

spotxcolor Package

Munehiro Yamamoto

2026/03/29 v1.7

Abstract

This package provides robust spot color (e.g., DIC, PANTONE) support for the `xcolor` package across all major \TeX engines. It resolves structural PDF issues found in legacy packages and provides explicit fallback mechanisms for various drivers.

1 Motivation & Features

While the legacy `spotcolor` package has been used for years, it suffers from structural PDF compatibility issues with modern `expl3` and `xcolor` updates. The `colorspace` package is an excellent modern alternative, but it lacks full support for the `dvipdfmx` driver. To address these gaps, the `spotxcolor` package natively hooks into the `xcolor` package to provide a universal, print-safe solution. Its core features include:

- **Native `xcolor` Integration:** Use spot colors exactly like standard colors (e.g., `\textcolor`, `\pagecolor`), generating perfect PDF structures for `pdftex` and `luatex`.
- **Universal Engine Support & Safe Fallbacks:** Ensures safe CMYK fallback for `dvipdfmx` and `xetex` when using standard macros, while providing explicit injection commands (`\SpotColor`) for true spot color output in `dvipdfmx`.
- **Advanced Graphics (TikZ/PGF) Support:** Internally patches PGF driver macros and pattern primitives to emit true spot color PDF operators. It supports fill/stroke separation, fadings, blend modes, and uncolored patterns (`patterns` and `patterns.meta`). Spot color tints are auto-detected from CMYK values throughout the entire PGF rendering pipeline.
- **Seamless Decorations:** Works flawlessly with `colortbl` (zebra-striped tables) and `tcolorbox` (frames, backgrounds, shadows).
- **Complete Backward Compatibility:** Perfectly emulates user commands from both `spotcolor` and `colorspace` packages.

2 Requirements

This package requires a modern \TeX Live environment (with an up-to-date `expl3/l3kernel`), alongside the `xcolor` and `iftex` packages.

3 Loading the `spotxcolor` Package

Load the `spotxcolor` package in your preamble. It automatically detects your engine via the `iftex` package.

```
\usepackage{spotxcolor}
```

4 Usage

4.1 Defining and Using Spot Colors

Use `\definespotcolor` to register a spot color. The arguments are: \LaTeX name, PDF name, and CMYK values. Once registered, you can use standard `xcolor` commands:

```
\definespotcolor{DIC161s}{DIC 161s*}{0, 0.64, 1, 0}
```

```
\textcolor{DIC161s}{This text is DIC 161.}
```

PANTONE colors work the same way.

```
% PANTONE Metallic Coated (true spot color)
```

```
\definespotcolor{PANTONE876C}{PANTONE 876 C}{0.4467, 0.7463, 0.9732, 0}
```

```
\textcolor{PANTONE876C}{This text is PANTONE 876 C (Metallic Gold).}
```

```
\textcolor{PANTONE876C!50}{50% tint of PANTONE 876 C.}
```

The PDF Separation array will contain `/PANTONE#20876#20C`, matching the encoding produced by Adobe Illustrator.

Live example: **PANTONE 876 C (100%)** 50% 25%

4.2 `Dvipdfmx` Explicit Spot Colors

All standard macros (`\color`, `\textcolor`, `\pagecolor`) produce true spot color output on `dvipdfmx/xetex` as well, via automatic `\special{pdf:code}` injection.

The `\SpotColor` command remains available as a lightweight, low-level alternative for direct PDF literal injection:

```
\SpotColor{DIC161s}{1.0}
This text is 100\% DIC 161.
```

```
\SpotColor{DIC161s}{0.5}
This text is 50\% DIC 161.
```

4.3 Note for (u)p \TeX Users

If you compile your document with `platex` or `uplatex` and generate a PDF via `dvipdfmx`, you **must** specify the `dvipdfmx` driver option globally in your document class:

```
\documentclass[dvipdfmx]{article}
```

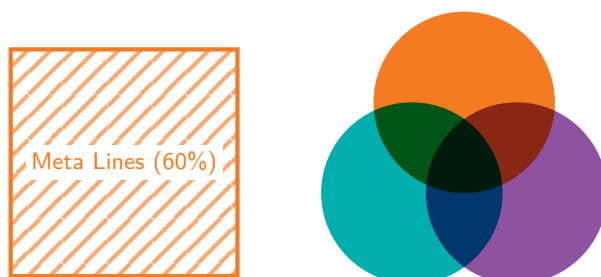
Without this option, modern `expl3` and `xcolor` will default to the `dvips` driver, and the PDF objects for spot colors will not be generated correctly.

