The LaTeX PDF management bundle

The LaTeX Project*

Version 0.96z, released 2026-04-15

Abstract

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

The PDF management code is loaded automatically if you use `\DocumentMetadata` before `\documentclass`.

```
\DocumentMetadata      % loads also the PDF management
    {
      % options, see documentmetadata-support.pdf
    }

\documentclass {...}
```

It is also possible to load the PDF as package, e.g. in a class.

```
\RequirePackage{pdfmanagement}
```

If the PDF management has already been loaded, e.g. in `\DocumentMetadata` this will do nothing. The package has one option, `backend` that allows to set a special backend like `dvipdfmx`. It will do nothing if the backend code has already been loaded.

Other options, e.g., the PDF standard or the PDF version should then be set with a `\SetKeys` command:

```
\RequirePackage{pdfmanagement}
\SetKeys[document/metadata]
  {
    pdfversion=2.0,
    lang=de,
    xmp=false,
    ...
  }
```

Note that PDF management should be loaded as early as possible.

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

Feedback wanted!

Bug reports and feedback are welcome. Please open an issue at https://github.com/latex3/pdfresources.

While the code targets PDF as output format, feedback about the effect on other formats is needed too.

1 Introduction

The LaTeX format contained for a long time nearly no code specific to the now quite central output format, PDF. It also offered 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 were delegated to external packages which had to recourse to the primitive low-level commands in their code.

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

Despite the potential problems, the number of conflicts were small and could be resolved in an ad-hoc fashion. But the plans for LaTeX regarding support for tagged PDF and PDF standards mean that much more PDF specific code has 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 done with the l3pdf package which has then been integrated as a module 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 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 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.

The code is there not loaded by the kernel but only if the trigger command `\DocumentMetadata` is used, or if the package `pdfmanagement` is loaded manually.

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 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 PDF management should be loaded as early as possible. When using `\DocumentMetadata`, various PDF related options can be set in the key-val argument. The options of `\DocumentMetadata` are described in the documentation of ltdocinit and in documentmetadata-support.

```
\DocumentMetadata % activates the PDF management interface
    {
       %options
    }
\documentclass {...}
```

When loading the PDF management as package, options can be set with `\SetKeys`:

```
\RequirePackage{pdfmanagement} %loads the PDF management interface
\SetKeys[document/metadata]{
       %options
    }
\documentclass {...}
```

To test if the PDF management has been loaded `\IfPDFManagementActiveTF` can be used.

5 Requirements

The new PDF management is developed parallel to the LaTeX format and should be updated together with the format. 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 majority of the modules don't have a stand alone `sty`, their code is combined in one file and loaded by the main package. 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 files to disappear.

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

l3pdfdict This module provides commands for PDF dictionaries. Its main purpose is to create name spaces. The code 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.

l3pdfmanagement This is the core code of the PDF management.

ltdocinit This module provides the `\DocumentMetadata` command.

hyperref-generic This module provides a new generic hyperref driver. The driver will be loaded automatically by hyperref if the PDF management code is active.

l3backend-pdfmanagement This module contains backend code needed by the PDF management. It will in due time be integrated into l3backend.

l3pdfmeta This module contains code to handle PDF standards. Currently it handles pdf/A and colorprofiles/outputintents.

l3pdfxform Commands for form XObjects (xforms).

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.

l3pdffile This module provides commands for to embed files.

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.

l3pdffield Commands for form fields. Currently it only provides commands for checkboxes. It must be loaded explicitly as with `\usepackage{l3pdffield}`.

7 Incompabilities

As described in section 2, 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.

The PDF management offers support for standards natively. It also writes XMP-metadata See the documentation of l3pdfmeta for details.

7.3 hyperxmp

hyperxmp uses `\pdfcatalog` to insert the `/MetaData` reference and also relies on some hyperref internals which are not present in the new generic driver used by hyperref when the pdfmanagement is active. This makes hyperxmp incompatible.

For some time some patch code was provided by the bundle to keep hyperxmp working but starting with version 0.95s XMP-metadata are handled directly by the pdfmanagement bundle (see the documentation of l3pdfmeta) and the patch code has been removed and the loading of hyperxmp has been disabled.

