Binding Activities to Services

When an Activity is bound to a Service, it maintains a reference to the Service instance itself, allowing you to make method calls on the running Service as you would any other instantiated class.

Binding is available for Activities that would benefi t from a more detailed interface with a Service. To support binding for a Service, implement the onBind method as shown in the simple example below:

```
private final IBinder binder = new MyBinder();
@Override
public IBinder onBind(Intent intent) {
return binder;
}
public class MyBinder extends Binder {
MyService getService() {
return MyService.this;
}
}
```

The connection between the Service and Activity is represented as a ServiceConnection. You'll need to implement a new ServiceConnection, overriding the onServiceConnected and onServiceDisconnected methods to get a reference to the Service instance once a connection has been established.

```
// Reference to the service
private MyService serviceBinder;
// Handles the connection between the service and activity
private ServiceConnection mConnection = new ServiceConnection() {
public void onServiceConnected(ComponentName className, IBinder service) {
// Called when the connection is made.
serviceBinder = ((MyService.MyBinder)service).getService();
}
public void onServiceDisconnected(ComponentName className) {
// Received when the service unexpectedly disconnects.
serviceBinder = null;
}
};
```

To perform the binding, call bindService, passing in an Intent (either explicit or implicit) that selects the Service to bind to and an instance of your new ServiceConnection implementation, as shown in this skeleton code:

```
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
// Bind to the service
Intent bindIntent = new Intent(MyActivity.this, MyService.class);
bindService(bindIntent, mConnection, Context.BIND_AUTO_CREATE);
        }
```

Once the Service has been bound, all of its public methods and properties are available through the serviceBinder object obtained from the onServiceConnected handler.

Android applications do not (normally) share memory, but in some cases, your application may want to interact with (and bind to) Services running in different application processes.

You can communicate with a Service running in a different process using broadcast Intents or through the extras Bundle in the Intent used to start the Service. If you need a more tightly coupled connection, you can make a Service available for binding across application boundaries using AIDL. AIDL defi nes the Service's interface in terms of OS level primitives, allowing Android to transmit objects across process boundaries. AIDL defi nitions are covered in Chapter 11.