The everypage package*

Sergio Callegari†

2020/10/17

WARNING

This package is now in *legacy status*. Functionality similar to that provided by this package has been directly implemented in LaTeX since its 2020 Fall release. Do not use everypage in new documents and do not rely on it in new packages or classes of yours.

When running on a pre-2020-10-01 version of LaTeX, everypage will now fall back to everypage-1x, its own past code base.

When running on a modern LaTeX, everypage will strive to provide its legacy interfaces by using the newer LaTeX facilities. However, full equivalence is not possible and breakage may occur. When truly needed, try loading everypage-1x in place of everypage to force usage of the old code base. This is expected to keep working for a few more LaTeX release cycles after Fall 2020.

Abstract

The everypage package was meant to extend LaTeX providing hooks to do actions on every page or on the current page. Currently, similar functionality is directly provided by LaTeX. Specifically, everypage supports actions performed *before* the page is shipped, so they can be used to put watermarks *in the background* of a page, or to set the page layout. The package is in many ways similar to bobhook by Karsten Tinnefeld, but it differs in the way in which the hooks are implemented. In some sense, it may also be related to package everyshi by Martin Schroeder, but again the implementation is different.

1 Introduction

Until a recent past, LaTeX provided no explicit hooks to be run as documents pages were finalized and output to the dvi or pdf file. Consequently, many solutions were developed to work around this limitation and to offer features (e.g., watermarks) that would

*This file (`everypage.dtx`) has version number 2.0b, last revised 2020/10/17.

†Sergio Callegari can be reached at `sergio.callegar at gmail dot com`

otherwise be impossible. These solution included packages such as everyshi, bobhook and watermark. Package everypage was a solution providing plumbing that could be used in higher level packages such as draftwatermark (watermarking) and flippdf (print preprocessing) to mention a couple of them.

All of these extensions relied on applying modifications to some LaTeX internals and as such were prone to subtle interactions with other packages and breakage. The situation has been cleared by the 2020 Fall LaTeX release where an official and generic support for hooks has been introduced.

As of today, everypage can be considered obsolete. It needs to remain around because older releases of LaTeX are still in use and legacy code exist that is based on the interfaces it exposes. However, no new document, class or package should be based on it. Furthermore, it can be expected that existing packages that use everypage will progressively learn to rely directly on the new LaTeX hook mechanism.

This manual is meant to aid the transition by showing how everypage is now being updated to cater both for older and newer LaTeX kernels. Specifically, it illustrates:

1. how everypage is now actually split in two packages: everypage-1x, providing the old code base; and everypage itself, that strives to implement the legacy interface on top of the new mechanisms offered by LaTeX;

2. how everypage can automatically import the old code base as needed, how loading of the latter can be forced (if absolutely needed or for comparison) and for how long the old code can be expected to keep working with newer releases of LaTeX;

3. how the legacy interface of everypage, once implemented/emulated on top of the new LaTeX facilities actually differs from its nominal behavior.

2 The original code base

The original implementation of everypage, now available as everypage-1x, adds two LaTeX hooks that get run when document pages are finalized and output to the dvi or pdf file. Specifically, one hook gets executed on every page, while the other is executed for the current page. Hook actions are are performed *before* the page is output on the medium, and this is important to be able to play with the page layout or to put things *behind* the page contents (e.g., watermarks).

2.1 User interface

`\AddEverypageHook` `\AddThispageHook`

The `\AddEverypageHook` command accepts one argument and adds it to the list of hooks that are run before every page is output. Similarly, the `\AddThispageHook` command accepts one argument, however it adds it to the list of hooks that are run before the *current* page is output.

The following rules apply:

1. once hooks are introduced and stacked in a series, there is no way to unstack them, nor to clear them;

2. in order to have hooks that get run only when specific conditions are met, conditionals must be included in the hooks;

3. there is no formatting inherent in the hooks. If one wants to put some watermark on a page, it is his own duty to put in the hook the code to place the watermark in the right position and with the appropriate appearance and style. When the hooks are run, the page is still empty, so that one begins filling it at point (1 inch, 1 inch) from the top left corner;

4. the hook code should *eat up no space*. This means that hooks meant to place some material on the page, need to have the material placed in a box with zero width and height beforehand.

5. no particular assumption is made on the LaTeX output driver, so everypage should work equally well with LaTeX, pdfLaTeX, LuaLaTeX, or XeLaTeX. Similarly, the package should work equally well with dvi, dvips, etc. output drivers. Obviously, the final compatibility with the different output drivers depends on the actual code that is placed in the hooks.

