PDFmanagement (testphase) bundle

The LaTeX Project*

Version 0.95a, released 2021-02-22

Abstract

This is a temporary package created to allow the manual loading of the new LaTeX PDF management code during the test phase. It will disappear when the code is integrated into the LaTeX format

During the test phase the package should be used like this:

```
\RequirePackage{pdfmanagement-testphase} %load the package
\DeclareDocumentMetadata      %activate the PDF management
 {
   %options
 }
 \documentclass {...}
```


1 Introduction

The LaTeX format currently contains nearly no code specific to the now quite central output format, PDF. It also offers nearly no interfaces to important PDF related primitive commands for package writers.

Important tasks like supporting PDF standards, creating links, adding special colors, managing the content of central PDF-directories or even simple tasks like setting the PDF version are delegated to external packages which have to recourse to the primitive low-level commands in their code.

This is problematic for three reasons:

- At first using primitives directly can lead to clashes and duplicate settings with conflicting values—nothing prevent packages to add for example the `/Title` twice to the Info dictionary, the `/Lang` entry twice to the Catalog, or to add two `/ExtGState` resources to a page. The PDF normally doesn't break in such cases—the format is quite robust—but it will ignore one of the duplicates and the output can be wrong.
- At second the primitives differ between the various engines and backends with which LaTeX is used. To support the engines and backend packages have to write and maintain "driver" files which they did to a varying degree. This makes it difficult for users to assess if a package will work with their work-flow and is a strain for package writers as they have to keep track of engine and backend changes.

*E-mail: latex-team@latex-project.org

- And at last generic hooks and configuration points to various PDF related structures are missing and difficult to add.

Despite the potential problems, until now the number of conflicts were small and could be resolved in an ad-hoc fashion. But the future plans for LaTeX regarding support for tagged PDF and PDF standards mean that much more PDF specific code will have to be written by the kernel directly and this can not be done without proper, well-defined and well-behaving interfaces and hooks.

Some first steps for better support of PDF related commands have been already done with the l3pdf package which has now been integrated into l3kernel. It offers backend independent commands to create PDF objects and destination, to set the compress level and the PDF version.

The PDF management bundle extends this to more PDF related areas and provides interfaces to them in a backend independent way.

The new PDF management has three main objectives connected with the problems identified above:

- For commands with "clash potential" it implements commands to replace the primitives and so to resolve potential conflicts.
- It implements commands for a variety of PDF related tasks and supports a well-defined set of backends.
- If sensible this commands are enhanced by hooks from the new LaTeX hook system. This has been e.g. done for annotations in the l3pdfannot bundle.

2 "Change Strategy": The integration into LaTeX

The central module of this bundle, l3pdfmanagement, defines an interface for the (pdfTeX) primitives `\pdfcatalog`, `\pdfinfo`, `\pdfpagesattr`, `\pdfpagesattr` and `\pdfpageresources` and the analog commands from the other engines and backends.

All these commands have a "clash potential", this means that the new interface is incompatible with a parallel use of the primitive commands which it targets to replace and supersede. This doesn't affect many packages, but the list of package using such primitives contains central and important packages like hyperref, tikz, pdfx and more.

So while the goal is to integrate the code into the LaTeX format directly, this can not be done immediately without conflicts with existing documents and packages.

As an intermediary step the package pdfmanagement-testphase has been created which loads the code manually. With it package authors and users can test the new code, give feedback and packages can be adapted.

Loading the package will only *load* the modules, to *activate* the core PDF management the trigger command `\DeclareDocumentMetadata` has to be used too. The loading and activation has to be done *before* the `\documentclass` command.

We hope that this setup will allow packages writers and users to test the PDF management code and adapt packages and documents safely.

3 Backend support

The supported backends are pdflatex, lualatex, (x)dvipdfmx (latex, xelatex, dvilualatex (in texlive 2021)) and dvips with ps2pdf (not completely yet). dvips with distiller could work too but is untested.

