The `ltshipout` documentation*

Frank Mittelbach, LaTeX Project Team

December 3, 2022

Contents

---

*This file has version v1.0n dated 2022/11/08, © 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   This box register is called \ShipoutBox (alternatively available via the L3 name \l_-
\l_shipout_box   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 LaTeX2_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 2ε 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
\g_shipout_readonly_int `\ifnum\ReadonlyShipoutCounter=...`
`\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
\g_shipout_totalpages_int `\arabic{totalpages}`
`\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 `\thetotalpages/\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 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{2020/10/01}%
4 ⟨latexrelease⟩                    {\shipout}{Hook management (shipout)}%
5 \ExplSyntaxOn
```

3.1 Debugging

`\g__shipout_debug_bool` Holds the current debugging state.

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

(*End 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`.

```
7 \cs_new_eq:NN \__shipout_debug:n  \use_none:n
8 \cs_new_protected:Npn \shipout_debug_on:
9   {
10     \bool_gset_true:N \g__shipout_debug_bool
11     \__shipout_debug_gset:
12   }
13 \cs_new_protected:Npn \shipout_debug_off:
14   {
15     \bool_gset_false:N \g__shipout_debug_bool
16     \__shipout_debug_gset:
17   }
18 \cs_new_protected:Npn \__shipout_debug_gset:
19   {
20     \cs_gset_protected:Npx \__shipout_debug:n ##1
21       { \bool_if:NT \g__shipout_debug_bool {##1} }
22   }
```

(End 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).

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

(End 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.

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

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

__shipout_finalize_box:

For LuaTeX invoke the `pre_shipout_filter` callback.

```
26 \sys_if_engine_luatex:TF
27   {
28     \newprotectedluacmd \__shipout_finalize_box:
29     \exp_args:Nx \everyjob {
30       \exp_not:V \everyjob
31       \exp_not:N \lua_now:n {
32         luatexbase.create_callback('pre_shipout_filter', 'list')
33         local~call, getbox, setbox = luatexbase.call_callback, tex.getbox, tex.setbox~
34         lua.get_functions_table()[\the \allocationnumber] = function()
35           local~head = getbox(\the \l_shipout_box)
36           local~result = call('pre_shipout_filter', head)
37           if~not (result == head) then~
38             setbox(\the \l_shipout_box, result~or~nil)
39           end~
40         end
41       }
42     }
43   } {
44     \cs_set_eq:NN \__shipout_finalize_box: \scan_stop:
45   }
```

(End 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.

```
46 \cs_set_protected:Npn \__shipout_execute: {
47   \tl_set:Nx \l_shipout_group_level_tl
48     { \int_value:w \tex_currentgrouplevel:D }
49   \tex_afterassignment:D \__shipout_execute_test_level:
50   \tex_setbox:D \l_shipout_box
51 }
```

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

\shipout

Overloading the `\shipout` primitive:

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

(End 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.

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

(*End 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`.

```
54 \cs_new:Npn \__shipout_execute_test_level: {
55   \int_compare:nNnT
56     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
57     \tex_aftergroup:D \__shipout_execute_cont:
58 }
```

(*End 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.

```
59 \cs_new:Npn \__shipout_execute_cont: {
60   \__shipout_execute_main_cont:Nnnn
61     \l_shipout_box
62     { \hook_use:n {shipout/before} }
63     { \hook_if_empty:nF {shipout/foreground}
64       { \__shipout_add_foreground_picture:n
65         { \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).

```
66       \bool_lazy_and:nnF
67         { \hook_if_empty_p:n {shipout/background} }
68         { \tl_if_empty_p:N \@kernel@before@shipout@background }
69         { \__shipout_add_background_picture:n
70           { \@kernel@before@shipout@background
71             \hook_use:n {shipout/background}
72             \@kernel@after@shipout@background }
73         }
74     }
75     { \hook_use:n {shipout/after} }
76 }
```

(*End 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`.