2.2 Comparison with other packages

The purpose of the original code base is better understood by comparing it to other packages that were meant to solve related problems at the time when LaTeX had no hook mechanism of its own:

- In comparison with bobhook by Karsten Tinnefeld, Lpackeverypage (in its legacy incarnation) used to differ in purpose and in the hook implementation. Package everypage was meant to make no assumption on what one could want to do on every page, whether to add text, images, or some low-level instruction for the output driver, or even to perform some accounting task. This made the package lighter and more flexible at the cost of being more difficult to use. In other words, everypage was meant to be more of a plumbing to be employed in higher level packages;

- in comparison with watermark by Alexander I. Rozhenko, similar considerations could me made. Specifically, watermark is explicitly targeted at making it easy to put watermarks on document pages, while everypage is lower level.

- in comparison to both bobhook and watermark, everypage used to employ a similar low level mechanism, by manipulation of the internal LaTeX macro `\@begindvi` to do the job. However, the redefinition of `\@begindvi` in everypage was postponed as much as possible, striving to avoid interference with other packages using `\AtBeginDvi` or anyway manipulating `\@begindvi`. Furthermore, everypage made no special assumption on the initial code that `\@begindvi` might contain.

- in comparison with everyshi by Martin Schroeder, everypage used to be similarly low level, but relied on changing the `\@begindvi` macro, rather than the even lower-level `\shipout` macro.

2.3 Known applications of the everypage code

Package everypage has historically found application in at least two higher level packages, namely:

- draftwatermark, meant at providing extended facilities for page watermarking on all LaTeX engines and output drivers; and
- flippdf, meant at catering for a frequent preprint requirement, where publisher may require a document with *mirrored* pages to get the best results out of a photographic printout process.

3 The new code base

The new code base differs from the old one because it does not touch any LaTeX internals. Conversely it relies on the hook mechanism that is officially provided by LaTeX since its 2020 fall release. Obviously, this has no other purpose than to provide back compatibility for older LaTeX code written before such LaTeX release and relying on the original everypage interfaces.

To this aim, the new code base does the following:

1. complains out loud, reminding that new code should not be based on everypage, but rather make direct usage of the new LaTeX interfaces;
2. checks whether the new LaTeX interfaces are actually available. If this is not the case, it resorts to loading everypage-1x that provides the old code base;
3. implements/emulates the `\AddEverypageHook` and `\AddThispageHook` commands on top of the new `\AddToHook` and `\AddToHookNext` LaTeX commands.

With specific reference to point 3 above, note that the new hook mechanism provided by LaTeX is extensively documented in issue 32 of *LaTeX News* and in file `lthooks-doc.pdf`. Furthermore, consider that everypage employs hooks in the *shipout* class, which are documented in file `ltshipout-doc.pdf`.

3.1 Compatibility of the new code with the original everypage interface

Because the implementation is different and due to choices (in fact, quite reasonable ones) by the LaTeX developers, the new implementation of everypage cannot be exactly equivalent to the original one (hence, please, do not open bugs for this!).

The main difference is the hook code provided by the user now gets wrapped in a `\put` command, inside a `picture` environment with `\unitlength` at 1 pt. This is because everypage relies on a `shipout/background` type hook. The `\put` command typesets material at (1 in, -1 in) to emulate the original coordinate system of everypage. This means that some of the points about the interface in section 2.1 do not apply anymore (or at least do not apply in the same way). Specifically, point 3 about lack of any pre-canned

formatting is not completely true anymore, because of the implicit picture environment. Furthermore, point 4 about the need not to eat up space can now be completely ignored.

While the changes described above seem to go in the direction of less rules and less concern, this might not be always true and subtle breakage of legacy code may happen in corner cases.

4 Forcing usage of the older code base

The usage of the older code base of everypage can be forced by simply substituting `\usepackage{everypage-1x}` for `\usepackage{everypage}`. In this case, no warning about the package obsolescence is emitted, because it is assumed that the user knows what he/she is doing. However, explicitly requiring everypage-1x is obviously discouraged.

The old code base actually works just fine with the Fall 2020 LaTeX release and it will probably keep to do so for a few more LaTeX release cycles. This is mostly up to the LaTeX developers and their will to maintain untouched some internal deprecated interfaces. In any case, `\usepackage{everypage-1x}` will eventually stop working and is now declared in an *unmaintained state*.

