Chapter 6

6. Android Telephony

6.1. Telephony Manager

The **android.telephony.TelephonyManager** class provides information about the telephony services such as subscriber id, sim serial number, phone network type etc. Moreover, you can determine the phone state etc.

Example: the following example shows how to display information of your telephony services using TelephoneManager class.

activity_main.xml

```
<RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context=".MainActivity" >

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_alignParentTop="true"
      android:layout_marginLeft="38dp"
      android:layout_marginTop="30dp"
      android:text="Phone Details:" />

</RelativeLayout>
```

MainActivity.java

```
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.telephony.TelephonyManager;
import android.view.Menu;
import android.widget.TextView;

public class MainActivity extends Activity {
  TextView textView1;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
           textView1=(TextView)findViewById(R.id.textView1);

        //Get the instance of TelephonyManager
TelephonyManager  tm=(TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
         //Calling the methods of TelephonyManager the returns the information
        String IMEINumber=tm.getIMEINumber();
        String subscriberID=tm.getDeviceId();
        String SIMSerialNumber=tm.getSimSerialNumber();
        String networkCountryISO=tm.getNetworkCountryIso();
        String SIMCountryISO=tm.getSimCountryIso();
        String softwareVersion=tm.getDeviceSoftwareVersion();
        String voiceMailNumber=tm.getVoiceMailNumber();

        //Get the phone type
        String strphoneType="";

        int phoneType=tm.getPhoneType();

        switch (phoneType)
        {
                case (TelephonyManager.PHONE_TYPE_CDMA):
                        strphoneType="CDMA";
                           break;
                case (TelephonyManager.PHONE_TYPE_GSM):
                        strphoneType="GSM";
                           break;
                case (TelephonyManager.PHONE_TYPE_NONE):
                         strphoneType="NONE";
                            break;
         }

        //getting information if phone is in roaming
        boolean isRoaming=tm.isNetworkRoaming();

        String info="Phone Details:\n";
        info+="\n IMEI Number:"+IMEINumber;
        info+="\n SubscriberID:"+subscriberID;
        info+="\n Sim Serial Number:"+SIMSerialNumber;
        info+="\n Network Country ISO:"+networkCountryISO;
        info+="\n SIM Country ISO:"+SIMCountryISO;
        info+="\n Software Version:"+softwareVersion;
        info+="\n Voice Mail Number:"+voiceMailNumber;
        info+="\n Phone Network Type:"+strphoneType;
        info+="\n In Roaming? :"+isRoaming;

        textView1.setText(info);//displaying the information in the textView
    }
}
```

AndroidManifest.xml

You need to provide **READ_PHONE_STATE** permission in the AndroidManifest.xml file.

AndroidManifest.xml

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

6.2. Phone Call

In android, we can easily make a phone call from our android applications by invoking built-in phone calls app using Intents action (**ACTION_CALL**). To make a phone call using Intent object in android application, we need to write the code like as shown below.

```
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:" + txtPhone.getText().toString()));
startActivity(callIntent);
```

If you observe above code, we are using Intent object **ACTION_CALL** action to make a phone call based on our requirements.

Example

Following is the example of making a phone call by invoking default phone calls app using Intent object in android application. Create a new android application using android studio and give names as **PhoneCallExample**.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical" android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/fstTxt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:layout_marginTop="150dp"
    android:text="Mobile No"
    />
  <EditText
    android:id="@+id/mblTxt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:ems="10">
  </EditText>
  <Button
    android:id="@+id/btnCall"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:text="Call" />
</LinearLayout>
```

Now open our main activity

file **MainActivity.java** from **\src\main\java\com.example.phonecallexample** path and write the code like as shown below

MainActivity.java

```
import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {
   private EditText txtPhone;
   private Button btn;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      txtPhone = (EditText)findViewById(R.id.mblTxt);
      btn = (Button)findViewById(R.id.btnCall);
      btn.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            callPhoneNumber();
         }
      });
   }
   @Override
   public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults)
   {
      if(requestCode == 101)
      {
         if(grantResults[0] == PackageManager.PERMISSION_GRANTED)
         {              callPhoneNumber();          }
      }
   }

   public void callPhoneNumber()
```

```
  {
     try
     {
        if(Build.VERSION.SDK_INT > 22)
        {
           if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CALL_PHONE) != Package
Manager.PERMISSION_GRANTED) {
              ActivityCompat.requestPermissions(MainActivity.this, new String[]{Manifest.permission.CA
LL_PHONE}, 101);
              return;
           }

