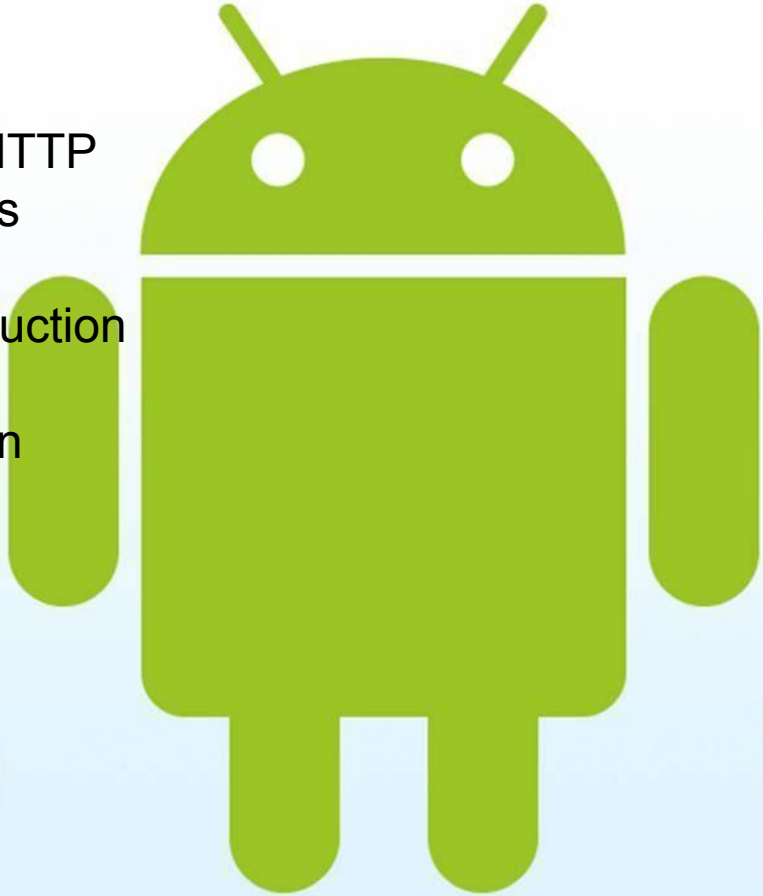
# Partie IV : Développement avancé

- Téléphonie
- Webservice && HTTP
- Location && Maps
- Thread
- Bluetooth - Introduction
- BatteryManager
- NFC - Introduction



# Développement avancé : Téléphonie

*Telephone The term telephone means "speech over a distance." The Greek* roots are tele, which means "distant," and *phone, which means "speech."*

- ***Package***
- ***Telephony background and terms***
- ***Accessing telephony information***
- ***Interacting with the phone***
- Android Telephony Internals
- ***Working with messaging: SMS***

## PACKAGE OVERVIEW

**CellLocation**
Methods to request location information.

**PhoneNumberFormattingTextWatcher**
Callback methods that notify an application about changes to phone numbers themselves. When used with a TextView object, it formats the text as a phone number using methods in the PhoneNumberUtils class.

**PhoneNumberUtils**
A utility class that contains methods for processing strings containing phone Numbers.

**PhoneStateListener**
Callback methods that track changes in the state of the mobile network connection, the call, or other telephony objects.

**ServiceState**
Methods that return information about the current mobile network service provider and the availability of service.

**TelephonyManager**
Methods that provide information about the state of mobile service, the call state, the SIM, the network, the subscriber to the mobile service, voicemail, and other related information.

**GSM / Code division multiple access (CDMA)**

## PACKAGE OVERVIEW

**Limitations on What Applications Can Do with the Phone**

Some developers may be disappointed to see that the android.telephony package limits their access to getting information, and does not provide direct control over the dialer or the state of a call. There are good reasons for this. Essentially, providing a low-level telephony API that can be shared among multiple applications is perilous.

# Développement avancé : Téléphonie

Android Telephony Internals : android.internal.telephony && android.internal.telephony.gsm

ATParseEx
ATResponseParser
Call
CallerInfo
CallerInfoAsyncQuery
CallStateException
Connection
DefaultPhoneNotifier
IPhoneStateListener
IPhoneSubInfo
Itelephony
ITelephonyRegistry
…

*Layers of telephony packages :*

This is a large and complex package, and a complete understanding will require reading the Android source code

## *Telephony background and terms*

A SIM card is a small, removable, and secure smart card. Every device that operates on a GSM network has specific unique identifiers, which are stored on the SIM card:

- ***Integrated Circuit Card ID*** *(ICCID)*
- *Identifies a SIM card (also known as a SIM* Serial Number, or SSN).

- ***International Mobile Equipment Identity*** *(IMEI)*
- *Identifies a physical device. (The* number is usually printed underneath the battery).

- ***International Mobile Subscriber Identity*** *(IMSI)*
- *Identifies a subscriber (and the* network that subscriber is on).

- ***Location Area Identity*** *(LAI)*
- *Identifies the region the device is in within a provider* network.

- ***Authentication Key*** *(Ki)*
- *A 128-bit key used to authenticate a SIM card on this* provider network. A 128-bit key.

## ***Interacting with the phone*** **:** Quick and Easy Phone Calls

Through the use of Intent objects, Android enables applications to tell other applications to perform certain operations, such as initiating a phone call

```
package example.dialing;
import …;
public class dialing extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
call();
}
private void call() {
try {
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:9785551212"));
startActivity(callIntent);
} catch (ActivityNotFoundException activityException) {
Log.e("dialing-example", "Call failed", activityException);
}
}
```

## *Interacting with the phone : phone state*

This section shows how to track the state of a phone call :

```
private class ListenToPhoneState extends PhoneStateListener
{
public void onCallStateChanged(int state, String incomingNumber) {
Log.i("telephony-example", "State changed: " + stateName(state));
}
String stateName(int state) {
switch (state) {
    case TelephonyManager.CALL_STATE_IDLE: return "Idle";
    case TelephonyManager.CALL_STATE_OFFHOOK: return "Off hook";
    case TelephonyManager.CALL_STATE_RINGING: return "Ringing";
}
return Integer.toString(state);
}
}
```

1) Overrides the onCallStateChanged method of the PhoneStateListener class.
2) Adds a message to Android's log whenever the state changes.
3) Chooses meaningful strings to represent call states.

## *Interacting with the phone : Accessing telephony information*

Full example :

```
Package telephonyexplorer;

Import …

public class TelephonyManagerExample extends Activity
 {

     private TextView telMgrOutput;

    @Override
    public void onCreate(final Bundle icicle)
  {
          Log.d(Constants.LOGTAG, "TelephonyManagerExample onCreate");

         super.onCreate(icicle);
         this.setContentView(R.layout.telmgrexample);
         this.telMgrOutput = (TextView) findViewById(R.id.telmgroutput);
    }
```

Classical activity

# Développement avancé : Téléphonie

```
@Override
public void onStart()
 {
        super.onStart();

        // TelephonyManager
        final TelephonyManager telMgr = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
        this.telMgrOutput.setText(telMgr.toString());

        // PhoneStateListener
        PhoneStateListener phoneStateListener = new PhoneStateListener()
        {

                @Override
                public void onCallStateChanged(final int state, final String incomingNumber)
                {
                    telMgrOutput.setText(getTelephonyOverview(telMgr));
                    Log.d(Constants.LOGTAG, "phoneState updated - incoming number - " + incomingNumber);
                }
        };

telMgr.listen(phoneStateListener, PhoneStateListener.LISTEN_CALL_STATE);
String telephonyOverview = getTelephonyOverview(telMgr);
this.telMgrOutput.setText(telephonyOverview);

}
```

Tel manager from the phone service

listener

action

Apply listener

# Développement avancé : Téléphonie

```
public String getTelephonyOverview(final TelephonyManager telMgr)
        {
                int callState = telMgr.getCallState();
                String callStateString = "NA";


                switch (callState)
                {
                case TelephonyManager.CALL_STATE_IDLE: /* en attente */
                        callStateString = "IDLE";
                        break;
                case TelephonyManager.CALL_STATE_OFFHOOK: /* décroché */
                        callStateString = "OFFHOOK";
                        break;
                case TelephonyManager.CALL_STATE_RINGING: /* sonne */
                        callStateString = "RINGING";
                        break;
                }


                GsmCellLocation cellLocation = (GsmCellLocation) telMgr.getCellLocation();
                String cellLocationString = cellLocation.getLac() + " " + cellLocation.getCid();


                String deviceId = telMgr.getDeviceId();
                String deviceSoftwareVersion = telMgr.getDeviceSoftwareVersion();


                String line1Number = telMgr.getLine1Number();


                String networkCountryIso = telMgr.getNetworkCountryIso();
                String networkOperator = telMgr.getNetworkOperator();
                String networkOperatorName = telMgr.getNetworkOperatorName();
```

Phone state

GSM info

# Développement avancé : Téléphonie

```
int phoneType = telMgr.getPhoneType();

        String phoneTypeString = "NA";

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

        String simCountryIso = telMgr.getSimCountryIso();
        String simOperator = telMgr.getSimOperator();
        String simOperatorName = telMgr.getSimOperatorName();
        String simSerialNumber = telMgr.getSimSerialNumber();
        String simSubscriberId = telMgr.getSubscriberId();
        int simState = telMgr.getSimState();
        String simStateString = "NA";

        switch (simState)
        {
        case TelephonyManager.SIM_STATE_ABSENT:
                simStateString = "ABSENT";
                break;
        case TelephonyManager.SIM_STATE_NETWORK_LOCKED:
                simStateString = "NETWORK_LOCKED";
                break;
```

Phone type

SIM info

# Développement avancé : Téléphonie

```
        case TelephonyManager.SIM_STATE_PIN_REQUIRED:
            simStateString = "PIN_REQUIRED";
            break;
        case TelephonyManager.SIM_STATE_PUK_REQUIRED:
            simStateString = "PUK_REQUIRED";
            break;
        case TelephonyManager.SIM_STATE_READY:
            simStateString = "STATE_READY";
            break;
        case TelephonyManager.SIM_STATE_UNKNOWN:
            simStateString = "STATE_UNKNOWN";
            break;
        }


        StringBuilder sb = new StringBuilder();
        sb.append("telMgr - ");
        sb.append("  \ncallState = " + callStateString);
        sb.append("  \ncellLocationString = " + cellLocationString);
        sb.append("  \ndeviceId = " + deviceId);
        sb.append("  \ndeviceSoftwareVersion = " + deviceSoftwareVersion);
        sb.append("  \nline1Number = " + line1Number);
        sb.append("  \nnetworkCountryIso = " + networkCountryIso);
        sb.append("  \nnetworkOperator = " + networkOperator);
        sb.append("  \nnetworkOperatorName = " + networkOperatorName);
        sb.append("  \nphoneTypeString = " + phoneTypeString);
        sb.append("  \nsimCountryIso = " + simCountryIso);
        sb.append("  \nsimOperator = " + simOperator);
        sb.append("  \nsimOperatorName = " + simOperatorName);
        sb.append("  \nsimSerialNumber = " + simSerialNumber);
        sb.append("  \nsimSubscriberId = " + simSubscriberId);
        sb.append("  \nsimStateString = " + simStateString);


        return sb.toString();
    }
```

