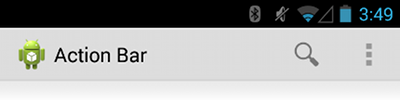
**Adding the Action Bar**

The action bar is one of the most important design elements you can implement for your app's activities. It provides several user interface features that make your app immediately familiar to users by offering consistency between other Android apps. Key functions include:

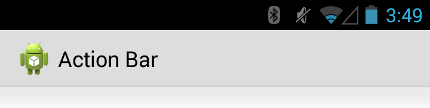
* A dedicated space for giving your app an identity and indicating the user's location in the app.
* Access to important actions in a predictable way (such as Search).
* Support for navigation and view switching (with tabs or drop-down lists).



This training class offers a quick guide to the action bar's basics. For more information about action bar's various features, see the [Action Bar](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\ui\actionbar.html) guide.

# Setting Up the Action Bar

In its most basic form, the action bar displays the title for the activity and the app icon on the left. Even in this simple form, the action bar is useful for all activities to inform users about where they are and to maintain a consistent identity for your app.



Setting up a basic action bar requires that your app use an activity theme that enables the action bar. How to request such a theme depends on which version of Android is the lowest supported by your app. So this lesson is divided into two sections depending on which Android version is your lowest supported.

## Support Android 3.0 and Above Only

Beginning with Android 3.0 (API level 11), the action bar is included in all activities that use the[Theme.Holo](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\R.style.html#Theme_Holo) theme (or one of its descendants), which is the default theme when either **the**[**targetSdkVersion**](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\manifest\uses-sdk-element.html#target)**or [minSdkVersion](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\guide\\topics\\manifest\\uses-sdk-element.html" \l "min) attribute is set to "11" or greater.**

So to add the action bar to your activities, simply set either attribute to 11 or higher. For example:

<manifest ... >  
    <uses-sdk android:minSdkVersion="11" ... />  
    ...  
</manifest>

**Note:** If you've created a custom theme, be sure it uses one of the [Theme.Holo](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.style.html" \l "Theme_Holo) themes as its parent. For details, see [Styling the Action Bar](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\training\basics\actionbar\styling.html).

Now the [Theme.Holo](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.style.html" \l "Theme_Holo) theme is applied to your app and all activities show the action bar. That's it.

## Support Android 2.1 and Above

dding the action bar when running on versions older than Android 3.0 (down to Android 2.1) requires that you include the Android Support Library in your application.

To get started, read the [Support Library Setup](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\tools\support-library\setup.html) document and set up the **v7 appcompat** library (once you've downloaded the library package, follow the instructions for [Adding libraries with resources](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\tools\support-library\setup.html#libs-with-res)).

Once you have the Support Library integrated with your app project:

1. Update your activity so that it extends [ActionBarActivity](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\support\\v7\\app\\ActionBarActivity.html). For example:

public class MainActivity extends ActionBarActivity { ... }

1. In your manifest file, update either the [<application>](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\manifest\application-element.html) element or individual [<activity>](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\manifest\application-element.html) elements to use one of the [Theme.AppCompat](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\support\\v7\\appcompat\\R.style.html" \l "Theme_AppCompat) themes. For example:

<activity android:theme="@style/Theme.AppCompat.Light" ... >

**Note:** If you've created a custom theme, be sure it uses one of the [Theme.AppCompat](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\support\\v7\\appcompat\\R.style.html" \l "Theme_AppCompat) themes as its parent. For details, see [Styling the Action Bar](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\training\basics\actionbar\styling.html).

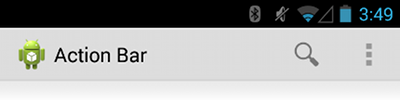
Now your activity includes the action bar when running on Android 2.1 (API level 7) or higher.

Remember to properly set your app's API level support in the manifest:

<manifest ... >  
    <uses-sdk android:minSdkVersion="7"  android:targetSdkVersion="18" />  
    ...  
</manifest>

# Adding Action Buttons

The action bar allows you to add buttons for the most important action items relating to the app's current context. Those that appear directly in the action bar with an icon and/or text are known as action buttons. Actions that can't fit in the action bar or aren't important enough are hidden in the action overflow.



