Android Application Development, COMP 10073

Mohawk College, Winter 2021

Broadcast Senders and Receivers

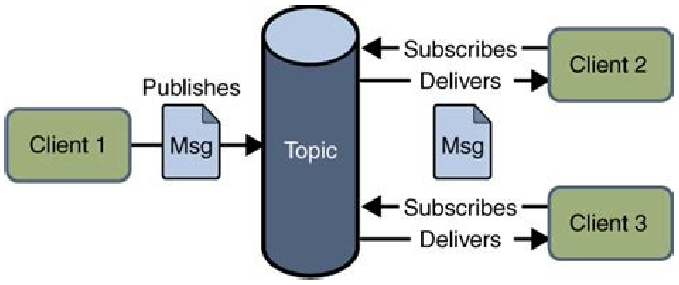
https://developer.android.com/guide/components/broadcasts.html

Android apps can send or receive broadcast messages from the Android system and other Android apps, similar to the publish-subscribe design pattern. The idea is that when a special event occurs a "sender" puts out a generic message that can wake up any receiver that is interested in the message. Example broadcasts:

- The device starts charging
- The device switches into airplane mode
- A picture is taken
- An SMS message is received

Broadcasts in Computing

https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern



Where

- Senders of messages, called publishers, do not program the messages to be sent directly to specific receivers.
- Instead publishers categorize messages into classes without knowledge of subscribers.
- Subscribers express interest in one or more classes and only receive messages that are of interest.
- Subscribers don't know for sure whether there are publishers.

Broadcast Receiver for new SMS messages.

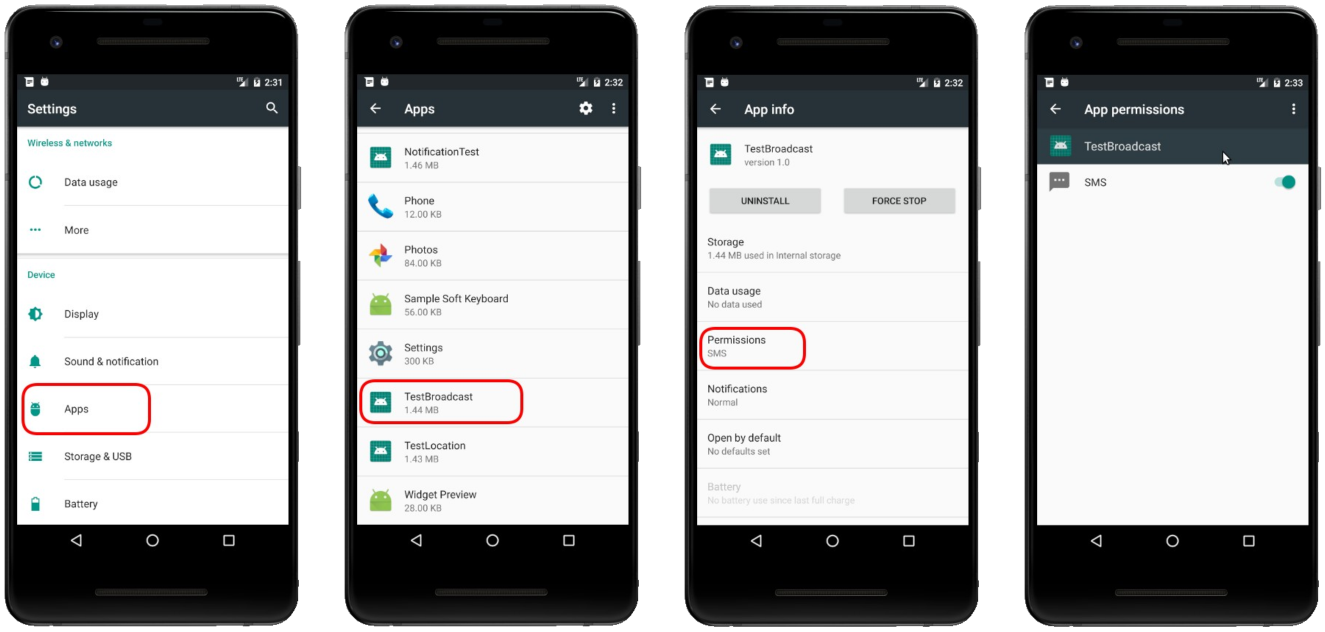
Set up a Broadcast Receiver for new SMS messages. Start by creating an application with the empty template. The default “hello world layout” will work for this example.

Add the following permission to the Android Manifest near the top of the file.

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

This line needs to appear as part of the <manifest> element. If you install the application and then check “apps” under “settings” you will find that this application now lists SMS under permissions, and by default, the permission is not granted. Normally when you install an application from the app store you will get a list of Android features that the application requires permissions for.