SIM info

Build ,string info

*

## Android Telephony Internals with AIDL

```
public class DialerDemo extends Activity {
  IPhone phone=null;

  @Override
  public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);

    IServiceManager svcMgr=ServiceManagerNative.getDefault();

    try {
      phone=IPhone.Stub.asInterface(svcMgr.getService("phone"));
    }
    catch (DeadObjectException e) {
      android.util.Log.e("DialerDemo", "Error in dial()", e);
      Toast.makeText(DialerDemo.this, e.toString(), 2000).show();
      finish();
    }

    final EditText number=(EditText)findViewById(R.id.number);
    Button dial=(Button)findViewById(R.id.dial);

    dial.setOnClickListener(new Button.OnClickListener() {
      public void onClick(View v) {
        try {
          if (phone!=null) {
            phone.dial(number.getText().toString());
          }
```

Get a service with AIDL

Dial phone

## Android Telephony Internals with AIDL

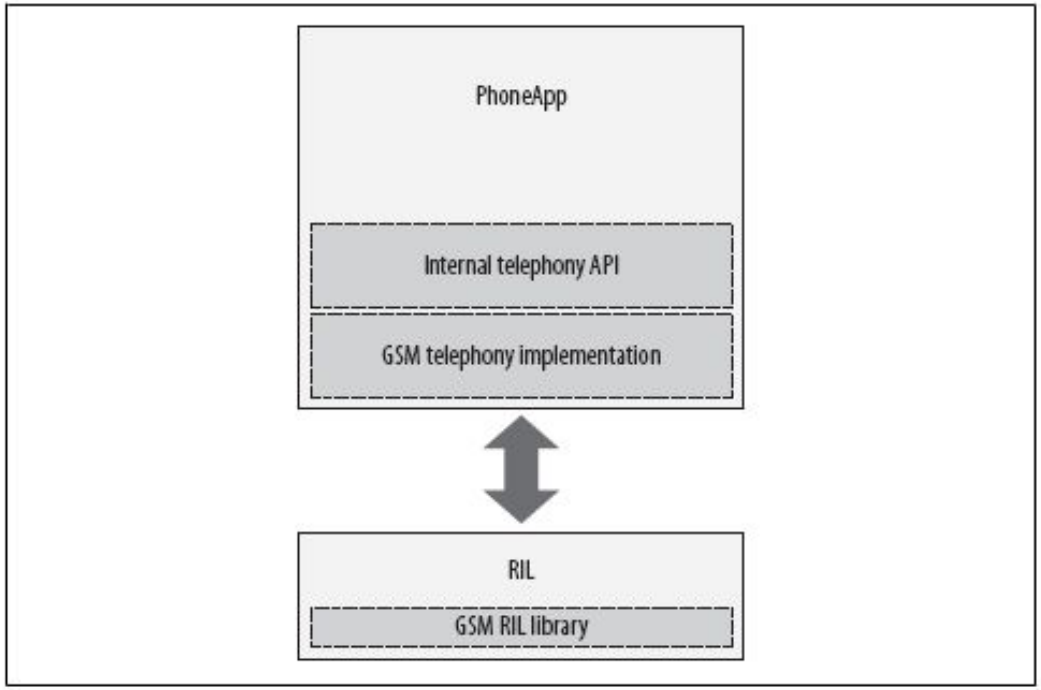
```
        }
        catch (DeadObjectException e) {
          android.util.Log.e("DialerDemo", "Error in dial()", e);
          Toast.makeText(DialerDemo.this, e.toString(), 2000).show();
        }
      }
    });

    Button call=(Button)findViewById(R.id.call);

    call.setOnClickListener(new Button.OnClickListener() {
      public void onClick(View v) {
        try {
          if (phone!=null) {
            phone.call(number.getText().toString());
          }
        }
        catch (DeadObjectException e) {
          android.util.Log.e("DialerDemo", "Error in dial()", e);
          Toast.makeText(DialerDemo.this, e.toString(), 2000).show();
        }
      }
    });
  }
}
```

Call phone with remote procedure

## Android Telephony Internals

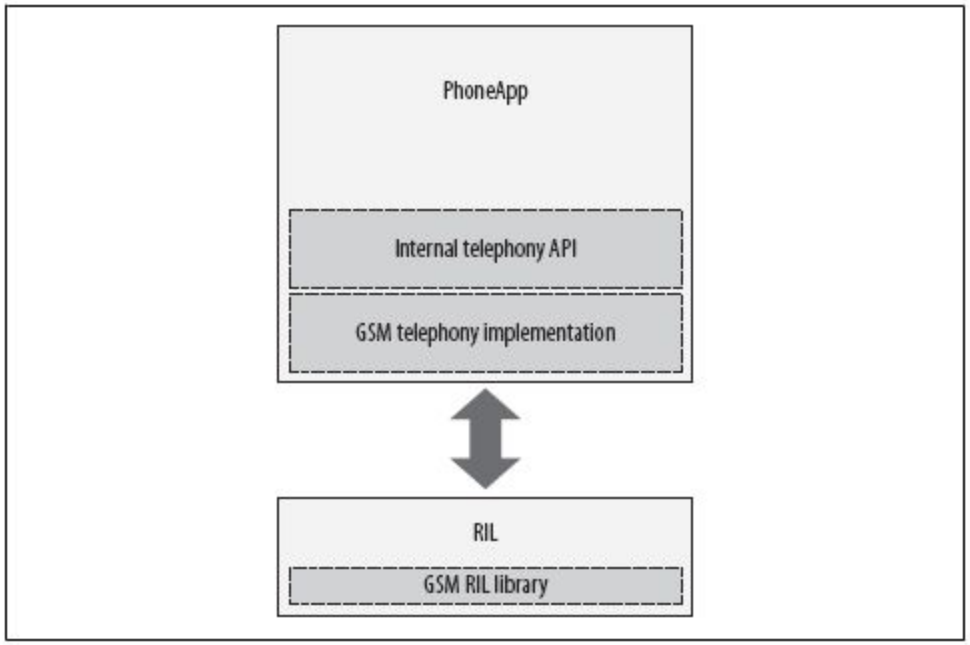



*Layers of telephony packages*

This package is layered over an implementation of telephony internals for a particular telephony technology, such as GSM or CDMA. That layer, in turn, communicates with a Radio Interface Layer (RIL) that is implemented as a daemon in Android.

## Android Telephony Internals : android.internal.telephony Package

*Layers of telephony packages*




Many of the internal packages use the remote methods feature discussed in previous chapter.

For instance, the following AIDL comes from *IPhoneStateListener.aidl:*

```
oneway interface IPhoneStateListener {
void onServiceStateChanged(in ServiceState
serviceState);
void onSignalStrengthChanged(int asu);
void onMessageWaitingIndicatorChanged(boolean mwi);
void onCallForwardingIndicatorChanged(boolean cfi);
// we use bundle here instead of CellLocation so it can get
the right subclass
void onCellLocationChanged(in Bundle location);
void onCallStateChanged(int state, String
incomingNumber);
void onDataConnectionStateChanged(int state);
void onDataActivity(int direction);
}
```

## Android Telephony Internals : Exploring Android Telephony Internals

```
./adb devices

./adb -s emulator-5554 shell

# logcat -b radio

(startActivity(callIntent); //
Eclipse

D/GSM ( 85): [GSMConn] update: parent=DIALING,
hasNewParent=false,
wasConnectingInOrOut=true, wasHolding=false,
isConnectingInOrOut=true,changed=false
D/RILJ ( 85): [0161]> SET_MUTE false
D/RIL ( 22): onRequest: SET_MUTE
D/RILJ ( 85): [0161]< SET_MUTE error:
com.android.internal.telephony.gsm.CommandException:
REQUEST_NOT_SUPPORTED
D/RILJ ( 85): [UNSL]< CALL_STATE_CHANGED
D/RILJ ( 85): [0162]> GET_CURRENT_CALLS
D/RIL ( 22): onRequest: GET_CURRENT_CALLS
D/AT ( 22): AT> AT+CLCC
```

*Layers of telephony packages :*

This is a large and complex package, and a complete understanding will require reading the Android source code

## Emulation de l'état du téléphone

Il est possible d'émuler l'état du téléphone en se connectant directement sur le port TCP. Le numéro de port est indiqué par la machine virtuelle à son lancement. Ensuite utiliser telnet pour une connexion directe:

```
> telnet localhost 5554

> help gsm
```

gsm list : liste des appels en cours
gsm call : Crée un appel entrant vers l'émulateur , gsm call 25412565
gms busy : Ferme un appel sortent car occupé : gsm busy 12345
gsm hold : Met un appel sortant en pause
gsm accept : Change l'état d'un appel sortant à actif
gsm cancel : Interrompt un appel (entrant ou sortant)
gsm data : Modifie l'état de connexion données
gsm voice : Modifie l'état de connexion voix
gsm status : Affiche l'état voix et données de l'émulateur

http://developer.android.com/tools/devices/emulator.html

## Android Telephony : Working with the **PhoneNumberUtils**

```
. . .

private TextView pnOutput;
private EditText pnInput;
private EditText pnInPlaceInput;
private Button pnFormat;

. . .

this.pnFormat.setOnClickListener(new OnClickListener() {
  public void onClick(View v) {
     String phoneNumber = PhoneNumberUtils.formatNumber(
       pnInput.getText().toString());
     phoneNumber = PhoneNumberUtils.convertKeypadLettersToDigits(
       pnInput.getText().toString());

   StringBuilder result = new StringBuilder();
   result.append(phoneNumber);
   result.append("\nisGlobal - "
       + PhoneNumberUtils.isGlobalPhoneNumber(phoneNumber));
   result.append("\nisEmergency - "
       + PhoneNumberUtils.isEmergencyNumber(phoneNumber));

     pnOutput.setText(result.toString());
     pnInput.setText("");
  }
});
```

The PhoneNumberUtils class has a number of static helper methods for parsing phone numbers, the simplest of which is formatNumber.