5 Examples in Advanced Graphics

The true power of the `spotxcolor` package lies in its flawless integration with modern graphics libraries. For the following examples, we have defined three spot colors forming a triadic color harmony: **DIC161s** (Vermilion), **DIC256s** (Green), and **DIC200s** (Purple).

5.1 TikZ: Patterns and Multiply Blend Mode

`spotxcolor` safely forces PGF's hardcoded RGB patterns into the CMYK color space to ensure print safety. It also flawlessly handles PDF blend modes (e.g., multiply) with spot colors.



5.2 tcolorbox Integration

Spot colors and their tints (e.g., `DIC200s!10`) can be used directly within complex `tcolorbox` environments, including frames, backgrounds, and drop shadows.

Triadic Harmony with tcolorbox

This box is styled entirely using **DIC 200s** and its tints. The frame is 100% spot color, the title background is 80%, the content background is 5%, and even the drop shadow is rendered safely using a 30% tint.

You can also mix text colors easily: **Vermilion Text** and **Green Text**.

6 Spot Color Dictionaries and Tools

6.1 Included Dictionary (`dic.def`)

For your convenience, the package includes `dic.def` in the `doc` directory, which contains definitions for the widely used DIC Color Guide in Japanese commercial printing. The CMYK values in this file are natively extracted from Adobe's official legacy definitions (`.acbl`), ensuring absolute print-accurate fallback values.

! IMPORTANT WARNING

Please do **not** load the entire `dic.def` file (or any other large dictionary) into your preamble using `\input`.

Calling `\definespotcolor` immediately generates PDF objects (ColorSpace dictionaries and Tint Transform Functions) to guarantee strict PDF specification compliance and robustness across all rendering engines.

Loading thousands of unused spot colors will bloat your PDF with unnecessary objects and resource aliases, which may cause PDF viewers (including Adobe Acrobat) to freeze or fail to render the document.

Instead, open `dic.def` in a text editor, copy **only** the specific `\definespotcolor` lines you actually need, and paste them into your document preamble.

6.2 Extraction Tools

If you need spot colors from other color books (e.g., PANTONE, TOYO, HKS), you can extract them from Adobe Illustrator's color book files using the provided Ruby scripts located in the `doc` directory:

- **`acbl2def.rb`**: Extracts native, perfect CMYK values from Adobe Color Book Legacy (`.acbl`) XML files. (*Highly Recommended*)
- **`ase2def.rb`**: Extracts CMYK values from Adobe Swatch Exchange (`.ase`) files exported directly from Illustrator.

- **acb2def.rb**: Parses modern binary Adobe Color Book (.acb) files. Note that many .acb files store colors in the Lab color space. This script applies a 3rd-Degree Polynomial Regression Model (trained on native Adobe data) to predict highly accurate CMYK fallback values while minimizing K-channel muddiness. While extremely close to native values, it remains an approximation. For exact manufacturer values, use **acb12def.rb** or **ase2def.rb**.

6.3 A Note on PANTONE Color Families

Not all PANTONE color books produce true spot colors in PDF. When using **acb2def.rb** to generate PANTONE definitions, be aware of the distinction:

PANTONE Metallic Coated, PANTONE Solid Coated/Uncoated, etc.

These represent actual spot inks that require a dedicated printing plate. Adobe Illustrator creates a `/Separation` color space for these colors, and they appear as true spot colors in the output preview. **acb2def.rb** converts their Lab values to CMYK via the built-in ML model to produce accurate fallback colors for on-screen display.

PANTONE+ CMYK Coated / PANTONE+ CMYK Uncoated These are *process color* formulations—CMYK recipes designed to approximate PANTONE colors using only the four standard process inks. Adobe Illustrator writes them as plain CMYK operators (e.g., `1 0.94 0 0.5 k`) without creating `/Separation` entries. While **acb2def.rb** can generate `.def` files for these families, the resulting colors will function as process colors in practice.

7 Backward Compatibility

spotxcolor allows you to safely reuse legacy code and dictionaries written for both the **spotcolor** and **colorspace** packages without raising errors.

7.1 Wrapper for **spotcolor**

The legacy **spotcolor** package defined colors using a three-step process. **spotxcolor** seamlessly maps these to its modern implementation:

```
% Legacy syntax:
\NewSpotColorSpace{DIC}
\AddSpotColor{DIC}{DIC161s}{DIC\SpotSpace 161s*}{0 0.64 1 0}
\SetPageColorSpace{DIC}

% spotxcolor interpretation:
% The \AddSpotColor macro safely acts as a wrapper for:
% \definespotcolor{DIC161s}{DIC 161s*}{0, 0.64, 1, 0}
```

Note that `\NewSpotColorSpace` and `\SetPageColorSpace` are now defined as dummy macros. In modern environments, `spotxcolor` globally and safely manages PDF resource dictionaries.