11

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]

```
77 \cs_new:Npn \__shipout_execute_main_cont:Nnnn #1#2#3#4 {
78   \box_if_empty:NTF #1
79     { \@latex@warning@no@line{ Ignoring~ void~ shipout~ box } }
80     {
```

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.

```
81 %       \bool_gset_false:N \g__shipout_discard_bool  % setting this would disable
82                                         % \DiscardShipoutBox on doc-level
83       \cs_set_eq:NN \__shipout_saved_protect: \protect
84       \set@typeset@protect
```

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

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

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

```
86       #2
```

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).

```
87       \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.

```
88       \bool_if:NTF \g__shipout_discard_bool
89         { \@latex@info@no@line{Completed~ page~ discarded}
90           \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.

```
91           \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?

```
92 %           \group_begin:
93 %             \box_set_eq_drop:NN #1 #1
94 %           \group_end:
95 %           \cs_set_eq:NN \protect \exp_not:N
96         }
```

---

[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.

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.

```
97          { \box_if_empty:NTF #1
98              { \@latex@warning@no@line { Ignoring~ void~ shipout~ box.
99                  \MessageBreak The~ shipout~ box~ was~ voided~ by~ hook~ code }
100             }
```

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]

```
101             {
102                 \int_gincr:N \g_shipout_readonly_int
103                 \__shipout_debug:n {
104                     \typeout{Absolute~ page~ =~ \int_use:N \g_shipout_readonly_int
105                         \space (target:~ \@abspage@last)}
106                 }
```

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.

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

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 `\specials` 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:`.

```
108                 \__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`.

```
109                 #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.

```
110                 \__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.

```
111                 \int_compare:nNnT \@abspage@last = \g_shipout_readonly_int
112                     { \bool_lazy_and:nnF
113                         { \hook_if_empty_p:n {shipout/lastpage} }
114                         { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
115                         { \__shipout_debug:n { \typeout{Executing~ lastpage~ hook~
116                             on~ page~ \int_use:N \g_shipout_readonly_int } }
117                         \__shipout_add_foreground_box:n
118                             { \UseHook{shipout/lastpage}
119                                 \@kernel@after@shipout@lastpage }
```

---

[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.

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

```
120                        \bool_gset_true:N \g__shipout_lastpage_handled_bool
121                      }
122                  }

123              \hook_use:n {shipout}
124              \__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.

```
125              \cs_set_eq:NN \protect \exp_not:N
126              \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.

```
127              \__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.

```
128              \set@typeset@protect
129              #4
130          }
131        }
```

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).

```
132      \cs_set_eq:NN \protect \__shipout_saved_protect:
133    }
134 }
```

(*End 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.

```
135 \cs_set_protected:Npn \__shipout_execute_raw: {
136   \tl_set:Nx \l__shipout_group_level_tl
137     { \int_value:w \tex_currentgrouplevel:D }
138   \tex_afterassignment:D \__shipout_execute_test_level_raw:
139   \tex_setbox:D \l__shipout_raw_box
140 }

141 \cs_new:Npn \__shipout_execute_test_level_raw: {
142   \int_compare:nNnT
143     \l__shipout_group_level_tl < \tex_currentgrouplevel:D
144     \tex_aftergroup:D \__shipout_execute_nohooks_cont:
145 }
```

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

```
146 \cs_new:Npn \__shipout_execute_nohooks_cont: {
147   \__shipout_execute_main_cont:Nnnn \l__shipout_raw_box
148     {} { \__shipout_debug:n{ \typeout{Doing~ raw~ shipout~ ...} }
149           \box_set_eq:NN \l_shipout_box \l__shipout_raw_box } {}
150 }
```

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

\RawShipout The interface name for raw shipout.

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

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

__shipout_saved_protect: Remember the current \protect state.

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

(*End 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.

```
153 \hook_new:n{shipout/before}
154 \hook_new:n{shipout}
155 \hook_new:n{shipout/after}
156 \hook_new:n{shipout/foreground}
157 \hook_new:n{shipout/background}
158 \hook_new:n{shipout/firstpage}
159 \hook_new:n{shipout/lastpage}
```

(*End 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.