For Marshmallow follow: Settings -> Apps -> {app_name} -> Permissions



Add the declaration for the broadcast receiver to the <application> element:

```
<receiver
    android:name=".MyReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
```

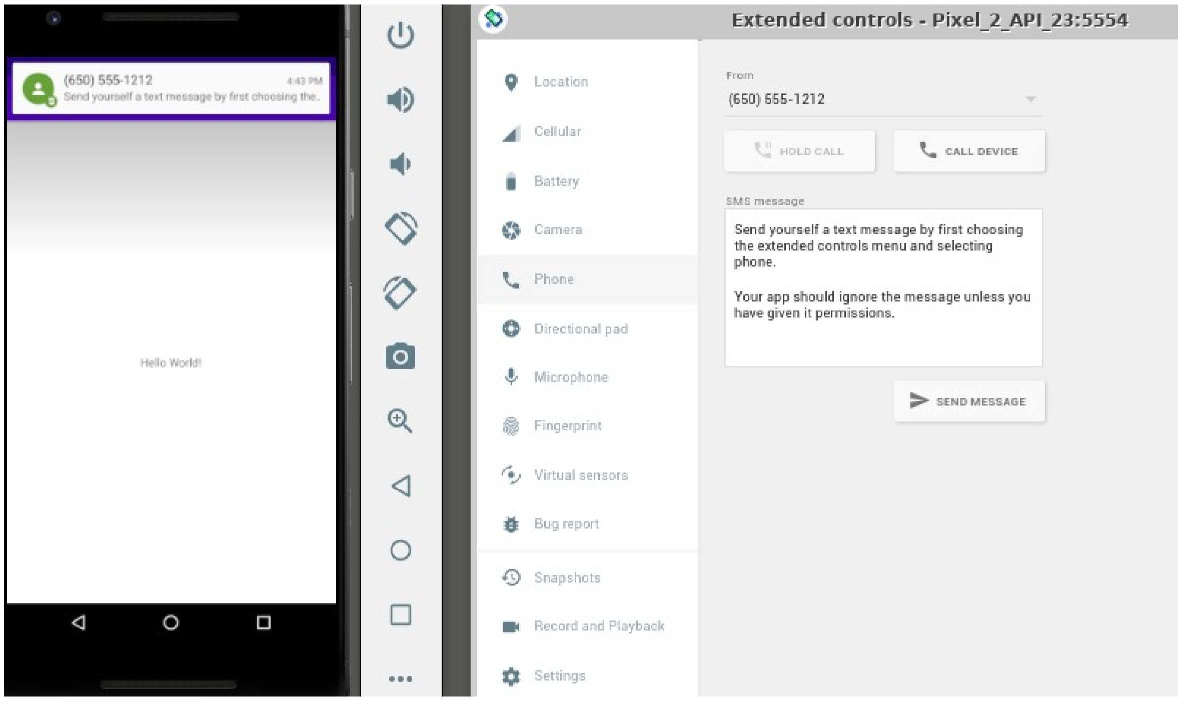
Add the MyReceiver Class

“MyReceiver” is a class that we must implement. It extends BroadcastReceiver and overrides onReceive(). For not just add the empty class to address the reference in the manifest. Include a logging message so we know whether the method is being called or not.

```
public class MyReceiver extends BroadcastReceiver {

    public static String tag = "==MyReceiver==";
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d(tag, "onReceive");
    }
}
```

Start your application and send yourself a text message and check the log.

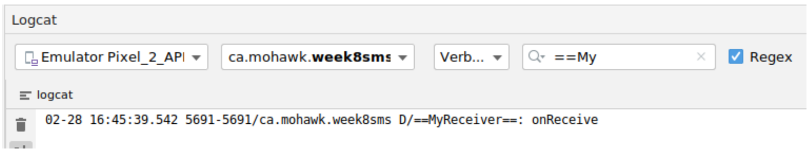


From: (650) 555-1212

SMS message: Send yourself a text message by first choosing the extended controls menu and selecting phone.

Your app should ignore the message unless you have given it permissions.

Give your app the permission and try the experiment again. This time you should get a log message.



```
02-28 16:45:39.542 5691-5691/ca.mohawk.week8sms D/==MyReceiver==: onReceive
```

The text message shown at the top of the phone isn’t part of your app. We have added our MyReceiver class to the list of Broadcast receivers.

Asking the User for Permission

https://developer.android.com/reference/android/support/v4/app/ActivityCompat#requestpermissions

Ideally your application should ask for permissions rather than expecting the user to know that they need to modify permissions. This functionality was included as part of Android SDK 23 which hardened the security model. Prior to SDK 23, it was enough to just include the permission in the manifest.

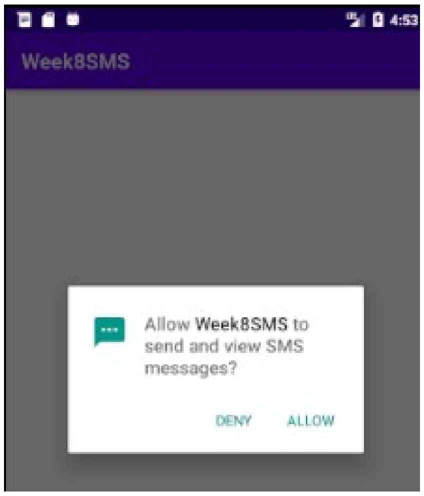
Subsequent versions of Android have modified the model further.

