MOBILE DEVELOPMENT

INTENT

CONTENTS

INTRODUCTION

An Android application could include any number of activities.

The app‘s Manifest designates one of the activities as the first one that should be shown to the user when the application is launched (android.intent.action.MAIN).

Usually, each activity is assocaited to a single screen.

An activity uses the setContentView(...) method to show a given UI.

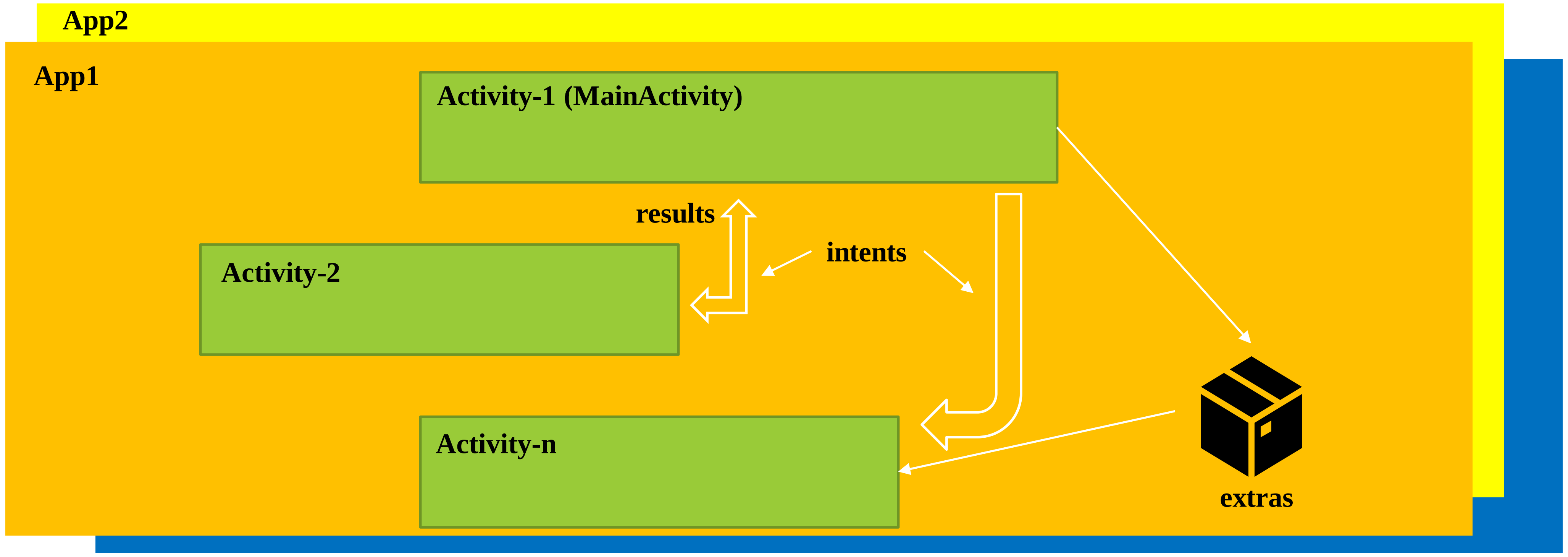
Activities are independent of each other; however they usually cooperate exchanging data and actions.

Activities interact with each other in an asynchronous mode.

Passing control and data from one activity to another is accomplished by asking the current activity to execute an intent.

INTRODUCTION

Activities call each other using Intents. An intent may include basic and extra data elements. The called activity may return a result to the caller.



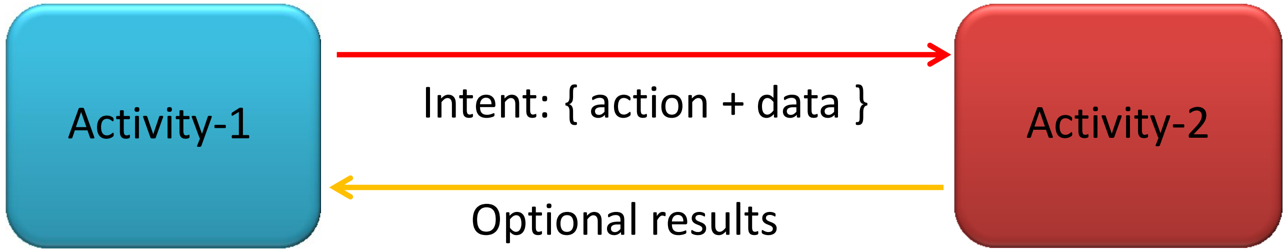
INTRODUCTION
(Invoking intents for execution)

Intents are roughly equivalent to a procedure call in Java (however the caller does not wait for the subroutine to complete). Intents are invoked using the following options

- startActivity(intent): launches an activity
- sendBroadcast(intent): sends an intent to any interested BroadcastReceivers
- startService(intent) or bindService(intent, …): communicates with a background service

The two main components of an Intent are:

- Action The built-in action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_CALL, ACTION_SENDTO,... or a user-created-activity
- Data Basic argument needed by the intent to work. For instance: a phone number to be called , a picture to be shown, a message to be sent, etc.



INTRODUCTION (PARTS OF AN INTENT)

Data Data is supplied as an URI, i.e. a string whose prefix indicates the composition of the data item. For instance: tel:/, http:/, mailto:/, file:/, content:, geo:, audio, media, vnd.android.cursor.dir are common URIs used by Android (For a detailed list of all Intents see http://www.openintents.org/intentsregistry/)

Initiating an Intent

```
Intent myOtherActivity = new Intent (action, data);
startActivity (myOtherActivity);
```

Primary data (as an URI)
tel://
http://
sendto://

Built-in or user-created activity

INTRODUCTION
(Examples of action/data pairs)

ACTION_DIAL tel://5551234 or tel:5551234: Display the phone dialer with the given number filled in.

ACTION_VIEW http://www.google.com: Show Google page in a browser view.

ACTION_EDIT content://contacts/people/2: Edit information about the contact person whose identifier is “2”.

ACTION_VIEW content://contacts/people/2: Used to start an activity to display contact person whose identifier is “2”.

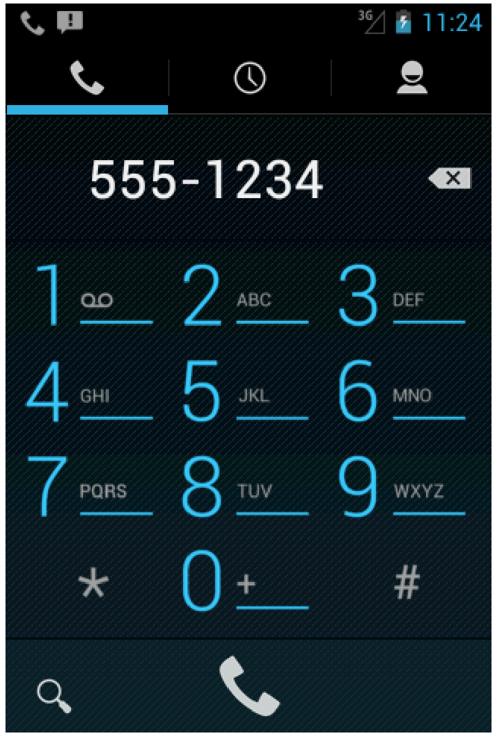
ACTION_VIEW content://contacts/ people/: Display a list of people, which the user can browse through. Selecting a particular person to view would result in a new intent

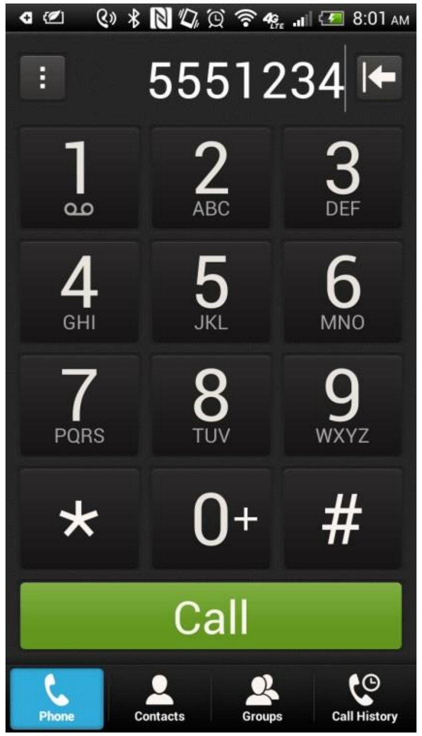
INTRODUCTION (Common built-in Android actions)

List of common actions that Intents can use for launching built-in activities [usually through startActivity(Intent)]

- ACTION_MAIN
- ACTION_VIEW
- ACTION_ATTACH_DATA
- ACTION_EDIT
- ACTION_PICK
- ACTION_CHOOSER
- ACTION_GET_CONTENT
- ACTION_DIAL
- ACTION_CALL
- ACTION_SEND
- ACTION_SENDTO
- ACTION_ANSWER
- ACTION_INSERT
- ACTION_DELETE
- ACTION_RUN
- ACTION_SYNC
- ACTION_PICK_ACTIVITY
- ACTION_SEARCH
- ACTION_WEB_SEARCH
- ACTION_FACTORY_TEST

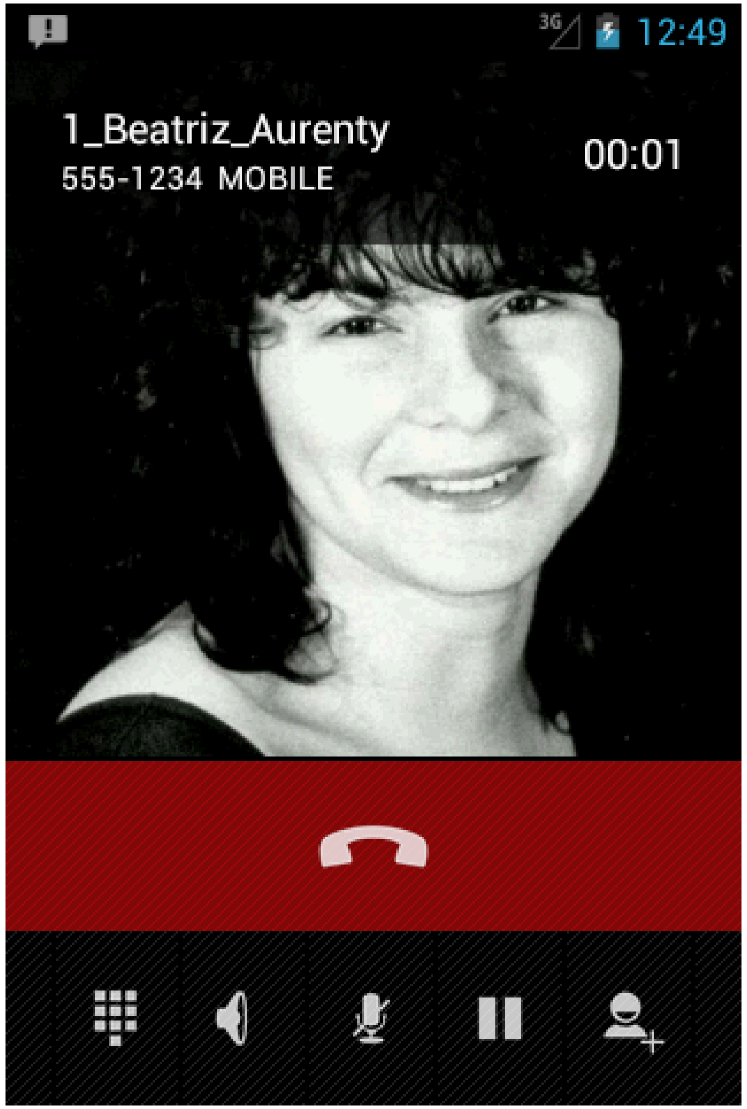
INTRODUCTION (EXAMPLE 1A: ACTION_DIAL)





Display phone dialer with the given number filled in

```
String myPhoneNumberUri =
“tel:555-1234”;
Intent myActivity2 = new
Intent(Intent.ACTION_DIAL,
Uri.parse(myPhoneNumberUri));
startActivity(myActivity2);
```



INTRODUCTION (EXAMPLE 1B: ACTION_CALL)

Placing an immediate phone call

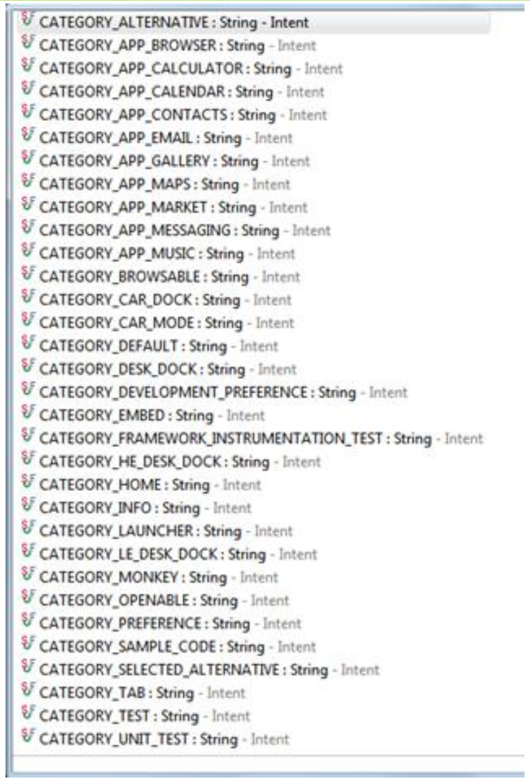
```
String myData = “tel:555-1234”;
Intent myActivity2 = new Intent(
Intent.ACTION_CALL,
Uri.parse(myData));
startActivity(myActivity2);
```

Needs Permission:
<uses-permission android:name=“android.permission.CALL_PHONE”/>

INTRODUCTION (Intents - secondary attributes)

In addition to action/data attributes, there're secondary attributes you can include with an intent: Category, Components, Type, and Extras

Category: additional information about the action to execute



```
CATEGORY_ALTERNATIVE : String - Intent
CATEGORY_APP_BROWSER : String - Intent
CATEGORY_APP_CALCULATOR : String - Intent
CATEGORY_APP_CALENDAR : String - Intent
CATEGORY_APP_CONTACTS : String - Intent
CATEGORY_APP_EMAIL : String - Intent
CATEGORY_APP_GALLERY : String - Intent
CATEGORY_APP_MAPS : String - Intent
CATEGORY_APP_MARKET : String - Intent
CATEGORY_APP_MESSAGING : String - Intent
CATEGORY_APP_MUSIC : String - Intent
CATEGORY_BROWSABLE : String - Intent
CATEGORY_CAR_DOCK : String - Intent
CATEGORY_CAR_MODE : String - Intent
CATEGORY_DEFAULT : String - Intent
CATEGORY_DESK_DOCK : String - Intent
CATEGORY_DEVELOPMENT_PREFERENCE : String - Intent
CATEGORY_EMBED : String - Intent
CATEGORY_FRAMEWORK_INSTRUMENTATION_TEST : String - Intent
CATEGORY_HE_DESK_DOCK : String - Intent
CATEGORY_HOME : String - Intent
CATEGORY_INFO : String - Intent
CATEGORY_LAUNCHER : String - Intent
CATEGORY_LE_DESK_DOCK : String - Intent
CATEGORY_MONKEY : String - Intent
CATEGORY_OPENABLE : String - Intent
CATEGORY_PREFERENCE : String - Intent
CATEGORY_SAMPLE_CODE : String - Intent
CATEGORY_SELECTED_ALTERNATIVE : String - Intent
CATEGORY_TAB : String - Intent
CATEGORY_TEST : String - Intent
CATEGORY_UNIT_TEST : String - Intent
```

Type
Set an explicit MIME data type
contacts/people
images/pictures
images/video
audio/mp3
MIME - Multipurpose Internet Mail Extensions

Extras
This is a Bundle of any additional information.
Typical methods include
    bundle.putInt(key, value) and
    bundle.getInt(key)

Component
Explicit name of a component class to use for the intent (eg. "MyMethod2")

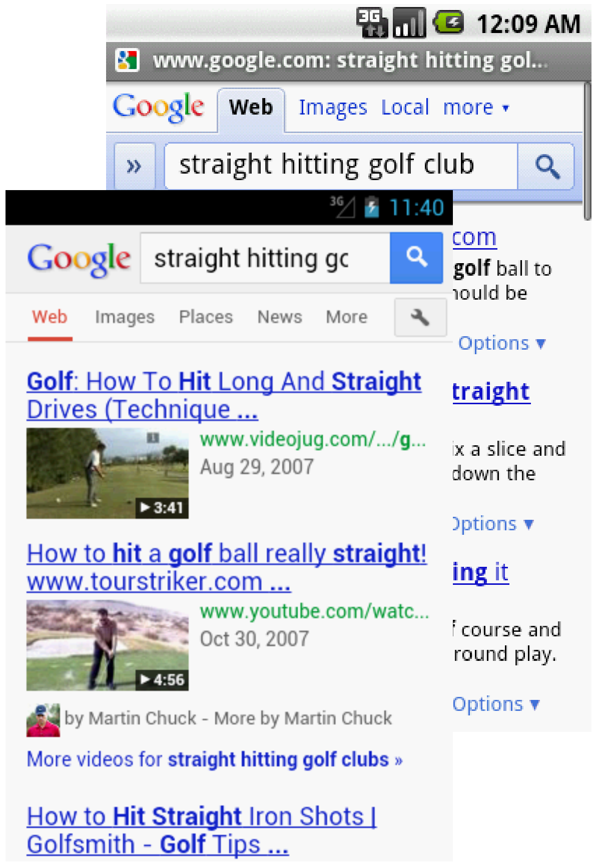
INTRODUCTION (Example 2: ACTION_WEB_SEARCH)