```
160 \let\@kernel@after@shipout@lastpage\@empty
161 \let\@kernel@before@shipout@background\@empty
162 \let\@kernel@after@shipout@background\@empty
```

(*End definition for* \@kernel@after@shipout@lastpage, \@kernel@before@shipout@background, *and* \@kernel@after@shipout@background. *These functions are documented on page* **??**.)

__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 \specials 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 \specials 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 \specials 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.

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

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

```
165        {
166            \cs_gset_eq:NN \__shipout_add_firstpage_specials:  \prg_do_nothing:
167            \cs_gset_eq:NN \__shipout_drop_firstpage_specials: \prg_do_nothing:
168        }
```

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).

```
169        {
170            \hbox_set:Nn \l__shipout_firstpage_box { \UseHook{shipout/firstpage} }
171        }
```

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.

```
172    \cs_gset_eq:NN \__shipout_run_firstpage_hook: \prg_do_nothing:
173    \cs_gset:Npn \__shipout_add_firstpage_material:Nn ##1 ##2 {
174        \@latex@warning{ First~ page~ is~ already~ shipped~ out,~ ignoring
175                            \MessageBreak \string##1 }
176      }
177  }
```

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

`\__shipout_add_firstpage_specials:`
`\__shipout_drop_firstpage_specials:`

The `\__shipout_add_firstpage_specials:` then adds the `\specials` 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 meta data declaration, etc. so by default we assume there is something to do.

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

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

```
179      \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.

```
180      \__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.

```
181      \cs_gset_eq:NN \__shipout_add_firstpage_specials: \prg_do_nothing:
182  }
```

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.

```
183  \cs_new:Npn \__shipout_drop_firstpage_specials: {
184      \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.

```
185      \cs_gset_eq:NN \__shipout_drop_firstpage_specials:  \prg_do_nothing:
186  }
```

(*End definition for* __shipout_add_firstpage_specials: *and* __shipout_drop_firstpage_specials:*.*)

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

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

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

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

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

(*End 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.

```
189 \cs_new:Npn \__shipout_add_firstpage_material:Nn #1#2 {
190     \AddToHook{shipout/firstpage}{#2}
191 }
```

(*End 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.*

```
192 \cs_new:Npn \__shipout_get_box_size:N #1 {
193     \dim_set:Nn \l_shipout_box_ht_dim { \box_ht:N #1 }
194     \dim_set:Nn \l_shipout_box_dp_dim { \box_dp:N #1 }
195     \dim_set:Nn \l_shipout_box_wd_dim { \box_wd:N #1 }
196     \dim_set:Nn \l_shipout_box_ht_plus_dp_dim
197         { \l_shipout_box_ht_dim + \l_shipout_box_dp_dim }
198 }
```

(*End definition for* __shipout_get_box_size:N*.*)

\l_shipout_box_ht_dim · And here are the variables set by __shipout_get_box_size:N.
\l_shipout_box_dp_dim
\l_shipout_box_wd_dim
\l_shipout_box_ht_plus_dp_dim

```
199 \dim_new:N \l_shipout_box_ht_dim
200 \dim_new:N \l_shipout_box_dp_dim
201 \dim_new:N \l_shipout_box_wd_dim
202 \dim_new:N \l_shipout_box_ht_plus_dp_dim
```

(*End 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

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

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

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

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

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

17

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.

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

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

```
208     \box_if_vertical:NTF \l_shipout_box
209       {
```

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

```
210         \tl_set:Nx \l__shipout_saved_badness_tl
211           { \vfuzz=\the\vfuzz\relax
212             \vbadness=\the\vbadness\relax }
213         \vfuzz=\c_max_dim
214         \vbadness=\c_max_int
```

Then we reconstruct \l_shipout_box . . .

```
215         \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
216           {
```

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

```
217             \hbox_set:Nn \l__shipout_tmp_box
218               { \l__shipout_saved_badness_tl #1 }
219             \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
220             \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
221             \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).