7.2 Wrapper for `colorspace`

The `colorspace` package required authors to manually manipulate page resource dictionaries using specific commands:

- `\pagecolorspace{<name>}`
- `\resetpagecolorspace`

`spotxcolor` defines both of these as **empty dummy macros** (they do nothing). Why? Because `spotxcolor` utilizes robust, modern techniques (like hooking into PGF’s resource management API) to inject color spaces globally and safely. Attempting to manually manipulate the `/ColorSpace` dictionary on a per-page basis is dangerous in modern L^AT_EX and can easily corrupt the PDF structure. By neutralizing these legacy commands, `spotxcolor` guarantees that older documents will compile perfectly and safely without any structural PDF errors.

8 Compatibility with `pdfmanagement (l3pdf)`

8.1 Background

Modern L^AT_EX (since 2020) provides an optional PDF resource management layer called *pdfmanagement*, activated by placing `\DocumentMetadata{}` before `\documentclass`. When `pdfmanagement` is active, it takes exclusive control of all PDF page resources—including the `/ColorSpace` dictionary—and direct access via `\pdfpagemresources` or `\special{pdf:put}` is forbidden.

Furthermore, `pdfmanagement` redefines PGF’s `\pgfutil@addpdfresource@colorspaces` with its own parser (`__pdfmanagement_patch_pgfcOLORSPACES:w`), which expects a different argument format. Passing the raw object references that `spotxcolor` produces (e.g., `42 0 R`) through this rewritten macro causes a “Runaway argument” error.

8.2 Automatic Detection and Dispatch

`spotxcolor` detects `pdfmanagement` at load time using `\IfPDFManagementActiveTF` and automatically switches its resource registration strategy:

Mode	Resource Registration
Traditional (no <code>pdfmanagement</code>)	<code>\pgfutil@addpdfresource@colorspaces</code> or <code>\pdfpagemresources / \special{pdf:put}</code>
<code>pdfmanagement</code> active	<code>\pdfmanagement_add:nne</code> <code>{Page/Resources/ColorSpace}</code>

This switch is transparent to the user. All existing `\definespotcolor`, `\color`, `\textcolor`, `\pagecolor`, and `TikZ` commands work identically in both modes.

8.3 Usage

To use `spotxcolor` with `pdfmanagement`, simply add `\DocumentMetadata{}` as the very first line of your document:

```
\DocumentMetadata{}      % activates pdfmanagement
\documentclass{article}
\usepackage{spotxcolor}
\definespotcolor{DIC161s}{DIC 161s*}{0, 0.64, 1, 0}
\begin{document}
\textcolor{DIC161s}{Spot color text}
\end{document}
```

No additional options or workarounds are required.

9 Limitations

The `spotxcolor` package auto-detects spot colors by checking whether CMYK values in the PDF output are a *proportional scalar multiple* of a registered spot color's base CMYK values. This approach covers the vast majority of practical use cases, but has inherent limitations in the following areas.

9.1 Non-Proportional Color Mixes

When a spot color is mixed with another color model using `xcolor`'s `!` operator, the resulting CMYK values may no longer be a simple scalar multiple of the base. In such cases, the output correctly falls back to CMYK representation.

```
% Proportional tint - auto-detected as spot color (tint=0.5)
\textcolor{DIC161s!50}{50% tint}
% Base CMYK: (0, 0.64, 1, 0)
% Computed:  (0, 0.32, 0.5, 0) ← scalar multiple (×0.5)

% Non-proportional mix - falls back to CMYK
\textcolor{DIC161s!80!black}{80% DIC161s mixed with black}
% Computed:  (0, 0.512, 0.8, 0.2) ← K 0 breaks proportionality

% Another non-proportional mix
\textcolor{DIC161s!50!DIC256s}{50% DIC161s mixed with DIC256s}
% Computed:  (0.45, 0.32, 0.7, 0) ← no single spot color match
```

True representation of such mixes would require `/DeviceN` color spaces (e.g., `[/DeviceN [/DIC~161s* /Black] ...]`), which is beyond the scope of this package.

9.2 Shading (Gradients)

PGF’s `\shade` command generates Shading dictionary objects (PDF Type 2 and Type 3 shadings) with a hardcoded `/DeviceRGB` color space. The color values are resolved and frozen into RGB at the time the shading object is created, before any content stream operators are emitted.

