Flutter
New ColorScheme Roles for Material 3

SUMMARY

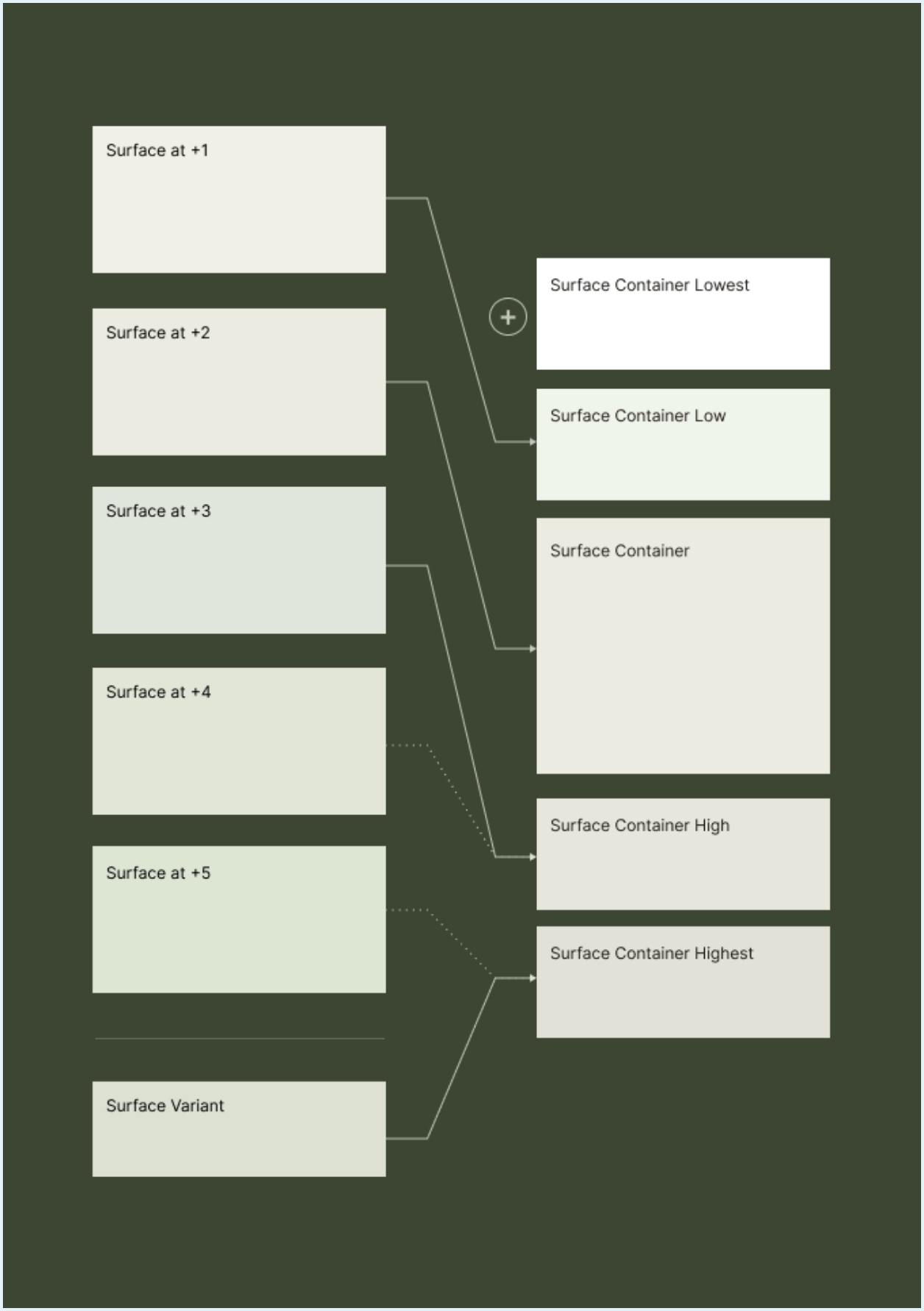
Proposal to add new ColorScheme roles to the Flutter framework.

Author: Qun Cheng (QuncCccccc)
Go Link: flutter.dev/go/{document-title}
Created: 12/2023 / **Last updated:** 12/2023

WHAT PROBLEM IS THIS SOLVING?

Provide support for the new `ColorScheme` roles based on the Material Design 3 guidelines.

The Material Design specification added tone-based surfaces, surface containers, and accent color roles. Tone-based surfaces and surface containers no longer tie to elevation and provide more flexibility. Therefore, these color roles help deprecate the use of surface tint color, which is hard to understand and used by few people. The following diagram shows what the migration looks like. For example, Surface Color with level 1 elevation maps to Surface Container low, Surface Color with level 2 elevation maps to Surface Container, and so on. Surface Colors with level 4 and 5 are not used in any widget defaults, but if there are any use cases for these two colors, Surface Container High and Surface Container Highest will be recommended to replace them.