❶ Format as phone number

❷ Convert alpha characters to digits

Alphabetic keypad letter characters into digits

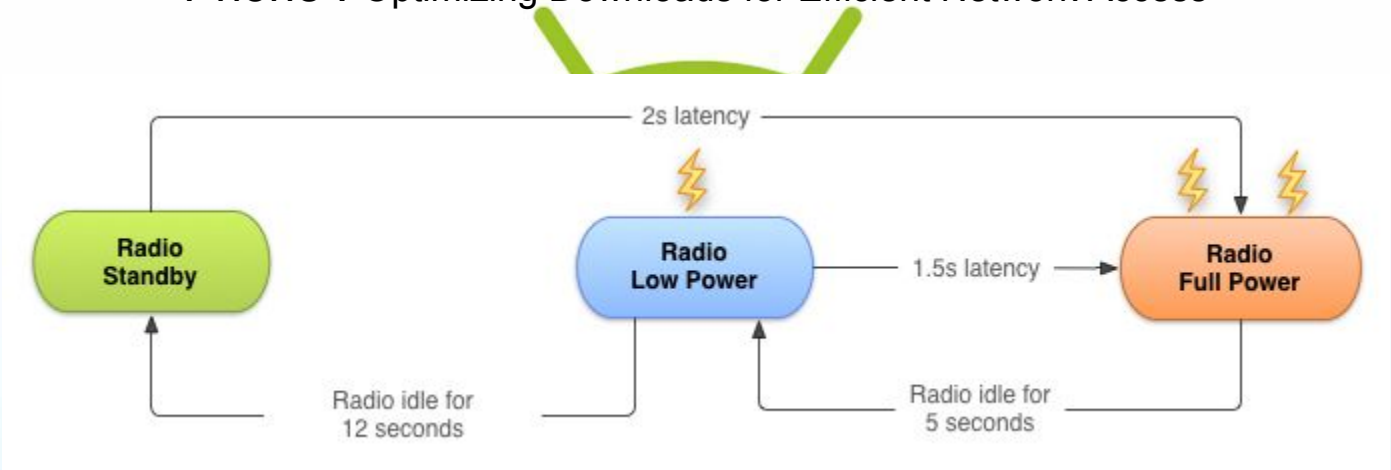
❸ Use additional phone number utilities

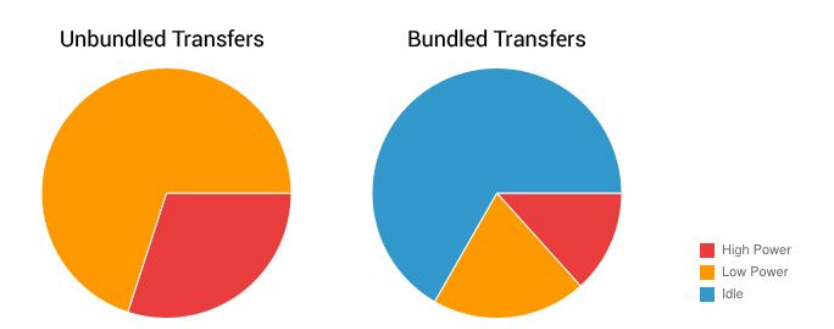
The number is global and whether it represents an emergency call

## Security : Phone-related manifest permissions and their purpose

| Phone-related permission | Purpose |
|---|---|
| android.permission.READ_PHONE_STATE | Allow application to read phone state |
| android.permission.MODIFY_PHONE_STATE | Allow application to modify phone state |
| android.permission.CALL_PHONE | Initiate a phone call without user confirmation in dialer |
| android.permission.CALL_PRIVILEGED | Call any number, including emergency, without confirmation in dialer |
| android.permission.PROCESS_OUTGOING_CALLS | Allow application to receive broadcast for outgoing calls and modify |

## Phone : Optimizing Downloads for Efficient Network Access




http://developer.android.com/training/efficient-downloads/efficient-network-access.html

## *Working with messaging: sending SMS*

```
public class SmsExample extends Activity {

    private EditText smsInputText;
    private EditText smsInputDest;
    private Button smsSend;

    private SmsManager smsManager;

    @Override
    public void onCreate(final Bundle icicle) {
        Log.d(Constants.LOGTAG, "SmsExample onCreate");

        super.onCreate(icicle);
        this.setContentView(R.layout.smsexample);

        this.smsInputDest = (EditText) findViewById(R.id.smsinputdest);
        this.smsInputText = (EditText) findViewById(R.id.smsinputtext);
        this.smsSend = (Button) findViewById(R.id.smssend_button);

        this.smsManager = SmsManager.getDefault();

        // pending intent request code NOT USED
        final PendingIntent sentIntent = PendingIntent.getActivity(this, 0, new Intent(this, SmsExample.class), 0);

        this.smsSend.setOnClickListener(new OnClickListener() {
```

SMS Manager

Pending Intent

## *Working with messaging: sending SMS*

```
public class SmsExample extends Activity {
….
        public void onClick(final View v) {
            Log.d(Constants.LOGTAG, "SmsExample sending SMS message via manager");

            String dest = smsInputDest.getText().toString();
            if (PhoneNumberUtils.isWellFormedSmsAddress(dest)) {

                // dest, serviceCenter (null for default), message,
                // sentIntent, deliveryIntent
                //
                // Set the second parameter to null. The scAddress relates
                // to the address of the server on the cellular network that will handle
                // the message, it is not the address of the sender.
                smsManager.sendTextMessage(smsInputDest.getText().toString(), null, smsInputText.getText()
                    .toString(), sentIntent, null);

                Toast.makeText(SmsExample.this, "SMS message sent", Toast.LENGTH_LONG).show();

            } else {
                Toast.makeText(SmsExample.this, "SMS destination invalid - try again",
Toast.LENGTH_LONG).show();
            }
        }
    });
}
```

check

Send SMS

## *Working with messaging: SMS security / manifest*

SMS-related manifest permissions and their purpose

| Phone-related permission | Purpose |
|---|---|
| android.permission.RECEIVE_SMS | Allow application to monitor incoming SMS messages |
| android.permission.READ_SMS | Allow application to read SMS messages |
| android.permission.SEND_SMS | Allow application to send SMS messages |
| android.permission.WRITE_SMS | Write SMS messages to the built-in SMS provider (not related to sending messages directly) |

## *Working with messaging: SMS security / manifest*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.msi.manning.telephonyexplorer">
        <application android:icon="@drawable/icon">
                <activity android:name=".Main" android:label="@string/app_name">
                        <intent-filter>
                                <action android:name="android.intent.action.MAIN" />
                                <category android:name="android.intent.category.LAUNCHER" />
                        </intent-filter>
                </activity>


      <activity android:name=".TelephonyManagerExample" android:label="@string/app_name" />
                <activity android:name=".PhoneNumberUtilsExample" android:label="@string/app_name" />
                <activity android:name=".SmsExample" android:label="@string/app_name" />
                <receiver android:name=".SmsReceiver">
                <intent-filter>
                        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
                </intent-filter>
                </receiver>
                <receiver android:name=".OutgoingCallReceiver">
                                </receiver>
        </application>
        <uses-permission android:name="android.permission.RECEIVE_SMS" />
        <uses-permission android:name="android.permission.READ_SMS" />
        <uses-permission android:name="android.permission.WRITE_SMS" />
        <uses-permission android:name="android.permission.SEND_SMS" />
</manifest>
```

receiver

rights

## *Working with messaging: Receiving SMS*

```
public class SmsReceiver extends BroadcastReceiver {
public static final String SMSRECEIVED = "SMSR";
private static final String SMS_REC_ACTION = "android.provider.Telephony.SMS_RECEIVED";


@Override
public void onReceive(fContext context, Intent intent)
{
if (intent.getAction().
equals(SmsReceiver.SMS_REC_ACTION))
{
StringBuilder sb = new StringBuilder();
Bundle bundle = intent.getExtras();


if (bundle != null)
{
  Object[] pdus = (Object[]) bundle.get("pdus");
  for (Object pdu : pdus)
 {
  SmsMessage smsMessage =
  SmsMessage.createFromPdu((byte[]) pdu);
  sb.append("body - “ + smsMessage.getDisplayMessageBody());
 }
}
Toast.makeText(context, "SMS RECEIVED - “ + sb.toString(), Toast.LENGTH_LONG).show();
}
}
}
```

To react to an incoming SMS message we once again are creating a BroadcastReceiver

receiver

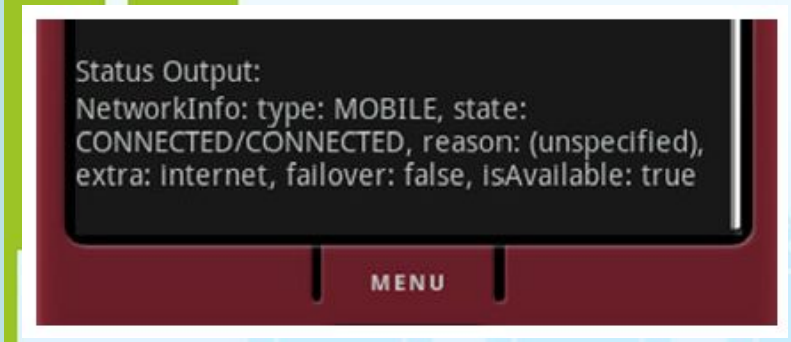
**Filter for action in receiver**

# Développement avancé : Networking

Android provides a host of utilities to determine the device configuration and the status of various services, including the network. You will typically use the ConnectivityManager class to determine whether there is network connectivity and to get notifications of network changes

The **onStart method of the NetworkExplorer main Activity**

```
@Override
public void onStart()
{
super.onStart();
ConnectivityManager cMgr = (ConnectivityManager)
this.getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo netInfo = cMgr.getActiveNetworkInfo();
this.status.setText(netInfo.toString());
}
```



# Développement avancé : Networking

TD

SimpleSMS

ANDROID_TELEPHONY_SERVICE (filtrer avec le tag PhoneService)

# Support Web - Web service

## Etat du réseau

Le type de connexion peut impacter le comportement de l'application. Aussi pour les applications mobiles il est nécessaire vérifier régulièrement le support réseau :

```
public boolean isOnline() {
        ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfoMob = cm.getNetworkInfo(cm.TYPE_MOBILE);
        NetworkInfo netInfoWifi = cm.getNetworkInfo(cm.TYPE_WIFI);
        if ((netInfoMob != null || netInfoWifi != null) && (netInfoMob.isConnectedOrConnecting() ||
netInfoWifi.isConnectedOrConnecting())) {

            Log.v(TAG, "OK");

            return true;
        }

        Log.v(TAG, "KO");
        return false;
    }
```

http://developer.android.com/reference/android/net/ConnectivityManager.html

## Utilisation TCP/UDP

L'utilisation des sockets TCP/UDP/IP est standard au package Java. Attention au support multicast qui n'est pas toujours porté ou qui fonctionne pas correctement.