Passing a string as Extra argument for Google Search. The string is a 'human' query with keywords

Goal: searching for golf clubs

```
Intent intent = new Intent(
Intent.ACTION_WEB_SEARCH);
intent.putExtra(SearchManager.QUERY,
"straight hitting golf clubs");
startActivity(intent);
```

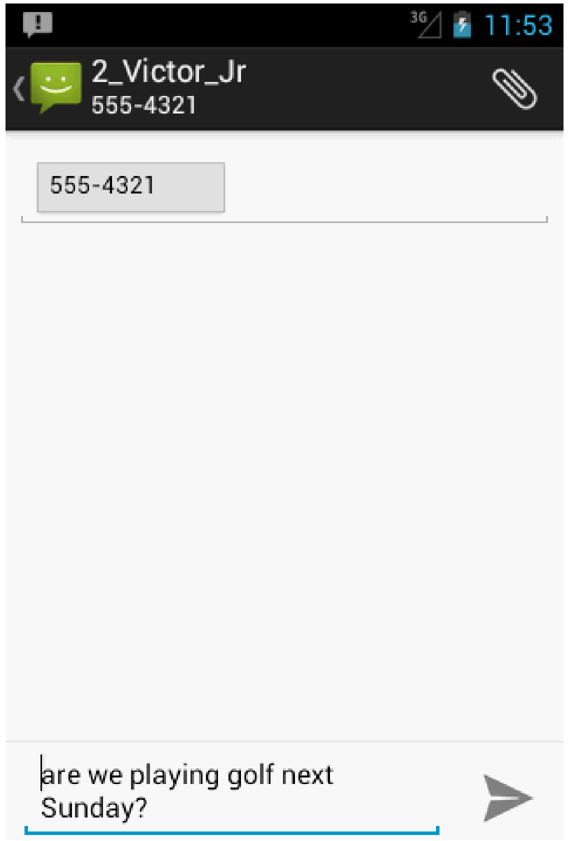
Secondary data


12:09 AM
www.google.com: straight hitting gol...
Google Web Images Local more
straight hitting golf club
11:40
Google straight hitting gc
Web Images Places News More
Golf: How To Hit Long And Straight Drives (Technique ...
www.videojug.com/.../g...
Aug 29, 2007
3:41
How to hit a golf ball really straight! www.tourstriker.com ...
www.youtube.com/watc...
Oct 30, 2007
4:56
by Martin Chuck - More by Martin Chuck
More videos for straight hitting golf clubs »
How to Hit Straight Iron Shots | Golfsmith - Golf Tips ...
Options

INTRODUCTION (Example 3: ACTION_SENDTO)

Preparing an SMS. The text is supplied as an Extra element. The intent expects such a value to be called “sms_body”

```
Intent intent = new Intent(
Intent.ACTION_SENDTO,
Uri.parse(“smsto:555-4321”));
intent.putExtra(“sms_body”,
“are we playing golf next Sunday?”);
startActivity(intent);
```

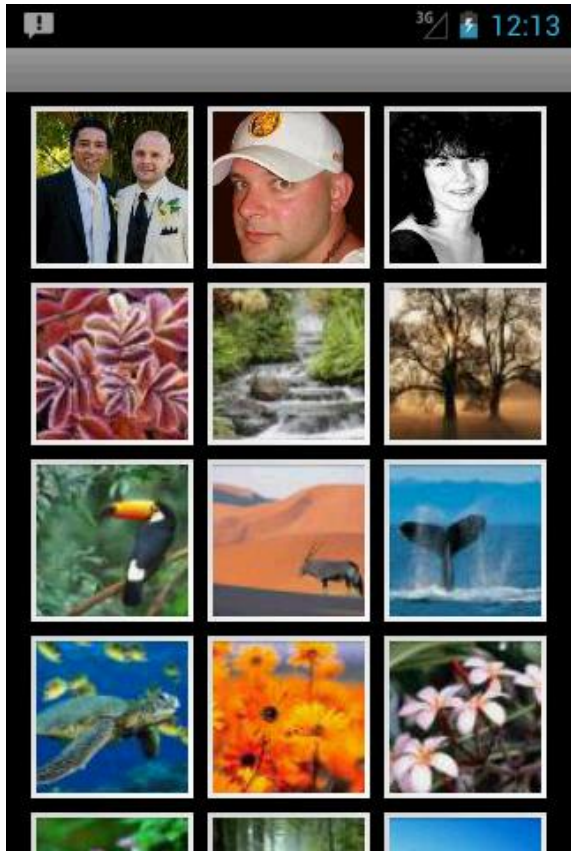


are we playing golf next Sunday?

INTRODUCTION (Example 4: ACTION_GET_CONTENT (Pictures))

Displaying the pictures contained in the device’s external storage. The content to be sought is determined by the MIME type given in .setType(…)

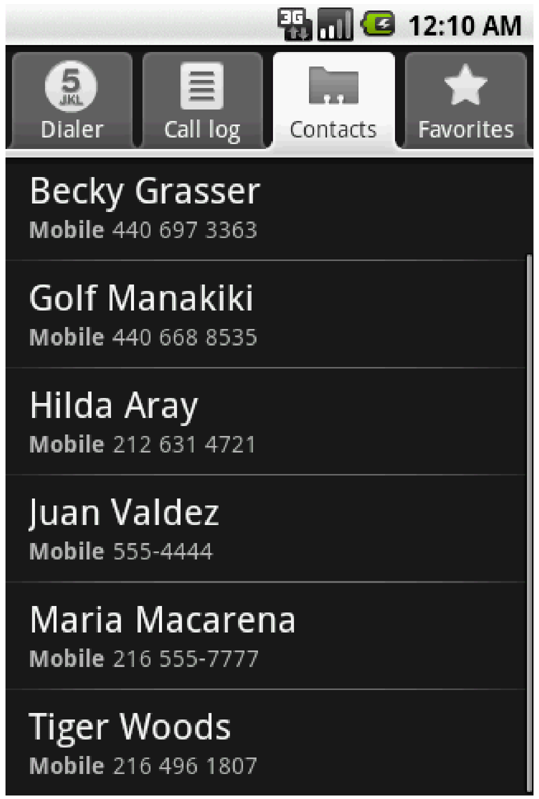
```
Intent intent = new Intent();
intent.setType(“image/pictures/*”);
intent.setAction(Intent.ACTION_GET_C
ONTENT);
startActivity(intent);
```



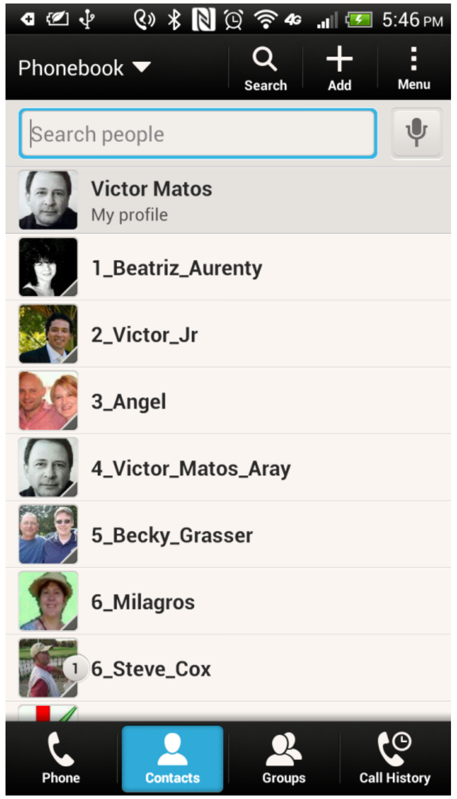
INTRODUCTION (Example 5: ACTION_VIEW (Contacts))

Showing all Contacts stored in your device

```
String myData = "content://contacts/people/";
Intent myActivity2 = new
Intent(Intent.ACTION_VIEW,
Uri.parse(myData));
startActivity(myActivity2);
```



Becky Grasser
Mobile 440 697 3363
Golf Manakiki
Mobile 440 668 8535
Hilda Aray
Mobile 212 631 4721
Juan Valdez
Mobile 555-4444
Maria Macarena
Mobile 216 555-7777
Tiger Woods
Mobile 216 496 1807



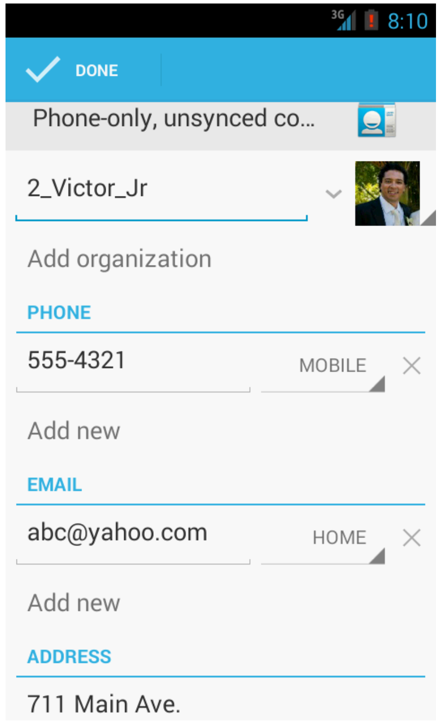
Victor Matos
My profile
1_Beatriz_Aurenty
2_Victor_Jr
3_Angel
4_Victor_Matos_Aray
5_Becky_Grasser
6_Milagros
6_Steve_Cox

INTRODUCTION (Example 6: ACTION_EDIT (Contacts))

Select a particular person (ID 2) from the contact list for editing purposes.

Later in this lesson we learn how to obtain the ID of stored contacts (music tracks, pictures, etc).

```
String myData = ContactsContract.Contacts
.CONTENT_URI + “/” + “2”;
Intent myActivity2 = new
Intent(Intent.ACTION_EDIT,
Uri.parse(myData));
startActivity(myActivity2);
```



2_Victor_Jr

Add organization

PHONE

555-4321 MOBILE

Add new

EMAIL

abc@yahoo.com HOME

Add new

ADDRESS

711 Main Ave.

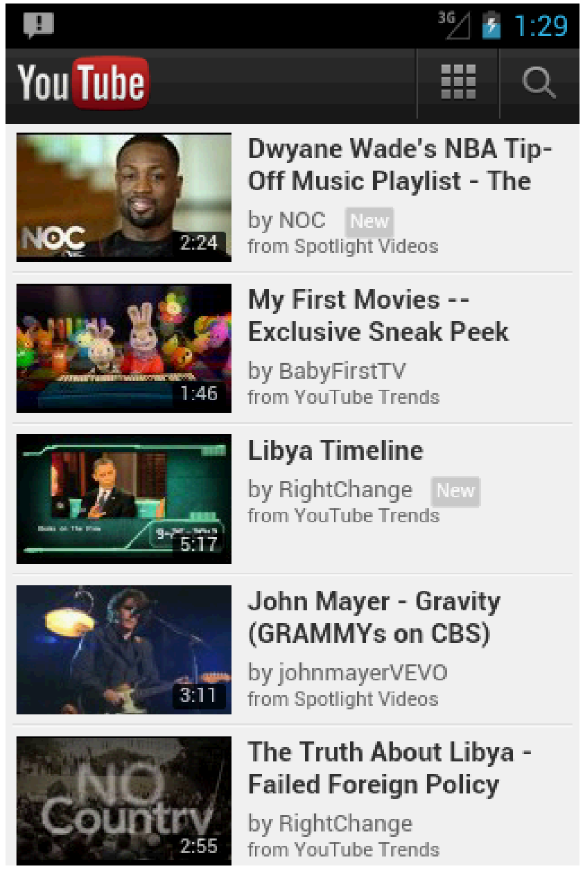
INTRODUCTION
(Example 7: ACTION_VIEW (Web page))

Viewing a web page. The user provides a valid URL pointing to the page.

```
String myUriString = “http://www.youtube.com”;
Intent myActivity2 = new Intent(Intent.ACTION_VIEW,
Uri.parse(myUriString));
startActivity(myActivity2);
```

Caution. Must add to the Manifest a request for permission to use the Internet:

```
<uses-permission android:name=“android.permission.INTERNET”/>
```



Dwyane Wade's NBA Tip-Off Music Playlist - The
by NOC New
from Spotlight Videos

My First Movies -- Exclusive Sneak Peek
by BabyFirstTV
from YouTube Trends

Libya Timeline
by RightChange New
from YouTube Trends

John Mayer - Gravity (GRAMMYs on CBS)
by johnmayerVEVO
from Spotlight Videos

The Truth About Libya - Failed Foreign Policy
by RightChange
from YouTube Trends

Try later with URI: “http://www.youtube.com/results?search_query=ping pong”

INTRODUCTION (Example 8: ACTION_VIEW (Maps - landmark)

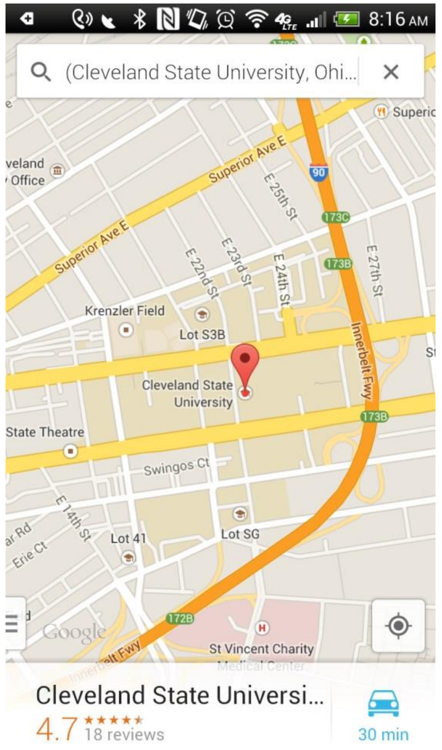
Geo Mapping an Address / Place

Provide a GeoCode expression holding a street address (or place, such as 'golden gate ca')

```
// (you may get multiple results...)
String thePlace = "Cleveland State University, Ohio";
Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
Uri.parse("geo:0,0?q=(" + thePlace + ")"));
startActivity(intent);
```

Modify the Manifest adding the following requests:

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


8:16 AM
(Cleveland State University, Ohi...
Superior Ave E
Krenzler Field
Lot S3B
Cleveland State University
State Theatre
Swingos Ct
Lot 41
Lot SG
St Vincent Charity Medical Center
Google
Cleveland State Universi...
4.7 18 reviews
30 min

INTRODUCTION (Example 9: ACTION_VIEW (Maps - Coordinates))

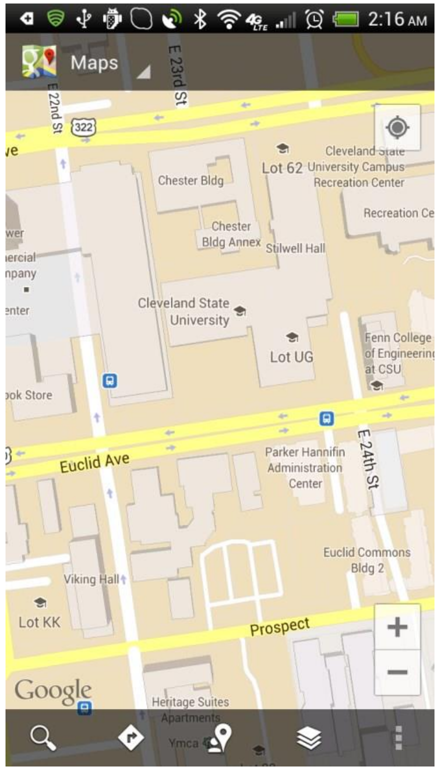
Geo Mapping Coordinates (latitude, longitude)

Provide a GeoCode holding latitude and longitude (also an addittional zoom ‘&z=xx’ with xx in range 1..23)

```
// map is centered aroung given Lat, Long
String geoCode = “geo:41.5020952,-81.6789717&z=16”;
Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse(geoCode));
startActivity(intent);
```

Modify the Manifest adding the following requests:

```
<uses-permission android:name=“android.permission.ACCESS_COARSE_LOCATION”/>
<uses-permission android:name=“android.permission.INTERNET”/>
```

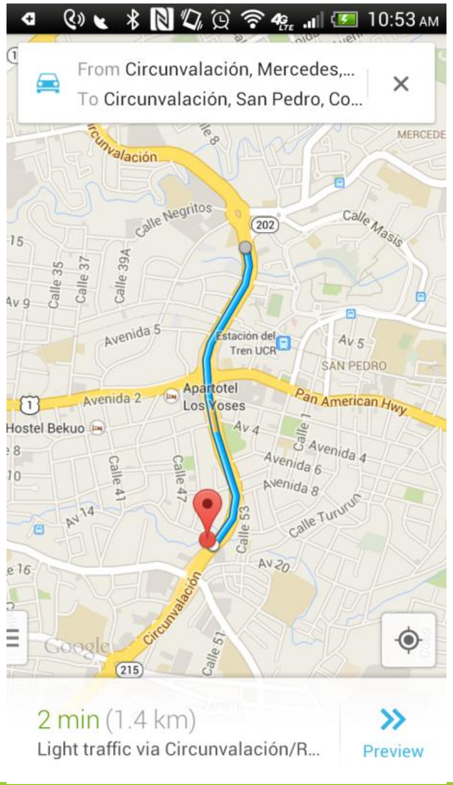

2:16 AM
Maps
E 23rd St
E 22nd St
Chester Bldg
Lot 62
Cleveland State University Campus Recreation Center
Chester Bldg Annex
Stilwell Hall
Cleveland State University
Lot UG
Fenn College of Engineering at CSU
Euclid Ave
Parker Hannifin Administration Center
E 24th St
Euclid Commons Bldg 2
Viking Hall
Lot KK
Prospect
Google
Heritage Suites Apartments
Ymca

INTRODUCTION (Example 10: ACTION_VIEW (Maps - Directions))

Getting driving directions

User provides GeoCodes (latitude,Longitude) for the starting and ending locations

```
Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
                                    Uri.parse(“http://maps.google.com/maps?”
                                                +“saddr=9.938083,-84.054430&”
                                                +“daddr=9.926392,-84.055964”));
startActivity(intent);
```


10:53 AM

From Circunvalación, Mercedes,...