## Specify the Actions in XML

All action buttons and other items available in the action overflow are defined in an XML [menu resource](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\resources\menu-resource.html). To add actions to the action bar, create a new XML file in your project's res/menu/directory.

Add an <item> element for each item you want to include in the action bar. For example:

res/menu/main\_activity\_actions.xml

<menu xmlns:android="http://schemas.android.com/apk/res/android" >  
    <!-- Search, should appear as action button -->  
    <item android:id="@+id/action\_search"  
          android:icon="@drawable/ic\_action\_search"  
          android:title="@string/action\_search"  
          android:showAsAction="ifRoom" />  
    <!-- Settings, should always be in the overflow -->  
    <item android:id="@+id/action\_settings"  
          android:title="@string/action\_settings"  
          android:showAsAction="never" />  
</menu>

### Download action bar icons

To best match the Android[iconography](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\design\style\iconography.html#action-bar) guidelines, you should use icons provided in the [Action Bar Icon Pack](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\design\downloads\index.html#action-bar-icon-pack).

This declares that the Search action should appear as an action button when room is available in the action bar, but the Settings action should always appear in the overflow. (By default, all actions appear in the overflow, but it's good practice to explicitly declare your design intentions for each action.)

The icon attribute requires a resource ID for an image. The name that follows @drawable/ must be the name of a bitmap image you've saved in your project's res/drawable/ directory. For example,"@drawable/ic\_action\_search" refers to ic\_action\_search.png. Likewise, the title attribute uses a string resource that's defined by an XML file in your project's res/values/ directory, as discussed in [Building a Simple User Interface](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\training\basics\firstapp\building-ui.html#Strings).

**Note:** When creating icons and other bitmap images for your app, it's important that you provide multiple versions that are each optimized for a different screen density. This is discussed more in the lesson about [Supporting Different Screens](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\training\basics\supporting-devices\screens.html).

**If your app is using the Support Library** for compatibility on versions as low as Android 2.1, theshowAsAction attribute is not available from the android: namespace. Instead this attribute is provided by the Support Library and you must define your own XML namespace and use that namespace as the attribute prefix. (A custom XML namespace should be based on your app name, but it can be any name you want and is only accessible within the scope of the file in which you declare it.) For example:

res/menu/main\_activity\_actions.xml

<menu xmlns:android="http://schemas.android.com/apk/res/android"  
      **xmlns:yourapp="http://schemas.android.com/apk/res-auto"** >  
    <!-- Search, should appear as action button -->  
    <item android:id="@+id/action\_search"  
          android:icon="@drawable/ic\_action\_search"  
          android:title="@string/action\_search"  
          **yourapp:showAsAction="ifRoom"**  />  
    ...  
</menu>

## Add the Actions to the Action Bar

To place the menu items into the action bar, implement the [onCreateOptionsMenu()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\app\\Activity.html" \l "onCreateOptionsMenu(android.view.Menu)) callback method in your activity to inflate the menu resource into the given [Menu](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\view\Menu.html) object. For example:

@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu items for use in the action bar  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.main\_activity\_actions, menu);  
    return super.onCreateOptionsMenu(menu);  
}

## Respond to Action Buttons

When the user presses one of the action buttons or another item in the action overflow, the system calls your activity's [onOptionsItemSelected()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\app\\Activity.html" \l "onOptionsItemSelected(android.view.MenuItem)) callback method. In your implementation of this method, call [getItemId()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\view\\MenuItem.html" \l "getItemId()) on the given [MenuItem](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\view\\MenuItem.html) to determine which item was pressed—the returned ID matches the value you declared in the corresponding <item> element's android:id attribute.

@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    // Handle presses on the action bar items  
    switch (item.getItemId()) {  
        case R.id.action\_search:  
            openSearch();  
            return true;  
        case R.id.action\_settings:  
            openSettings();  
            return true;  
        default:  
            return super.onOptionsItemSelected(item);  
    }  
}

## Add Up Button for Low-level Activities



**Figure 4.** The Up button in Gmail.

All screens in your app that are not the main entrance to your app (activities that are not the "home" screen) should offer the user a way to navigate to the logical parent screen in the app's hierarchy by pressing the Up button in the action bar.

When running on Android 4.1 (API level 16) or higher, or when using [ActionBarActivity](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\support\\v7\\app\\ActionBarActivity.html) from the Support Library, performing Upnavigation simply requires that you declare the parent activity in the manifest file and enable the Up button for the action bar.