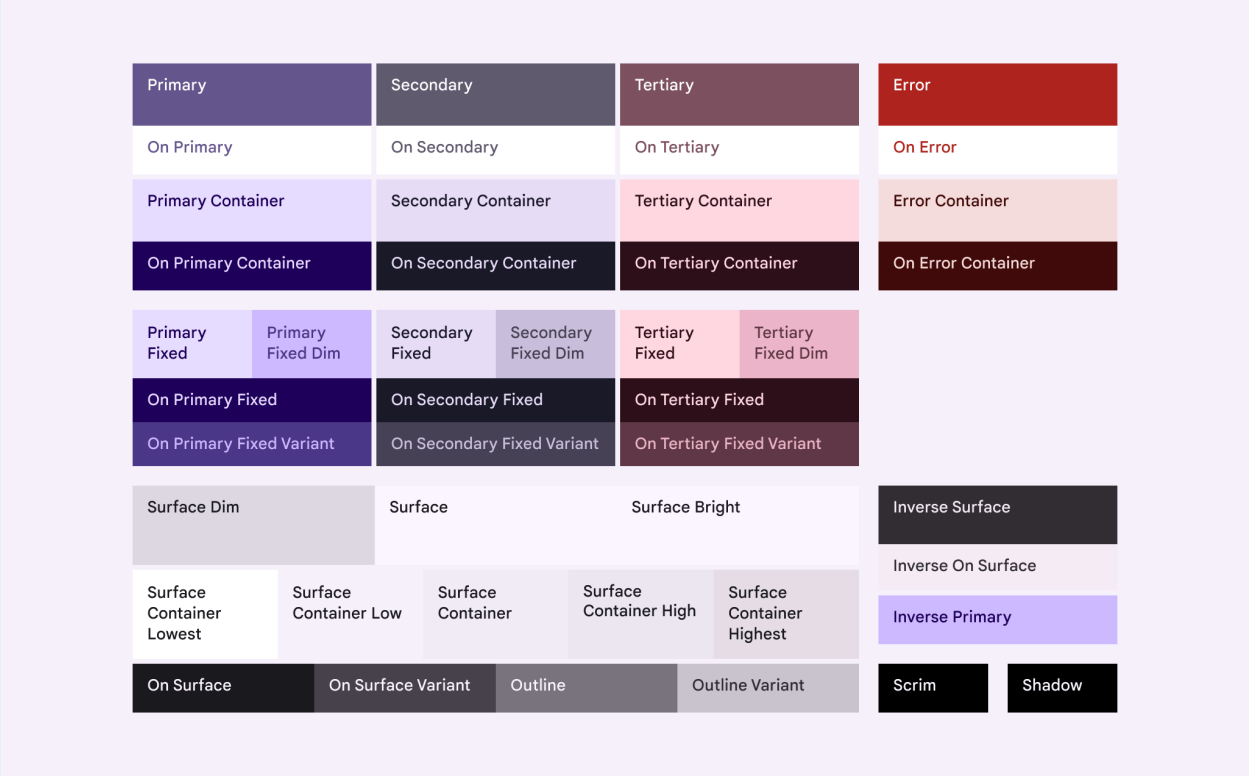


Secondly, the color token system has applied the new tone-based surfaces and surface containers to all the existing widgets and stopped using surface tint color. To keep in sync with Material Design, these new colors are needed.

Also, we are going to support more widgets created by Material Design, such as Carousel. The token system has disconnected with the deprecated tokens and keeps in sync with new roles. New widgets will also depend on the updated color scheme.

BACKGROUND

Material Design 3 added 19 new color roles to the Color Scheme and 3 colors will be deprecated.



The newly added colors can be divided into two groups:

New Surface roles:

- Surface Dim
- Surface Bright
- Surface Container Lowest
- Surface Container low
- Surface Container
- Surface Container High
- Surface Container Highest

The updated roles remove the use of opacity overlays. These updated roles are also no longer tied to elevation, and remove the use of surface tint, which is hard to understand and not easy to use.

Accent color add-ons:

- Primary Fixed
- Primary Fixed Dim
- On Primary Fixed
- On Primary Fixed Variant
- Secondary Fixed
- Secondary Fixed Dim
- On Secondary Fixed
- On Secondary Fixed Variant
- Tertiary Fixed
- Tertiary Fixed Dim
- On Tertiary Fixed
- On Tertiary Fixed Variant

These colors are add-on roles for accent colors, providing alignment with the Android team. In the past, Material and Android used two different sets of color roles (or tokens). These roles were inconsistent or conflicted across the sets in what colors they produced, how they were named, and how they were applied. By adding these extended accent color roles, the Material scheme can support all roles as a coherent set.

