*, *3*, *4*, *4*
\shipout 65, 113, 484, 487, *4*
shipout 214
shipout commands:
\l_shipout_box 84, 96, 99, 111, 122, 187, 210, 240, 245, 268, 269, 276, 287, 290, 291, 295, 302, 312, 320, 327, 337, 343, 344, 347, 354, 356, 357, 363, 365, *15*
\l_shipout_box_dp_dim 255, 258, 260, 291, 344, 586, *3*
\l_shipout_box_ht_dim 254, 258, 260, 290, 310, 343, 585, *3*
\l_shipout_box_ht_plus_dp_dim 257, 260, 276, 327, 338, 340, *3*
\l_shipout_box_wd_dim 256, 260, 302, 354, 587, *3*
\shipout_debug_off: . . . 68, 74, 468, *7*
\shipout_debug_on: 68, 69, 467, *7*
\shipout_discard: . . . 390, 390, 464, *5*
\g_shipout_readonly_int 163, 165, 172, 177, 393, 402, 411, 415, 419, 423, *6*
\g_shipout_totalpage_int *6*
\g_shipout_totalpages_int 148, 395, *6*
shipout internal commands:
__shipout_add_background_box:n 241, 267, 267, 385, 473
__shipout_add_background_picture:n 130, 384, 384, 477
__shipout_add_firstpage_material:Nn . 234, 250, 250, 466, 471
__shipout_add_firstpage_specials: 171, 227, 239, 239, 242, *15*
__shipout_add_foreground_box:n 178, 318, 318, 388, 475
__shipout_add_foreground_picture:n 125, 387, 387, 479
__shipout_debug:n 68, 68, 81, 164, 176, 209, *11*
\g__shipout_debug_bool . 67, 71, 76, 82
__shipout_debug_gset: . 68, 72, 77, 79
\g__shipout_discard_bool 142, 149, 151, 264, 391
__shipout_drop_firstpage_specials: 188, 228, 239, 244, 246, *17*
__shipout_excuse_extra_page: 437, 445, 445
__shipout_execute: 107, 107, 113, *12*
__shipout_execute_cont: 118, 120, 120
__shipout_execute_main_cont:Nnnn 121, 138, 138, 208, *16*
__shipout_execute_nohooks_cont: 205, 207
__shipout_execute_raw: 196, 196, 212, *16*
__shipout_execute_test_level: 110, 115, 115
__shipout_execute_test_level_raw: 196, 199, 202
__shipout_finalize_box: 87, 89, 105, 185
\l__shipout_firstpage_box 231, 241, 248, *17*
__shipout_force_immediate_writes: 406, 521, 524, 526, 536, 538
__shipout_get_box_size:N 146, 168, 253, 253, 268, *19*
\l__shipout_group_level_tl 108, 114, 117, 197, 204
\c__shipout_horigin_tl . . . 25, 50, 371
__shipout_init_page_origins: 24, 49, 371, 371, 382

\g__shipout_lastpage_handled_bool 181, 249, 417
__shipout_picture_overlay:n 4, 6, 23, 46, 48, 385, 388
__shipout_picture_overlay_linkoff: 7, 19, 32
__shipout_picture_overlay_linkon: 6, 18, 34
__shipout_picture_overlay_linkstate: 8, 15, 29
\l__shipout_raw_box 86, 200, 208, 210, 240, 245, *15*
__shipout_run_firstpage_hook: 169, 224, 224, 233, *17*
\l__shipout_saved_badness_tl 265, 271, 279, 292, 297, 305, 314, 322, 330, 342, 349, 359, 367
__shipout_saved_protect: 144, 193, 213, 213
\l__shipout_tmp_box 30, 36, 37, 38, 54, 56, 57, 58, 265, 278, 280, 281, 282, 286, 304, 306, 307, 308, 311, 329, 331, 332, 333, 339, 358, 360, 361, 362, 364
\c__shipout_vorigin_tl . . . 27, 52, 371
shipout/... (hook) *2*, *5*
shipout/after (hook) *6*, *6*, *2*, *2*, *2*, *15*, *4*, *4*, *4*
shipout/after 214, *3*
shipout/afterpage (hook) *18*
shipout/background (hook) *5*, *7*, *7*, *8*, *3*, *13*, *3*, *14*, *17*, *17*, *4*
shipout/background 214, *3*
shipout/before (hook) . . . *5*, *5*, *6*, *6*, *2*, *8*, *2*, *8*, *2*, *3*, *3*, *13*, *14*, *3*, *3*, *4*, *4*, *4*, *5*, *5*
shipout/before 214, *3*
shipout/firstpage (hook) *2*, *2*, *15*, *15*, *17*, *17*, *18*, *4*, *18*, *4*, *4*, *5*, *2*, *29*
shipout/firstpage 214, *3*
shipout/foreground (hook) *5*, *7*, *7*, *8*, *3*, *14*, *3*, *4*
shipout/foreground 214, *3*
shipout/lastpage (hook) . . . *2*, *13*, *15*, *15*, *18*, *4*, *23*, *23*, *24*, *24*, *4*, *4*, *4*, *5*, *2*
shipout/lastpage 214, *3*
\ShipoutBox 84, 492, 568, *8*
\ShipoutBoxDepth 512, 585
\ShipoutBoxHeight 511, 585, *30*
\ShipoutBoxWidth 513, 585
skip commands:
\skip_zero:N 283, 284, 285, 334, 335, 336
\space 166, 422
\special *30*
\string 236, 415
sys commands:
\sys_if_engine_luatex:TF 13, 87