For example, here's how you can declare an activity's parent in the manifest:

<application ... >  
    ...  
    <!-- The main/home activity (it has no parent activity) -->  
    <activity  
        android:name="com.example.myfirstapp.MainActivity" ...>  
        ...  
    </activity>  
    <!-- A child of the main activity -->  
    <activity  
        android:name="com.example.myfirstapp.DisplayMessageActivity"  
        android:label="@string/title\_activity\_display\_message"  
        android:parentActivityName="com.example.myfirstapp.MainActivity" >  
        <!-- Parent activity meta-data to support 4.0 and lower -->  
        <meta-data  
            android:name="android.support.PARENT\_ACTIVITY"  
            android:value="com.example.myfirstapp.MainActivity" />  
    </activity>  
</application>

Then enable the app icon as the Up button by calling [setDisplayHomeAsUpEnabled()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\app\\ActionBar.html" \l "setDisplayHomeAsUpEnabled(boolean)):

@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity\_displaymessage);  
  
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
    // If your minSdkVersion is 11 or higher, instead use:  
    // getActionBar().setDisplayHomeAsUpEnabled(true);  
}

Because the system now knows MainActivity is the parent activity for DisplayMessageActivity, when the user presses the Up button, the system navigates to the parent activity as appropriate—you **do not**need to handle the Up button's event.

For more information about up navigation, see [Providing Up Navigation](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\training\implementing-navigation\ancestral.html).

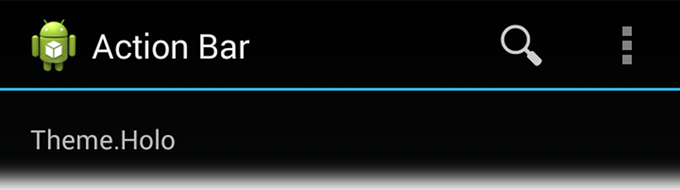
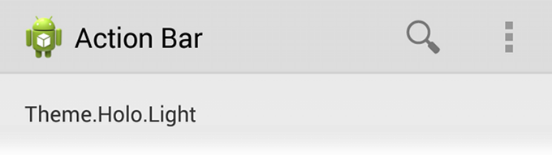
# Styling the Action Bar

The action bar provides your users a familiar and predictable way to perform actions and navigate your app, but that doesn't mean it needs to look exactly the same as it does in other apps. If you want to style the action bar to better fit your product brand, you can easily do so using Android's [style and theme](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\ui\themes.html) resources.

Android includes a few built-in activity themes that include "dark" or "light" action bar styles. You can also extend these themes to further customize the look for your action bar.

**Note:** If you are using the Support Library APIs for the action bar, then you must use (or override) the[Theme.AppCompat](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v7\appcompat\R.style.html#Theme_AppCompat) family of styles (rather than the[Theme.Holo](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\R.style.html#Theme_Holo) family, available in API level 11 and higher). In doing so, each style property that you declare must be declared twice: once using the platform's style properties (the [android:](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.attr.html)properties) and once using the style properties included in the Support Library (the[appcompat.R.attr](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v7\appcompat\R.attr.html) properties—the context for these properties is actually your app). See the examples below for details.

## Use an Android Theme

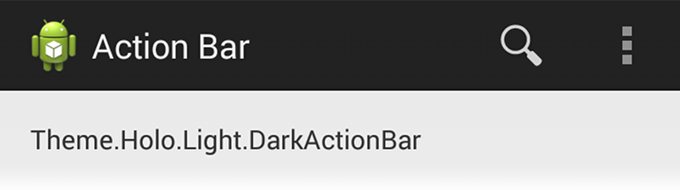
Android includes two baseline activity themes that dictate the color for the action bar:

* [Theme.Holo](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\R.style.html#Theme_Holo) for a "dark" theme.
* [Theme.Holo.Light](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\R.style.html#Theme_Holo_Light) for a "light" theme.

You can apply these themes to your entire app or to individual activities by declaring them in your manifest file with the android:themeattribute for the [<application>](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\manifest\application-element.html) element or individual [<activity>](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\manifest\application-element.html) elements.