The 3 colors being removed:

- Background
- On Background
- Surface Variant

The Material Design 3 removes these 3 colors. Background should be replaced with Surface; On Background should be replaced with On Surface; Surface Variant should be migrated to Surface Container Highest.

Glossary

- **Surface Dim** - Surface Dim is the darkest surface color in both light and dark themes. While the default surface color automatically inverts between light and dark themes, the surface dim color keeps its relative brightness across both light and dark themes.
- **Surface Bright** - Surface Bright is the lightest surface color in both light and dark themes. Similar to Surface Dim, this color also does not invert its brightness between light and dark.
- **Fixed roles** - Roles with the "Fixed" keyword, such as primaryFixed, stay the same color between light and dark themes, as appropriate in places where this behavior is desired.

OVERVIEW

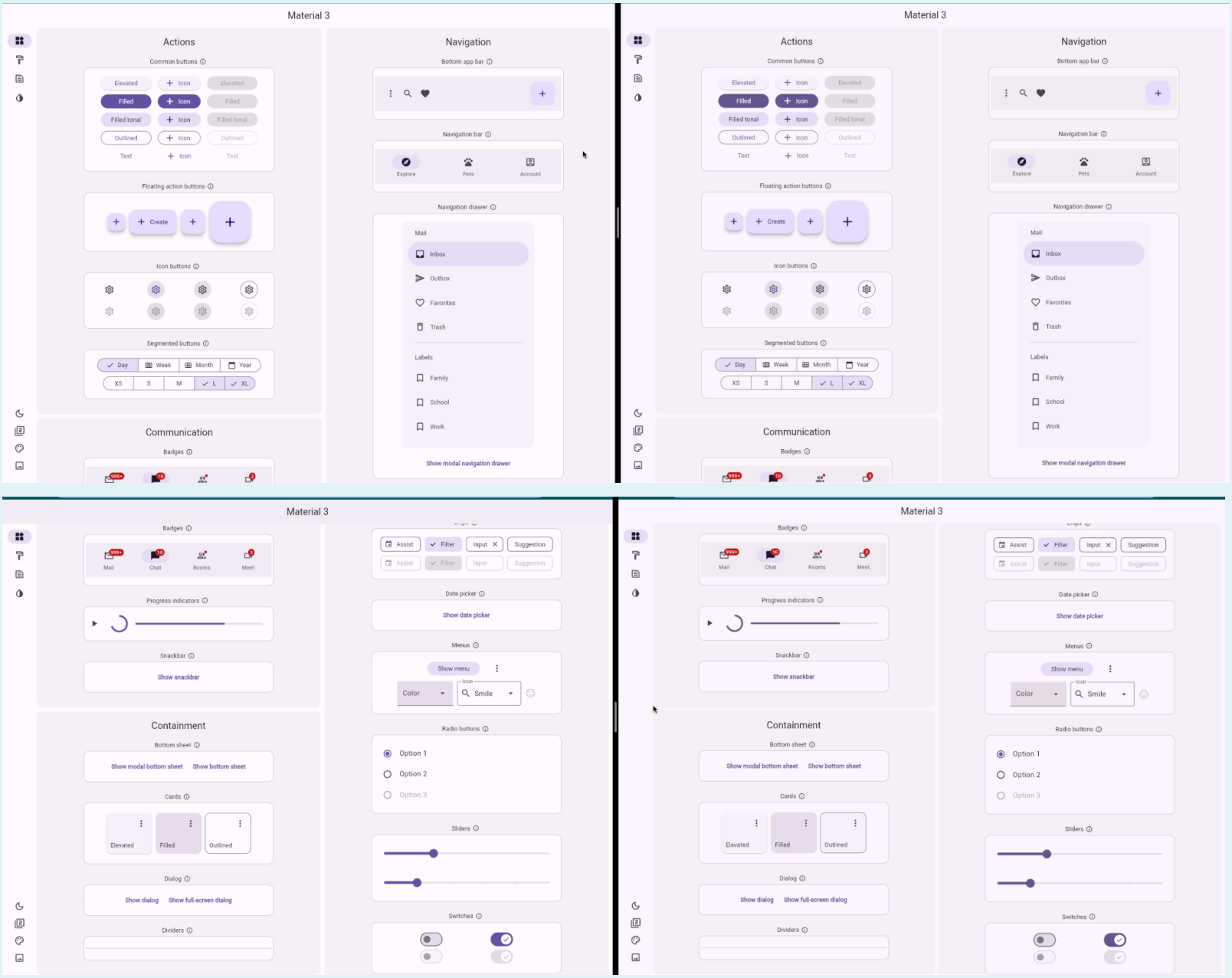
To support these new colors without breaking existing applications, we propose to add the new colors to the existing `ColorScheme` class, while deprecating the removed colors. The solution mostly follows the design doc which establishes a pattern for adding new colors and deprecating old colors.