```
222             \skip_zero:N \baselineskip
223             \skip_zero:N \lineskip
224             \skip_zero:N \lineskiplimit
225             \box_use:N \l__shipout_tmp_box
226             \vbox_unpack:N \l_shipout_box
```

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

```
227             \kern \c_zero_dim
228           }
229         \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
230         \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.

```
231         \l__shipout_saved_badness_tl
232       }
233       {
```

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.

```
234         \box_if_horizontal:NT \l_shipout_box
235           {
236             \tl_set:Nx \l__shipout_saved_badness_tl
237               { \hfuzz=\the\hfuzz\relax
238                 \hbadness=\the\hbadness\relax }
239             \hfuzz=\c_max_dim
```

```
240        \hbadness=\c_max_int
241        \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
242           {
243              \hbox_set:Nn \l__shipout_tmp_box
244                 { \l__shipout_saved_badness_tl #1 }
245              \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
246              \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
247              \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
248              \box_move_up:nn
249                 \l_shipout_box_ht_dim
250                 { \box_use:N \l__shipout_tmp_box }
251              \hbox_unpack:N \l_shipout_box
252           }
253        \l__shipout_saved_badness_tl
254        }
255    }
256 }
```

(End 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.

```
257 \cs_new:Npn \__shipout_add_foreground_box:n #1
258 {
259    \box_if_vertical:NTF \l_shipout_box
260       {
261          \tl_set:Nx \l__shipout_saved_badness_tl
262             { \vfuzz=\the\vfuzz\relax
263                \vbadness=\the\vbadness\relax }
264          \vfuzz=\c_max_dim
265          \vbadness=\c_max_int
266          \vbox_set_to_ht:Nnn \l_shipout_box \l_shipout_box_ht_plus_dp_dim
267             {
268                \hbox_set:Nn \l__shipout_tmp_box
269                   { \l__shipout_saved_badness_tl #1 }
270                \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
271                \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
272                \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
273                \skip_zero:N \baselineskip
274                \skip_zero:N \lineskip
275                \skip_zero:N \lineskiplimit
276                \vbox_unpack:N \l_shipout_box
277                \kern -\l_shipout_box_ht_plus_dp_dim
278                \box_use:N \l__shipout_tmp_box
279                \kern  \l_shipout_box_ht_plus_dp_dim
280             }
281          \l__shipout_saved_badness_tl
282          \box_set_ht:Nn \l_shipout_box \l_shipout_box_ht_dim
283          \box_set_dp:Nn \l_shipout_box \l_shipout_box_dp_dim
284       }
285       {
286          \box_if_horizontal:NT \l_shipout_box
287             {
288                \tl_set:Nx \l__shipout_saved_badness_tl
```

```
289            { \hfuzz=\the\hfuzz\relax
290              \hbadness=\the\hbadness\relax }
291          \hfuzz=\c_max_dim
292          \hbadness=\c_max_int
293          \hbox_set_to_wd:Nnn \l_shipout_box \l_shipout_box_wd_dim
294            {
295              \hbox_unpack:N \l_shipout_box
296              \kern -\box_wd:N \l_shipout_box
297              \hbox_set:Nn \l__shipout_tmp_box
298                { \l__shipout_saved_badness_tl #1 }
299              \box_set_wd:Nn \l__shipout_tmp_box \c_zero_dim
300              \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
301              \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
302              \box_move_up:nn { \box_ht:N \l_shipout_box }
303                             { \box_use:N \l__shipout_tmp_box }
304              \kern \box_wd:N \l_shipout_box
305            }%
306            \l__shipout_saved_badness_tl
307          }
308        }
309 }
```

(*End definition for* __shipout_add_foreground_box:n.)

__shipout_init_page_origins:
\c__shipout_horigin_tl
\c__shipout_vorigin_tl

Two constants holding the offset of the top-left with respect to the media box.

Setting the constants this way is courtesy of Bruno.