5 Implementation

5.1 Implementation of everypage

Announce the name and version of the package, which requires LaTeX 2_ε.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{everypage}%
  [2020/10/17 R2.0b Hooks to run on every page]
```

Complain out loud about the package being obsolete.

```
\PackageWarningNoLine{everypage}{%
 Functionality similar to this package has recently\MessageBreak
 been implemented in LaTeX. This package is now in\MessageBreak
 legacy status.\MessageBreak
 Please, don't use it in new documents and packages}
```

Depending on the actual functionalities provided by LaTeX consider loading everypage-1x. If so doing, warn about this too and stop. Otherwise give some more warnings.

```
\@ifundefined{AddToHook}{%
  \PackageWarningNoLine{everypage}{%
    You appear to be running a version of LaTeX\MessageBreak
    too old to provide the new functionality.\MessageBreak
    Forcing fallback to `everypage-1x' that\MessageBreak
    uses an older code base}
  \RequirePackage{everypage-1x}
  \endinput}{%
  \PackageWarningNoLine{everypage}{%
    You appear to be running a version of LaTeX\MessageBreak
```

```
    providing the new functionality.\MessageBreak
    Doing the best to deliver the original ‘everypage‘\MessageBreak
    interface on top of it. Strict equivalence is\MessageBreak
    not possible, breakage may occur.\MessageBreak
    If truly needed, Use ‘everypage-1x‘ to force the\MessageBreak
    loading of an older code base}}
```

\AddEverypageHook
\AddThispageHook

Finally, implement the everypage interface on top of the LaTeX `shipout/background` hooks.

```
\newcommand*{\AddEverypageHook}[1]{%
  \AddToHook{shipout/background}{\put(1in,-1in){#1}}}
\newcommand*{\AddThispageHook}[1]{%
  \AddToHookNext{shipout/background}{\put(1in,-1in){#1}}}
```

5.2 Implementation of everypage-1x

Announce the name and version of the package, which requires LaTeX 2_ε.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{everypage-1x}%
  [2020/10/10 1.2 Hooks to run on every page]
```

\sc@everypage@hook
\sc@everypage@hook

Define internal macros to hold the hooks and initialize them to contain nothing.

```
\newcommand{\sc@everypage@hook}{}
\newcommand{\sc@thispage@hook}{}
```

\AddEverypageHook
\AddThispageHook

Define the commands used to add the hooks.

```
\newcommand*{\AddEverypageHook}[1]{%
  \g@addto@macro\sc@everypage@hook{#1}}
\newcommand*{\AddThispageHook}[1]{%
  \g@addto@macro\sc@thispage@hook{#1}}
```

\sc@ep@init

The internal initialization code of the package. The package works by redefining the normal `\@outputpage` routine that takes care of outputting pages, so that the modified version first calls a special preamble, and then calls the original `\@outputpage` code and finally a postamble.

```
\newcommand*{\sc@ep@init}{%
  \let\sc@op@saved\@outputpage
  \def\@outputpage{%
    \sc@op@preamble
    \sc@op@saved
    \sc@op@postamble}}
```

\sc@op@preamble

The outputpage preamble contains instructions to redefine the `\@begindvi` macro that is called at every page output by the original `\@outputpage` code. Specifically: first the original `\@begindvi` is saved; then the hooks are called; then the hooks for the current page are cleared; eventually, the saved `\@begindvi` is called.

```
\newcommand*{\sc@op@preamble}{%
  \let\sc@begindvi\@begindvi
```

```
\def\@begindvi{%
  \sc@begindvi
  \sc@everypage@hook
  \sc@thispage@hook
  \gdef\sc@thispage@hook{}}
```

`\sc@op@postamble` The outputpage postamble simply restores the `\@begindvi` command to the saved value.

```
\newcommand*{\sc@op@postamble}{%
  \let\@begindvi\sc@begindvi}
```

Note that in exceptional cases this might not be the intended behaviour. For instance, consider situations where the `\@begindvi` is hacked by some other package to modify itself. By restoring the saved value, one might lose the modifications. This potential issue is currently under investigation.

As the very last thing, the `\AtBeginDocument` macro is called to insert the everypage initialization routine in the queue of commands to be executed when the actual document begins. In this way, the everypage initialization code is run *after* other packages are loaded.

```
\AtBeginDocument{\sc@ep@init}
```

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.