Modify your MainActivity so it looks like this:

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Ask for permission (SDK >= 23)
        ActivityCompat.requestPermissions(this,
          new String[]{Manifest.permission.RECEIVE_SMS}, 1);
    }
}
```



Allow Week8SMS to send and view SMS messages?

Now we get a handy popup each time the application starts asking whether we want to allow this application to receive SMS messages. This popup will adjust our settings for us. If we deny the request we get the option to never ask again.

Read the SMS message

Android will start your service to process the SMS. Your app does not actually need to be running to receive the SMS Intent.

The idea is this:

- MyReceive is registered through the manifest file with the android system
- It will execute so long as you have permission
- We will use the call to MyReceive to start the MainActivity
  - This will work whether the app is already running or not
- We will pass the text message to the MainActivity through an Intent
- The MainActivity will update the TextView widget to show the message

MyReceiver.java

```java
public class MyReceiver extends BroadcastReceiver {

    public static String tag = "==MyReceiver==";
    public static String MSG = "MSG";
    @Override
    public void onReceive(Context context, Intent intent) {

        Log.d(tag, "onReceive");

        // Pull the SmsMessage information from the intent
        // This is an array of messages
        SmsMessage[] messages = getMessagesFromIntent(intent);

        // Get the sender from the first message block
        String sender = messages[0].getOriginatingAddress();

        // Get the first message body text -
        String msg = messages[0].getMessageBody();

        // Log the message
        Log.d(tag, sender + " : " + msg);

        // Prepare to launch the main activity
        Intent myIntent = new Intent(context,
                MainActivity.class);

        myIntent.putExtra(MSG, msg);

        // Set the New Activity Flags
        myIntent.addFlags( Intent.FLAG_ACTIVITY_CLEAR_TASK
                | Intent.FLAG_ACTIVITY_CLEAR_TOP
                | Intent.FLAG_ACTIVITY_NEW_TASK  );

        context.startActivity(myIntent);
    }
}
```

MainActivity.java

```java
public class MainActivity extends AppCompatActivity {

    public static String tag = "==MainActivity==";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Ask for permission (SDK >= 23)
        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.RECEIVE_SMS}, 1);

        Intent myIntent = getIntent();
        String msg = myIntent.getStringExtra(MyReceiver.MSG);
        TextView tv = findViewById(R.id.msgview);
        if (msg != null ) {
            tv.setText(msg);
        }
    }
}
```

Intent Flags

FLAG_ACTIVITY_NEW_TASK (*must have*)

This activity becomes the start of a new task on history stack.

FLAG_ACTIVITY_CLEAR_TOP

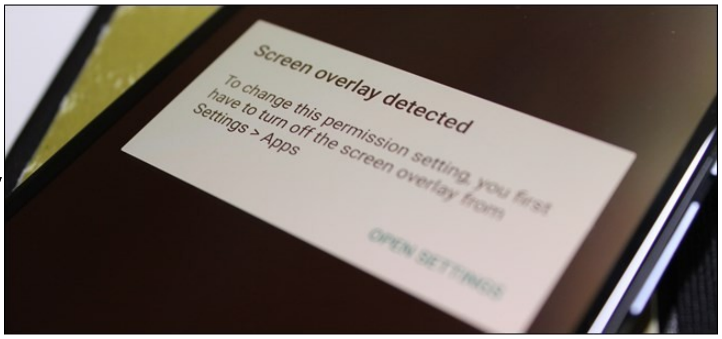
If set, and the activity being launched is already running in the current task, then instead of launching a new instance of that activity, all of the other activities on top of it will be closed and this Intent will be delivered to the (now on top) old activity as a new Intent.

FLAG_ACTIVITY_CLEAR_TASK

If set in an Intent passed to Context#startActivity, this flag will cause any existing task that would be associated with the activity to be cleared before the activity is started.

This way, when you load that FLAG_ACTIVITY_NEW_TASK, and you hit the back button, you won't end up back at a login or sign up screen. That'd be a little awkward for our users if they were already logged in and hit it by accident.

Peculiarities



Screen overlay detected

To change this permission setting, you first have to turn off the screen overlay from Settings > Apps

OPEN SETTINGS

At this point, so long as you have set up the permission the Android system will start your app when an SMS message is received. Try stopping the app

If there are multiple popups being displayed while you are asking the user for permissions you may see a warning about “Screen Overlay Detected”. This happens with this app if a notification for the received text message, or a toast, is visible while you are asking for permissions. You can address this in the short term by turning off the notifications for your AVD’s messaging app.

The security issue involves attempting to confuse a user by asking for permissions, but hiding the exact message with another overlay.

The fact that our application always asks for permission is a bit of a nuissance as well. We will address these details in the next lecture.