To Circunvalación, San Pedro, Co...

2 min (1.4 km)

Light traffic via Circunvalación/R...

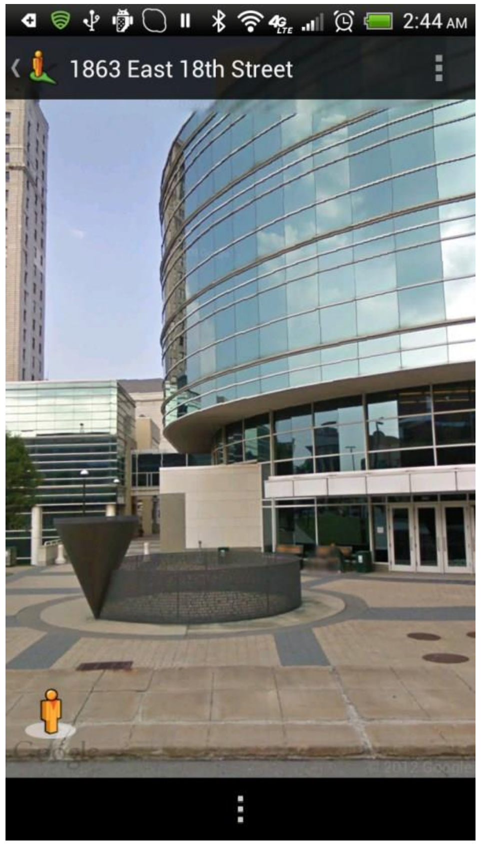
INTRODUCTION
(Example 11: ACTION_VIEW (Maps - StreetView))

GeoCode Uri structure: google.streetview:cbll=latitude,longitude&cbp=1,yaw,,pitch,zoom&mz=mapZoom

```
String geoCode = “google.streetview:”
                          + “cbll=41.5020952,-81.6789717&”
                          + “cbp=1,270,,45,1&mz=7”;
Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse(geoCode));
startActivity(intent);
```

Modify the Manifest adding the following requests:

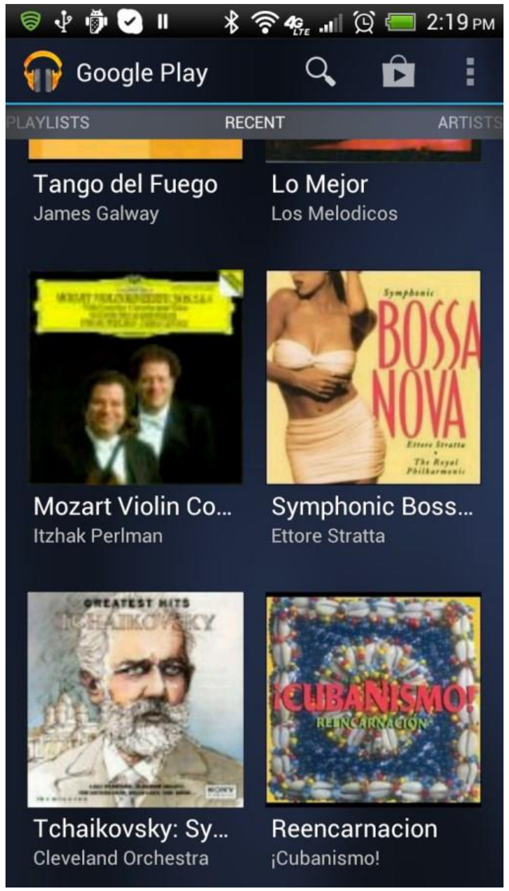
```
<uses-permission android:name=“android.permission.ACCESS_COARSE_LOCATION”/>
<uses-permission android:name=“android.permission.INTERNET”/>
```



INTRODUCTION (Example 12: ACTION_MUSIC_PLAYER)

Launching the Music Player

```
Intent myActivity2 = new Intent("android.intent.action.MUSIC_PLAYER");
startActivity(myActivity2);
```



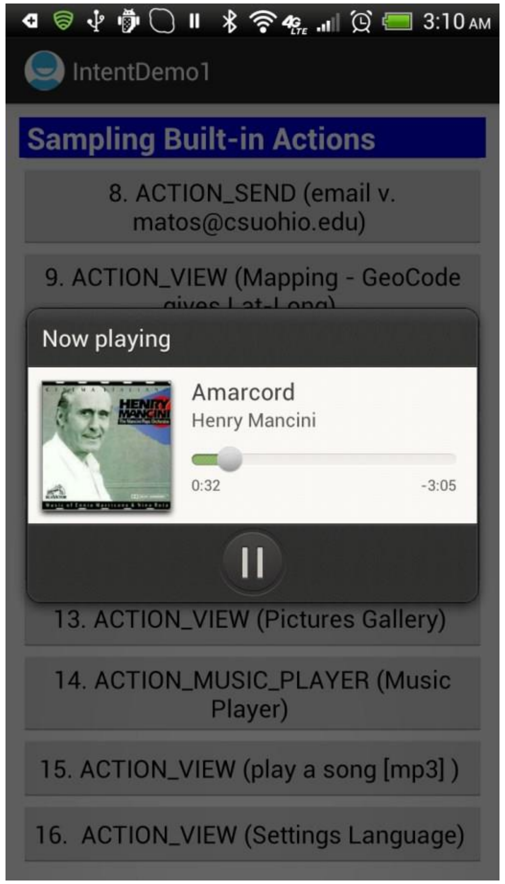
INTRODUCTION (Example 13: ACTION_VIEW (Music))

Playing a song stored in the SD card

```
Intent myActivity2 = new Intent(Intent.ACTION_VIEW);
Uri data = Uri.parse("file://" + Environment
                                  .getExternalStorageDirectory()
                                  .getAbsolutePath()
                                + "/Music/Amarcord.mp3");
myActivity2.setDataAndType(data, "audio/mp3");
startActivity(myActivity2);
```

Add to Manifest:

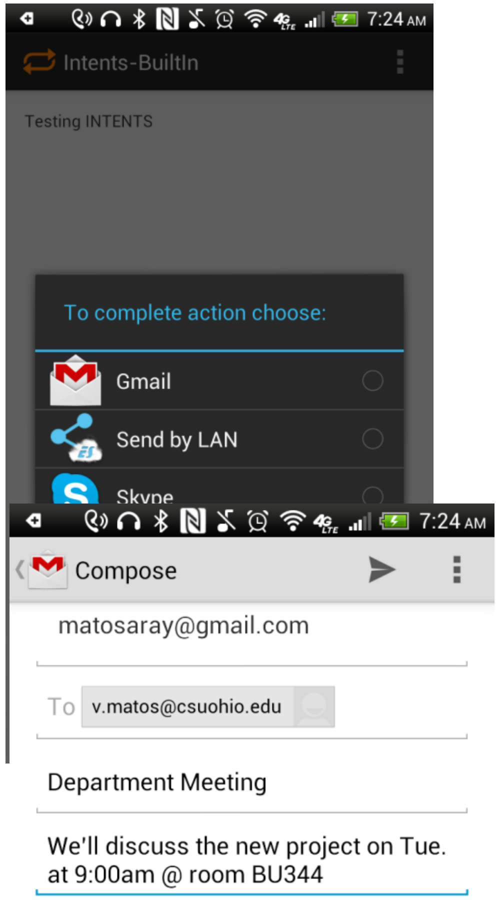
```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```


3:10 AM
IntentDemo1
Sampling Built-in Actions
8. ACTION_SEND (email v. matos@csuohio.edu)
9. ACTION_VIEW (Mapping - GeoCode gives Lat-Long)
Now playing
Amarcord
Henry Mancini
0:32
-3:05
13. ACTION_VIEW (Pictures Gallery)
14. ACTION_MUSIC_PLAYER (Music Player)
15. ACTION_VIEW (play a song [mp3])
16. ACTION_VIEW (Settings Language)

INTRODUCTION (Example 14: ACTION_SEND (Email))

Sending Email

```
// send email
String emailSubject = “Department Meeting”;
String emailText = “We’ll discuss the new project on Tue. at 9:00am @ room BU344”;
String[] emailReceiverList = {“v.matos@csuohio.edu”};
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType(“vnd.android.cursor.dir/email”);
intent.putExtra(Intent.EXTRA_EMAIL, emailReceiverList);
intent.putExtra(Intent.EXTRA_SUBJECT, emailSubject);
intent.putExtra(Intent.EXTRA_TEXT, emailText);
startActivity(Intent.createChooser(intent, “To complete action choose:”));
```



matosaray@gmail.com

To v.matos@csuohio.edu

Department Meeting

We'll discuss the new project on Tue. at 9:00am @ room BU344

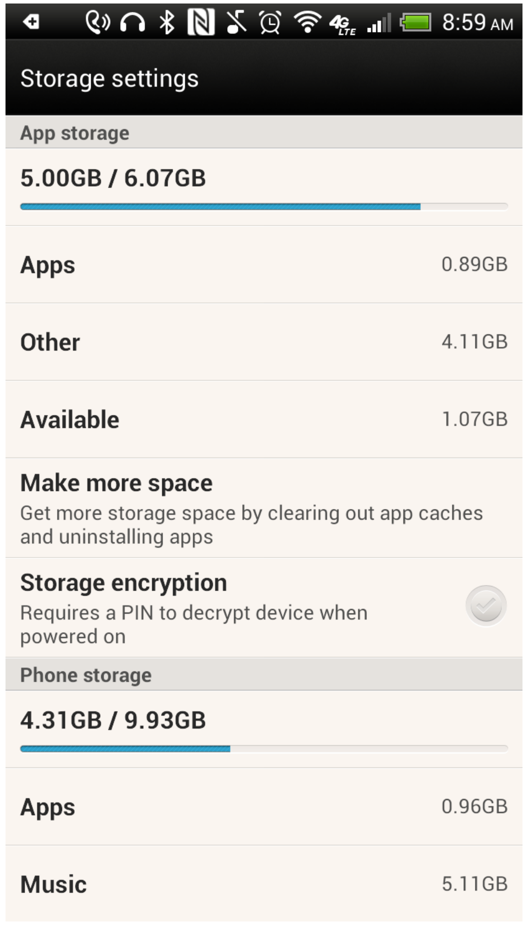
INTRODUCTION (Example 15: Device Settings)

System Settings

Almost all configurable features of an Android device can be accessed through built-in actions. For example ,an intent using android.provider.Settings.XXX

where XXX is as in Appendix A, invokes an app where the corresponding set of parameters defining XXX-settings could be adjusted. For a list of selected built-in actions see Appendix A.

```
startActivity(new Intent(android.provider.Settings.ACTION_INTERNAL_STORAGE_SETTINGS));
```



8:59 AM

Storage settings

App storage

5.00GB / 6.07GB

Apps	0.89GB
Other	4.11GB
Available	1.07GB

Make more space
Get more storage space by clearing out app caches and uninstalling apps

Storage encryption
Requires a PIN to decrypt device when powered on

Phone storage

4.31GB / 9.93GB

Apps	0.96GB
Music	5.11GB

Reference: https://developer.android.com/reference/android/provider/Settings.html

INTER-PROCESS COMMUNICATION

A typical Java program runs in a single thread. There, the program calls its methods using a synchronous stop-and-go protocol. Parameters are supplied to a called function, the caller passes control to the sub-routine, and waits for the function to complete. When it finally ends, the caller grabs any returned values , and proceeds with the rest of its work.

Android apps may include several independent but usually cooperative activities. Each activity works in its own thread with one of them designated as the Main.

Android uses The startActivity(Intent) method to initiate an activity, which will become active and (perhaps) visible; however the caller continues to execute in its own thread.

The next examples illustrate the basic inter-process communication mechanism used in Android for apps that consists of several collaborative activities. We will see how the calls are made, how input data is supplied to the called activity, and how results are returned to the caller.

INTER-PROCESS COMMUNICATION (Starting activities and getting results)

For a parent activity to trigger the execution of a child activity, and eventually get results back we use the method: startActivityForResult(Intent, requestCodeID)

Where requestCodeID is an arbitrary value you choose to identify the caller (similar to a ‘nickname’)

The results returned by the child-activity (if any) could be asynchronously picked up by a listener method defined in the parent activity: onActivityResult(requestCodeID, resultCode, Intent)

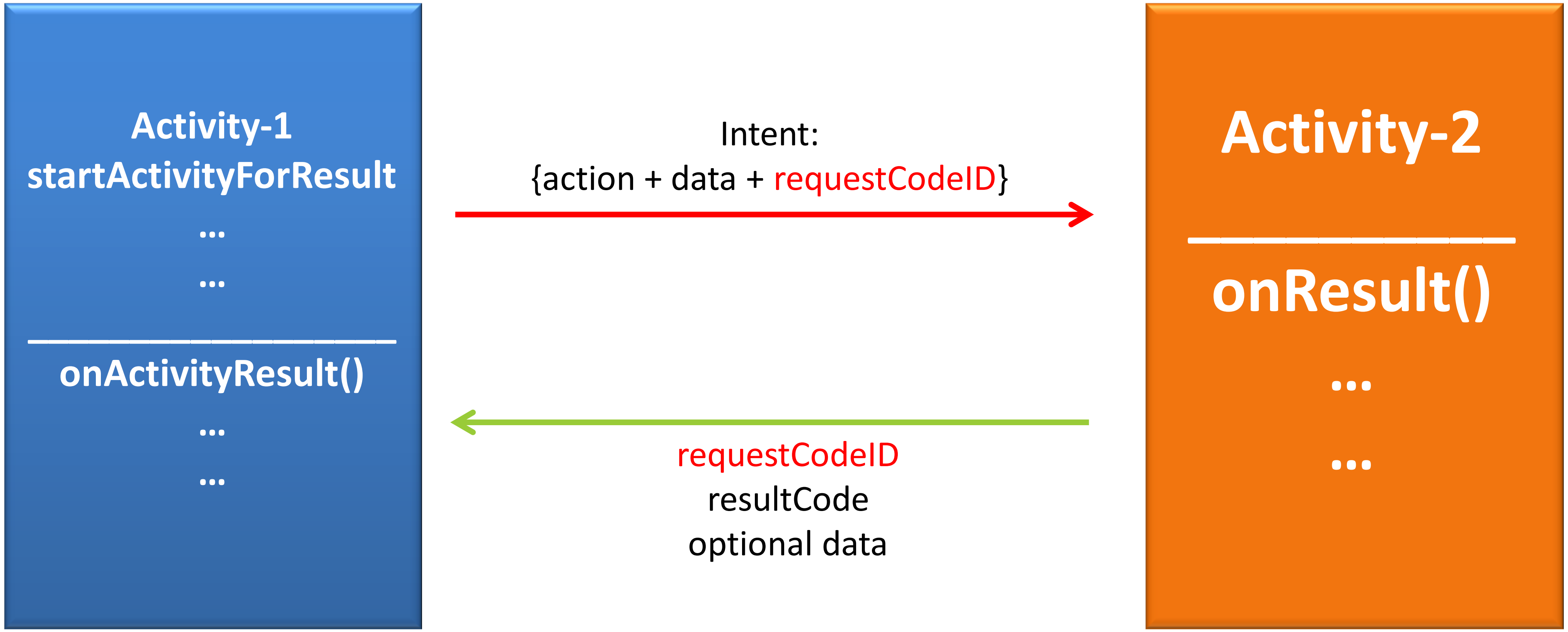
When the called activity is ready to finish, it could return an optional resultCode to the caller to summarize the success of its execution setResult(resultCode)

Standard resultCodes include

- Activity.RESULT_CANCELED (something bad happened),
- Activity.RESULT_OK (a happy ending),
- or any custom values.

The brief resultCode as well as any additional extra data can be collected back on the parent's using onActivityResult(int requestCodeID, int resultCode, Intent data)

INTER-PROCESS COMMUNICATION (Starting activities and getting results)

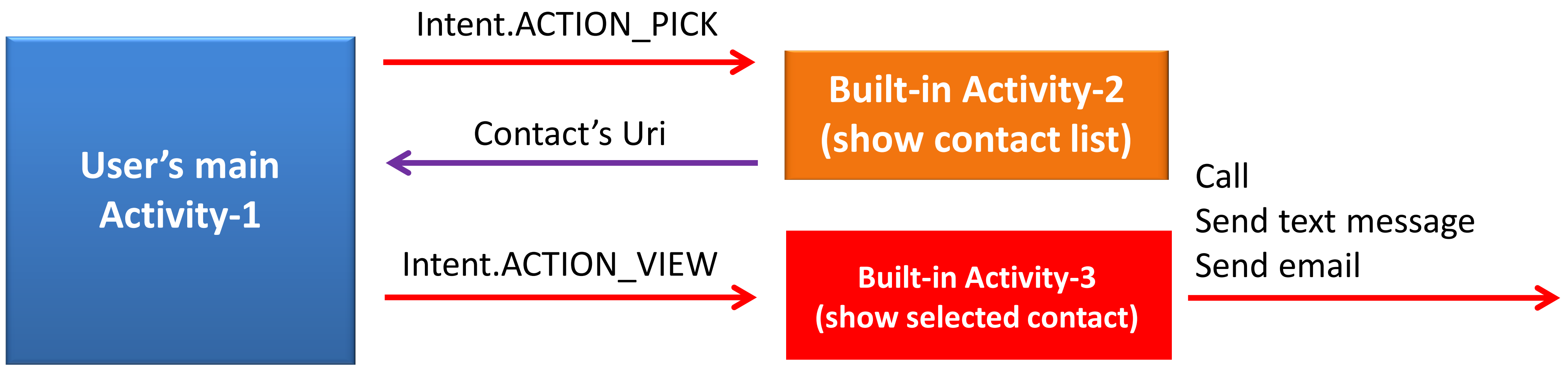


INTER-PROCESS COMMUNICATION (Example 16: Let’s play golf - Call for a tee-time)

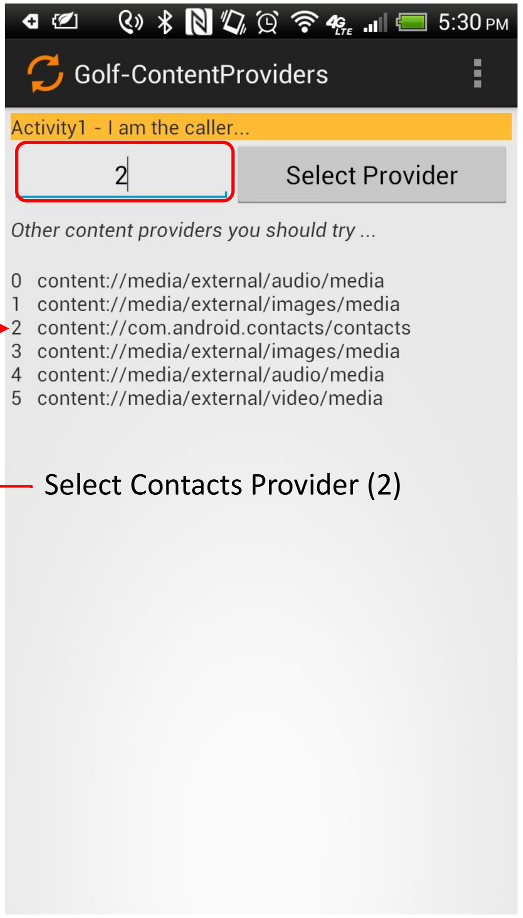
Show all our contacts and pick a particular golf course using the Intent.ACTION_PICK on the URI: android.provider.ContactsContract.Contacts.CONTENT_URI

Use the returned URI identifying the place we want to call for a tee-time reservation.

‘Nicely’ show the selected contact’s entry allowing calling, texting, emailing actions (use Intent.ACTION_VIEW).



INTER-PROCESS COMMUNICATION (Example 16: Let's play golf - Call for a tee-time)



Golf-ContentProviders

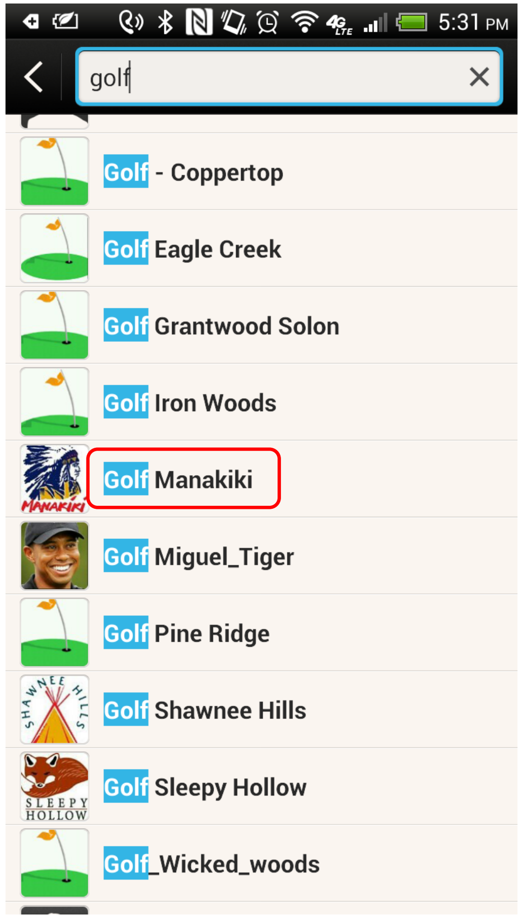
Activity1 - I am the caller...

2 Select Provider

Other content providers you should try ...

