## Retaining an Object During a Configuration Change

If restarting your activity requires that you recover large sets of data, re-establish a network connection, or perform other intensive operations, then a full restart due to a configuration change might be a slow user experience. Also, it might not be possible for you to completely restore your activity state with the [Bundle](https://developer.android.com/reference/android/os/Bundle.html) that the system saves for you with the [onSaveInstanceState()](https://developer.android.com/reference/android/app/Activity.html" \l "onSaveInstanceState(android.os.Bundle)) callback—it is not designed to carry large objects (such as bitmaps) and the data within it must be serialized then deserialized, which can consume a lot of memory and make the configuration change slow. In such a situation, you can alleviate the burden of reinitializing your activity by retaining a [Fragment](https://developer.android.com/reference/android/app/Fragment.html) when your activity is restarted due to a configuration change. This fragment can contain references to stateful objects that you want to retain.

When the Android system shuts down your activity due to a configuration change, the fragments of your activity that you have marked to retain are not destroyed. You can add such fragments to your activity to preserve stateful objects.

To retain stateful objects in a fragment during a runtime configuration change:

Extend the [Fragment](https://developer.android.com/reference/android/app/Fragment.html) class and declare references to your stateful objects.

Call [setRetainInstance(boolean)](https://developer.android.com/reference/android/app/Fragment.html" \l "setRetainInstance(boolean)) when the fragment is created.

Add the fragment to your activity.

Use [FragmentManager](https://developer.android.com/reference/android/app/FragmentManager.html) to retrieve the fragment when the activity is restarted.

For example, define your fragment as follows:

|  |
| --- |
| public class RetainedFragment extends Fragment {      // data object we want to retain     private MyDataObject data;      // this method is only called once for this fragment     @Override     public void onCreate(Bundle savedInstanceState) {         super.onCreate(savedInstanceState);         // retain this fragment         setRetainInstance(true);     }      public void setData(MyDataObject data) {         this.data = data;     }      public MyDataObject getData() {         return data;     } } |

Then use [FragmentManager](https://developer.android.com/reference/android/app/FragmentManager.html) to add the fragment to the activity. You can obtain the data object from the fragment when the activity starts again during runtime configuration changes. For example, define your activity as follows:

|  |
| --- |
| public class MyActivity extends Activity {      private RetainedFragment dataFragment;      @Override     public void onCreate(Bundle savedInstanceState) {         super.onCreate(savedInstanceState);         setContentView(R.layout.main);          // find the retained fragment on activity restarts         FragmentManager fm = getFragmentManager();         dataFragment = (DataFragment) fm.findFragmentByTag(“data”);          // create the fragment and data the first time         if (dataFragment == null) {             // add the fragment             dataFragment = new DataFragment();             fm.beginTransaction().add(dataFragment, “data”).commit();             // load the data from the web             dataFragment.setData(loadMyData());         }          // the data is available in dataFragment.getData()         ...     }      @Override     public void onDestroy() {         super.onDestroy();         // store the data in the fragment         dataFragment.setData(collectMyLoadedData());     } } |

## Handling the Configuration Change Yourself

If your application doesn't need to update resources during a specific configuration change *and* you have a performance limitation that requires you to avoid the activity restart, then you can declare that your activity handles the configuration change itself, which prevents the system from restarting your activity.

To declare that your activity handles a configuration change, edit the appropriate [<activity>](https://developer.android.com/guide/topics/manifest/activity-element.html) element in your manifest file to include the [android:configChanges](https://developer.android.com/guide/topics/manifest/activity-element.html" \l "config) attribute with a value that represents the configuration you want to handle. Possible values are listed in the documentation for the [android:configChanges](https://developer.android.com/guide/topics/manifest/activity-element.html" \l "config) attribute (the most commonly used values are "orientation" to prevent restarts when the screen orientation changes and "keyboardHidden" to prevent restarts when the keyboard availability changes). You can declare multiple configuration values in the attribute by separating them with a pipe | character.

For example, the following manifest code declares an activity that handles both the screen orientation change and keyboard availability change:

|  |
| --- |
| <activity android:name=".MyActivity"           android:configChanges="orientation|keyboardHidden"           android:label="@string/app\_name"> |

Now, when one of these configurations change, MyActivity does not restart. Instead, the MyActivity receives a call to [onConfigurationChanged()](https://developer.android.com/reference/android/app/Activity.html" \l "onConfigurationChanged(android.content.res.Configuration)). This method is passed a [Configuration](https://developer.android.com/reference/android/content/res/Configuration.html) object that specifies the new device configuration. By reading fields in the [Configuration](https://developer.android.com/reference/android/content/res/Configuration.html), you can determine the new configuration and make appropriate changes by updating the resources used in your interface. At the time this method is called, your activity's [Resources](https://developer.android.com/reference/android/content/res/Resources.html) object is updated to return resources based on the new configuration, so you can easily reset elements of your UI without the system restarting your activity.

|  |
| --- |
| @Override public void onConfigurationChanged(Configuration newConfig) {     super.onConfigurationChanged(newConfig);      // Checks the orientation of the screen     if (newConfig.orientation == Configuration.ORIENTATION\_LANDSCAPE) {         Toast.makeText(this, "landscape", Toast.LENGTH\_SHORT).show();     } else if (newConfig.orientation == Configuration.ORIENTATION\_PORTRAIT){         Toast.makeText(this, "portrait", Toast.LENGTH\_SHORT).show();     } } |