T

tag internal commands:
__tag_lastpage_label: 407
TEX and LATEX 2ε commands:
\@abspage@last 166, 172, 398, 400, 402, 415, 419, 422, 462, 463, *25*
\@auxout 414
\@begindvi *9*
\@begindvibox 502, 503, *4*
\@cclv *8*
\@empty 221, 222, 223, 569
\@expl@@@shipout@add@background@box@@n 470, 576
\@expl@@@shipout@add@background@picture@@n 470, 580
\@expl@@@shipout@add@firstpage@material@@Nn 470, 573
\@expl@@@shipout@add@foreground@box@@n 470, 578
\@expl@@@shipout@add@foreground@picture@@n 470, 582
\@extra@page@added 440
\@kernel@after@enddocument 399
\@kernel@after@enddocument@afterlastpage 410
\@kernel@after@shipout@background 133, 221, *13*
\@kernel@after@shipout@lastpage 175, 180, 221, 430, 436, *24*
\@kernel@before@begindocument . . 460
\@kernel@before@enddocument@afterlastpage 405
\@kernel@before@shipout@background 129, 131, 221, *13*
\@latex@info@no@line 150
\@latex@warning 235
\@latex@warning@no@line 140, 159, 421
\@ne 432
\@undefined 491, 492, 493, 494, 495, 497, 498, 499, 506, 507, 509, 511, 512, 513, 516, 560
\c@page 432, 530
\c@totalpages 395, 494
\declare@file@substitution 591
\disable@package@load 549
\g@addto@macro 399, 405, 410, 461
\if@filesw 413
\m@ne 530
\reenable@package@load 517, *27*
\set@typeset@protect 145, 189
\undeclare@... . . . . *27*

tex commands:
\tex_afterassignment:D 110, 199
\tex_aftergroup:D 118, 205, *12*
\tex_currentgrouplevel:D 109, 117, 198, 204
\tex_deadcycles:D 152
\tex_immediate:D 529
\tex_setbox:D 111, 200
\tex_shipout:D 187, 433, 584
\tex_vss:D 39, 59
\tex_write:D 529
\textheight 433
\the 95, 96, 99, 272, 273, 298, 299, 323, 324, 350, 351
\thepage *23*
\thetotalpages 397, 495, *23*
tl commands:
\tl_const:Nn 372, 377
\tl_if_empty_p:N 129, 175, 430
\tl_new:N 114, 266
\tl_set:Nn . 108, 197, 271, 297, 322, 349
totalpages *6*
\typeout 165, 176, 209, *4*

U

\unitlength 28, 53, *9*
\unvbox 503
use commands:
\use_none:n 68
\UseHook 179, 231, 435
\usepackage *7*

V

\vbadness 273, 275, 324, 326, *19*
\vbox 433, 503, *19*
vbox commands:
\vbox_set_to_ht:Nnn 276, 327
\vbox_to_zero:n 26, 51
\vbox_unpack:N 287, 337
\vfil 446, 458
\vfuzz 272, 274, 323, 325, *19*

W

\write 529, *15*

X

\xdef 398, 402, 463