```
0  content://media/external/audio/media
1  content://media/external/images/media
2  content://com.android.contacts/contacts
3  content://media/external/images/media
4  content://media/external/audio/media
5  content://media/external/video/media
```

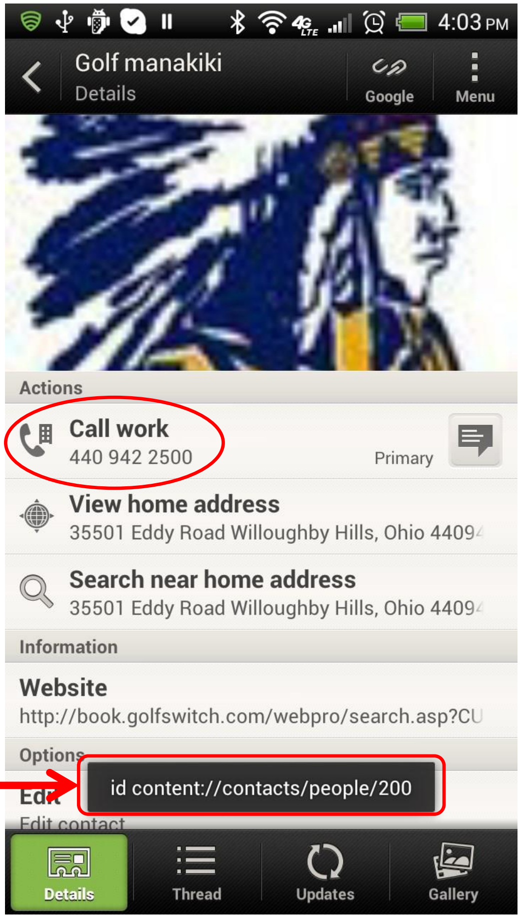
Select Contacts Provider (2)



golf

Golf - Coppertop
Golf Eagle Creek
Golf Grantwood Solon
Golf Iron Woods
Golf Manakiki
Golf Miguel_Tiger
Golf Pine Ridge
Golf Shawnee Hills
Golf Sleepy Hollow
Golf_Wicked_woods

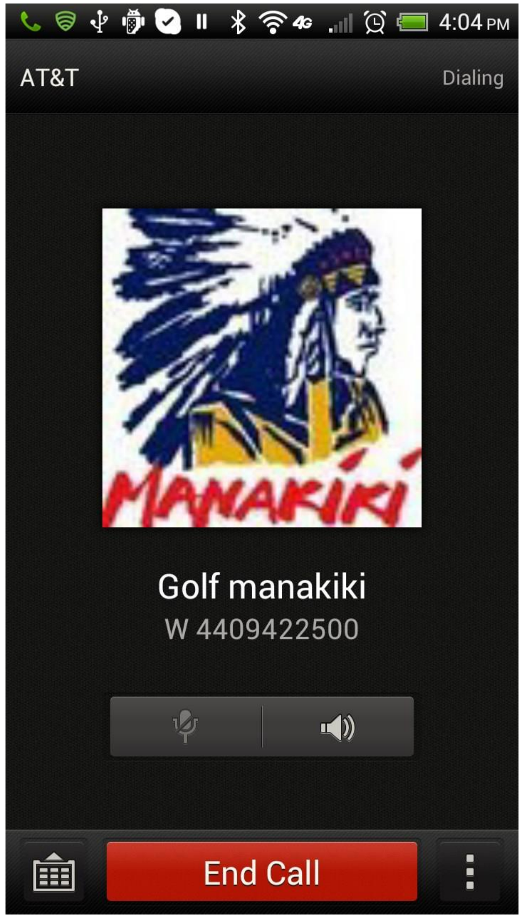
Complete the phone call



Golf manakiki
Details

Actions
Call work 440 942 2500 Primary
View home address 35501 Eddy Road Willoughby Hills, Ohio 4409
Search near home address 35501 Eddy Road Willoughby Hills, Ohio 4409
Information
Website http://book.golfswitch.com/webpro/search.asp?CU
Options
Edit contact

id content://contacts/people/200



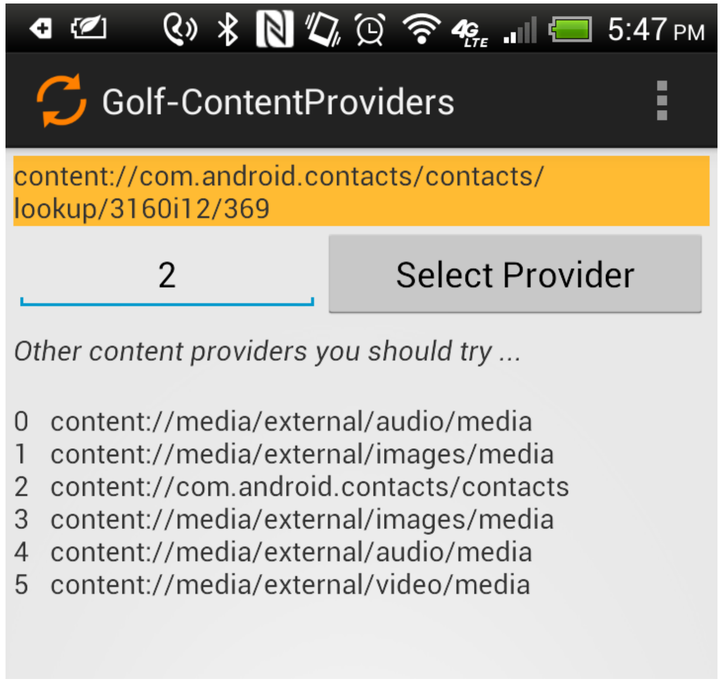
AT&T Dialing

Golf manakiki
W 4409422500

End Call

INTER-PROCESS COMMUNICATION (Example 16: Let’s play golf - Call for a tee-time)

The image on the left shows the URI returned from the interaction with the content provider. The path allows a lookup operation targeting the selected golf course. The image on the right shows the result of cancelling the search for data in the contact list.

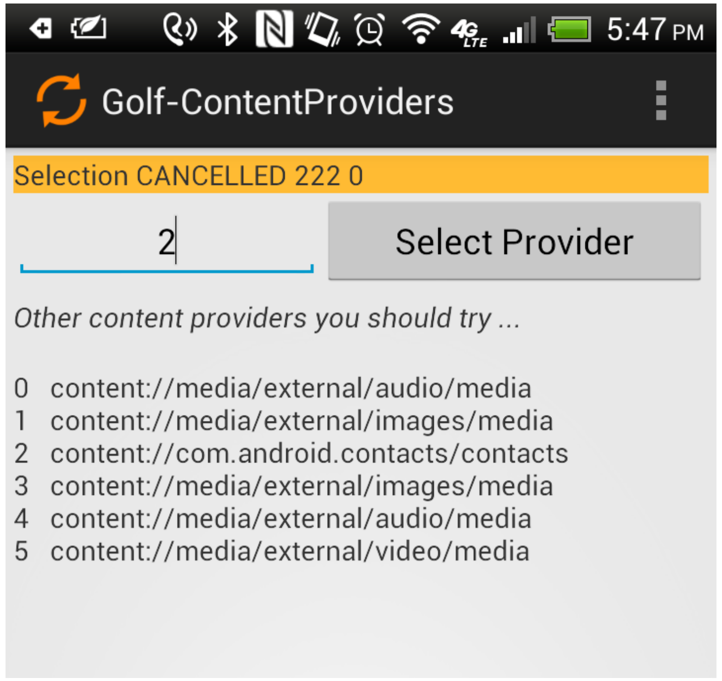


content://com.android.contacts/contacts/lookup/3160i12/369

2

Other content providers you should try ...

```
0  content://media/external/audio/media
1  content://media/external/images/media
2  content://com.android.contacts/contacts
3  content://media/external/images/media
4  content://media/external/audio/media
5  content://media/external/video/media
```



Selection CANCELLED 222 0

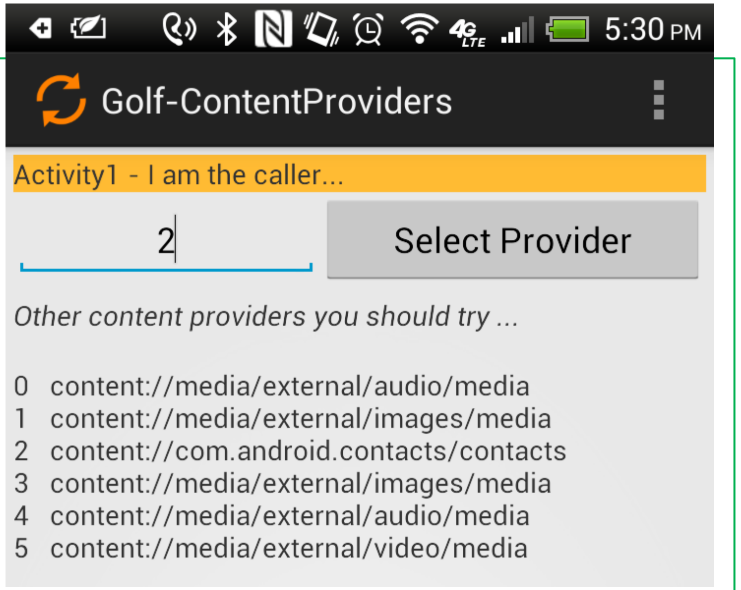
2

Other content providers you should try ...

```
0  content://media/external/audio/media
1  content://media/external/images/media
2  content://com.android.contacts/contacts
3  content://media/external/images/media
4  content://media/external/audio/media
5  content://media/external/video/media
```

INTER-PROCESS COMMUNICATION (Example 16: activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/LinearLayout1" android:layout_width="match_parent"
              android:layout_height="match_parent" android:orientation="vertical" android:padding="4dp" >
  <TextView android:id="@+id/txtMsg" android:layout_width="match_parent"
            android:layout_height="wrap_content" android:background="@android:color/holo_orange_light"
            android:text="Activity1 - I am the caller..." />
  <LinearLayout android:layout_width="match_parent" android:layout_height="wrap_content" >
    <EditText android:id="@+id/txtProviderOption" android:layout_width="wrap_content"
              android:layout_height="wrap_content" android:layout_weight="1"
              android:ems="5" android:gravity="center_horizontal"
              android:inputType="number" android:text="2">
      <requestFocus />
    </EditText>
    <Button android:id="@+id/btnOption" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_weight="1" android:text="Select Provider"/>
  </LinearLayout>
  <TextView android:layout_width="match_parent" android:layout_height="wrap_content"
            android:layout_marginTop="5dp" android:text="Other content providers you should try ..." android:textStyle="italic" />
  <TextView android:id="@+id/txtProviders" android:layout_width="match_parent" android:layout_height="wrap_content" android:text="" />
</LinearLayout>
```



Activity1 - I am the caller...

