Getting started with Material Components for Android

1. Depend on our library

Material Components for Android is available through Google's Maven Repository. To use it:

1. Open the `build.gradle` file for your application.
2. Make sure that the `repositories` section includes Google's Maven Repository `google()`. For example:

```
allprojects {
  repositories {
    google()
    jcenter()
  }
}
```

3. Add the library to the `dependencies` section:

```
dependencies {
  // ...
  implementation 'com.google.android.material:material:<version>'
  // ...
}
```

Visit Google's Maven Repository or MVN Repository to find the latest version of the library.

New Namespace and AndroidX

If your app currently depends on the original Design Support Library, you can make use of the `Refactor to AndroidX…` option provided by Android Studio. Doing so will update your app's dependencies and code to use the newly packaged `androidx` and `com.google.android.material` libraries.

If you don't want to switch over to the new `androidx` and `com.google.android.material` packages yet, you can use Material Components via the `com.android.support:design:28.0.0` dependency.

Note: You should not use the `com.android.support` and `com.google.android.material` dependencies in your app at the same time.

2. Compile your app with Android 9

In order to use Material Components for Android, and the latest versions of the Support Libraries, you will have to update your app's `compileSdkVersion` to `28` and download the Android 9 using the SDK manager. For more information on Android 9 and its timeline, take a look at the Program Overview page.

3. Ensure you are using `AppCompatActivity`

Using `AppCompatActivity` will ensure that all the components work correctly. If you are unable to extend from `AppCompatActivity`, update your activities to use `AppCompatDelegate`. This will enable the `AppCompat` versions of components to be inflated among other important things.

4. Change your app theme to inherit from a Material Components theme

Doing an app-wide migration by changing your app theme to inherit from a Material Components theme is the recommended approach. However, be sure to test thoroughly afterwards, as components in existing layouts may change their looks and behavior.

Note: If you **can't** change your theme, you can do one of the following:

- Inherit from one of our Material Components **Bridge** themes. See the **Bridge Themes** section for more details.
- Continue to inherit from an AppCompat theme and add some new theme attributes to your theme. See the **App Compat Themes** section for more details.

Material Components themes

The following is the list of Material Components themes you can use to get the latest component styles and theme-level attributes.

- `Theme.MaterialComponents`
- `Theme.MaterialComponents.NoActionBar`
- `Theme.MaterialComponents.Light`
- `Theme.MaterialComponents.Light.NoActionBar`
- `Theme.MaterialComponents.Light.DarkActionBar`
- `Theme.MaterialComponents.DayNight`
- `Theme.MaterialComponents.DayNight.NoActionBar`
- `Theme.MaterialComponents.DayNight.DarkActionBar`

Update your app theme to inherit from one of these themes, e.g.:

```
<style name="Theme.MyApp" parent="Theme.MaterialComponents.DayNight">
    <!-- ... -->
</style>
```

For more information on how to set up theme-level attributes for your app, take a look at our Theming guide, as well as our Dark Theme guide for why it's important to inherit from the `DayNight` theme.

Note: Using a Material Components theme enables a custom view inflater which replaces default components with their Material counterparts. Currently, this only replaces `<Button>` XML components with `<MaterialButton>`.

Bridge Themes {#bridge-themes}

If you cannot change your theme to inherit from a Material Components theme, you can inherit from a Material Components **Bridge** theme.

```
<style name="Theme.MyApp" parent="Theme.MaterialComponents.Light.Bridge">
    <!-- ... -->
</style>
```

Both `Theme.MaterialComponents` and `Theme.MaterialComponents.Light` have `.Bridge` themes:

- `Theme.MaterialComponents.Bridge`
- `Theme.MaterialComponents.Light.Bridge`
- `Theme.MaterialComponents.NoActionBar.Bridge`
- `Theme.MaterialComponents.Light.NoActionBar.Bridge`
- `Theme.MaterialComponents.Light.DarkActionBar.Bridge`

Bridge themes inherit from AppCompat themes, but also define the new Material Components theme attributes for you. If you use a bridge theme, you can start using Material Design components without changing your app theme.