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, the PDF management will load the patches automatically. This can be disabled by using `debug={firstaidoff=pgf}` in `\DocumentMetadata`

7.5 transparent

The package transparent is compatible.

7.6 pdflscape

The package pdflscape is compatible.

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 added 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 ocgx2, animate, media9

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

7.10 acrotex

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

7.11 fancytooltips

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

7.12 bidi

The package bidi (loaded e.g. by polyglossia for right-to-left scripts) patches and redefines internals of hyperref and overwrites definitions of the generic driver. It is therefore not guarantied to be compatible with the PDF management if hyperref is used.

8 Implementation

```
⟨@@=pdfmanagement⟩
⟨*headinit⟩
\ProvidesExplPackage{pdfmanagement-init}{2026-04-15}{0.96z}
  {LaTeX PDF management bundle}

\DeclareOption { debug }
  {
    \msg_redirect_module:nnn { pdf } { none } { warning }
  }

\ProcessOptions\relax
⟨/headinit⟩
⟨*headtestphase⟩
\ProvidesExplPackage{pdfmanagement-testphase}{2026-04-15}{0.96z}
  {LaTeX PDF management bundle}

\DeclareOption { debug }
  {
    \msg_redirect_module:nnn { pdf } { none } { warning }
  }

\ProcessOptions\relax
⟨/headtestphase⟩
```

<*standalone>

```
\ProvidesExplPackage{pdfmanagement}{2026-04-15}{0.96z}
  {LaTeX PDF management bundle}
```

The only package option is for the backend. Other options can be set after loading the package with `\SetKeys[document/metadata]`

```
\DeclareKeys[pdfmanagement]
 {
   backend .code:n =
    {
      \str_if_exist:NTF \c_sys_backend_str
       {
        \PackageWarning{pdfmanagement}
         {
           backend~is~already~loaded.\MessageBreak
           'backend=#1'~ignored.
         }
       }
       {
         \tl_new:N  \l__pdfmanagement_backend_tl
         \tl_set:Nn \l__pdfmanagement_backend_tl {#1}
       }
```

```
    },
   backend .usage = load,
 }
\ProcessKeyOptions[pdfmanagement]

\IfPDFManagementActiveT
 {
   \endinput
 }

\RequirePackage{pdfmanagement-init}
% the backend key must be processed first.
%    \begin{macrocode}
\tl_if_exist:NTF \l__pdfmanagement_backend_tl
  {
    \exp_args:No \sys_load_backend:n { \l__pdfmanagement_backend_tl }
  }
  {
    \sys_ensure_backend:
  }
\file_input:n {l3backend-pdf-\c_sys_backend_str.def}
\hook_use_once:n {pdfmanagement/add}
```

firstaid should be loaded only after the backend has been set as it can contain backend dependant code.

```
\RequirePackage{pdfmanagement-firstaid}
```

⟨/standalone⟩

8.1 Loading the core file

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

⟨∗header⟩

```
\ProvidesExplFile{pdfmanagement.ltx}{2026-04-15}{0.96z}
  {PDF~management~code}
```

⟨/header⟩

⟨∗package⟩

```
\RequirePackage{tagpdf-base}
\input{pdfmanagement.ltx}
```

These keys are also needed in documentmetadata-support, so need to be in the init-file:

```
\keys_define:nn { document / metadata }
  {
    ,pdfversion .code:n =
      {
        \pdf_version_gset:n { #1 }
        \AddToDocumentProperties[document]{pdfversion}{#1}
      }
    ,uncompress .code:n =
      {
        \pdf_uncompress:
      }
```

```
    ,uncompress .value_forbidden:n = true
    ,lang .code:n =
      {
        \pdfmanagement_add:nnn {Catalog} {Lang}{(#1)}
        \AddToDocumentProperties[document]{lang}{#1}
      }
    ,pdfstandard .code:n =
      {
        \clist_map_inline:nn{#1}
         {
           \keys_set:ne {document / metadata} {_pdfstandard=\str_uppercase:n{##1}}
         }
      }
  }
⟨/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.