2

Other content providers you should try ...

```
0  content://media/external/audio/media
1  content://media/external/images/media
2  content://com.android.contacts/contacts
3  content://media/external/images/media
4  content://media/external/audio/media
5  content://media/external/video/media
```

INTER-PROCESS COMMUNICATION (Example 16: MainActivity.java)

```
public class MainActivity extends Activity {
  TextView txtMsg; EditText txtProvider, txtExample; Button btnCallActivity2;
  Uri[] uriProvider = {Uri.parse("content://media/external/audio/media"),
                       Uri.parse("content://media/external/images/media"),
                       android.provider.ContactsContract.Contacts.CONTENT_URI,
                       android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
                       android.provider.MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
                       android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI};
  @SuppressLint("NewApi")
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    try {
      txtMsg = (TextView) findViewById(R.id.txtMsg);
      txtProvider = (EditText) findViewById(R.id.txtProviderOption);
      // show some examples of built-in content providers
      txtExample = (EditText) findViewById(R.id.txtExamples);
      for (int i=0; i<uriProvider.length; i++) txtExample.append( "\n" + i + " " + uriProvider[i].toString());
      btnCallActivity2 = (Button) findViewById(R.id.btnOption);
      btnCallActivity2.setOnClickListener(new ClickHandler());
    }
    catch (Exception e) { Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();}
}// onCreate
```

```
private class ClickHandler implements OnClickListener {
  @Override
  public void onClick(View v) {
    try {// start myActivity2. Tell it that my nickname is 222
      int option = Integer.parseInt(txtProvider.getText().toString());
      Intent myActivity2 = new Intent( Intent.ACTION_PICK,uriProvider[option]);
      startActivityForResult(myActivity2, 222);
    }
    catch (Exception e) {
      Toast.makeText( getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show();
    }
  }// onClick
}// ClickHandler
```

INTER-PROCESS COMMUNICATION (Example 16: MainActivity.java)

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data){
  super.onActivityResult(requestCode, resultCode, data);
  try {// use requestCode to find out who is talking to us
    switch (requestCode) {
      case (222): {// 222 is our friendly contact-picker activity
        if (resultCode == Activity.RESULT_OK) {
          Toast.makeText(getApplication(), “id ” + data.getDataString(), 1).show();
          // it will return an URI that looks like: content://contacts/people/n, where n is the selected contact’s ID
          txtMsg.setText(returnedData.toString());
          // show a ‘nice’ screen with the selected contact
          Toast.makeText( getApplication(), “Nice UI for\n ” + returnedData, 1).show();
          Intent myAct3 = new Intent( Intent.ACTION_VIEW, Uri.parse(returnedData) );
          startActivity(myAct3);
        }
        else { // user pressed the BACK button to end called activity
          txtMsg.setText(“Selection CANCELLED ” + requestCode + “ ” + resultCode);
        }
        break;
      }
    }// switch
  }
  catch (Exception e) { Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show(); }
}// onActivityResult
}// MainActivity
```

INTER-PROCESS COMMUNICATION (Example 16: Comments)

The app offers a list of content providers to choose from (Contacts can be reached from: android.provider.ContactsContract.Contacts.CONTENT_URI)

An intent object is assembled combining ACTION_PICK with the chosen URI.

The caller app identifies itself with the requestCode 222, starts the intent and waits for ACTION_PICK to send results back to it.

The app's listener onActivityResult verifies that a resultCode was sent back to itself (222). Then it checks that ACTION_PICK satisfactorily completed its work and returned an Activity.RESULT_OK completion code.

The URI string coming back from ACTION_PICK can be obtained from the returned intent with data.getDataString().

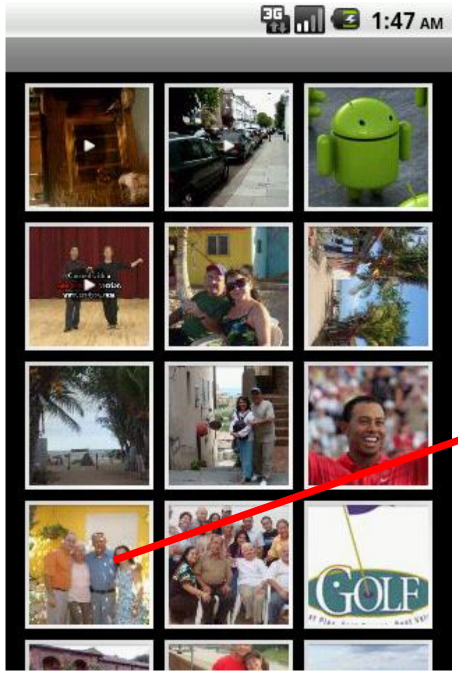
The previous result URI that can be passed to other actions, such as ACTION_VIEW. This will complete the user's request.

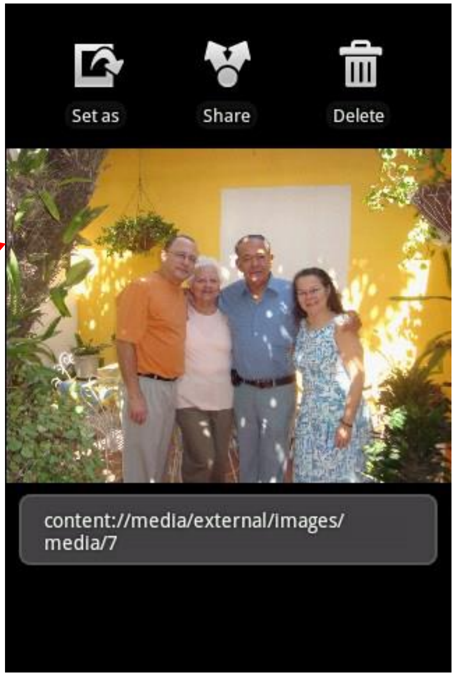
INTER-PROCESS COMMUNICATION (Example 16: Screenshots)

Showing Pictures and Video - for this example we selected:

- ACTION_PICK & content://media/external/images/media
- followed by ACTION_VIEW & content://media/external/images/media/media7

video





content://media/external/images/media/7

USING BUNDLES TO PASS DATA

A Bundle is an Android data-exchange mechanism used for efficient interprocess communications on either in-process or cross-process calls.

A Bundle is conceptually similar to a Java HashMap. It associates a string key to a parcelable (exchangeable) data element. Data could be either primitive data types or object-references. Bundles are functionally equivalent to a collection of <name, value> pairs.

There is a set of putXXX and getXXX methods to store and retrieve (single and array) values of primitive data types from/to the bundles. For example

```
Bundle myBundle = new Bundle();
myBundle.putDouble (“var1”, 3.1415);
…
Double v1 = myBundle.getDouble(“var1”);
```

USING BUNDLES TO PASS DATA (Intents and bundles - Calling a receiver)

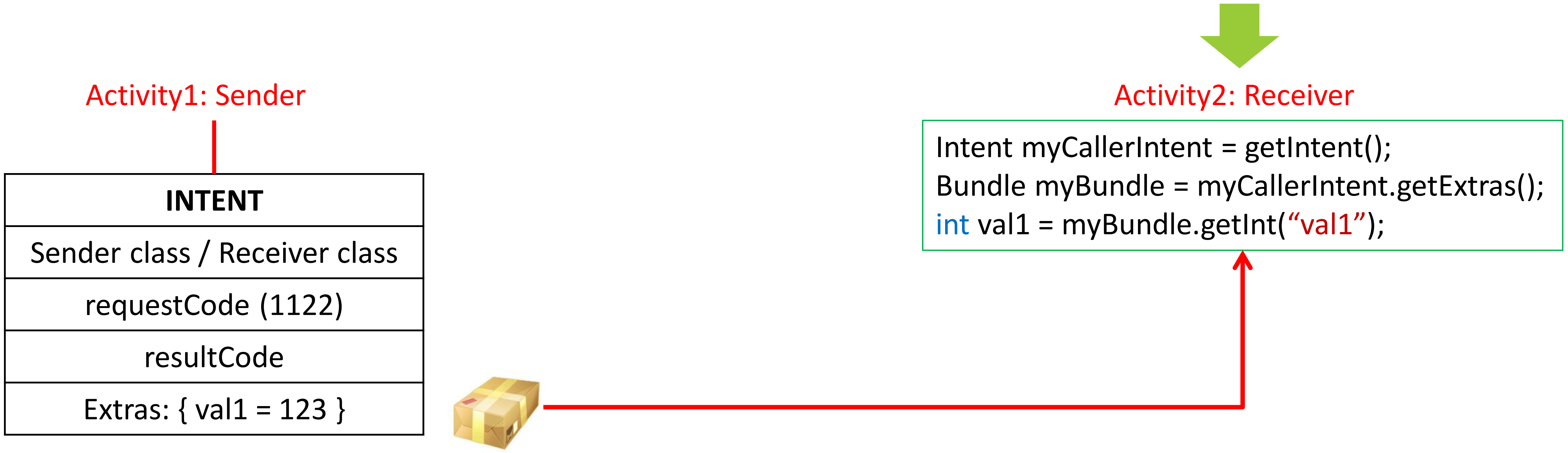
A single Bundle could contain an unlimited number of <key, value> items. They offer an elegant solution to Android IPC exchanges; observe it is sufficient to attach a single extra bundle to an intent for two interacting activities to move any amount of data.

Activity1: Sender $\longrightarrow$ Activity2: Receiver

```
Intent myIntentA1A2 = new Intent (Activity1.this, Activity2.class);
Bundle myBundle1 = new Bundle();
myBundle1.putInt ("val1", 123);
myIntentA1A2.putExtras(myBundle1);
startActivityForResult(myIntentA1A2, 1122);
```

INTENT
Sender class / Receiver class
requestCode (1122)
resultCode
Extras: { val1 = 123 }

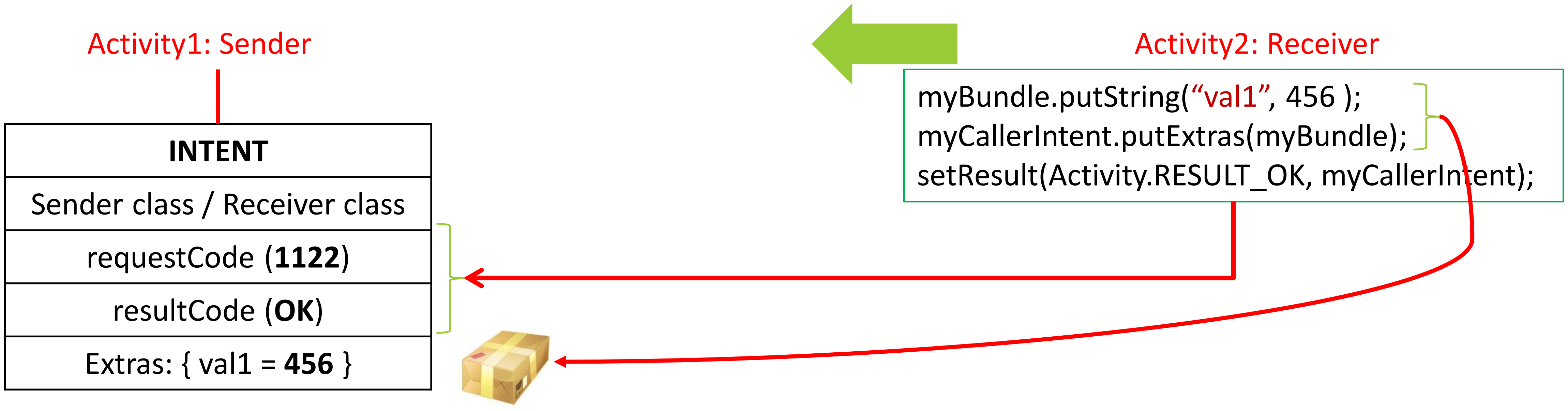
USING BUNDLES TO PASS DATA (Intents and bundles - Receiver is awoken)



INTENT
Sender class / Receiver class
requestCode (1122)
resultCode
Extras: { val1 = 123 }

```
Intent myCallerIntent = getIntent();
Bundle myBundle = myCallerIntent.getExtras();
int val1 = myBundle.getInt(“val1”);
```

USING BUNDLES TO PASS DATA (Intents and bundles - Receiver is awoken)



```
myBundle.putString("val1", 456 );
myCallerIntent.putExtras(myBundle);
setResult(Activity.RESULT_OK, myCallerIntent);
```

USING BUNDLES TO PASS DATA (Common bundle methods)

.clear(): removes all elements from the mapping of this Bundle.

.clone(): clones the current Bundle.

.containsKey(String key): returns true if the given key is contained in the mapping of this Bundle.

.putIntArray(String key, int[] value)/.getIntArray(String key): inserts/replaces/retrieves an int array value into the mapping of this Bundle

.putString(String key, String value)/.getString(String key): inserts/replaces/retrieves a String value into the mapping of this Bundle

.putStringArray(String key, String[] value)/.getStringArray(String key): inserts/replaces/retrieves a String array value into the mapping of this Bundle

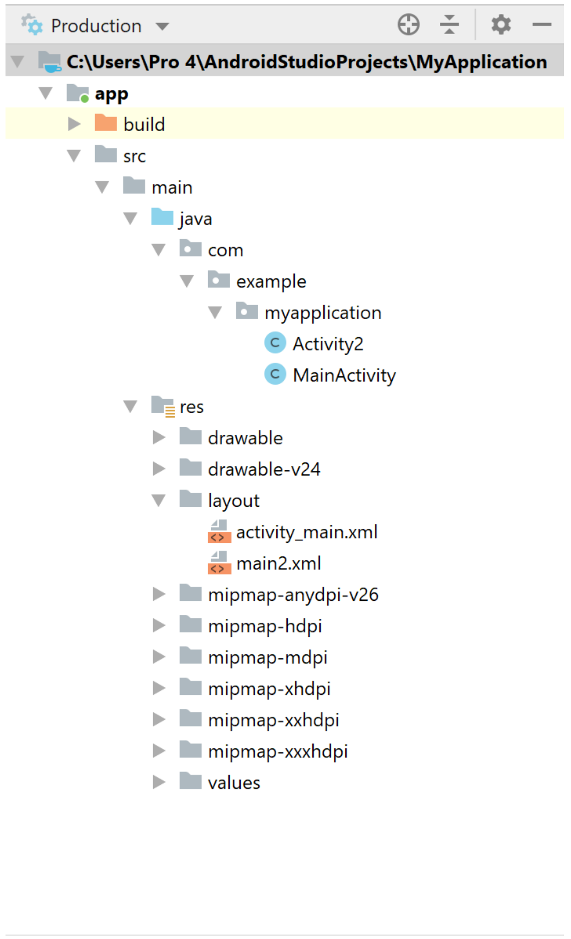
.putStringArrayList(String key, ArrayList<String> value): inserts/replaces an ArrayList value into the mapping of this Bundle.

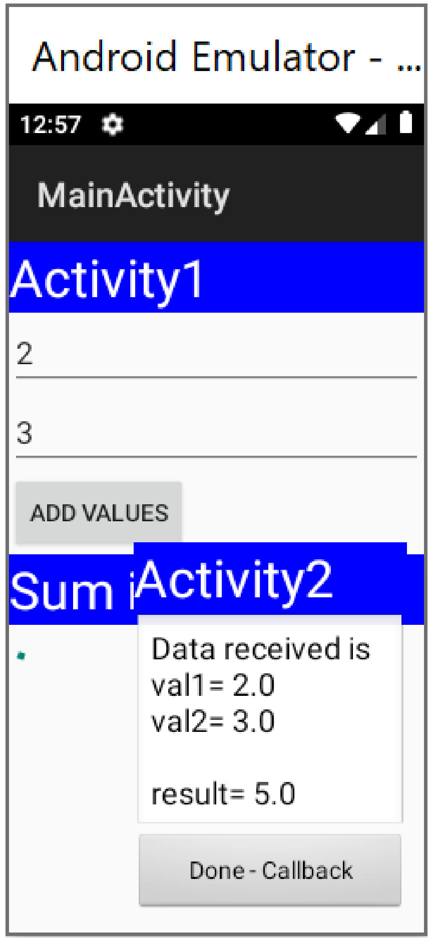
.remove(String key): removes any entry with the given key from the mapping of this Bundle.

. size(): returns the number of mappings contained in this Bundle.

Reference: http://developer.android.com/reference/android/os/Bundle.html

USING BUNDLES TO PASS DATA (Example 17: XML Layout – activity_main.xml)



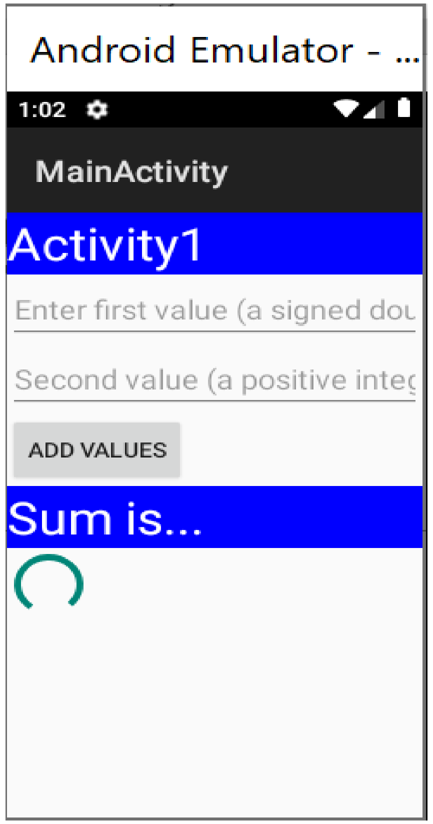


Data received is
val1= 2.0
val2= 3.0

result= 5.0

In this example Activity1 passes two numbers to Activity2 which will add them up and return the result. All data is passed back and forth in a Bundle.

USING BUNDLES TO PASS DATA (Example 17: Screenshots)


Android Emulator - ...

1:02

MainActivity

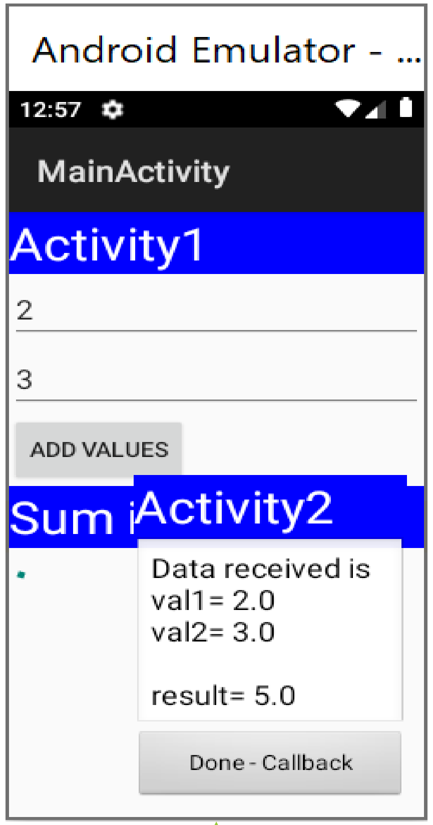
Activity1

Enter first value (a signed dou

Second value (a positive integ

ADD VALUES

Sum is...


Android Emulator - ...

12:57

MainActivity

Activity1

2

3

ADD VALUES

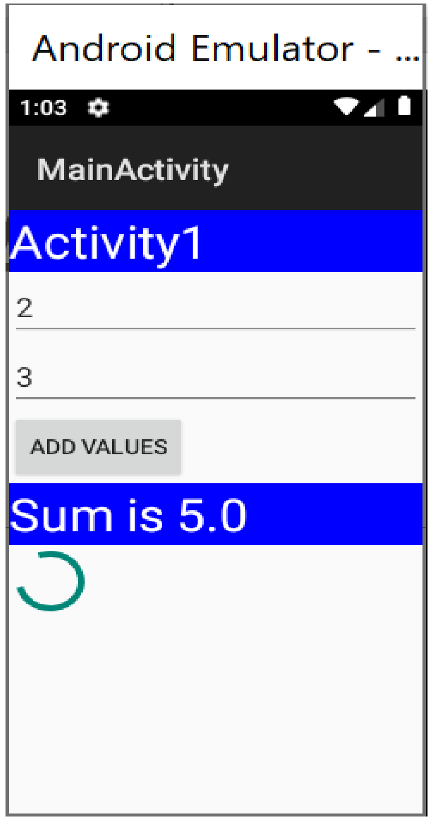
Sum i

Activity2

Data received is
val1= 2.0
val2= 3.0

result= 5.0

Done - Callback


Android Emulator - ...

1:03

MainActivity

Activity1

2

3

ADD VALUES

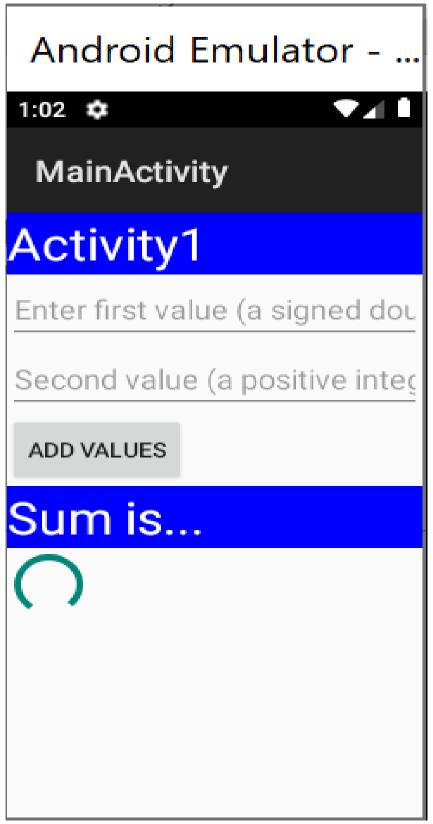
Sum is 5.0

Data values collected by Activity1 are placed into a bundle and sent to Activity2. Observe that progress bar in Activity1 remains active however this process has no focus (its button and textboxes do not respond) Activity2 is visible and has focus.

Activity1 receives the result of the operation carried out by Activity2.

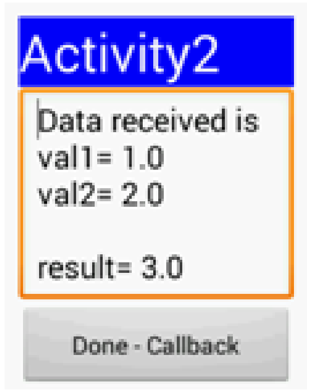
USING BUNDLES TO PASS DATA (Example 17: XML Layout – MainActivity.xml)