A draft PR that implements the changes is available.

USAGE EXAMPLES

The new changes in the draft PR are deployed to the experimental M3 demo (demo on the right side). Compared with the current M3 demo (demo on the left side), the difference is subtle.

The Material Design spec has removed surface tint tokens for all widgets. Even though the widgets' `surfaceTint` colors are not yet deprecated, their defaults are changed to transparent color. So the demo with the new changes is not affected by the surface tint color.



DETAILED DESIGN/DISCUSSION

Starting from version 0.162, Material Design tokens have been gradually updated to use tonal surfaces. We will keep updating the component tokens to the latest version, and also apply the tokens changes to Material widgets.

Add new color properties to ColorScheme

In the current `ColorScheme`, some color parameters are required, such as `ColorScheme.primary`, but some later-added color parameters are nullable and have getters for each that provide a non-null default if the color is null, such as `ColorScheme.primaryContainer`.

```
/// Create a ColorScheme instance.
const ColorScheme({
 required this.primary,
 Color? primaryContainer,
 // ...
}) : _primaryContainer = primaryContainer,
   // ...

  final Color? _primaryContainer;
  Color get primaryContainer => _primaryContainer ?? primary;
```

To add new `ColorScheme` roles, we cannot just add them as required parameters because that would break all the current apps that use the const constructor without the new colors. Similar to how we added `ColorScheme.primaryContainer`, we can add these colors using the same approach.

```
  /// Create a ColorScheme instance.
  const ColorScheme({
    ...
    Color? primaryFixed,
    Color? primaryFixedDim,
    Color? onPrimaryFixed,
    Color? onPrimaryFixedVariant,

    // ...
  }) : _primaryFixed = primaryFixed,
       _primaryFixedDim = primaryFixedDim,
       _onPrimaryFixed = onPrimaryFixed,
       _onPrimaryFixedVariant = onPrimaryFixedVariant,

  // ...

  final Color? _primaryFixed;
  Color get primaryFixed => _primaryFixed ?? primary;
  // ...
```

Deprecate unused colors

We can use a similar solution to deprecate the colors being removed. For required parameters, like `background` and `onBackground` colors, they can be changed to nullable parameters so that apps can remove these colors without causing compilation issues for apps that haven't. For the nullable parameter, `surfaceVariant`, we can just mark it as deprecated. The draft PR also adds dart fix for these unused colors.

```
const ColorScheme({
 required this.primary,

 // ...

 @Deprecated(
   'Use surface instead. '
   'This feature was deprecated after 3.18.0-0.0.pre.'
 )
 Color? background,
 @Deprecated(
   'Use onSurface instead. '
   'This feature was deprecated after 3.18.0-0.0.pre.'
 )
 Color? onBackground,
 @Deprecated(
   'Use surfaceContainerHighest instead. '
   'This feature was deprecated after 3.18.0-0.0.pre.'
 )
 Color? surfaceVariant,
}) : _background = background,
     _onBackground = onBackground,
     _surfaceVariant = surfaceVariant;

// ...

final Color? _background;
@Deprecated(
 'Use surface instead. '
 'This feature was deprecated after 3.18.0-0.0.pre.'
)
Color get background => _background ?? surface;

// ...
```

Fallback defaults

For the new properties that are nullable, we need to provide appropriate default values. Rather than picking hard coded values, we can express them in terms of the existing colors that are required. Here is the proposed list we will use if a property

isn't provided:

Property	Fallback
surfaceDim	surface
surfaceBright	surface
surfaceContainerLowest	surface
surfaceContainerLow	surface
surfaceContainer	surface
surfaceContainerHigh	surface
surfaceContainerHighest	surface
primaryFixed	primary
primaryFixedDim	primary
onPrimaryFixed	onPrimary
onPrimaryFixedVariant	onPrimary
secondaryFixed	secondary
secondaryFixedDim	secondary
onSecondaryFixed	onSecondary
onSecondaryFixedVariant	onSecondary
tertiaryFixed	tertiary
tertiaryFixedDim	tertiary
onTertiaryFixed	onTertiary
onTertiaryFixedVariant	onTertiary

OPEN QUESTIONS

- If a class `xColorScheme` extends `ColorScheme` and overrides `ColorScheme`'s methods, such as `copyWith()`, adding new parameters will cause breaking change. I see this problem on the 1P side, other than broadcasting this change to users, are there any better ways to avoid this?

TESTING PLAN

Unit tests for new colors will be added to expand the current color scheme tests.