```
public void run()
        {
                DatagramSocket wSocket = null;
                DatagramPacket wPacket = null;

                try
                {
                        wSocket = new DatagramSocket(listenPort);
                        wBuffer = new byte[128];
                        wPacket = new DatagramPacket( wBuffer, wBuffer.length );
                }
                catch (SocketException e)
                {
                        System.err.println( "Could not open the socket: \n" + e.getMessage() );
                        System.exit( 1 );
                }

                while (isRunning)
                {
                        try
                        {
                                wSocket.receive( wPacket );
....
```

http://developer.android.com/reference/android/net/ConnectivityManager.html

## Embedding the WebKit Browser

WebView is not significantly different than any other widget in Android

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >
<WebView android:id="@+id/webkit"
android:layout_width="fill_parent"
android:layout_height="fill_parent" />
</LinearLayout>
```

## Embedding the WebKit Browser

The Java code is equally simple

```
package com.commonsware.android.browser1;
import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;

public class BrowserDemo1 extends Activity
{
WebView browser;
@Override
public void onCreate(Bundle icicle)
{
super.onCreate(icicle);
setContentView(R.layout.main);
browser=(WebView)findViewById(R.id.webkit);
browser.loadUrl("http://commonsware.com");
}
}
```

AndroidManifest.xml

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.commonsware.android.browser1">
<uses-permission android:name="android.permission.INTERNET"
/>
<application android:icon="@drawable/cw">
<activity android:name=".BrowserDemo1"
android:label="BrowserDemo1">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

## Service Web

Un service web est un élément de logique applicative informatique distribuée (le plus souvent disponible sur Internet ou sur un intranet) permettant l'échange de données entre applications et systèmes hétérogènes.

Autour de cette définition, beaucoup de termes circulent : UUDI, WSDL, SOAP; XML-RPC, REST, service web etc.

Android propose un support pour SOAP : protocole bâti sur XML qui permet la transmission de message. Il est notamment utilisé dans le cadre d'architecture de type SOA (Service Oriented Architecture) pour le web. SOAP n'est pas de type REST.

Le transfert s'effectue le plus souvent à l'aide du protocole HTTP, mais un autre protocole (FTP, SMTP, etc.) peut également être utilisé.

Un message SOAP est composé de trois parties :

- une enveloppe contenant des informations sur le message afin de permettre son acheminement et son traitement ;
- un en-tête, inséré dans l'enveloppe et contenant, par exemple, des directives ou des informations contextuelles liées au traitement du message ;
- un modèle de données au format XML, pour les informations à transmettre, lui aussi intégré à l'enveloppe.

# Service Web

## Choix et présentation d'un service REST

REST (REpresentational State Transfer) est un style d?architecture pour les systèmes hypermédia distribués.

**Client-Serveur** : Les contraintes sont séparées entre le client et le serveur.
**Stateless** : Chaque requête d'un client vers un serveur doit contenir toute l'information nécessaire
**Cache** : Le serveur envoie une réponse qui donne l'information sur la propension de cette réponse à être mise en cache,
**Une interface uniforme** : Cette contrainte est sous-définie en 4 règles essentielles.

- L'identification des ressources : Chaque ressource est identifiée uniquement
- La manipulation des ressources à travers des représentations : Les ressources ont des représentations définies.
- Un message auto-descriptif : Les messages expliquent leur nature. Par exemple, si une représentation en HTML est codé en UTF-8, le message contient l'information nécessaire pour dire que c'est le cas.
- Hypermédia comme le moteur de l'état de l'application : Chaque accès aux états suivants de l'application est décrit dans le message courant.

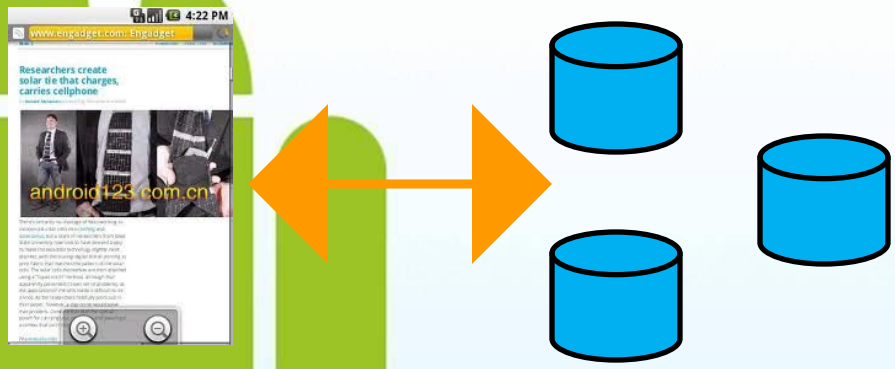
**Un système hiérarchisé** : Les états de l'application sont identifiées par des ressources individuelles. Toute l'information n'est pas envoyée dans une seule ressource unique. Les requêtes/réponses entre le client et le serveur augmentent et donc peut faire baisser les performances d'où l'importance du cache, etc. Le bénéfice est que cela rend beaucoup plus flexible l'évolution du système.

**Code-On-Demand** (Optionnel) : La possibilité pour les clients d'exécuter des scripts obtenus depuis le serveur. Cela permet d'éviter que le processing ne se fasse que du côté serveur et permet donc de faire évoluer les fonctionnalités du client au cours du temps.

## HTTP GET communication (java.net / android.net)

In this recipe, the HTTP GET is used to retrieve XML or JSON data (see http://www.json.org/ for an overview)

Android includes three XML parsers: the traditional W3C DOM parser (org.w3c.dom), a SAX parser (org.xml.sax), and the XML pull parser. It also has a JSON parser (org.json).




To search any topic, the topic just needs to be appended to the query. For example, to search information on the National Basketball Association (NBA), the following query returns JSON data:

**http://ajax.googleapis.com/ajax/services/search/web?v=1.0&q=NBA**

The activity needs Internet permission to run. So, the following should be added to the AndroidManifest XML file:

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

## HTTP GET communication (java.net / android.net)

### Retrieving data on the web :

```
public String SearchRequest(String searchString)
throws MalformedURLException, IOException
 {
String newFeed=url+searchString;
StringBuilder response = new StringBuilder();
Log.v("gsearch","gsearch url:"+newFeed);
URL url = new URL(newFeed);
HttpURLConnection httpconn = (HttpURLConnection) url.openConnection();

if(httpconn.getResponseCode()==HttpURLConnection.HTTP_OK)
{
BufferedReader input = new BufferedReader(
new InputStreamReader(httpconn.getInputStream()),8192);
String strLine = null;
while ((strLine = input.readLine()) != null) {
response.append(strLine);
}
input.close();
}
return response.toString();
}
```

Open connection

Get response from server

Read data from connection

## HTTP GET communication (java.net / android.net)

### Parsing data :

```
public void ProcessResponse(String resp) throws IllegalStateException,
IOException, JSONException, NoSuchAlgorithmException
{
StringBuilder sb = new StringBuilder();
Log.v("gsearch","gsearch result:"+resp);
JSONObject mResponseObject = new JSONObject(resp);
JSONObject responObject = mResponseObject.getJSONObject("responseData");
JSONArray array = responObject.getJSONArray("results");
Log.v("gsearch","number of resultst:"+array.length());


for(int i = 0; i<array.length(); i++)
{
Log.v("result",i+"] "+array.get(i).toString());
String title = array.getJSONObject(i).getString("title");
String urllink = array.getJSONObject(i)
.getString("visibleUrl");
sb.append(title);
sb.append("\n");
sb.append(urllink);
sb.append("\n");
}
tv1.setText(sb.toString());
}
…
```

JASON object

Parsing JASON data

Display text

## HTTP POST communication (java.net / android.net)

Les requêtes de type POST permettent d'envoyer des données au composant logiciel situé à l'URL utilisée, par exemple la soumission d'un formulaire.

```
URL url = new URL(newFeed);

HttpURLConnection httpconn = (HttpURLConnection) url.openConnection();

httpconn.setRequestMethod(POST);
```

# Développement avancé : Networking

TP

ANDROID_ACTIVITY_WebViewActivity
ANDROID_WEBSERVICE_JSON

# Localisation && Géolocalisation

https://developers.google.com/maps/documentation/android/intro

http://glennsayers.com/android-mapping/

http://stackoverflow.com/questions/16596715/how-can-i-create-an-android-application-in-android-studio-that-uses-the-google-m

http://www.todroid.com/how-to-create-a-google-map-application-using-android-studio/
http://fr.slideshare.net/apps4ucoza/meetup-google-maps-v2

# Développement avancé : Location

**A mobile device with accurate location awareness is very powerful. Combining location awareness with network data access is world changing**

# Localisation

**Déterminer la localisation**
La plupart des appareils Android permettent de déterminer la géolocalisation actuelle. Cela peut être fait par l'intermédiaire d'un GPS (Global Positioning System) du module, par triangulation tour de téléphonie cellulaire ou via les réseaux wifi. Android contient le paquet android.location qui fournit l'API pour déterminer la position géographique actuelle.

**LocationManager**

La classe LocationManager donne accès à un service de localisation Android. Ce service permet aux fournisseurs de localisation, de mettre à jour une localisation avec un listener.

**LocationProvider**

La classe LocationProvider est la superclasse des fournisseurs de moyen de localisation (wifi, GPS)l. Cette information est stockée dans  Location class.
L'appareil Android peut avoir plusieurs LocationProvider disponibles et vous pouvez sélectionner celui que vous souhaitez utiliser (en principe GPS).