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.

```
310 \cs_new:Npn \__shipout_init_page_origins: {
311   \tl_const:Nx \c__shipout_horigin_tl
312     {
313       \cs_if_exist_use:NTF \pdfvariable { horigin }
314         { \cs_if_exist_use:NF \pdfhorigin { 1in } }
315     }
316   \tl_const:Nx \c__shipout_vorigin_tl
317     {
318       \cs_if_exist_use:NTF \pdfvariable { vorigin }
319         { \cs_if_exist_use:NF \pdfvorigin { 1in } }
320     }
```

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.

```
321   \cs_gset_eq:NN \__shipout_init_page_origins: \prg_do_nothing:
322 }
```

(*End definition for* __shipout_init_page_origins:, \c__shipout_horigin_tl, *and* \c__shipout_-
vorigin_tl.)

__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.*

```
323 \cs_new:Npn \__shipout_picture_overlay:n #1 {
```

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

```
324        \__shipout_init_page_origins:
325        \kern -\c__shipout_horigin_tl \scan_stop:
326        \vbox_to_zero:n {
327          \kern -\c__shipout_vorigin_tl \scan_stop:
328          \unitlength 1pt \scan_stop:
```

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.

```
329          \hbox_set_to_wd:Nnn \l__shipout_tmp_box \c_zero_dim
330                            { \ignorespaces #1 \hss }
331          \box_set_ht:Nn \l__shipout_tmp_box \c_zero_dim
332          \box_set_dp:Nn \l__shipout_tmp_box \c_zero_dim
333          \box_use:N \l__shipout_tmp_box
334          \tex_vss:D
335        }
336      }
```

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

`\__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.

```
337 \cs_new:Npn \__shipout_add_background_picture:n #1 {
338    \__shipout_add_background_box:n { \__shipout_picture_overlay:n {#1} }
339 }
```

(*End 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.

```
340 \cs_new:Npn \__shipout_add_foreground_picture:n #1 {
341    \__shipout_add_foreground_box:n { \__shipout_picture_overlay:n {#1} }
342 }
```

(*End 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).

```
343 \cs_new_protected:Npn \shipout_discard: {
344    \bool_gset_true:N \g__shipout_discard_bool
345 }
```

(*End 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.

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

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

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

(*End 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.

```
348 \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.

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

(*End 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.

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

(*End definition for* \@abspage@last. *This function is documented on page* **??**.)

\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
```
352 \g@addto@macro \@kernel@after@enddocument {
353   \int_compare:nNnT \@abspage@last = \maxdimen
354     {
```

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

```
355       \xdef\@abspage@last{ \int_eval:n {\g_shipout_readonly_int + 1} }
356     }
357 }
```

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.

```
358    \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.

```
359    \int_compare:nNnF \g_shipout_readonly_int = 0
360      {
```

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!*

```
361      \if@filesw
362        \iow_now:Nx \@auxout {
363          \gdef\string\@abspage@last {\int_use:N \g_shipout_readonly_int}}
364      \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.

```
365      \bool_if:NTF \g__shipout_lastpage_handled_bool
366        {
```

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.

```
367        \int_compare:nNnF \@abspage@last = \g_shipout_readonly_int
368          {
369          \@latex@warning@no@line{Hook~ 'shipout/lastpage'~ executed~
370            on~ wrong~ page~ (\@abspage@last\space not~
371            \int_use:N\g_shipout_readonly_int).\MessageBreak
372            Rerun~ to~ correct~ this}%
373          }
374        }
375        {
```

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.

```
376        \bool_lazy_and:nnF
377          { \hook_if_empty_p:n {shipout/lastpage} }
378          { \tl_if_empty_p:N \@kernel@after@shipout@lastpage }
379          {
380          \tex_shipout:D\vbox to\textheight
381            {
382            \hbox:n { \UseHook{shipout/lastpage}
383              \@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.

```
384                      \__shipout_excuse_extra_page:
385                      \null
386                    }
```

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.

```
387                    \cs_gset_eq:NN \@extra@page@added \relax
388                  }
389              }
390          }
391 }
```

(*End definition for* \enddocument *,* \@kernel@after@enddocument *, and* \@kernel@after@enddocument@afterlastpage*. These functions are documented on page* **??***.*)