That the interfaces and commands are backend independent doesn't mean that the results and even the compilation behavior is identical. The backends are too different to allow this. Some backends expand arguments e.g. in a `\special` while other don't. Some backends can insert a resource at the first compilation, while another uses the aux-file and a label and so needs at least two compilation runs. Some backends manage some of the resources through side-effects, some manage them automatically. All this mean that package writers will still have to keep an eye on backend requirements and run tests for all variants. Also backend specific code will still be needed in some cases.

4 Use

The package should be loaded before `\documentclass`. To activate the resource management it should be followed by `\DeclareDocumentMetadata{`⟨*key-val*⟩`}`. The options of `\DeclareDocumentMetadata` are described in the documentation of ltdocinit.

```
\RequirePackage{pdfmanagement-testphase} %load the package
\DeclareDocumentMetadata %activates the l3management interface
  {
     %options
  }
 \documentclass {...}
```

The PDF management can be deactivated either setting the key `pdfmanagement` to `false` or by commenting out the whole `\DeclareDocumentMetadata` declaration.

To test if the PDF management is active the predicate `\pdfmanagement_if_active:TF` can be used, see the documentation of l3pdfmanagement.

5 Requirements

The new PDF management requires a LaTeX format from 2020/10/01 or later and an L3 programming layer of 2021-02-18 or later. It currently depends on the experimental packages l3ref-tmp and l3bitset. In some places, e.g. when writing strings to the pdf it assumes that the file is utf8 encoded – ascii will naturally work too, but legacy 8bit encodings are not supported.

6 Modules

The bundle contains a number of modules. The organization and naming is bound to change over time: For almost all modules the goal is to integrate them into the format and the individual style will disappear.

The description items give the name of the documentation of the bundles. There doesn't exist in all cases a related `.sty`.

l3pdfdict This modules provides commands for PDF dictionaries. Its main purpose is to create name spaces. It is used e.g. by l3pdfmanagement and l3pdfannot.

l3pdfannot This module provides commands for annotations. Currently mainly link annotations, widget annotations will be added later. It doesn't require the PDF management to be active so it is possible to load and test it individually.

l3pdfmanagement This is the code code of the PDF management. It should not be loaded directly, but only as described in section 4.

ltdocinit This package provides the `\DeclareDocumentMetadata` command. It should not be loaded directly.

hyperref-generic This package provides a new generic hyperref driver. The driver will be loaded automatically by hyperref if the PDF management code is active. It should not be loaded directly.

l3backend-pdf-extra This module contains backend code needed by the PDF management. It will in due time be integrated into l3backend. It should not be loaded directly.

l3pdfmeta This module contains code to handle PDF standards. Currently it handles pdf/A and colorprofiles/outputintents. It is loaded by the bundle, and should not be loaded independently.

l3pdfxform Commands for form XObjects (xforms). The code is loaded by the bundle.

l3pdftool A number of commands like text conversion commands and bcd/emc. The commands will at some time be moved into the l3pdf module of l3kernel. It should not be loaded independently.

l3pdffile This module provides commands for to embed files. It is currently not loaded automatically, but it requires the PDF management.

pdfmanagement-firstaid This module provides a number of patches for external incompatible packages. This patches will disappear as soon as the packages are natively compatible. It is loaded automatically.

7 Incompabilities

As described in section 2, if activated the new PDF management takes over the management of core PDF dictionaries. All packages that bypass the PDF management and access these dictionaries with primitives like `\pdfcatalog`, `\pdfinfo`, `\pdfpageresources`, `\pdfpagesattr` and `\pdfpageattr` or similar commands from other engines and backends are basically incompatible: values can get lost or will be wrong.

The following describes known incompatible packages along with some suggestions how this should or will be handled in future. The list is not exhaustive.

7.1 hyperref

A generic driver that can be used as replacement has been developed and is provided by this bundle. It will be loaded automatically if the pdf management is active.

The generic driver differs in some points from other hyperref drivers:

- The code for bookmarks has been removed from this driver, instead the bookmarks is loaded and used.
- The driver isn't yet fully integrated into hyperref. This means that it doesn't react to a number of package options. Instead `\hypersetup` should be used.
- Incomplete is the support for form fields. Quite probably form fields will be extracted in a dedicated package.
- The driver uses for the color handling the l3color package. While normally it should be able to use colors defined with color and xcolor, there could be edge cases where it fails.
- The colors have been changed (this counts probably as an improvement ...).