# Géolocalisation

Toutes les APIs  Google de cartographie, imagerie, web service

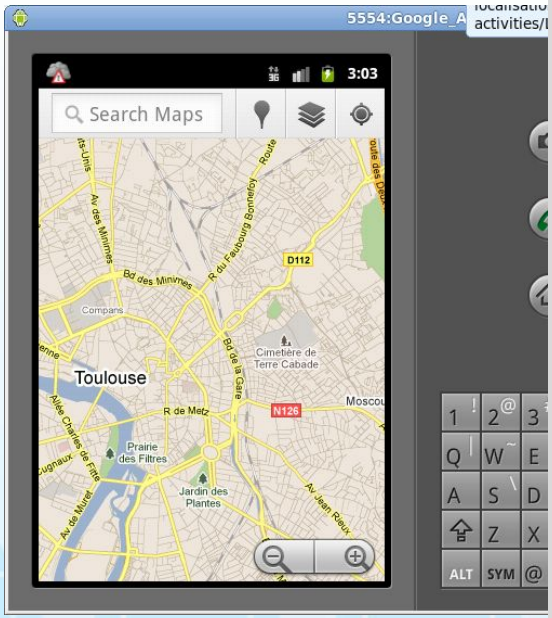
JavaScript API v3Embed an interactive Google Map in your webpage using JavaScript.
Maps for BusinessGoogle Maps API for Business provides Enterprise-ready application support for your mapping needs.
Google Maps SDK for iOSAdd Google Maps to your iOS apps, with rotation, tilt & 3D buildings. Launch the Google Maps app using a URL scheme
Maps Android APIAdd interactive, vector-based maps to your Android application.
Web ServicesUse URL requests to access geocoding, directions, elevation, place and time zone information.

Places APIAccess information about establishments, geographic locations, or prominent points of interest.
Maps Image APIsEmbed a Google Maps image or Street View panorama in your web page without requiring JavaScript.
Static Maps - Street ViewGoogle Earth APITake your visitors anywhere on the Earth without leaving your web page.

**Map v2 est Disponible pour Android et IOS**

https://developers.google.com/maps/documentation/

## Simple intent with location API




```
public class LaunchDemo extends Activity
{
      private EditText lat;
      private EditText lon;

      @Override
      public void onCreate(Bundle icicle)
      {
            super.onCreate(icicle);
            setContentView(R.layout.main);

            Button btn=(Button)findViewById(R.id.map);
            lat=(EditText)findViewById(R.id.lat);
            lon=(EditText)findViewById(R.id.lon);

            btn.setOnClickListener(new View.OnClickListener()
            {
                  public void onClick(View view) {
                        String _lat=lat.getText().toString();
                        String _lon=lon.getText().toString();
                        Uri uri=Uri.parse("geo:"+_lat+","+_lon);
                        startActivity(new Intent(Intent.ACTION_VIEW, uri));
                  }
            });
      }
}
```

## Google Play Service

**Principe** : Google Play service offre un ensemble de service tel que les Maps, Google+. Le client distant configure des interfaces avec les différents services Google et accède ensuite à ces services avec un crédit. Chaque projet doit être enregistrée auprès du service Google Play. Dans un projet on peut avoir plusieurs applications qui sont alors identifiées par le certificat du développeur et le package.

L'accès est limitée par le "courtesy limits" de chaque API, avec une facturation si l'on veut dépasser cette limite.

**Conditions d'accès et configuration**

1) Avoir un compte Gmail
2) Inscrire un projet : https://code.google.com/apis/console
3) S'abonner à des services
4) Obtenir les clés d'activation avec son certificat
5) Intégrer les clés dans le manifest de l'application
6) Ajouter la librairie GooglePlayService

https://developers.google.com/maps/documentation/android/

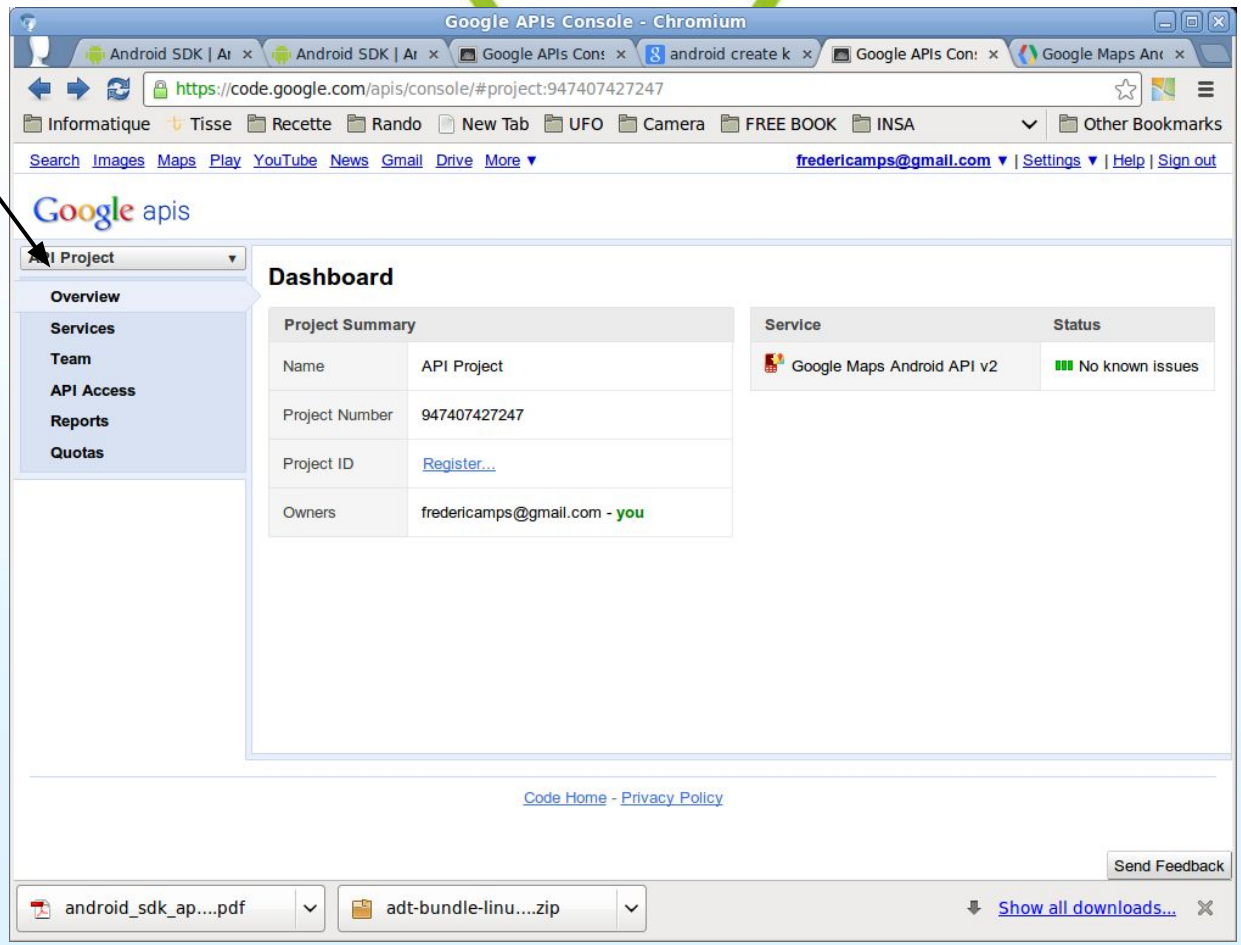
## Conditions d'utilisation de Google Maps et de Google Earth

http://www.google.com/intl/fr/help/legalnotices_maps.html

# Google MAP - Enregistrement du projet

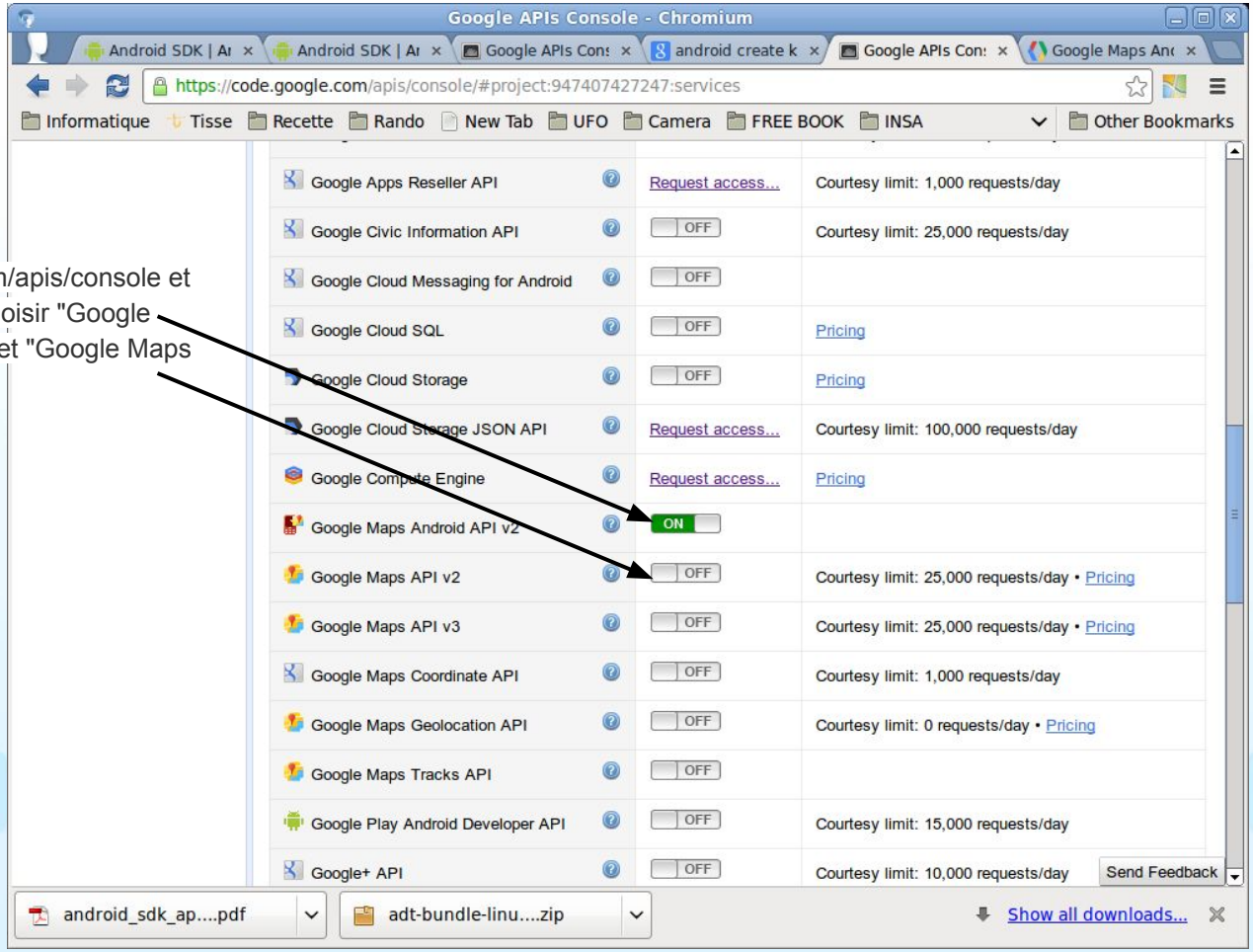
https://code.google.com/apis/console

Déclarer un projet



## Google MAP - Enregistrement du service Google

Aller sur https://code.google.com/apis/console et dans l'onglet service choisir "Google Maps Android API V2" et "Google Maps v2"