```
<?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:orientation="vertical" >
 <TextView      android:layout_width="match_parent" android:layout_height="wrap_content"
                    android:background="#ff0000ff" android:text="Activity1"
                    android:textColor="#ffffffff" android:textSize="30sp" />
 <EditText      android:id="@+id/EditText01" android:layout_width="match_parent"
                    android:layout_height="wrap_content" android:hint="Enter first value (a signed double)"
                    android:inputType="numberDecimal|numberSigned|number" />
 <EditText      android:id="@+id/EditText02" android:layout_width="match_parent"
                    android:layout_height="wrap_content" android:hint="Second value (a positive integer)"
                    android:inputType="number" />
 <Button        android:id="@+id/btnAdd" android:layout_width="wrap_content"
                    android:layout_height="wrap_content" android:text="Add Values" />
 <TextView      android:id="@+id/txtResult" android:layout_width="match_parent"
                    android:layout_height="wrap_content" android:background="#ff0000ff"
                    android:text="Sum is..." android:textColor="#ffffffff" android:textSize="30sp" />
 <ProgressBar android:id="@+id/progressBar1"
                   android:layout_width="wrap_content"
                   android:layout_height="wrap_content" />
</LinearLayout>
```


Android Emulator - ...
1:02
MainActivity
Activity1
Enter first value (a signed dou
Second value (a positive integ
ADD VALUES
Sum is...

USING BUNDLES TO PASS DATA (Example 17: XML Layout – main2.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_gravity="right|bottom"
              android:layout_margin="10dp"
              android:background="@android:color/transparent"
              android:orientation="vertical" >
 <TextView    android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:background="#ff0000ff"
              android:text="Activity2"
              android:textColor="#ffffffff"
              android:textSize="30sp" />
 <EditText    android:id="@+id/etDataReceived"
              android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:text="Data reveived..." />
 <Button      android:id="@+id/btnDone"
              android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:text="Done – Callback" />
</LinearLayout>
```


Activity2

```
Data received is
val1= 1.0
val2= 2.0

result= 3.0
```

USING BUNDLES TO PASS DATA (Example 17: MainActivity.java)

```
public class MainActivity extends Activity {
 EditText txtValue1, txtValue2; TextView txtResult; Button btnAdd;
 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main1);
  txtValue1 = (EditText)findViewById(R.id.EditText01); txtValue2 = (EditText)findViewById(R.id.EditText02); txtResult = (TextView) findViewById(R.id.txtResult);
  btnAdd = (Button) findViewById(R.id.btnAdd);
  btnAdd.setOnClickListener(new OnClickListener() {
   @Override
   public void onClick(View v) {
    // get values from the UI
    Double v1 = Double.parseDouble(txtValue1.getText().toString()), v2 = Double.parseDouble(txtValue2.getText().toString());
    // create intent to call Activity2
    Intent myIntentA1A2 = new Intent (MainActivity.this, Activity2.class);
    // create a Bundle (MAP) container to ship data & add <key,value> data items to the container
    Bundle myDataBundle = new Bundle(); myDataBundle.putDouble("val1", v1); myDataBundle.putDouble("val2", v2);
    // attach the container to the intent
    myIntentA1A2.putExtras(myDataBundle);
    // call Activity2, tell your local listener to wait a response sent to a listener known as 101
    startActivityForResult(myIntentA1A2, 101);
   }});
 }//onCreate
```

```
 @Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  try {
   if ((requestCode == 101 ) && (resultCode == Activity.RESULT_OK)){
    Bundle myResultBundle = data.getExtras();
    txtResult.setText("Sum is " + myResultBundle.getDouble("vresult"));
   }
  }
  catch (Exception e) {
   txtResult.setText("Problems – " + requestCode + " " + resultCode);
  }
 }//onActivityResult
}//MainActivity
```

USING BUNDLES TO PASS DATA (Example 17: Activity2.java)

```
public class Activity2 extends Activity implements OnClickListener{
  EditText dataReceived; Button btnDone;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main2);
    dataReceived = (EditText) findViewById(R.id.etDataReceived); btnDone = (Button) findViewById(R.id.btnDone);
    btnDone.setOnClickListener(this);
    // pick call made to Activity2 via Intent
    Intent myLocalIntent = getIntent();
    // look into the bundle sent to Activity2 for data items
    Bundle myBundle = myLocalIntent.getExtras();
    Double v1 = myBundle.getDouble("val1"), v2 = myBundle.getDouble("val2");
    // operate on the input data
    Double vResult = v1 + v2;
    // for illustration purposes. show data received & result
    dataReceived.setText("Data received is \n" + "val1= " + v1 + "\nval2= " + v2 + "\n\nresult= " + vResult);
    // add to the bundle the computed result & attach updated bumble to invoking intent
    myBundle.putDouble("vresult", vResult);
    myLocalIntent.putExtras(myBundle);
    // return sending an OK signal to calling activity
    setResult(Activity.RESULT_OK, myLocalIntent);
  }//onCreate
  @Override public void onClick(View v) { finish(); }
}
```

USING BUNDLES TO PASS DATA (Example 17: Manifest.xml)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example.intentdemo2b"
          android:versionCode="1"
          android:versionName="1.0" >
 <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="18" />
 <application android:icon="@drawable/ic_launcher" android:label="@string/app_name" android:theme="@style/AppTheme" >
  <activity android:name=".Activity1" android:label="@string/title_activity_intent_demo2_b" >
   <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
  </activity>
  <activity android:name=".Activity2" android:theme="@android:style/Theme.Translucent.NoTitleBar"/>
 </application>
</manifest>
```

USING BUNDLES TO PASS DATA (Example 17: Comments)

The continuous rotation of this circular progress bar will visibly indicate the working state of Activity1.

Activity2 has a small layout and a transparent background. When displayed it will be partially super-imposed on top of Activity1's screen.

Activity1 prepares an Intent to invoke Activity2. The statement myIntentA1A2.putExtras(myDataBundle) attaches the bundle to the intent.

startActivityForResult(…) passes the intent and its id 101 to Activity2.

The listener in Activity1 waits for the result wrapped into an extra bundle. When it arrives it is extracted and displayed to the user.

Activity2 issues .getIntent() to grab the incoming intent and the extra bundle it carries. The two numeric variables are combined to produce a result (vResult).

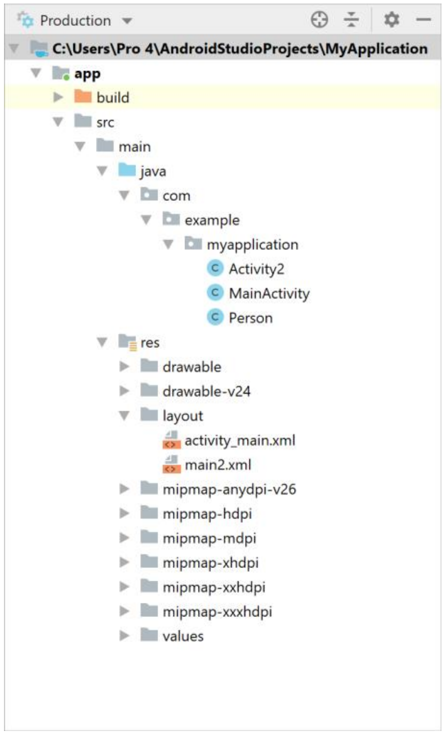
Activity2 issues .putDouble(…) to store the result into the outgoing bundle.

Activity2 releases the outgoing bundle together with a RESULT_OK flag.

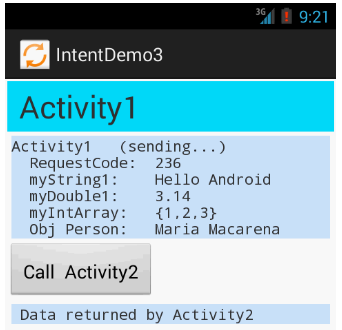
The manifest defines both activities and applies a translucent, NoTitleBar theme to Activity2.

USING BUNDLES TO PASS DATA (Example 18: Exchanging data between 2 activities in the same app)

In this example the caller Activity1 sends a variety of arguments to Activity2 including simple types, arrays, and serialized objects. Activity2 applies changes on the input data and returns new values. Both activities are part of the same app.

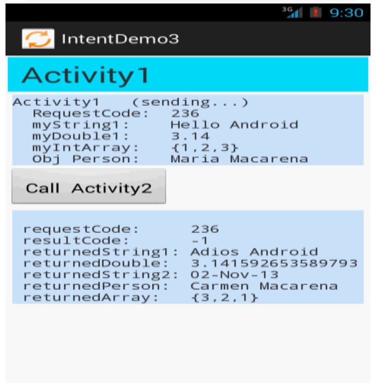


```
Production
C:\Users\Pro 4\AndroidStudioProjects\MyApplication
  app
    build
    src
      main
        java
          com
            example
              myapplication
                Activity2
                MainActivity
                Person
        res
          drawable
          drawable-v24
          layout
            activity_main.xml
            main2.xml
          mipmap-anydpi-v26
          mipmap-hdpi
          mipmap-mdpi
          mipmap-xhdpi
          mipmap-xxhdpi
          mipmap-xxxhdpi
          values
```



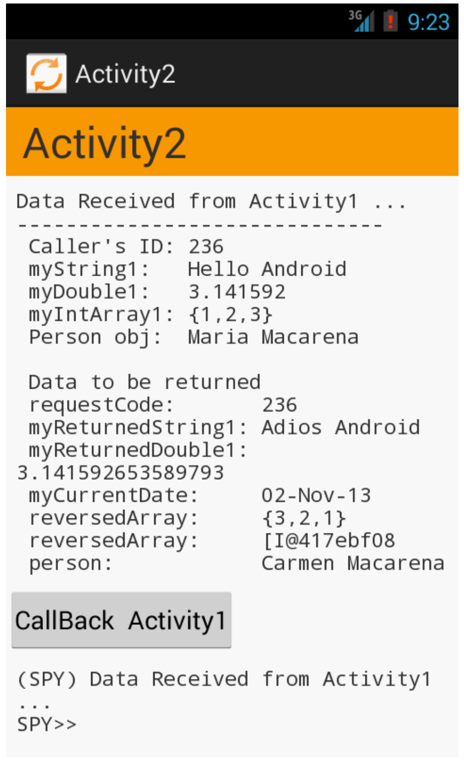
```
9:21
IntentDemo3
Activity1
Activity1   (sending...)
  RequestCode:   236
  myString1:     Hello Android
  myDouble1:     3.14
  myIntArray:    {1,2,3}
  Obj Person:    Maria Macarena
[Call Activity2]
Data returned by Activity2
```

Send all of these items to Activity2 and wait for results

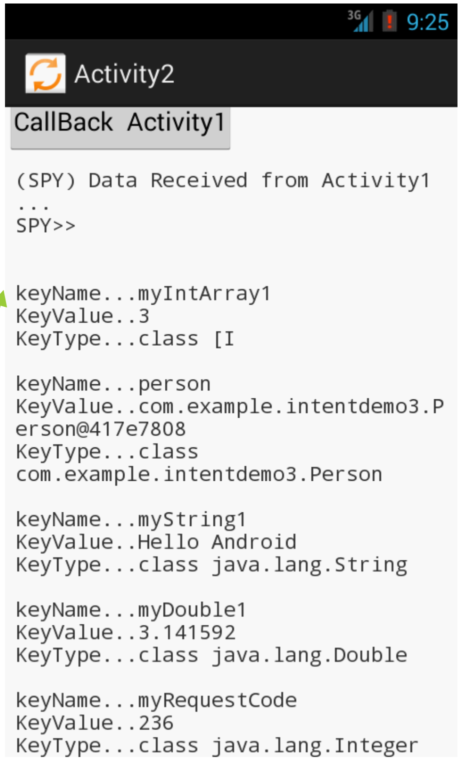


```
9:30
IntentDemo3
Activity1
Activity1   (sending...)
  RequestCode:   236
  myString1:     Hello Android
  myDouble1:     3.14
  myIntArray:    {1,2,3}
  Obj Person:    Maria Macarena
[Call Activity2]

 requestCode:       236
 resultCode:        -1
 returnedString1:  Adios Android
 returnedDouble:   3.141592653589793
 returnedString2:  02-Nov-13
 returnedPerson:   Carmen Macarena
 returnedArray:    {3,2,1}
```

Returned values sent back from Activity2



```
9:23
Activity2
Activity2
Data Received from Activity1 ...
------------------------------
 Caller's ID: 236
 myString1:   Hello Android
 myDouble1:   3.141592
 myIntArray1: {1,2,3}
 Person obj:  Maria Macarena

 Data to be returned
 requestCode:          236
 myReturnedString1:  Adios Android
 myReturnedDouble1:
3.141592653589793
 myCurrentDate:        02-Nov-13
 reversedArray:        {3,2,1}
 reversedArray:        [I@417ebf08
 person:               Carmen Macarena
[CallBack Activity1]

(SPY) Data Received from Activity1
...
SPY>>
```



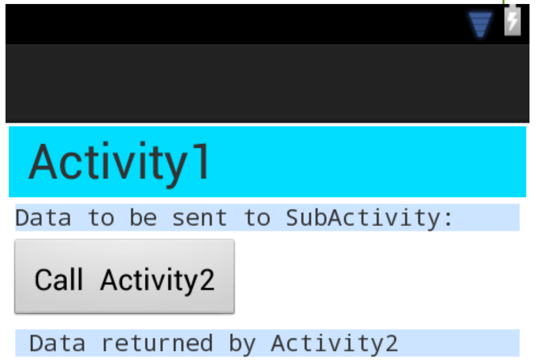
```
9:25
Activity2
[CallBack Activity1]

(SPY) Data Received from Activity1
...
SPY>>

keyName...myIntArray1
KeyValue..3
KeyType...class [I

keyName...person
KeyValue..com.example.intentdemo3.P
erson@417e7808
KeyType...class
com.example.intentdemo3.Person

keyName...myString1
KeyValue..Hello Android
KeyType...class java.lang.String

keyName...myDouble1
KeyValue..3.141592
KeyType...class java.lang.Double

keyName...myRequestCode
KeyValue..236
KeyType...class java.lang.Integer
```

Echo data received
Do local operations (reverse array, change Person's name, return PI and current date)

Explore Bundle to obtain:
keyName, keyValue, & keyType of each arriving item

USING BUNDLES TO PASS DATA (Example 18: XML Layout – activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="match_parent"
           android:layout_height="match_parent" android:layout_margin="2dp" >
 <LinearLayout android:layout_width="match_parent" android:layout_height="wrap_content" android:orientation="vertical" >
  <TextView android:layout_width="match_parent" android:layout_height="wrap_content"
            android:background="@color/holo_blue_bright" android:padding="4sp"
            android:text="Activity1" android:textSize="30sp" />
  <TextView android:id="@+id/txtTop" android:layout_width="match_parent"
            android:layout_height="wrap_content" android:layout_margin="4dip"
            android:background="#330077ff" android:text="Data to be sent to SubActivity:" android:typeface="monospace"/>
  <Button   android:id="@+id/btnCallActivity2" android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:background="@android:drawable/btn_default"
            android:padding="15dp" android:text="Call Activity2" />
  <TextView android:id="@+id/txtReturnedValues" android:layout_width="match_parent"
            android:layout_height="wrap_content" android:layout_margin="4dip"
            android:background="#330077ff" android:text="Data returned by Activity2" android:typeface="monospace" />
 </LinearLayout>
</ScrollView>
```