For example:

<application android:theme="@android:style/Theme.Holo.Light" ... />



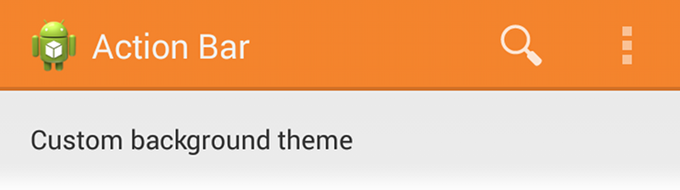
You can also use a dark action bar while the rest of the activity uses the light color scheme by declaring the [Theme.Holo.Light.DarkActionBar](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.style.html" \l "Theme_Holo_Light_DarkActionBar)theme.

When using the Support Library, you must instead use the [Theme.AppCompat](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\support\\v7\\appcompat\\R.style.html" \l "Theme_AppCompat) themes:

* [Theme.AppCompat](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v7\appcompat\R.style.html#Theme_AppCompat) for the "dark" theme.
* [Theme.AppCompat.Light](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v7\appcompat\R.style.html#Theme_AppCompat_Light) for the "light" theme.
* [Theme.AppCompat.Light.DarkActionBar](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v7\appcompat\R.style.html#Theme_AppCompat_Light_DarkActionBar) for the light theme with a dark action bar.

Be sure that you use action bar icons that properly contrast with the color of your action bar. To help you, the [Action Bar Icon Pack](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\design\downloads\index.html#action-bar-icon-pack) includes standard action icons for use with both the Holo light and Holo dark action bar.

## Customize the Background



To change the action bar background, create a custom theme for your activity that overrides the [actionBarStyle](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.attr.html" \l "actionBarStyle) property. This property points to another style in which you can override the [background](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\R.attr.html#background) property to specify a drawable resource for the action bar background.

If your app uses [navigation tabs](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\ui\actionbar.html#Tabs) or the [split action bar](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\ui\actionbar.html#SplitBar), then you can also specify the background for these bars using the [backgroundStacked](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.attr.html" \l "backgroundStacked) and [backgroundSplit](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.attr.html" \l "backgroundSplit) properties, respectively.

**Caution:** It's important that you declare an appropriate parent theme from which your custom theme and style inherit their styles. Without a parent style, your action bar will be without many style properties unless you explicitly declare them yourself.

### For Android 3.0 and higher only

When supporting Android 3.0 and higher only, you can define the action bar's background like this:

res/values/themes.xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <!-- the theme applied to the application or activity -->  
    <style name="CustomActionBarTheme"  
           parent="@style/Theme.Holo.Light.DarkActionBar">  
        <item name="android:actionBarStyle">@style/MyActionBar</item>  
    </style>  
  
    <!-- ActionBar styles -->  
    <style name="MyActionBar"  
           parent="@style/Widget.Holo.Light.ActionBar.Solid.Inverse">  
        <item name="android:background">@drawable/actionbar\_background</item>  
    </style>  
</resources>

Then apply your theme to your entire app or individual activities:

<application android:theme="@style/CustomActionBarTheme" ... />

### For Android 2.1 and higher

When using the Support Library, the same theme as above must instead look like this:

res/values/themes.xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <!-- the theme applied to the application or activity -->  
    <style name="CustomActionBarTheme"  
           parent="@style/Theme.**AppCompat**.Light.DarkActionBar">  
        <item name="android:actionBarStyle">@style/MyActionBar</item>  
  
        <!-- Support library compatibility -->  
        <item name="actionBarStyle">@style/MyActionBar</item>  
    </style>  
  
    <!-- ActionBar styles -->  
    <style name="MyActionBar"  
           parent="@style/Widget.**AppCompat**.Light.ActionBar.Solid.Inverse">  
        <item name="android:background">@drawable/actionbar\_background</item>  
  
        <!-- Support library compatibility -->  
        <item name="background">@drawable/actionbar\_background</item>  
    </style>  
</resources>

Then apply your theme to your entire app or individual activities:

<application android:theme="@style/CustomActionBarTheme" ... />

## Customize the Text Color

To modify the color of text in the action bar, you need to override separate properties for each text element:

* Action bar title: Create a custom style that specifies the textColor property and specify that style for the [titleTextStyle](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.attr.html" \l "titleTextStyle) property in your custom [actionBarStyle](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.attr.html" \l "actionBarStyle).