## Google MAP - Génération du certificat

La génération du certificat se fait par keytool, utilitaire livré dans le JDK Java.

```
keytool -genkey -v -keystore ~/.android/debug.keystore -alias android -keyalg RSA -keysize 2048 -validity 10000

Entrez le mot de passe du fichier de clés :
Ressaisissez le nouveau mot de passe :
...
Est-ce CN=android, OU="CN=Android Debug, O=Android, C=US", O=android, L=NY, ST=NYC, C=FR ?
[non]:

Génération d'une paire de clés RSA de 2 048 bits et d'un certificat auto-signé (SHA256withRSA) d'une validité de
10 000 jours
      ...
Key:  Sun RSA public key, 2048 bits
modulus: 18912886958337454000364282356476209226240389420416461806846300580064...94254618994859940217 53
public exponent: 65537
Validity: [From: Fri Jan 18 14:37:05 CET 2013,To: Tue Jun 05 15:37:05 CEST 2040]
Issuer: CN=android, OU="CN=Android Debug, O=Android, C=US", O=android, L=NY, ST=NYC, C=FR SerialNumber: [4b1a126a]
```

# Google MAP - Type de signature

## Signature en mode Debug

Les outils de compilation Android fournissent un mode de signature en mode debug qui permet de déboguer tout en répondant aux exigences du système Android pour signer les fichiers APK. Lorsque vous utilisez le mode debug pour construire votre application, les outils SDK invoquent keytool pour créer automatiquement un fichier de clés de débogage et une clés.

Les outils de débogage SDK créent le fichier de clés  et la clés avec des noms et  mot de passe prédéfinis:
Nom de fichier de clés: "debug.keystore"
Mot de passe keystore: "android"
Alias clés: "androiddebugkey"
Mot de passe clé: "android"
CN: "CN = Android Debug, O = Android, C = US"

## Signature en mode Release

En prévision de la signature de votre application, vous devez d'abord vous assurer que vous avez une clé privée adéquate pour signer. Une clé privée adéquate est :

- la clé doit être en votre possession
- la clé représente l'entité personnelle, organisationnelle ou de l'organisation à être identifié avec l'application
- elle a une période de validité qui dépasse la durée de vie prévue de votre d'applications. Une période de validité de plus de 25 ans est recommandé.
- Si vous envisagez de publier votre application (s) sur Google Play, notez que la période de validité se termine après le 22 Octobre 2033. Vous ne pouvez pas uploader  une application si elle est signée avec une clé dont la validité expire avant cette date.

## Google MAP - Génération de la signature du certificat

```
keytool -list -v -keystore ~/.android/debug.keystore
Tapez le mot de passe du Keystore : xxxxx

Type Keystore : JKS
Fournisseur Keystore : SUN

Votre Keystore contient 1 entree(s)

Nom d'alias : androiddebugkey
Date de création : 15 fevr. 2012
Type dentrée: {0}
Longueur de chaine du certificat : 1
Certificat[1]:
Propriétaire: CN=Android Debug, O=Android, C=US
emetteur: CN=Android Debug, O=Android, C=US
Numero de serie: 52f320f2
Valide du: Wed Feb 15 16:30:31 CET 2012 au: Fri Feb 07 16:30:31 CET 2042
Empreintes du certificat:
        MD5:  82:94:BA:0D:E6:58:E7:2D:01:50:90:D1:FD:6E:79:CD
        SHA1: A2:31:10:69:AF:94:3E:7C:80:0B:5A:E1:BE:FA:E1:EC:4A:F3:2B:58
        Nom de lalgorithme de signature: {7}
        Version: {8}
```

1) Le fichier des clés debug.keystore se trouve dans $HOME/.android/

2) Utiliser keytool (utilitaire de Oracle, installé avec Java) pour générer une signature de certificat de type SHA1.

## Google MAP - Génération de la clé API

Demande d'une clé , donner ici la signature SHA1 de votre certificat suivi du package :

```
E3:29:52:6B:94:4E:6E:20:59:DD:0
F:41:3D:6B:6D:C1:3E:92:29:9B;co
m.example.mapdemo
```

Google APIs Console - Chromium

https://code.google.com/apis/console/#project:947407427247:access

Google apis

API Project

- Overview
- Services
- Team
- API Access
- Reports
- Quotas

**API Access**

To prevent abuse, Google places limits on API requests. Using a valid OAuth token or API key allows you to exceed anonymous limits by connecting requests back to your project.

**Authorized API Access**

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private. A single project may contain up to 20 client IDs. Learn more

Create an OAuth 2.0 client ID...

**Simple API Access**

Use API keys to identify your project when you do not need to access user data. Learn more

**Key for Android apps (with certificates)**

API key: AIzaSyA5E9hPcDU9OKbCwbAgsQvlnNtD-V8km9M

Android apps: A2:31:10:69:AF:94:3E:7C:80:0B:5A:E1:BE:FA:E1:EC:4A:F3:2B:58;Android

Activated on: Jan 16, 2013 3:37 AM

Activated by: fredericamps@gmail.com – you

Generate new key... | Edit allowed Android apps... | Delete key...

**Key for browser apps (with referers)**

API key: AIzaSyCI90HHgu0WU5-94_DNCA9FnfcafeQaEqQ

Referers: Any referer allowed

Activated on: Jan 16, 2013 3:25 AM

Activated by: fredericamps@gmail.com – you

Generate new key... | Edit allowed referers... | Delete key...

Create new Server key... | Create new Browser key... | Create new Android key...

https://developers.google.com/maps/documentation/android/start#getting_the_google_maps_android_api_v2

## Google MAP - Intégration de la clé dans le manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.mapdemo"
   android:versionCode="1"
   android:versionName="1.0" >

   <permission
   android:name="com.example.mapdemo.permission.MAPS_RECEIVE"
   android:protectionLevel="signature"/>
  <uses-permission android:name="com.example.mapdemo.permission.MAPS_RECEIVE"/>

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="17" />

   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
   <!-- External storage for caching. -->
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <!-- My Location -->
   <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
   <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

   <uses-feature
      android:glEsVersion="0x00020000"
      android:required="true" />
```

permissions

OpenGL ES v2

## Google MAP - Intégration de la clé dans le manifest (suite)

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <meta-data
      android:name="com.google.android.maps.v2.API_KEY"

android:value="AIzaSyDQvLf3ICBMdiGT-kvmK7kQAFVOShZqdFY" />

    <activity
      android:name="com.example.mapdemo.BasicMapActivity"
      android:label="@string/app_name" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
```

clé API Google Map v2

## Google MAP - Librairie - Configuration Eclipse

Google API 4.1.2

Import de Google Play Service

Android Support v4

Java - Eclipse

File Edit Run Source Navigate Search Project Refactor Window Help

Package Explorer

- ENSICA_JAVA
- GMM
- google-play-services_lib
- MainActivity [MainActivity]
  - src
    - com.example.mapdemo
    - com.example.mapdemo.view
  - gen [Generated Java Files]
  - Google APIs [Android 4.1.2]
    - android.jar - /home/fcamps/Softs/android-sdk/platforms/android-16
    - effects.jar - /home/fcamps/Softs/android-sdk/add-ons/addon-google_apis-google-16/libs
    - usb.jar - /home/fcamps/Softs/android-sdk/add-ons/addon-google_apis-google-16/libs
    - maps.jar - /home/fcamps/Softs/android-sdk/add-ons/addon-google_apis-google-16/libs
  - Android Dependencies
    - google-play-services_lib.jar - /home/fcamps/Softs/android-sdk/extras/google/google_play_services/libproject/google-play-services_lib/bin
    - google-play-services.jar - /home/fcamps/Softs/android-sdk/extras/google/google_play_services/libproject/google-play-services_lib/libs
  - Referenced Libraries
    - android-support-v4.jar - /home/fcamps/Softs/android-sdk/extras/android/support/v4
  - assets
  - bin
  - res
  - AndroidManifest.xml 3 17/01/13 22:49 frederiCamps@gmail.com
  - proguard.cfg 3 17/01/13 22:49 frederiCamps@gmail.com
  - project.properties 3 17/01/13 22:49 frederiCamps@gmail.com
- TD_ENSICA
- test2Map
- testMap

Task List

Connect Mylyn

Connect to your task and ALM tools or create a local task.

Outline

An outline is not available.

Log

| ID | TID | Application |
|---|---|---|
| 032 | 8032 | com.example.mapdem |
| 032 | 8032 | com.example.mapdem |
| 032 | 8032 | com.example.mapdem |
| 032 | 8032 | com.example.mapdem |
| 032 | 8036 | com.example.mapdem |
| 032 | 8036 | com.example.mapdem |
| 032 | 8036 | com.example.mapdem |
| 032 | 8032 | com.example.mapdem |
| 032 | 8057 | com.example.mapdem |
| 032 | 8049 | com.example.mapdem |

MainActivity

## Google MAP - Librairie - Configuration Gladle/IntelliJ

http://developer.android.com/google/play-services/setup.html

## Google MAP - Librairie - Configuration Eclipse

Google API 4.1.2

Import de Google Play Service

Properties for MainActivity

type filter text

- Resource
- Android
- Android Lint Preferen
- Builders
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Project References
- Run/Debug Settings
- Subversion
- Task Repository
- Task Tags
- Validation
- WikiText

**Android**

Project Build Target

| Target Name | Vendor | Platform | API Level |
|---|---|---|---|
| Android 2.3.3 | Android Open Source Project | 2.3.3 | 10 |
| Google APIs | Google Inc. | 2.3.3 | 10 |
| Intel Atom x86 System Image | Intel Corporation | 2.3.3 | 10 |
| Android 3.0 | Android Open Source Project | 3.0 | 11 |
| Android 4.1.2 | Android Open Source Project | 4.1.2 | 16 |
| Google APIs | Google Inc. | 4.1.2 | 16 |
| Android 4.2 | Android Open Source Project | 4.2 | 17 |
| Google APIs | Google Inc. | 4.2 | 17 |

Android + Google APIs

Library

Is Library

| Reference | Project |
|---|---|
| ../../libproject/google-play-services_lib | google-play-services_lib |

Add... Remove Up Down

Restore Defaults Apply

Cancel OK

## Google MAP - Librairie - Configuration Eclipse

Android Support v4

Import de Google Play Service

Google API 4.1.2

Properties for MainActivity

type filter text

- Resource
- Android
- Android Lint Preferen
- Builders
- Java Build Path
- Java Code Style
- Java Compiler
- Java Editor
- Javadoc Location
- Project References
- Run/Debug Settings
- Subversion
- Task Repository
- Task Tags
- Validation
- WikiText

Java Build Path

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

- android-support-v4.jar - /home/fcamps/Softs/android-sdk/extras/android/support/v4
  - Source attachment: (None)
  - Javadoc location: (None)
  - Native library location: (None)
  - Access rules: (No restrictions)
- Android Dependencies
  - Access rules: No rules defined
  - Native library location: (None)
  - google-play-services_lib.jar - /home/fcamps/Softs/android-sdk/extras/google/google_play_services/libproject/goo
  - google-play-services.jar - /home/fcamps/Softs/android-sdk/extras/google/google_play_services/libproject/google-
- Google APIs [Android 4.1.2]
  - Access rules: No rules defined
  - Native library location: (None)
  - android.jar - /home/fcamps/Softs/android-sdk/platforms/android-16
  - effects.jar - /home/fcamps/Softs/android-sdk/add-ons/addon-google_apis-google-16/libs
  - usb.jar - /home/fcamps/Softs/android-sdk/add-ons/addon-google_apis-google-16/libs
  - maps.jar - /home/fcamps/Softs/android-sdk/add-ons/addon-google_apis-google-16/libs

Add JARs... | Add External JARs... | Add Variable... | Add Library... | Add Class Folder... | Add External Class Folder... | Edit... | Remove | Migrate JAR File...

