

# The `ltshipout` documentation\*

Frank Mittelbach, L<sup>A</sup>T<sub>E</sub>X Project Team

October 31, 2024

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overloading the <code>\shipout</code> primitive	2
1.2	Provided hooks	3
1.3	Legacy L <sup>A</sup> T <sub>E</sub> X commands	4
1.4	Special commands for use inside the hooks	5
1.5	Provided LuaT <sub>E</sub> X callbacks	5
1.6	Information counters	6
1.7	Debugging shipout code	6
<b>2</b>	<b>Emulating commands from other packages</b>	<b>7</b>
2.1	Emulating <code>atbegshi</code>	7
2.2	Emulating <code>everyshi</code>	8
2.3	Emulating <code>atenddvi</code>	8
2.4	Emulating <code>everypage</code>	8
	<b>Index</b>	<b>9</b>

## 1 Introduction

The code provides an interface to the `\shipout` primitive of T<sub>E</sub>X 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.<sup>1</sup>

---

\*This file has version v1.0n dated 2024/10/22, © L<sup>A</sup>T<sub>E</sub>X Project.

<sup>1</sup>Heiko’s interfaces are emulated by the kernel code, if a document requests his packages, so older documents will continue to work.

## 1.1 Overloading the `\shipout` primitive

---

`\shipout` With this implementation T<sub>E</sub>X’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 L<sup>A</sup>T<sub>E</sub>X 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 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).

---

`\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 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> names).<sup>2</sup> These variables can be used inside the hook code for `shipout/before`, `shipout/foreground` and `shipout/background` if needed.

---

<sup>2</sup>Might need changing, but HO’s version as strings is not really helpful I think).

## 1.2 Provided hooks

---

<code>shipout/before</code>	The code for <code>\shipout</code> offers a number of hooks into which packages (or the user) can add code to support different use cases. These are:
<code>shipout/after</code>	
<code>shipout/foreground</code>	
<code>shipout/background</code>	
<code>shipout/firstpage</code>	
<code>shipout/lastpage</code>	

---

**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!

**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.<sup>3</sup>

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

---

<sup>3</sup>In  $\text{\LaTeX}2_{\epsilon}$  that was already existing, but implemented using a box register with the name `\@begindivibox`.

**shipout/lastpage** The corresponding hook to add `\specials` 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:

<code>shipout/firstpage</code>	only on the first page
<code>shipout/background</code>	
<code>\boxed content of \ShipoutBox</code>	
<code>shipout/foreground</code>	
<code>shipout/lastpage</code>	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 `\typeouts`, 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.

### 1.3 Legacy $\LaTeX$ commands

---

`\AtBeginDvi` `\AtBeginDvi` `{(code)}`

`\AtEndDvi`

`\AtBeginDvi` is the existing  $\LaTeX 2_\epsilon$  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

---

<code>\DiscardShipoutBox</code>	<code>\AddToHookNext {shipout/before} {...\DiscardShipoutBox...}</code>
<code>\shipout_discard:</code>	

---

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`.<sup>4</sup> If `atbegshi` gets loaded then those commands become available as public functions with their original names as given below.

## 1.5 Provided Lua $\TeX$ callbacks

---

<code>pre_shipout_filter</code>	Under Lua $\TeX$ the <code>pre_shipout_filter</code> Lua callback is provided which gets called directly after the <code>shipout</code> hook, immediately before the <code>shipout</code> 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`.

---

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

## 1.6 Information counters

---

<code>\ReadonlyShipoutCounter</code>	<code>\ifnum\ReadonlyShipoutCounter=...</code>
<code>\g_shipout_readonly_int</code>	<code>\int_use:N \g_shipout_readonly_int % expl3 usage</code>

---

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.

---

<code>totalpages</code>	<code>\arabic{totalpages}</code>
<code>\g_shipout_totalpages_int</code>	<code>\int_use:N \g_shipout_totalpage_int % expl3 usage</code>

---

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!

---

<code>\PreviousTotalPages</code>	<code>\PreviousTotalPages</code>
----------------------------------	----------------------------------

---

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

---

<code>\DebugShipoutsOn</code>	<code>\DebugShipoutsOn</code>
<code>\DebugShipoutsOff</code>	Turn the debugging of shipout code on or off. This displays changes made to the shipout data structures.
<code>\shipout_debug_on:</code>	
<code>\shipout_debug_off:</code>	

---

*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`

---

<code>\AtBeginShipoutUpperLeft</code>	<code>\AddToHook {shipout/before} {...\AtBeginShipoutUpperLeft{&lt;code&gt;}...}</code>
<code>\AtBeginShipoutUpperLeftForeground</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.

---

<code>\AtBeginShipoutAddToBox</code>	<code>\AddToHook {shipout/before} {...\AtBeginShipoutAddToBox{&lt;code&gt;}...}</code>
<code>\AtBeginShipoutAddToBoxForeground</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.

---

<code>\AtBeginShipoutBox</code>	This is the name of the shipout box as <code>atbegshi</code> knows it.
---------------------------------	--

---

---

<code>\AtBeginShipoutOriginalShipout</code>
---

---

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` `\AtBeginShipout{<code>}`  $\equiv$  `\AddToHook{shipout/before}{<code>}`  
`\AtBeginShipoutNext` `\AtBeginShipoutNext{<code>}`  $\equiv$  `\AddToHookNext{shipout/before}{<code>}`

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

---

`\AtBeginShipoutFirst` The `atbegshi` names for `\AtBeginDvi` and `\DiscardShipoutBox`.  
`\AtBeginShipoutDiscard`

## 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>}`  $\equiv$  `\AddToHook{shipout/before}{<code>}`

---

`\AtNextShipout` `\AtNextShipout{<code>}`  $\equiv$  `\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>}`  $\equiv$   
`\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>}`  $\equiv$   
`\AddToHookNext{shipout/background}{\put(1in,-1in){<code>}}`

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