More details can be found in the documentation hyperref-generic.pdf.

7.2 pdfx

pdfx is not compatible. It uses the commands `\pdfpagesattr`, `\pdfpageattr`, `\pdfinfo` and `\pdfcatalog`. The needed changes are not many, but can not be done by external patches.

It is also one goal of the pdfmanagement project to offer support for standards natively. The code is under development, see the documentation of l3pdfmeta.

7.3 hyperxmp

hyperxmp uses `\pdfcatalog` to insert the `/MetaData` reference. This makes it incompatible, but adjusting this is even possible with external patches. hyperxmp also relies on some hyperref internals, so changes in hyperref must be coordinated.

Some patch code is provided by the bundle and loaded automatically, but it is not complete currently. Failures are e.g. possible with complicated author or title settings. The handling of dates isn't correct either yet. The patch code can be disabled by using `firstaidoff=hyperxmp` in `\DeclareDocumentMetadata`

7.4 tikz/pgf

pgf writes to the page resources too and so is incompatible. The needed changes are rather small and will be done in coordination with the maintainer. Until this works, pagemanagement will load the patches automatically. This can be disabled by using `firstaidoff=pgf` in `\DeclareDocumentMetadata`

7.5 transparent

The package transparent is incompatible. A replacement has been written (transparent-ltx) and is loaded automatically. It requires a very recent L3 programming layer! This can be disabled by using `firstaidoff=transparent` in `\DeclareDocumentMetadata`

7.6 pdflscape

The package pdflscape is incompatible. A replacement has been written (pdflscape-ltx) and is loaded automatically. This can be disabled by using `firstaidoff=pdflscape` in `\DeclareDocumentMetadata`

7.7 colorspace

The package is incompatible. Some patches have been added to pdfmanagement-firstaid. Alternative code for spot colors is in the l3color package which has now been add to l3kernel.

7.8 embedfile, attachfile, attachfile2

Tools needed to be able to write a replacement to replace these packages have been developed in the l3pdffile package. Full replacements for the packages don't exist yet.

7.9 tagpdf

The development code is compatible and will be uploaded in time.

7.10 ocgx2, animate, media9

These package all make use of low-level PDF command and will have to be reviewed.

7.11 acrotex

The acrotex makes heavy use of PDF commands and so must be reviewed and adapted, including the currently untested route dvips + distiller.

7.12 fancytooltips

This package uses `\pdfpageattr` and acrotex and so must be reviewed.

8 Implementation

```
⟨@@=pdf⟩
⟨*package⟩
\ProvidesExplPackage {pdfmanagement-testphase} {2021-02-22} {0.95a}
  {experimental PDF resource management}
\providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
\IfFormatAtLeastTF{2020-10-01}{}{
 \PackageWarning{pdfmanagement-testphase}
   {This~package~needs~LaTeX~2020-10-01~or~newer.
    \MessageBreak Loading~is~aborted.}{}
 \DeclareOption { debug }{}
 \newcommand\DeclareDocumentMetadata[1]{}
 \ProcessOptions\relax
 }
\IfFormatAtLeastTF{2020-10-01}{}{\endinput}

\DeclareOption { debug }
```

```
  {
    \msg_redirect_module:nnn { pdf } { none } { warning }
  }

\ProcessOptions\relax
```
⟨/package⟩

8.1 Loading the core files.

This loads the core files. The backend should not be loaded to allow to set it in the document.

⟨*package⟩
```
\RequirePackage{l3pdfdict}       % needed by l3pdfmanagement
\RequirePackage{l3pdfmanagement} % loads the core code with the boolean
\RequirePackage{ltdocinit}       % DeclareDocumentMetadata,
% can perhaps be combined or made optional ...
\RequirePackage{l3pdfannot}
\RequirePackage{l3pdfxform-beta}
\RequirePackage{l3pdfmeta}       %
\RequirePackage{l3pdftools}

```
⟨/package⟩

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.