Cancel | OK

## Google MAP - Activité de base

Fragment

Chargement du layout

```
public class BasicMapActivity extends
android.support.v4.app.FragmentActivity {
        private GoogleMap mMap;

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

    @Override
    protected void onResume() {
        super.onResume();
        setUpMapIfNeeded();
    }
```

```
private void setUpMapIfNeeded() {
  if (mMap == null) {
      mMap = ((SupportMapFragment)
      getSupportFragmentManager()
                                .findFragmentById(R.id.map))
                                .getMap();
       if (mMap != null) {
                setUpMap();
            }
        }
    }

private void setUpMap()
 {
   mMap.addMarker(new MarkerOptions().position(new
LatLng(0, 0)).title("Marker"));
    }
}
```

## Google MAP - Activité de base - Layout

Ici le layout intègre un fragment de type MapFragment qui est chargé par l'activité

Lien avec l'objet GoogleMap dans l'activité

```
<?xml version="1.0" encoding="utf-8"?>
<fragment
xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/map"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  class="com.google.android.gms.maps.SupportMapFragment"/>
```

## Google Map v2 Earth - API

Google préconise d'utiliser l'API `android.support.v4.app.FragmentActivity` pour une meilleure intégration dans les layout. En effet les fragement peuvent s'intégrer plus facilement en fonction du type d'équipement tablette/SmartPhone car on peut les gérer plus facilement en fonction de la taille de l'écran.

Fragment dans un layout dont la référence est un objet Class GoogleMap

Class Marker + Class LatLng

CameraUpdate center=CameraUpdateFactory sont utilisés pour la position de la caméra.

https://developers.google.com/maps/documentation/android/

https://developers.google.com/maps/documentation/android/reference/com/google/android/gms/maps/package-summary

# Google Map v2 Earth - Intégration

Procédure officielle pour l'intégration des lib (attention suivre rigoureusement celle-ci) :
https://docs.google.com/document/pub?id=19nQzvKP-CVLd7_VrpwnHfl-AE9fjbJySowONZZtNHzw

**Map Fragment / MapView v2 Quick Start**

The Maps APIs for Android have been updated in the latest release of the Google Play Services library and have been rolled out to Android devices with the Play Store. To test it out:

1. Create a new project
    1. File -> New -> Android Application Project
    2. You can leave all the options with their default values
    3. Choose BlankActivity in the Create Activity screen
1. Ensure you have the latest version (v2) of the Google Play Services installed.
    1. Select Window > Android SDK Manager or run android at the command line.
    2. Scroll to the bottom of the package list and select Extras > Google Play services. The Google Play services SDK is downloaded to your computer and installed in your Android SDK environment at `<android-sdk-folder>/extras/google/google_play_services/`
1. Add the Google Play Services project into your Eclipse workspace.
    1. Click File -> Import..., select Android -> Existing Android Code into Workspace
    2. Browse to and select `<android-sdk-folder>/extras/google/google_play_services/libproject/google-play-services_lib`
1. To add the dependency to Google Play Services into your project
    1. Project -> Properties -> Android -> Library, Add -> google-play-services_lib

# Google Map v2 Earth - Intégration

Procédure pour l'intégration des lib (attention suivre rigoureusement celle-ci) :

**Map Fragment / MapView v2 Quick Start**

The Maps APIs for Android have been updated in the latest release of the Google Play Services library and have been rolled out to Android devices with the Play Store. To test it out:

1. Project -> Properties -> Android -> Library, Add -> google-play-services_lib
1. Update `MainActivity.java` to extend `FragmentActivity` instead of `Activity`
2. Check that Google Play Services is available on the device at runtime to protect against error cases

```
GooglePlayServicesUtil.isGooglePlayServicesAvailable(getApplicationContext());
```

1. Update `res/layout/activity_main.xml` and replace the entire contents with

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/map"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 class="com.google.android.gms.maps.SupportMapFragment"/>
```

1. The map is controlled through the `GoogleMap` object, use `findFragmentById`to keep a reference to the `GoogleMap` from the layout.

```
GoogleMap map = ((SupportMapFragment)  getSupportFragmentManager().findFragmentById(R.id.map))
              .getMap();
```

# Google Map v2 Earth - Intégration

Procédure pour l'intégration des lib (attention suivre rigoureusement celle-ci) :

**Map Fragment / MapView v2 Quick Start**

To use Google Maps API you need an API Key:

1. Go to the Google APIs Console
2. In Services enable 'Google Maps Android API v2' and agree to the terms
3. In the left navigation bar, click API Access.
4. Click Create New Android Key....
5. In the resulting dialog, enter the SHA-1 fingerprint, then a semicolon, then your application's package name. For example:
    1. BB:0D:AC:74:D3:21:E1:43:67:71:9B:62:91:AF:A1:66:6E:44:5D:75;com.example.android.mapexample
1. If you have debug and production keys then add both in.
    1. To obtain the SHA1 of your debug certificate use keytool:

```
keytool -list -alias androiddebugkey -keystore ~/.android/debug.keystore -storepass android
-keypass android -v
```

    1. For Windows computers the .android directory is in your user directory
    2. Your new API key will be a 40 character string like `AIzbSyBaEqeUpYvlsbUe0OqTPn1-ABVMHVbdNxo`
1. Add the following tag into your `AndroidManifest.xml` just before the closing `</application>` tag

```
<meta-data
   android:name="com.google.android.maps.v2.API_KEY"
   android:value=" your_api_key"/>
```

# Google Map v2 Earth - Intégration

Procédure pour l'intégration des lib (attention suivre rigoureusement celle-ci) :

**Map Fragment / MapView v2 Quick Start**

Your application now also needs the following permissions in the `AndroidManifest.xml`

```
<permission android:name="your_package_name.permission.MAPS_RECEIVE"
 android:protectionLevel="signature"/>
<uses-permission  android:name="your_package_name.permission.MAPS_RECEIVE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
```

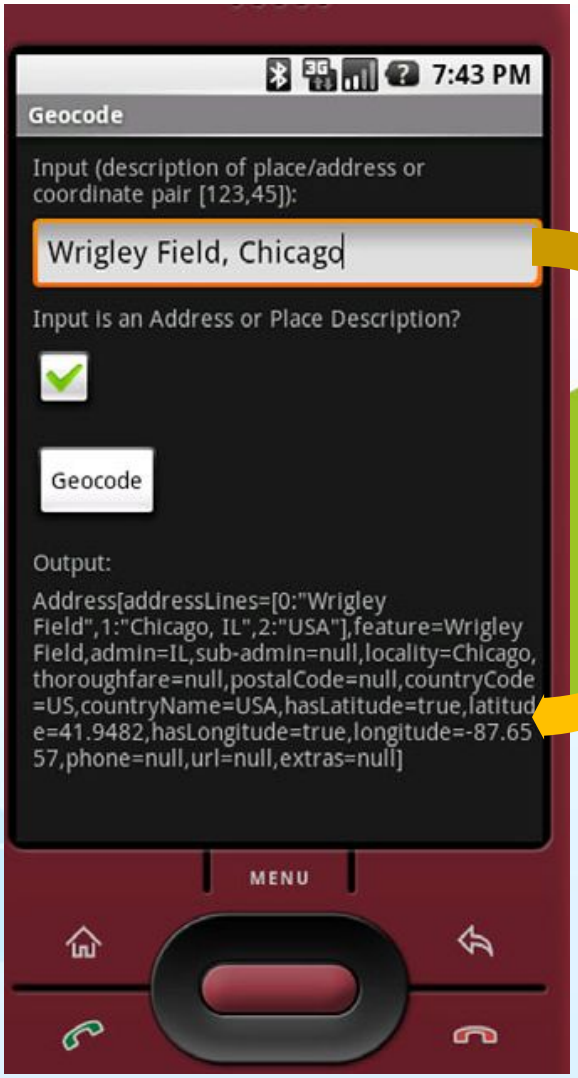
Optionally you can include one or both of these permissions to enable auto-location

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

Maps v2 uses OpenGl so the following uses-feature is also required