**Note:** The custom style applied to [titleTextStyle](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.attr.html" \l "titleTextStyle) should use[TextAppearance.Holo.Widget.ActionBar.Title](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\R.style.html#TextAppearance_Holo_Widget_ActionBar_Title) as the parent style.

* Action bar tabs: Override [actionBarTabTextStyle](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.attr.html" \l "actionBarTabTextStyle) in your activity theme.
* Action buttons: Override [actionMenuTextColor](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.attr.html" \l "actionMenuTextColor) in your activity theme.

### For Android 3.0 and higher only

When supporting Android 3.0 and higher only, your style XML file might look like this:

res/values/themes.xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <!-- the theme applied to the application or activity -->  
    <style name="CustomActionBarTheme"  
           parent="@style/Theme.Holo">  
        <item name="android:actionBarStyle">@style/MyActionBar</item>  
        <item name="android:actionBarTabTextStyle">@style/MyActionBarTabText</item>  
        <item name="android:actionMenuTextColor">@color/actionbar\_text</item>  
    </style>  
  
    <!-- ActionBar styles -->  
    <style name="MyActionBar"  
           parent="@style/Widget.Holo.ActionBar">  
        <item name="android:titleTextStyle">@style/MyActionBarTitleText</item>  
    </style>  
  
    <!-- ActionBar title text -->  
    <style name="MyActionBarTitleText"  
           parent="@style/TextAppearance.Holo.Widget.ActionBar.Title">  
        <item name="android:textColor">@color/actionbar\_text</item>  
    </style>  
  
    <!-- ActionBar tabs text styles -->  
    <style name="MyActionBarTabText"  
           parent="@style/Widget.Holo.ActionBar.TabText">  
        <item name="android:textColor">@color/actionbar\_text</item>  
    </style>  
</resources>

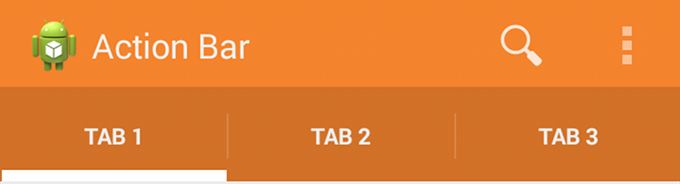
### For Android 2.1 and higher

When using the Support Library, your style XML file might look like this:

res/values/themes.xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <!-- the theme applied to the application or activity -->  
    <style name="CustomActionBarTheme"  
           parent="@style/Theme.**AppCompat**">  
        <item name="android:actionBarStyle">@style/MyActionBar</item>  
        <item name="android:actionBarTabTextStyle">@style/MyActionBarTabText</item>  
        <item name="android:actionMenuTextColor">@color/actionbar\_text</item>  
  
        <!-- Support library compatibility -->  
        <item name="actionBarStyle">@style/MyActionBar</item>  
        <item name="actionBarTabTextStyle">@style/MyActionBarTabText</item>  
        <item name="actionMenuTextColor">@color/actionbar\_text</item>  
    </style>  
  
    <!-- ActionBar styles -->  
    <style name="MyActionBar"  
           parent="@style/Widget.**AppCompat**.ActionBar">  
        <item name="android:titleTextStyle">@style/MyActionBarTitleText</item>  
  
        <!-- Support library compatibility -->  
        <item name="titleTextStyle">@style/MyActionBarTitleText</item>  
    </style>  
  
    <!-- ActionBar title text -->  
    <style name="MyActionBarTitleText"  
           parent="@style/TextAppearance.**AppCompat**.Widget.ActionBar.Title">  
        <item name="android:textColor">@color/actionbar\_text</item>  
        <!-- The textColor property is backward compatible with the Support Library -->  
    </style>  
  
    <!-- ActionBar tabs text -->  
    <style name="MyActionBarTabText"  
           parent="@style/Widget.**AppCompat**.ActionBar.TabText">  
        <item name="android:textColor">@color/actionbar\_text</item>  
        <!-- The textColor property is backward compatible with the Support Library -->  
    </style>  
</resources>

## Customize the Tab Indicator



To change the indicator used for the [navigation tabs](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\ui\actionbar.html#Tabs), create an activity theme that overrides the [actionBarTabStyle](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\R.attr.html" \l "actionBarTabStyle) property. This property points to another style resource in which you override the [background](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\R.attr.html#background) property that should specify a state-list drawable.