           Intent callIntent = new Intent(Intent.ACTION_CALL);
           callIntent.setData(Uri.parse("tel:" + txtPhone.getText().toString()));
           startActivity(callIntent);

        }
        else {
           Intent callIntent = new Intent(Intent.ACTION_CALL);
           callIntent.setData(Uri.parse("tel:" + txtPhone.getText().toString()));
           startActivity(callIntent);
        }
     }
     catch (Exception ex)
     {
        ex.printStackTrace();
     }
  }
}
```

If you observe above code we are adding **Runtime** permissions to make sure our application work in both old / latest android OS versions and we used Intent action (**ACTION_CALL**) to make phone call on button click using default phone calls app. As discussed, we need to add a **CALL_PHONE** permission in our android manifest.

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

6.3. Send SMS

In android, we can send SMS from our android application in two ways either by using **SMSManager** api or Intents based on our requirements. If we use **SMSManager** api, it will directly send SMS from our application. In case if we use Intent with proper action (**ACTION_VIEW**), it will invoke built-in SMS app to send SMS from our application.

Sending SMS using SMSManager API

In android, to send SMS using SMSManager API we need to write the code like as shown below.

```
SmsManager smgr = SmsManager.getDefault();
smgr.sendTextMessage(MobileNumber,null,Message,null,null);
```

SMSManager API required **SEND_SMS** permission in our android manifest to send SMS. Following is the code snippet to set **SEND_SMS** permissions in manifest file.

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

Android Send SMS using Intent

To send SMS using Intent object, we need to write the code like as shown below.

```
Intent sInt = new Intent(Intent.ACTION_VIEW);
sInt.putExtra("address", new String[]{txtMobile.getText().toString()});
sInt.putExtra("sms_body",txtMessage.getText().toString());
sInt.setType("vnd.android-dir/mms-sms");
```

Even for Intent, it required a **SEND_SMS** permission in our android manifest to send SMS. Following is the code snippet to set **SEND_SMS** permissions in manifest file.

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

Now we will see how to send SMS in android application using **SMSManager** API with examples.

Example: The following example shows how to send sms from our android app using **SMSManager** api.

Create a new android application using android studio and give names as **SendSMSExample**.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical" android:layout_width="match_parent"
  android:layout_height="match_parent">
  <TextView
    android:id="@+id/fstTxt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:layout_marginTop="150dp"
    android:text="Mobile No" />
  <EditText
    android:id="@+id/mblTxt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="100dp"
    android:ems="10"/>
  <TextView
    android:id="@+id/secTxt"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Message"
    android:layout_marginLeft="100dp" />
```

```
    <EditText
        android:id="@+id/msgTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10" />
    <Button
        android:id="@+id/btnSend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:text="Send SMS" />
</LinearLayout>
```

Now open our main activity

file **MainActivity.java** from **\src\main\java\com.example.sendsmsexample** path and write the code like as shown below

MainActivity.java

```
import android.content.Intent;
import android.net.Uri;
import android.provider.Telephony;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private EditText txtMobile;
    private EditText txtMessage;
    private Button btnSms;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtMobile = (EditText)findViewById(R.id.mblTxt);
        txtMessage = (EditText)findViewById(R.id.msgTxt);
        btnSms = (Button)findViewById(R.id.btnSend);
        btnSms.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                try{
                    SmsManager smgr = SmsManager.getDefault();
                    smgr.sendTextMessage(txtMobile.getText().toString(),null,txtMessage.getText().toString(),
null,null);
                    Toast.makeText(MainActivity.this, "SMS Sent Successfully", Toast.LENGTH_SHORT).show();
```

```
            }
            catch (Exception e){
Toast.makeText(MainActivity.this, "SMS Failed to Send, Please try again", Toast.LENGTH_SHORT).show(
);
            }
        }
    });
  }
}
```

Do not forget to add a **SEND_SMS** permission in our android manifest.