Activity1

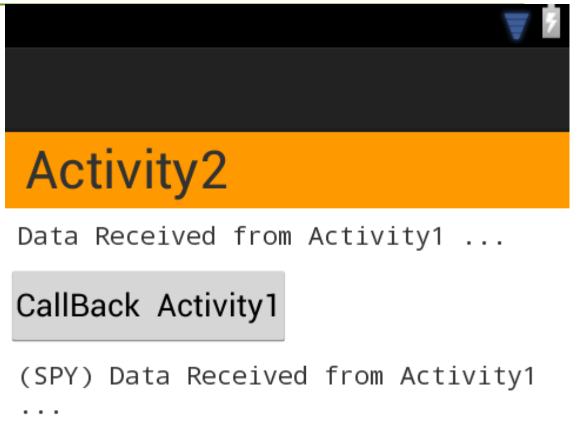
Data to be sent to SubActivity:

Call Activity2

Data returned by Activity2

USING BUNDLES TO PASS DATA (Example 18: XML Layout – main2.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="match_parent"
            android:layout_height="match_parent">
 <LinearLayout android:layout_width="match_parent" android:layout_height="match_parent"
               android:background="#ffffffff" android:orientation="vertical" >
  <TextView    android:layout_width="match_parent" android:layout_height="wrap_content"
               android:background="#ffff9900" android:padding="4sp"
               android:text="Activity2" android:textSize="30sp" />
  <TextView    android:id="@+id/txtIncomingData" android:layout_width="match_parent"
               android:layout_height="wrap_content" android:layout_margin="7dip"
               android:text="Data Received from Activity1 ..." android:typeface="monospace" />
  <Button      android:id="@+id/btnCallActivity1" android:layout_width="wrap_content"
               android:layout_height="wrap_content" android:padding="6sp" android:text="CallBack Activity1" />
  <TextView    android:id="@+id/spyBox" android:layout_width="match_parent"
               android:layout_height="wrap_content" android:layout_margin="7dip"
               android:text="(SPY) Data Received from Activity1 ..." android:typeface="monospace" />
</LinearLayout>
</ScrollView>
```


Activity2

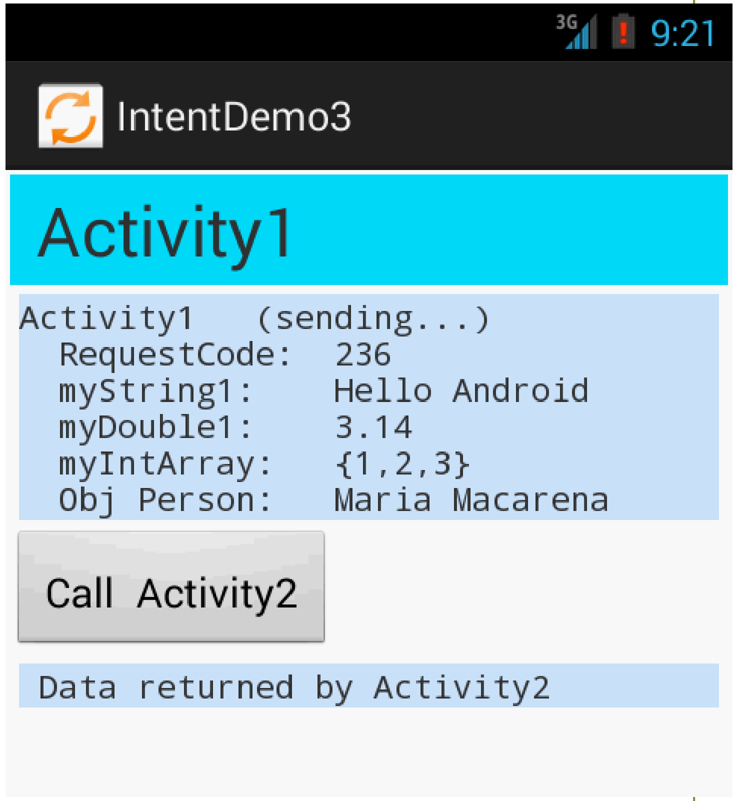
Data Received from Activity1 ...

CallBack Activity1

(SPY) Data Received from Activity1 ...

USING BUNDLES TO PASS DATA (Example 18: MainActivity.java – caller)

```
public class Activity1 extends Activity {
  TextView txtTop, txtReturnedValues;
  Button btnCallActivity2;
  // arbitrary interprocess communication ID (just a nickname!)
  private final int IPC_ID = (int) (10001 * Math.random());
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    try {
      setContentView(R.layout.main1);
      txtTop = (TextView) findViewById(R.id.txtTop);
      txtReturnedValues = (TextView) findViewById(R.id.txtReturnedValues);
      btnCallActivity2 = (Button) findViewById(R.id.btnCallActivity2);
      btnCallActivity2.setOnClickListener(new Clicker1());
      // for demonstration purposes- show in top textBox
      txtTop.setText("Activity1 (sending...)" + "\n RequestCode: " + IPC_ID
                                               + "\n myString1: Hello Android"
                                               + "\n myDouble1: 3.14 "
                                               + "\n myIntArray: {1,2,3} "
                                               + "\n Person: Maria Macarena");
    }
    catch (Exception e) { Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show(); }
}// onCreate
```

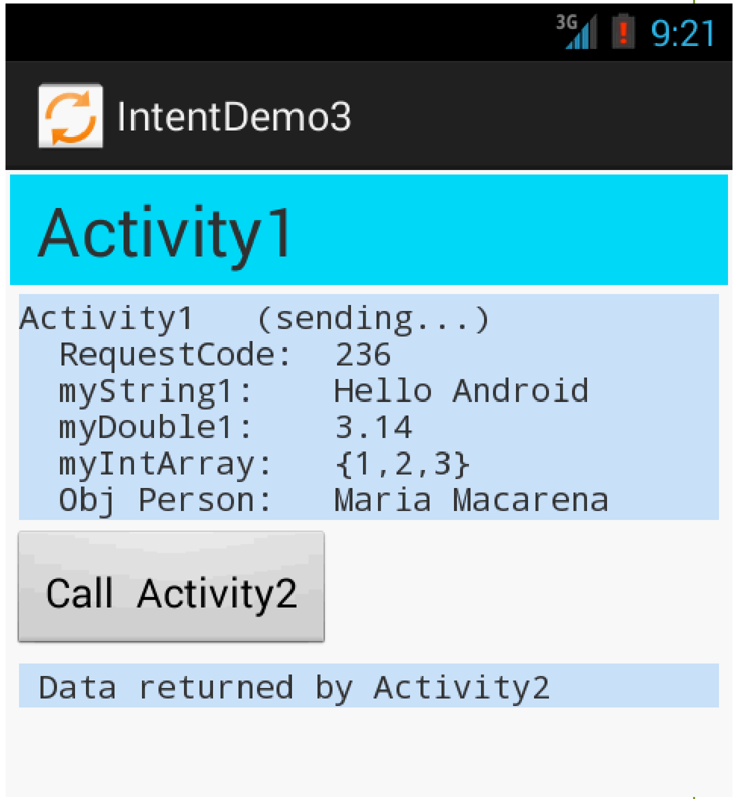


```
Activity1   (sending...)
  RequestCode:  236
  myString1:    Hello Android
  myDouble1:    3.14
  myIntArray:   {1,2,3}
  Obj Person:   Maria Macarena
```

```
Data returned by Activity2
```

USING BUNDLES TO PASS DATA (Example 18: MainActivity.java – caller)

```
private class Clicker1 implements OnClickListener {
 public void onClick(View v) {
  try {
   // create an Intent to talk to Activity2
   Intent myIntentA1A2 = new Intent(Activity1.this, Activity2.class);
   // prepare a Bundle and add the data pieces to be sent
   Bundle myData = new Bundle();
   myData.putInt(“myRequestCode”, IPC_ID);
   myData.putString(“myString1”, "Hello Android");
   myData.putDouble(“myDouble1”, 3.141592);
   int [] myLittleArray = { 1, 2, 3 };
   myData.putIntArray(“myIntArray1”, myLittleArray);
   // creating an object and passing it into the bundle
   Person p1 = new Person(“Maria”, “Macarena”);
   myData.putSerializable(“person”, p1);
   // bind the Bundle and the Intent that talks to Activity2
   myIntentA1A2.putExtras(myData);
   // call Activity2 and wait for results
   startActivityForResult(myIntentA1A2, IPC_ID);
  }
  catch (Exception e) { Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show(); }
 } // onClick
} // Clicker1
```

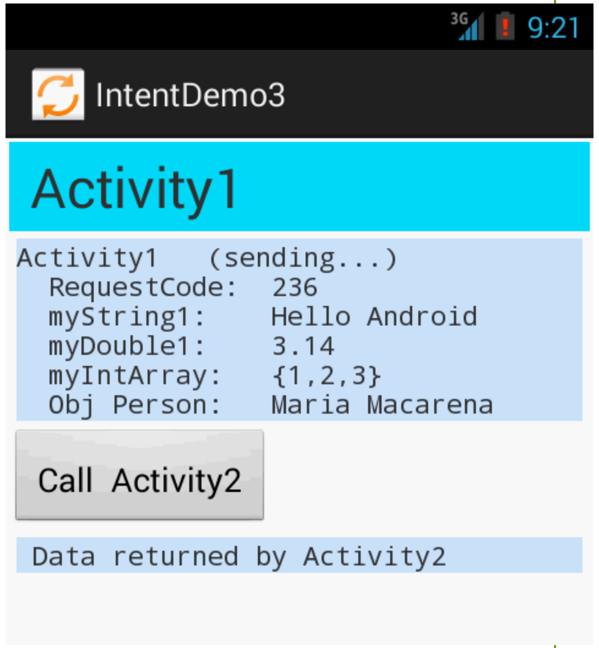


```
Activity1   (sending...)
  RequestCode:  236
  myString1:    Hello Android
  myDouble1:    3.14
  myIntArray:   {1,2,3}
  Obj Person:   Maria Macarena
```

```
Data returned by Activity2
```

USING BUNDLES TO PASS DATA (Example 18: MainActivity.java – caller)

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  try { // check that these results are for me
    if (IPC_ID == requestCode) { // Activity2 is over - see what happened
      if (resultCode == Activity.RESULT_OK) { // good - we have some data sent back from Activity2
        Bundle myReturnedData = data.getExtras();
        String myReturnedString1 = myReturnedData.getString("myReturnedString1");
        Double myReturnedDouble1 = myReturnedData.getDouble("myReturnedDouble1");
        String myReturnedDate = myReturnedData.getString("myCurrentDate");
        Person myReturnedPerson = (Person) myReturnedData.getSerializable("person");
        int[] myReturnedReversedArray = myReturnedData.getIntArray("myReversedArray");
        // display in the bottom label
        txtReturnedValues.setText("\n requestCode: " + requestCode + "\n resultCode: " + resultCode
                                            + "\n returnedString1: " + myReturnedString1
                                            + "\n returnedDouble: " + Double.toString(myReturnedDouble1)
                                            + "\n returnedString2: " + myReturnedDate
                                            + "\n returnedPerson: " + myReturnedPerson.getFullName()
                                            + "\n returnedArray: "  + Activity1.myConvertArray2String(myReturnedReversedArray));
      }
      else { /* user pressed the BACK button*/ txtTop.setText("Selection CANCELLED!");}
    }
  }
  catch (Exception e) { Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show(); } // try
}// onActivityResult
```

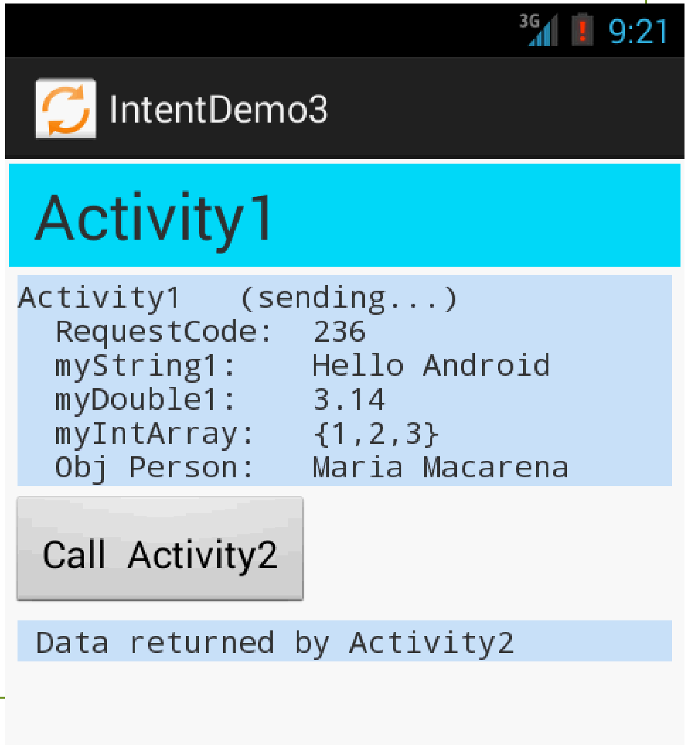


```
Activity1   (sending...)
  RequestCode:  236
  myString1:    Hello Android
  myDouble1:    3.14
  myIntArray:   {1,2,3}
  Obj Person:   Maria Macarena
```

```
Data returned by Activity2
```

USING BUNDLES TO PASS DATA (Example 18: MainActivity.java – caller)

```
  static String myConvertArray2String(int[] intArray) {
    if ( intArray == null) return “NULL”;
    String array2Str = “{” + Integer.toString(intArray[0]);
    for (int i=1; i<intArray.length; i++) {
      array2Str = array2Str + “,” + Integer.toString(intArray[i]);
    }
    return array2Str + “}”;
  }
} // Activity1
```

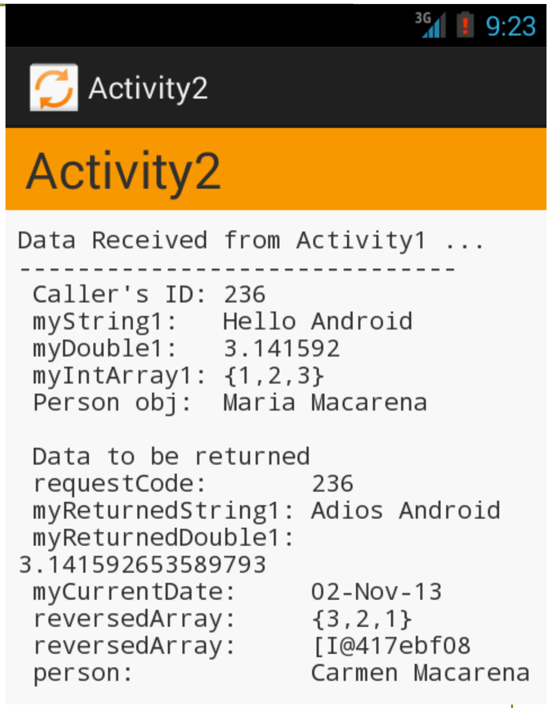


```
Activity1   (sending...)
  RequestCode:   236
  myString1:     Hello Android
  myDouble1:     3.14
  myIntArray:    {1,2,3}
  Obj Person:    Maria Macarena
```

```
Data returned by Activity2
```

USING BUNDLES TO PASS DATA (Example 18: Activity2.java – callee)

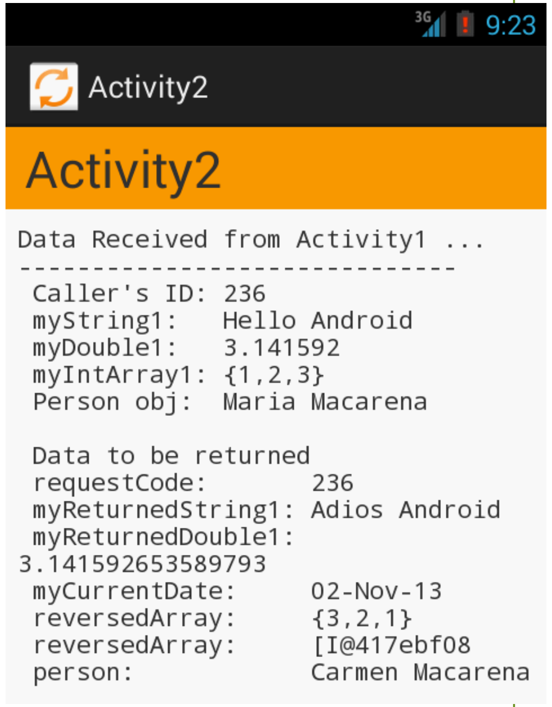
```
public class Activity2 extends Activity {
  TextView txtIncomingData, spyBox;
  Button btnCallActivity1;
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); setContentView(R.layout.main2);
    txtIncomingData = (TextView)findViewById(R.id.txtIncomingData);
    spyBox = (TextView)findViewById(R.id.spyBox);
    btnCallActivity1 = (Button)findViewById(R.id.btnCallActivity1);
    btnCallActivity1.setOnClickListener(new Clicker1());
    // create a local Intent handler – we have been called!
    Intent myCallerIntentHandler = getIntent();
    // grab the data package with all the pieces sent to us
    Bundle myBundle = myCallerIntentHandler.getExtras();
    // extract the individual data parts from the bundle, observe you know the individual keyNames
    int paramInt = myBundle.getInt("myRequestCode");
    String paramString = myBundle.getString("myString1");
    double paramDouble = myBundle.getDouble("myDouble1");
    int[] paramArray = myBundle.getIntArray("myIntArray1");
    Person paramPerson = (Person) myBundle.getSerializable("person");
    String personName = paramPerson.getFullName();
    //…
```



```
Data Received from Activity1 ...
------------------------------
 Caller's ID: 236
 myString1:   Hello Android
 myDouble1:   3.141592
 myIntArray1: {1,2,3}
 Person obj:  Maria Macarena

 Data to be returned
 requestCode:       236
 myReturnedString1: Adios Android
 myReturnedDouble1:
3.141592653589793
 myCurrentDate:     02-Nov-13
 reversedArray:     {3,2,1}
 reversedArray:     [I@417ebf08
 person:            Carmen Macarena
```