**Note:** A state-list drawable is important so that the tab currently selected indicates its state with a background different than the other tabs. For more information about how to create a drawable resource that handles multiple button states, read the [State List](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\resources\drawable-resource.html#StateList) documentation.

For example, here's a state-list drawable that declares a specific background image for several different states of an action bar tab:

res/drawable/actionbar\_tab\_indicator.xml

<?xml version="1.0" encoding="utf-8"?>  
<selector xmlns:android="http://schemas.android.com/apk/res/android">  
  
<!-- STATES WHEN BUTTON IS NOT PRESSED -->  
  
    <!-- Non focused states -->  
    <item android:state\_focused="false" android:state\_selected="false"  
          android:state\_pressed="false"  
          android:drawable="@drawable/tab\_unselected" />  
    <item android:state\_focused="false" android:state\_selected="true"  
          android:state\_pressed="false"  
          android:drawable="@drawable/tab\_selected" />  
  
    <!-- Focused states (such as when focused with a d-pad or mouse hover) -->  
    <item android:state\_focused="true" android:state\_selected="false"  
          android:state\_pressed="false"  
          android:drawable="@drawable/tab\_unselected\_focused" />  
    <item android:state\_focused="true" android:state\_selected="true"  
          android:state\_pressed="false"  
          android:drawable="@drawable/tab\_selected\_focused" />  
  
  
<!-- STATES WHEN BUTTON IS PRESSED -->  
  
    <!-- Non focused states -->  
    <item android:state\_focused="false" android:state\_selected="false"  
          android:state\_pressed="true"  
          android:drawable="@drawable/tab\_unselected\_pressed" />  
    <item android:state\_focused="false" android:state\_selected="true"  
        android:state\_pressed="true"  
        android:drawable="@drawable/tab\_selected\_pressed" />  
  
    <!-- Focused states (such as when focused with a d-pad or mouse hover) -->  
    <item android:state\_focused="true" android:state\_selected="false"  
          android:state\_pressed="true"  
          android:drawable="@drawable/tab\_unselected\_pressed" />  
    <item android:state\_focused="true" android:state\_selected="true"  
          android:state\_pressed="true"  
          android:drawable="@drawable/tab\_selected\_pressed" />  
</selector>

### For Android 3.0 and higher only

When supporting Android 3.0 and higher only, your style XML file might look like this:

res/values/themes.xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <!-- the theme applied to the application or activity -->  
    <style name="CustomActionBarTheme"  
           parent="@style/Theme.Holo">  
        <item name="android:actionBarTabStyle">@style/MyActionBarTabs</item>  
    </style>  
  
    <!-- ActionBar tabs styles -->  
    <style name="MyActionBarTabs"  
           parent="@style/Widget.Holo.ActionBar.TabView">  
        <!-- tab indicator -->  
        <item name="android:background">@drawable/actionbar\_tab\_indicator</item>  
    </style>  
</resources>

### For Android 2.1 and higher

When using the Support Library, your style XML file might look like this:

res/values/themes.xml

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <!-- the theme applied to the application or activity -->  
    <style name="CustomActionBarTheme"  
           parent="@style/Theme.**AppCompat**">  
        <item name="android:actionBarTabStyle">@style/MyActionBarTabs</item>  
  
        <!-- Support library compatibility -->  
        <item name="actionBarTabStyle">@style/MyActionBarTabs</item>  
    </style>  
  
    <!-- ActionBar tabs styles -->  
    <style name="MyActionBarTabs"  
           parent="@style/Widget.**AppCompat**.ActionBar.TabView">  
        <!-- tab indicator -->  
        <item name="android:background">@drawable/actionbar\_tab\_indicator</item>  
  
        <!-- Support library compatibility -->  
        <item name="background">@drawable/actionbar\_tab\_indicator</item>  
    </style>  
</resources>

**More resources**

* See more style properties for the action bar are listed in the [Action Bar](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\ui\actionbar.html#Style) guide.
* Learn more about how themes work in the [Styles and Themes](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\ui\themes.html) guide.
* For even more complete styling for the action bar, try the [Android Action Bar Style Generator](http://www.actionbarstylegenerator.com).