This contrasts with *patterns* (hatching), where the color is specified by operators in the content stream (`/pgfspot_DIC161s cs 0.6 /pgfpat21 scn`) and can be intercepted. For shadings, the content stream contains only `/Fm16 Do` (“paint this Form XObject”)—no color information to intercept.

```
% Both endpoints are converted to RGB inside the Shading object.
% The PDF will contain:
%   /ColorSpace /DeviceRGB
%   /Function << /C0 [r g b] /C1 [r g b] ... >>
\shade[left color=DIC161s, right color=DIC161s!10]
(0,0) rectangle (4, 1.5);
```

Since `spotxcolor`’s interception operates at the content stream operator level, it cannot modify these pre-built PDF objects. True spot color gradients would require generating Shading objects with `/ColorSpace [/Separation ...]` and tint-value function endpoints, which amounts to rewriting PGF’s shading generation code entirely.

9.3 Summary

Feature	Status	Notes
<code>\color</code> , <code>\textcolor</code>	Spot	All engines
<code>\pagecolor</code>	Spot	All engines
<code>\SpotColor</code>	Spot	Direct PDF literal
TikZ fill/stroke	Spot	Via PGF driver patch
Low-level PGF commands	Spot	Via PGF driver patch
Uncolored patterns	Spot	Via <code>[/Pattern [/Separation ...]]</code>
Fading / transparency	Spot	Fill color is spot
Opacity + blend modes	Spot	Fill color is spot
<code>colortbl</code> row colors	Spot	Via <code>\set@color</code> patch
<code>tcolorbox</code>	Spot	Frame, background, shadow
Proportional tints (<code>!N</code>)	Spot	Auto-detected
Non-proportional mixes (<code>!N!color</code>)	CMYK	Needs <code>/DeviceN</code>
Shading (gradients)	RGB	PGF hardcodes <code>/DeviceRGB</code>

10 Comparison with Other Spot Color Packages

Several L^AT_EX packages provide spot color support. This section compares them to help you choose the right tool for your workflow.

10.1 Package Overview

spotcolor (2006) The original spot color package, designed exclusively for pdfL^AT_EX.

It defines a custom xcolor color model (`spotcolor`) and provides the `\SpotColor` command for direct PDF literal injection via `\pdfliteral`. However, `\SpotColor` does not use the color stack, so spot color output does not survive page breaks. It only supports `pdftex` (no `luatex`, `dvipdfmx`, or `xetex` support) and has no PGF/TikZ integration.

xspotcolor (2014–2021) A spot color package for XeL^AT_EX and `dvipdfmx`. It provides basic PGF integration by defining custom PGF driver macros (`\pgfsys@color@spotcolor@fill`, etc.), but these only work when the user explicitly calls a spot color command—standard xcolor commands still produce CMYK. It does not support `pdftex` or `luatex`.

colorspace (2015–2019) A modern package supporting Separation and DeviceN color spaces. It hooks into xcolor’s internal color representation and provides PGF fill/stroke driver macros. It supports `pdftex` and `luatex`, but **not** `dvipdfmx` or XeL^AT_EX. It requires manual page resource management (`\pagecolorspace`, `\resetpagecolorspace`).

spotxcolor (2026) This package. It intercepts all color output paths—xcolor’s `\set@color`, PGF driver macros, and PGF pattern primitives—to transparently convert proportional spot color tints from CMYK to Separation operators. It supports all four major engines and requires no special commands beyond standard xcolor usage.

10.2 Feature Comparison

Feature	spotcolor	xespotcolor	colorspace	spotxcolor
pdftex	Yes	—	Yes	Yes
luatex	—	—	Yes	Yes
xetex	—	Yes	—	Yes
dvipdfmx	—	Yes	—	Yes
<code>\color</code> , <code>\textcolor</code> as spot	—	—	Yes	Yes
<code>\pagecolor</code> as spot	—	—	—	Yes
Color stack (page breaks)	—	—	Yes	Yes
Tinted expressions (<code>!50</code>)	—	—	Yes	Yes
PGF fill/stroke	—	Yes [†]	Yes	Yes
PGF pattern fills as spot	—	—	—	Yes
<code>colortbl</code> as spot	—	—	Yes [‡]	Yes
<code>tcolorbox</code> as spot	—	—	Yes [‡]	Yes
DeviceN	—	—	Yes	—

[†]: `xespotcolor` defines custom PGF driver macros, but they are only activated when the user explicitly calls spot color commands, not through standard `xcolor` usage.

[‡]: `colorspace` supports these through its `xcolor` integration, but only on `pdftex` and `luatex`.