USING BUNDLES TO PASS DATA (Example 18: Activity2.java – callee)

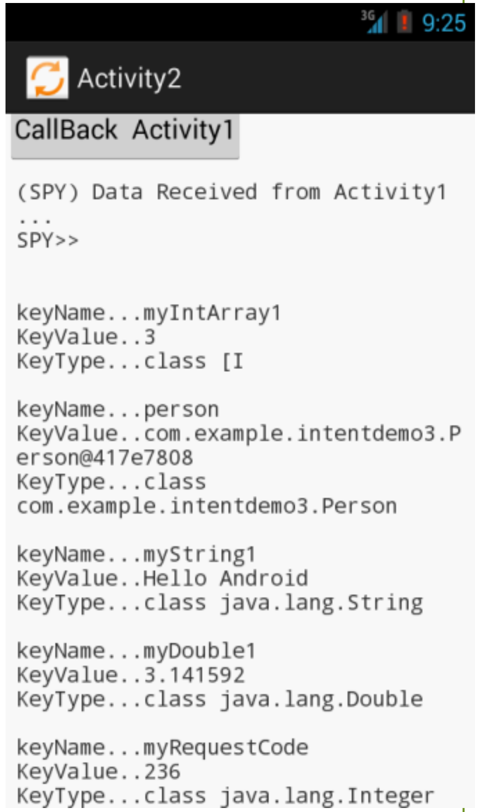
```
    //for debugging purposes - show arriving data
    txtIncomingData.append("\n----------" + "\n Caller's ID: " + paramInt + "\n myString1: " + paramString + "\n myDouble1: "
                                        + Double.toString(paramDouble) + "\n myIntArray1: "
                                        + Activity1.myConvertArray2String(paramArray) + "\n Person obj: " + paramPerson.getFullName());
    // next method assumes you do not know the data-items keyNames
    String spyData = extractDataFromBundle(myBundle); spyBox.append(spyData);
    // do here something with the extracted data. For example, reverse the values stored in the incoming integer array
    // int[] intReversedArray = myIntReverseArray( paramArray);
    int[] intReversedArray = myIntReverseArray(paramArray);
    String strReversedArray = Activity1.myConvertArray2String(intReversedArray);
    myBundle.putIntArray("myReversedArray", intReversedArray);
    // change the person's firstName
    paramPerson.setFirstName("Carmen"); myBundle.putSerializable("person", paramPerson);
    // Returning Results. Go back to myActivity1 with some new data made/change here.
    myBundle.putString("myReturnedString1", "Adios Android"); myBundle.putDouble("myReturnedDouble1", Math.PI);
    SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-yy");
    String now = formatter.format(new Date());
    myBundle.putString("myCurrentDate", now ); myCallerIntentHandler.putExtras(myBundle);
    // just debugging - show returning data
    txtIncomingData.append("\n\n Data to be returned " + "\n requestCode: " + paramInt + "\n myReturnedString1: " + myBundle.getString("myReturnedString1")
                                  + "\n myReturnedDouble1: " + myBundle.getDouble("myReturnedDouble1") + "\n myCurrentDate: " + myBundle.getString("myCurrentDate")
                                  + "\n reversedArray: " + strReversedArray + "\n reversedArray: " + myBundle.getIntArray("myReversedArray")
                                  + "\n person: " + ((Person) myBundle.getSerializable("person")).getFullName());
    setResult(Activity.RESULT_OK, myCallerIntentHandler);
}//onCreate
```



```
Data Received from Activity1 ...
-----------------------------
 Caller's ID: 236
 myString1:   Hello Android
 myDouble1:   3.141592
 myIntArray1: {1,2,3}
 Person obj:  Maria Macarena

 Data to be returned
 requestCode:        236
 myReturnedString1:  Adios Android
 myReturnedDouble1:
3.141592653589793
 myCurrentDate:      02-Nov-13
 reversedArray:      {3,2,1}
 reversedArray:      [I@417ebf08
 person:             Carmen Macarena
```

USING BUNDLES TO PASS DATA (Example 18: Activity2.java – callee)

```
private class Clicker1 implements OnClickListener { public void onClick(View v) { /*clear Activity2 screen so Activity1 could be seen*/ finish(); } }
private int[] myIntReverseArray( int[] theArray ) {
 int n = theArray.length; int[] reversedArray = new int[n];
 for (int i=0; i< theArray.length; i++ ) { reversedArray[i] = theArray[n -i -1]; }
 return reversedArray;
}
private String extractDataFromBundle(Bundle myBundle) {
 // What if I don't know the key names? what types are in the bundle?. This fragment shows
 // how to use bundle methods to extract its data. SOME ANDROID TYPES INCLUDE:
 // class [I (array integers) class [J (array long) class [D (array doubles) class [F (array floats) class java.lang.xxx (where xxx= Integer, Double, ...)
 // Remember, the Bundle is a set of <keyName, keyValue> pairs
 String spy = "\nSPY>>\n";
 Set<String> myKeyNames = myBundle.keySet(); //get all keyNames
 for (String keyName : myKeyNames){
  Serializable keyValue = myBundle.getSerializable(keyName);
  String keyType = keyValue.getClass().toString();
  if (keyType.equals("class java.lang.Integer")) keyValue = Integer.parseInt(keyValue.toString());
  else if (keyType.equals("class java.lang.Double")) keyValue = Double.parseDouble(keyValue.toString());
  else if (keyType.equals("class java.lang.Float")) keyValue = Float.parseFloat(keyValue.toString());
  else if (keyType.equals("class [I")){
   int[] arrint = myBundle.getIntArray(keyName); int n = arrint.length;
   keyValue = arrint[n-1]; // show only the last!
  } else keyValue = (String)keyValue.toString();
  spy += "\n\nkeyName..." + keyName + " \nKeyValue.." + keyValue + " \nKeyType..." + keyType ;
 } return spy; }}
```



```
(SPY) Data Received from Activity1
...
SPY>>


keyName...myIntArray1
KeyValue..3
KeyType...class [I

keyName...person
KeyValue..com.example.intentdemo3.P
erson@417e7808
KeyType...class
com.example.intentdemo3.Person

keyName...myString1
KeyValue..Hello Android
KeyType...class java.lang.String

keyName...myDouble1
KeyValue..3.141592
KeyType...class java.lang.Double

keyName...myRequestCode
KeyValue..236
KeyType...class java.lang.Integer
```

USING BUNDLES TO PASS DATA (Example 18: Person.java & manifest)

```java
public class Person implements Serializable {
  private static final long serialVersionUID = 1L;
  private String firstName;
  private String lastName;
  public Person(String firstName, String lastName) {
    super(); this.firstName = firstName;
    this.lastName = lastName;
  }
  public String getFirstName() { return firstName; }
  public void setFirstName(String value) {
    this.firstName = value;
  }
  public String getFullName() {
    return firstName + “ ” + lastName;
  }
}
```

```xml
<manifest xmlns:android=“http://schemas.android.com/apk/res/android”
              package=“com.example.intentdemo3”
              android:versionCode=“1” android:versionName=“1.0” >
  <uses-sdk android:minSdkVersion=“14” android:targetSdkVersion=“17” />
  <application android:icon=“@drawable/ic_launcher”
                    android:label=“@string/app_name”
                  android:theme=“@style/AppTheme” >
    <activity android:name=“.Activity1” android:label=“Activity1” >
      <intent-filter>
        <action android:name=“android.intent.action.MAIN” />
        <category android:name=“android.intent.category.LAUNCHER” />
      </intent-filter>
    </activity>
    <activity android:name=“.Activity2” android:label=“Activity2”/>
  </application>
</manifest>
```

USING BUNDLES TO PASS DATA (Example 18: Comments)

Various data items are inserted into an outgoing bundle, including simple types, an array, and a serialized object.

Activity1 invokes Activity2 and waits for its results.

The listener in Activity1 issues the statement data.getExtras() to get the returning bundle. Data items are extracted and displayed.

Activity2 uses getIntent() to capture the incoming intent request.

The arriving bundle is picked up and its data items are extracted. Observe the use of .getSerializable(…) to grab the object argument.

After operating on the input data, Activity2 packs all the outgoing extra items in a bundle and attaches it to the original intent.

Activity2 finishes with a RESULT_OK termination code.

The method extractDataFromBundle is used to get all the <key,value> pairs packed in a bundle.

This fragment defines a simple Person class, with is private data members, constructor, accessors, and custom method.

APPENDIXES

A complete list of built-in, broadcast, service actions, categories, and features for a particular SDK can be found in the folders: .../android-sdk/platforms/platform-YYY/data/

android.app.action.
ACTION_PASSWORD_CHANGED
ACTION_PASSWORD_EXPIRING
ACTION_PASSWORD_FAILED
ACTION_PASSWORD_SUCCEEDED
ADD_DEVICE_ADMIN
DEVICE_ADMIN_DISABLE_REQUESTED
DEVICE_ADMIN_DISABLED
DEVICE_ADMIN_ENABLED
SET_NEW_PASSWORD
START_ENCRYPTION
android.bluetooth.a2dp.profile.action.
CONNECTION_STATE_CHANGED
PLAYING_STATE_CHANGED
android.bluetooth.adapter.action.
CONNECTION_STATE_CHANGED
DISCOVERY_FINISHED
DISCOVERY_STARTED
LOCAL_NAME_CHANGED
REQUEST_DISCOVERABLE
REQUEST_ENABLE
SCAN_MODE_CHANGED
STATE_CHANGED
android.bluetooth.device.action.
ACL_CONNECTED
ACL_DISCONNECT_REQUESTED
ACL_DISCONNECTED
BOND_STATE_CHANGED
CLASS_CHANGED
FOUND
NAME_CHANGED
UUID
android.bluetooth.devicepicker.action.
DEVICE_SELECTED
LAUNCH
android.bluetooth.headset.
action.VENDOR_SPECIFIC_HEADSET_EVENT
profile.action.AUDIO_STATE_CHANGED
profile.action.CONNECTION_STATE_CHANGED
android.hardware.action.
NEW_PICTURE
NEW_VIDEO
input.action.QUERY_KEYBOARD_LAYOUTS
android.intent.action.
ACTION_POWER_CONNECTED
ACTION_POWER_DISCONNECTED
ACTION_SHUTDOWN
AIRPLANE_MODE
ALL_APPS
ANSWER
APP_ERROR
ASSIST
ATTACH_DATA
BATTERY_CHANGED
BATTERY_LOW
BATTERY_OKAY
BOOT_COMPLETED
BUG_REPORT
CALL
CALL_BUTTON
CAMERA_BUTTON
CHOOSER
CONFIGURATION_CHANGED
CREATE_LIVE_FOLDER
CREATE_SHORTCUT
DATE_CHANGED
DELETE
DEVICE_STORAGE_LOW
DEVICE_STORAGE_OK
DIAL
DOCK_EVENT
DREAMING_STARTED
DREAMING_STOPPED
EDIT

APPENDIXES

A complete list of built-in, broadcast, service actions, categories, and features for a particular SDK can be found in the folders: .../android-sdk/platforms/platform-YYY/data/

android.intent.action.
EVENT_REMINDER
EXTERNAL_APPLICATIONS_AVAILA
BLE
EXTERNAL_APPLICATIONS_UNAVAI
LABLE
FETCH_VOICEMAIL
GET_CONTENT
GTALK_CONNECTED
GTALK_DISCONNECTED
HEADSET_PLUG
INPUT_METHOD_CHANGED
INSERT
INSERT_OR_EDIT
INSTALL_PACKAGE
LOCALE_CHANGED
MAIN
MANAGE_NETWORK_USAGE
MANAGE_PACKAGE_STORAGE
MEDIA_BAD_REMOVAL
MEDIA_BUTTON

MEDIA_CHECKING
MEDIA_EJECT
MEDIA_MOUNTED
MEDIA_NOFS
MEDIA_REMOVED
MEDIA_SCANNER_FINISHED
MEDIA_SCANNER_SCAN_FILE
MEDIA_SCANNER_STARTED
MEDIA_SEARCH
MEDIA_SHARED
MEDIA_UNMOUNTABLE
MEDIA_UNMOUNTED
MUSIC_PLAYER
MY_PACKAGE_REPLACED
NEW_OUTGOING_CALL
NEW_VOICEMAIL
PACKAGE_ADDED
PACKAGE_CHANGED
PACKAGE_DATA_CLEARED
PACKAGE_FIRST_LAUNCH
PACKAGE_FULLY_REMOVED

android.intent.action.
PACKAGE_INSTALL
PACKAGE_NEEDS_VERIFICATION
PACKAGE_REMOVED
PACKAGE_REPLACED
PACKAGE_RESTARTED
PACKAGE_VERIFIED
PASTE
PHONE_STATE
PICK
PICK_ACTIVITY
POWER_USAGE_SUMMARY
PROVIDER_CHANGED
PROXY_CHANGE
REBOOT
RESPOND_VIA_MESSAGE
RINGTONE_PICKER
RUN
SCREEN_OFF
SCREEN_ON
SEARCH

SEARCH_LONG_PRESS
SEND
SEND_MULTIPLE
SENDTO
SET_ALARM
SET_WALLPAPER
SYNC
SYSTEM_TUTORIAL
TIME_SET
TIME_TICK
TIMEZONE_CHANGED
UID_REMOVED
UNINSTALL_PACKAGE
USER_PRESENT
VIEW
VOICE_COMMAND
WALLPAPER_CHANGED
WEB_SEARCH

APPENDIXES

A complete list of built-in, broadcast, service actions, categories, and features for a particular SDK can be found in the folders: .../android-sdk/platforms/platform-YYY/data/

android.media.
action.CLOSE_AUDIO_EFFECT_CONTROL_SESSION
action.DISPLAY_AUDIO_EFFECT_CONTROL_PANEL
action.OPEN_AUDIO_EFFECT_CONTROL_SESSION
ACTION_SCO_AUDIO_STATE_UPDATED
AUDIO_BECOMING_NOISY
RINGER_MODE_CHANGED
SCO_AUDIO_STATE_CHANGED
VIBRATE_SETTING_CHANGED

android.net.
conn.BACKGROUND_DATA_SETTING_CHANGED
conn.CONNECTIVITY_CHANGE
nsd.STATE_CHANGED
wifi.action.REQUEST_SCAN_ALWAYS_AVAILABLE
wifi.NETWORK_IDS_CHANGED
wifi.p2p.CONNECTION_STATE_CHANGE
wifi.p2p.DISCOVERY_STATE_CHANGE
wifi.p2p.PEERS_CHANGED
wifi.p2p.STATE_CHANGED
wifi.p2p.THIS_DEVICE_CHANGED
wifi.PICK_WIFI_NETWORK
wifi.RSSI_CHANGED
wifi.SCAN_RESULTS
wifi.STATE_CHANGE
wifi.supplicant.CONNECTION_CHANGE
wifi.supplicant.STATE_CHANGE
wifi.WIFI_STATE_CHANGED

android.nfc.action.
ADAPTER_STATE_CHANGED
NDEF_DISCOVERED
TAG_DISCOVERED
TECH_DISCOVERED

android.settings.
ACCESSIBILITY_SETTINGS
ADD_ACCOUNT_SETTINGS
AIRPLANE_MODE_SETTINGS
APN_SETTINGS
APPLICATION_DETAILS_SETTINGS
APPLICATION_DEVELOPMENT_SETTINGS
APPLICATION_SETTINGS
BLUETOOTH_SETTINGS
DATA_ROAMING_SETTINGS
DATE_SETTINGS
DEVICE_INFO_SETTINGS
DISPLAY_SETTINGS
DREAM_SETTINGS
INPUT_METHOD_SETTINGS
INPUT_METHOD_SUBTYPE_SETTINGS
INTERNAL_STORAGE_SETTINGS
LOCALE_SETTINGS
LOCATION_SOURCE_SETTINGS
MANAGE_ALL_APPLICATIONS_SETTINGS
MANAGE_APPLICATIONS_SETTINGS
MEMORY_CARD_SETTINGS
NETWORK_OPERATOR_SETTINGS
NFC_SETTINGS
NFCSHARING_SETTINGS
PRIVACY_SETTINGS
QUICK_LAUNCH_SETTINGS
SECURITY_SETTINGS
SETTINGS
SOUND_SETTINGS
SYNC_SETTINGS
USER_DICTIONARY_SETTINGS
WIFI_IP_SETTINGS
WIFI_SETTINGS
WIRELESS_SETTINGS

android.speech.tts.
engine.CHECK_TTS_DATA
engine.GET_SAMPLE_TEXT
engine.INSTALL_TTS_DATA
engine.TTS_DATA_INSTALLED
TTS_QUEUE_PROCESSING_COMPLETED