AppCompat Themes {#app-compat-themes}

You can also incrementally test new Material components without changing your app theme. This allows you to keep your existing layouts looking and behaving the same, while introducing new components to your layout one at a time.

However, you must add the following new theme attributes to your existing app theme, or you will encounter `ThemeEnforcement` errors:

```
<style name="Theme.MyApp" parent="Theme.AppCompat">

  <!-- Original AppCompat attributes. -->
  <item name="colorPrimary">@color/my_app_primary_color</item>
  <item name="colorSecondary">@color/my_app_secondary_color</item>
  <item name="android:colorBackground">@color/my_app_background_color</item>
  <item name="colorError">@color/my_app_error_color</item>
```

```
    <!-- New MaterialComponents attributes. -->
    <item name="colorPrimaryVariant">@color/my_app_primary_variant_color</item>
    <item name="colorSecondaryVariant">@color/my_app_secondary_variant_color</item>
    <item name="colorSurface">@color/my_app_surface_color</item>
    <item name="colorOnPrimary">@color/my_app_color_on_primary</item>
    <item name="colorOnSecondary">@color/my_app_color_on_secondary</item>
    <item name="colorOnBackground">@color/my_app_color_on_background</item>
    <item name="colorOnError">@color/my_app_color_on_error</item>
    <item name="colorOnSurface">@color/my_app_color_on_surface</item>
    <item name="scrimBackground">@color/mtrl_scrim_color</item>
    <item name="textAppearanceHeadline1">@style/TextAppearance.MaterialComponents.Headline1</item
    <item name="textAppearanceHeadline2">@style/TextAppearance.MaterialComponents.Headline2</item
    <item name="textAppearanceHeadline3">@style/TextAppearance.MaterialComponents.Headline3</item
    <item name="textAppearanceHeadline4">@style/TextAppearance.MaterialComponents.Headline4</item
    <item name="textAppearanceHeadline5">@style/TextAppearance.MaterialComponents.Headline5</item
    <item name="textAppearanceHeadline6">@style/TextAppearance.MaterialComponents.Headline6</item
    <item name="textAppearanceSubtitle1">@style/TextAppearance.MaterialComponents.Subtitle1</item
    <item name="textAppearanceSubtitle2">@style/TextAppearance.MaterialComponents.Subtitle2</item
    <item name="textAppearanceBody1">@style/TextAppearance.MaterialComponents.Body1</item>
    <item name="textAppearanceBody2">@style/TextAppearance.MaterialComponents.Body2</item>
    <item name="textAppearanceCaption">@style/TextAppearance.MaterialComponents.Caption</item>
    <item name="textAppearanceButton">@style/TextAppearance.MaterialComponents.Button</item>
    <item name="textAppearanceOverline">@style/TextAppearance.MaterialComponents.Overline</item>

  </style>
```

5. Add a Material component to your app

Take a look at our documentation for the full list of available Material components. Each component's page has specific instructions on how to implement it in your app.

Let's use text fields as an example.

Implementing a text field via XML

The default filled text field XML is defined as:

```
<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/textfield_label">

  <com.google.android.material.textfield.TextInputEditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"/>
</com.google.android.material.textfield.TextInputLayout>
```

Note: If you are **not** using a theme that inherits from a Material Components theme, you will have to specify the text field style as well,
via `style="@style/Widget.MaterialComponents.TextInputLayout.FilledBox"`

Other text field styles are also provided. For example, if you want an outlined text field in your layout, you can apply the Material Components `outlined` style to the text field in XML:

```
<com.google.android.material.textfield.TextInputLayout
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/textfield_label">

  <com.google.android.material.textfield.TextInputEditText
      android:layout_width="match_parent"
      android:layout_height="wrap_content"/>
</com.google.android.material.textfield.TextInputLayout>
```

Contributors

Material Components for Android welcomes contributions from the community. Check out our contributing guidelines as well as an overview of the directory structure before getting started.

Useful Links

- Theming Guide
- Contributing
- Building From Source
- Catalog App
- Class documentation
- MDC-Android on Stack Overflow
- Android Developer's Guide
- Material.io
- Material Design Guidelines