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

if we want to use Intents to send SMS replace button click code like as shown below.

```
btnSms.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    try{
      Intent i = new Intent(Intent.ACTION_VIEW);
      i.setData(Uri.parse("smsto:"));
      i.setType("vnd.android-dir/mms-sms");
      i.putExtra("address", new String(txtMobile.getText().toString()));
      i.putExtra("sms_body",txtMessage.getText().toString());
      startActivity(Intent.createChooser(i, "Send sms via:"));
    }
    catch(Exception e){
Toast.makeText(MainActivity.this, "SMS Failed to Send, Please try again", Toast.LENGTH_SHORT).show(
);
    }
  }
});
```

6.4. Send Email

In android, we can easily send an email from our android application using existing email clients such as **GMAIL**, **Outlook**, etc. instead of building an email client from scratch. The Intent object in android with proper action (**ACTION_SEND**) and data will help us to launch the available email clients to send an email

in our application. To send an email using Intent object in android application, we need to write the code like as shown below.

```
Intent it = new Intent(Intent.ACTION_SEND);
it.putExtra(Intent.EXTRA_EMAIL, new String[]{"abc@gmail.com "});
it.putExtra(Intent.EXTRA_SUBJECT, "test");
it.putExtra(Intent.EXTRA_TEXT, "Hello this is to inform you that....");
it.setType("message/rfc822");
```

If you observe above code we used multiple components to send email, those are

- ✓ **it** - Our local implicit intent
- ✓ **ACTION_SEND** - It's an activity action which specifies that we are sending some data.
- ✓ **putExtra** - we use this **putExtra()** method to add extra information to our Intent. Here we can add following things.
  - EXTRA_EMAIL - It's an array of email addresses
  - EXTRA_SUBJECT - The subject of the email that we want to send
  - EXTRA_TEXT - The body of the email

The android Intent object is having different options such as EXTRA_CC, EXTRA_BCC, EXTRA_HTML_TEXT, EXTRA_STREAM, etc. to add different options for email client.

- ✓ **setType** - We use this property to set the MIME type of data that we want to send. Here we used "**message/rfc822**" and other MIME types are "**text/plain**" and "**image/jpg**".

Now we will see how to send an email in android application using Intent object with examples.

Example: Following is the example of sending an email with existing email clients using Intent in android application. Create a new android application using android studio and give names as **SendMailExample**. In case if you are not aware of creating an app in android studio check this article Android Hello World App.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingLeft="20dp"
   android:paddingRight="20dp"
   android:orientation="vertical" >
   <EditText
      android:id="@+id/txtTo"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="To"/>
   <EditText
```

```
        android:id="@+id/txtSub"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Subject"/>
    <EditText
        android:id="@+id/txtMsg"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="Message"/>
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="Send"
        android:id="@+id/btnSend"/>
</LinearLayout>
```

Now open our

main activity file **MainActivity.java** from **\src\main\java\com.example.sendmailexample** path and write the code like as shown below

MainActivity.java

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    private EditText eTo;
    private EditText eSubject;
    private EditText eMsg;
    private Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        eTo = (EditText)findViewById(R.id.txtTo);
        eSubject = (EditText)findViewById(R.id.txtSub);
        eMsg = (EditText)findViewById(R.id.txtMsg);
        btn = (Button)findViewById(R.id.btnSend);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent it = new Intent(Intent.ACTION_SEND);
                it.putExtra(Intent.EXTRA_EMAIL, new String[]{eTo.getText().toString()});
```

```
            it.putExtra(Intent.EXTRA_SUBJECT,eSubject.getText().toString());
            it.putExtra(Intent.EXTRA_TEXT,eMsg.getText());
            it.setType("message/rfc822");
            startActivity(Intent.createChooser(it,"Choose Mail App"));
        }
    });
  }
}
```

We need to add **MIME type** in our android manifest file for that open android manifest file (**AndroidManifest.xml**) and write the code like as shown below

```
<manifest ...>
  <application
    ...
      <activity android:name=".MainActivity">
        <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
          <action android:name="android.intent.action.SEND"/>
          <category android:name="android.intent.category.DEFAULT"/>
          <data android:mimeType="message/rfc822"/>
        </intent-filter>
      </activity>
  </application>
</manifest>
```

If you observe above **AndroidManifest.xml** file we added following extra fields of Intent filters.

- **action** - we use this property to define that the activity can perform **SEND** action.
- **category** - we included the **DEFAULT** category for this activity to be able to receive implicit intents.
- **data** - the type of data the activity can send.