__shipout_excuse_extra_page: Say mea culpa . . .

```
392 \cs_new:Npn \__shipout_excuse_extra_page: {
393   \vfil
394   \begin{center}
395     \bfseries Temporary~ page!
396   \end{center}
397   \LaTeX{}~ was~ unable~ to~ guess~ the~ total~ number~ of~ pages~
398   correctly.~ ~ As~ there~ was~ some~ unprocessed~ data~ that~
399   should~ have~ been~ added~ to~ the~ final~ page~ this~ extra~
400   page~ has~ been~ added~ to~ receive~ it.
401   \par
402   If~ you~ rerun~ the~ document~ (without~ altering~ it)~ this~
403   surplus~ page~ will~ go~ away,~ because~ \LaTeX{}~ now~ knows~
404   how~ many~ pages~ to~ expect~ for~ this~ document.
405   \vfil
406 }
```

(*End definition for* __shipout_excuse_extra_page:*.*)

\PreviousTotalPages   In the preamble before the aux file was read \PreviousTotalPages is always zero.
\@kernel@before@begindocument

```
407 \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.

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

(*End 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.

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

(*End 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.

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

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

\DebugShipoutsOn
\DebugShipoutsOff

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

(*End 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.

```
416 ⟨@@=⟩
```

\@expl@@@shipout@add@firstpage@material@@Nn   Some internals needed elsewhere.
\@expl@@@shipout@add@background@box@@n
\@expl@@@shipout@add@foreground@box@@n
\@expl@@@shipout@add@background@picture@@n
\@expl@@@shipout@add@foreground@picture@@n

```
417 \cs_set_eq:NN \@expl@@@shipout@add@firstpage@material@@Nn
418                \__shipout_add_firstpage_material:Nn
419 \cs_set_eq:NN \@expl@@@shipout@add@background@box@@n
420                \__shipout_add_background_box:n
421 \cs_set_eq:NN \@expl@@@shipout@add@foreground@box@@n
422                \__shipout_add_foreground_box:n
423 \cs_set_eq:NN \@expl@@@shipout@add@background@picture@@n
424                \__shipout_add_background_picture:n
425 \cs_set_eq:NN \@expl@@@shipout@add@foreground@picture@@n
426                \__shipout_add_foreground_picture:n
```

(*End definition for* `\@expl@@@shipout@add@firstpage@material@@Nn` *and others. These functions are documented on page ??.*)

```
427 \ExplSyntaxOff
428 ⟨/2ekernel | latexrelease⟩
429 ⟨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.

```
430 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
431 ⟨latexrelease⟩                    {\shipout}{Hook management (shipout)}%
432 ⟨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!

```
433 ⟨latexrelease⟩\ifcsname tex_shipout:D\endcsname
434 ⟨latexrelease⟩\expandafter\let\expandafter\shipout
435 ⟨latexrelease⟩                    \csname tex_shipout:D\endcsname
436 ⟨latexrelease⟩\fi
437 ⟨latexrelease⟩
438 ⟨latexrelease⟩\let \RawShipout\@undefined
439 ⟨latexrelease⟩\let \ShipoutBox\@undefined
440 ⟨latexrelease⟩\let \ReadonlyShipoutCounter \@undefined
441 ⟨latexrelease⟩\let \c@totalpages \@undefined
442 ⟨latexrelease⟩\let \thetotalpages \@undefined
443 ⟨latexrelease⟩
444 ⟨latexrelease⟩\let \DiscardShipoutBox \@undefined
445 ⟨latexrelease⟩\let \DebugShipoutsOn \@undefined
446 ⟨latexrelease⟩\let \DebugShipoutsOff \@undefined
447 ⟨latexrelease⟩
448 ⟨latexrelease⟩\DeclareRobustCommand \AtBeginDvi [1]{%
449 ⟨latexrelease⟩  \global \setbox \@begindvibox
450 ⟨latexrelease⟩     \vbox{\unvbox \@begindvibox #1}%
451 ⟨latexrelease⟩}
452 ⟨latexrelease⟩
453 ⟨latexrelease⟩\let \AtBeginShipout \@undefined
454 ⟨latexrelease⟩\let \AtBeginShipoutNext \@undefined
455 ⟨latexrelease⟩
456 ⟨latexrelease⟩\let \AtBeginShipoutFirst \@undefined
457 ⟨latexrelease⟩
458 ⟨latexrelease⟩\let \ShipoutBoxHeight \@undefined
459 ⟨latexrelease⟩\let \ShipoutBoxDepth \@undefined
460 ⟨latexrelease⟩\let \ShipoutBoxWidth \@undefined
461 ⟨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).

```
462 ⟨latexrelease⟩
463 ⟨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).

