spotxcolor Package

Munehiro Yamamoto

2026/03/17 v1.4

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.}
```

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)pLATEX 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}
```

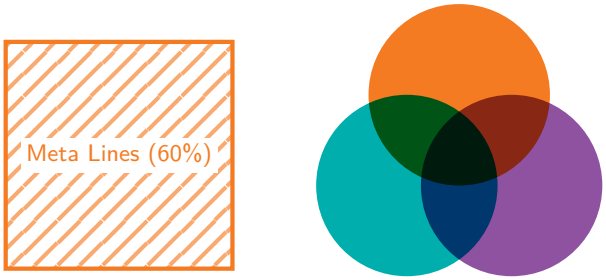
Without this option, modern expl3 and xcolor will default to the `dvips` driver, and the PDF objects for spot colors will not be generated correctly.

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 `acbl2def.rb` or `ase2def.rb`.

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 LaTeX 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 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.

8.1 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.

8.2 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 (`/pgfpspot_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.

8.3 Summary

Feature	Status	Notes
`\color`, `\textcolor`	**Spot**	All engines
`\pagecolor`	**Spot**	All engines
`\SpotColor`	**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 `[/Pattern [/Separation ...]]`
Fading / transparency	**Spot**	Fill color is spot
Opacity + blend modes	**Spot**	Fill color is spot
colortbl row colors	**Spot**	Via `\set@color` patch
tcolorbox	**Spot**	Frame, background, shadow
Proportional tints (`!N`)	**Spot**	Auto-detected
Non-proportional mixes (`!N!color`)	**CMYK**	Needs `/DeviceN`
Shading (gradients)	**RGB**	PGF hardcodes `/DeviceRGB`