```
464 %\reenable@package@load{atenddvi}
```

```
465 ⟨latexrelease⟩
466 ⟨latexrelease⟩\EndIncludeInRelease
467 ⟨∗2ekernel⟩
```

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.

```
468 ⟨/2ekernel⟩
469 ⟨*2ekernel | latexrelease⟩
470 ⟨latexrelease⟩\IncludeInRelease{2020/10/01}%
471 ⟨latexrelease⟩                    {\AtEndDvi}{atenddvi emulation}%
472 \ExplSyntaxOn
473 \cs_new_protected:Npn \AtEndDvi #1 {\AddToHook{shipout/lastpage}{#1}}
474 \ExplSyntaxOff
```

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

```
475 \disable@package@load{atenddvi}
476   {\PackageWarning{atenddvi}
477     {Functionality of this package is already\MessageBreak
478      provided by LaTeX.\MessageBreak\MessageBreak
479      It is there no longer necessary to load it\MessageBreak
480      and you can safely remove it.\MessageBreak
481      Found on}}
482 ⟨/2ekernel | latexrelease⟩

483 ⟨latexrelease⟩\EndIncludeInRelease
484 ⟨latexrelease⟩\IncludeInRelease{0000/00/00}%
485 ⟨latexrelease⟩                    {\AtEndDvi}{atenddvi emulation}%
486 ⟨latexrelease⟩\let \AtEndDvi \@undefined
487 ⟨latexrelease⟩\EndIncludeInRelease
488 ⟨*2ekernel⟩
```

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

```
489 ⟨/2ekernel⟩
```

6.2 Package **atbegshi** emulation

```
490 ⟨*atbegshi-ltx⟩
491 \ProvidesPackage{atbegshi-ltx}
492    [2021/01/10 v1.0c
493      Emulation of the original atbegshi^^Jpackage with kernel methods]
```

\AtBeginShipoutBox

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

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

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

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

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

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

(*End 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.

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

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

`\AtBeginShipoutDiscard` Just a different name.

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

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

`\AtBeginShipoutAddToBox`
`\AtBeginShipoutAddToBoxForeground`
`\AtBeginShipoutUpperLeft`
`\AtBeginShipoutUpperLeftForeground`

We don't expose them.

```
501 \let \AtBeginShipoutAddToBox
502              \@expl@@@shipout@add@background@box@@n
503 \let \AtBeginShipoutAddToBoxForeground
504              \@expl@@@shipout@add@foreground@box@@n
505 \let \AtBeginShipoutUpperLeft
506              \@expl@@@shipout@add@background@picture@@n
507 \let \AtBeginShipoutUpperLeftForeground
508              \@expl@@@shipout@add@foreground@picture@@n
```

(*End 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 `\special`s 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.

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

(*End 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.

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

(*End definition for* `\ShipoutBoxHeight`, `\ShipoutBoxWidth`, *and* `\ShipoutBoxDepth`. *These functions are documented on page* **??**.)

```
515 ⟨/atbegshi-ltx⟩
```

If the package is requested we substitute the one above:

```
516 ⟨∗2ekernel⟩
517 \declare@file@substitution{atbegshi.sty}{atbegshi-ltx.sty}
518 ⟨/2ekernel⟩
```

6.3   Package **everyshi** emulation

This is now directly handled in that package so emulation is not necessary any more.

Rather important :-)

₅₁₉ ⟨@@=⟩

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.

30

31