```
<uses-feature android:glEsVersion="0x00020000" android:required="true"/>
```

Test your application, checking the logcat output for any errors.




## *Converting places and addresses with Geocoder*

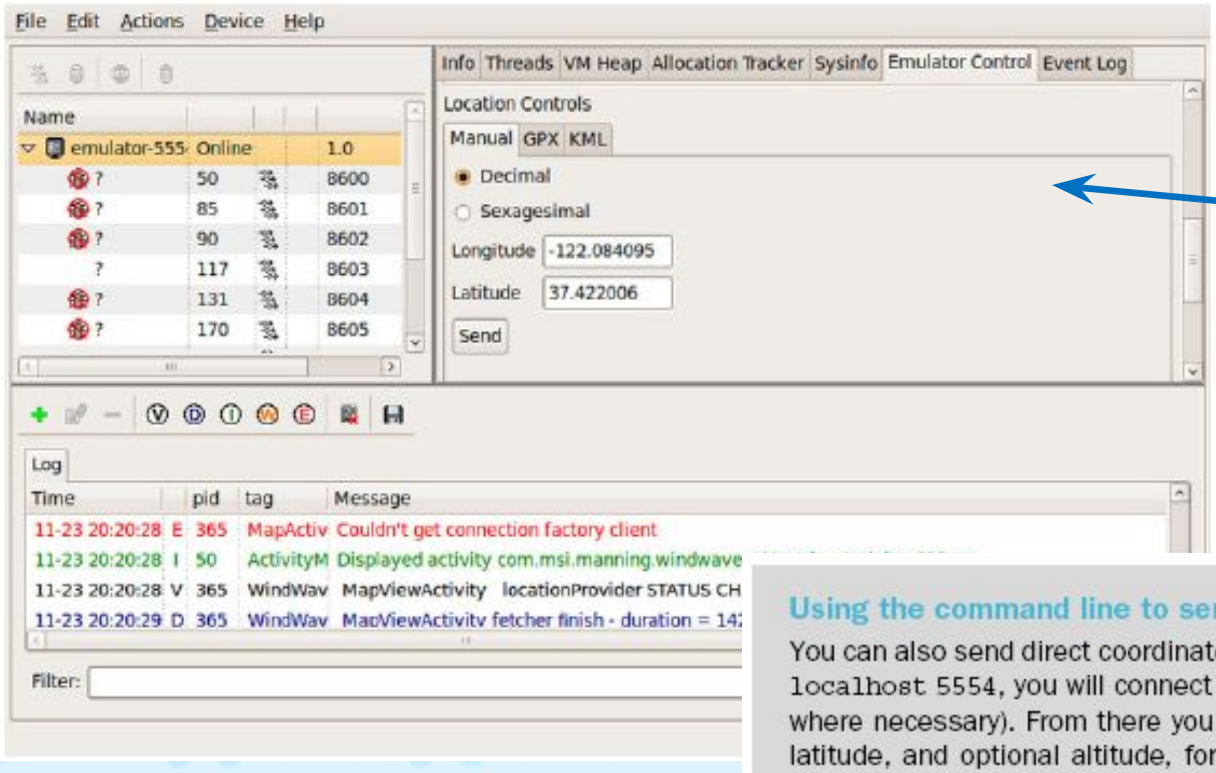
Reverse geocoding is the opposite, converting latitude and longitude into an address. To accomplish this, the Geocoder class makes a network call (automatically) to a web service.

```
Geocoder geocoder = new Geocoder(this);
if (isAddr) {
try {
List<Address> addresses =
geocoder.getFromLocationName(in, 1);
if (addresses != null) {
result = addresses.get(0).toString();
}
} catch
```

# Géolocalisation et émulation

## *Simulating your location within the emulator*

Using the DDMS tool or by using the command line within the shell. The fastest way to get started is to send in direct coordinates through the DDMS tool.




DDMS

Cmd line

### Using the command line to send coordinates

You can also send direct coordinates from within the emulator console. If you `telnet localhost 5554`, you will connect to the default emulator's console (adjust the port where necessary). From there you can use the geo fix command to send longitude, latitude, and optional altitude, for example, `geo fix -21.55 64.1`. Again keep in mind that the Android tools require that longitude be the first parameter

## *Simulating your location within the emulator*

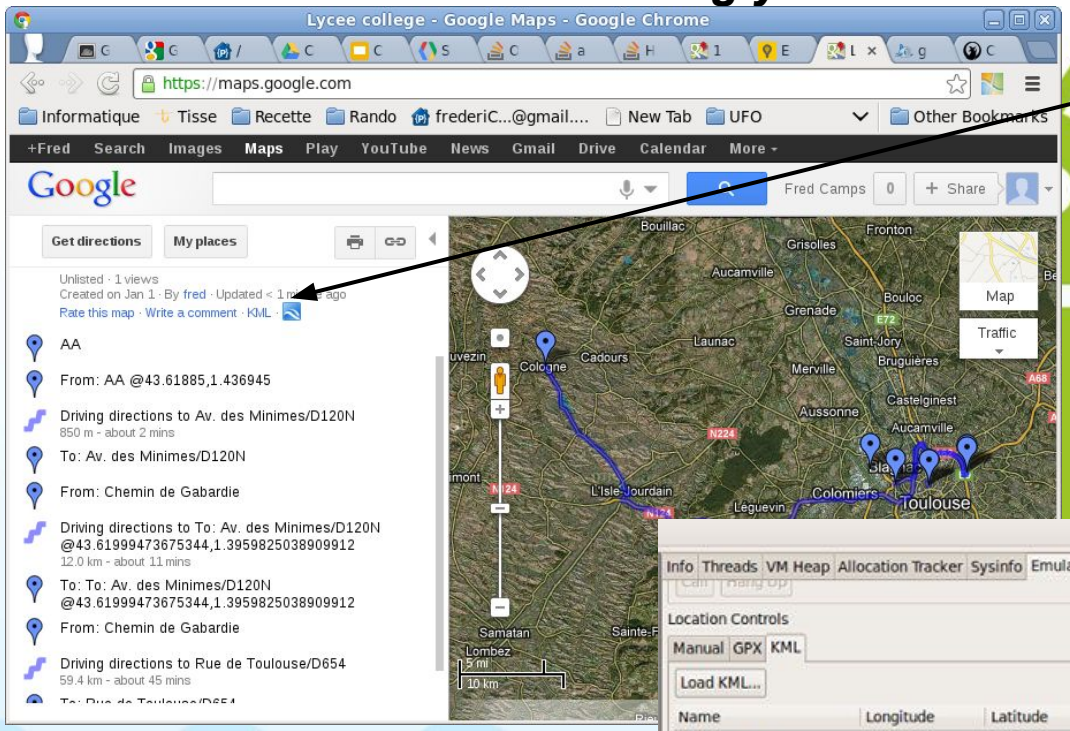
### *The GPS Exchange Format*

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx xmlns="http://www.topografix.com/GPX/1/1"
version="1.1"
creator="Charlie Collins - Hand Rolled"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.topografix.com/GPX/1/1
http://www.topografix.com/GPX/1/1/gpx.xsd">
<metadata>
<name>Sample Coastal California Waypoints</name>
<desc>Test waypoints for use with Android</desc>
<time>2008-11-25T06:52:56Z</time>
<bounds minlat="25.00" maxlat="75.00"
minlon="100.00" maxlon="-150.00" />
</metadata>
<wpt lat="41.85" lon="-124.38">
<ele>0</ele>
<name>Station 46027</name>
<desc>Off the coast of Lake Earl</desc>
</wpt>
…
```

Plugin Firefox GPX
http://www.elsewhere.org/journal/gmaptogpx

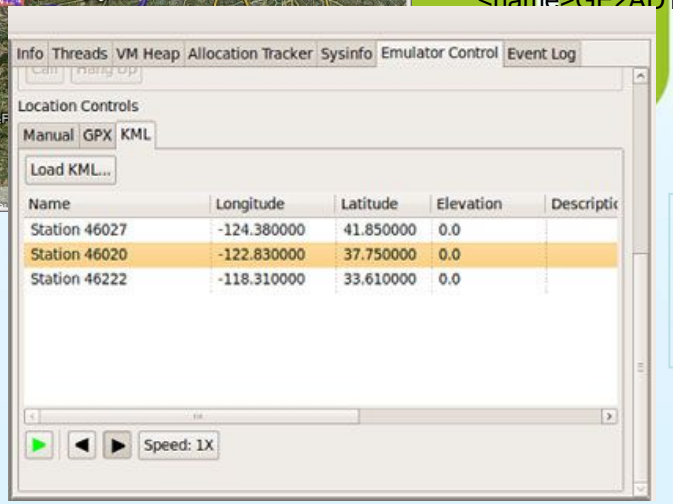
https://maps.google.com/ pour KML--> conversion pour adt : http://ge2adt.appspot.com/

## *Simulating your location within the emulator*



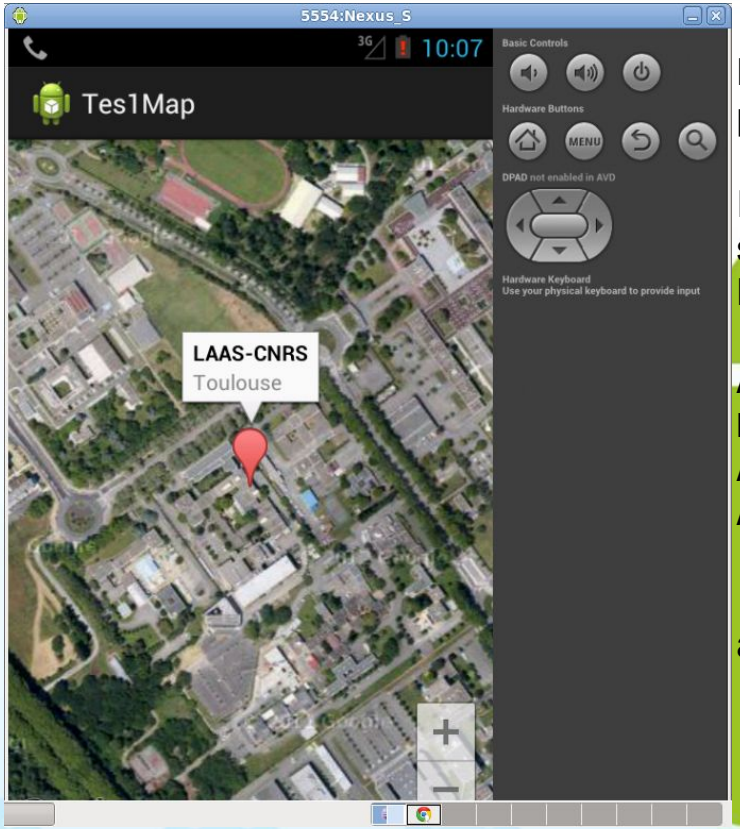

Fichier **KML** à partir de https://maps.google.com/

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Document>
  <name>GE2ADT</name>
    <Placemark><name>0</name><description>GE2ADT
Route</description><Point><coordinates>1.436945,43.618851
,0.000000</coordinates></Point></Placemark>
</Document>
</kml><?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Document>
  <name>GE2ADT</name>
```




GPX is an XML schema (http://www.topografix.com/GPX/1/1/) that allows you to store waypoints, tracks, and routes

# Développement avancé : MAPS & Earth




Par défaut les émulateurs actuels ne sont pas livrés avec Google play service, ce qui n'autorise pas l'utilisation de l'API Map.

Il est possible d'installer les bibliothèques à partir de la sauvegarde d'un équipement. Exemple à partir d'un émulateur Nexus S :

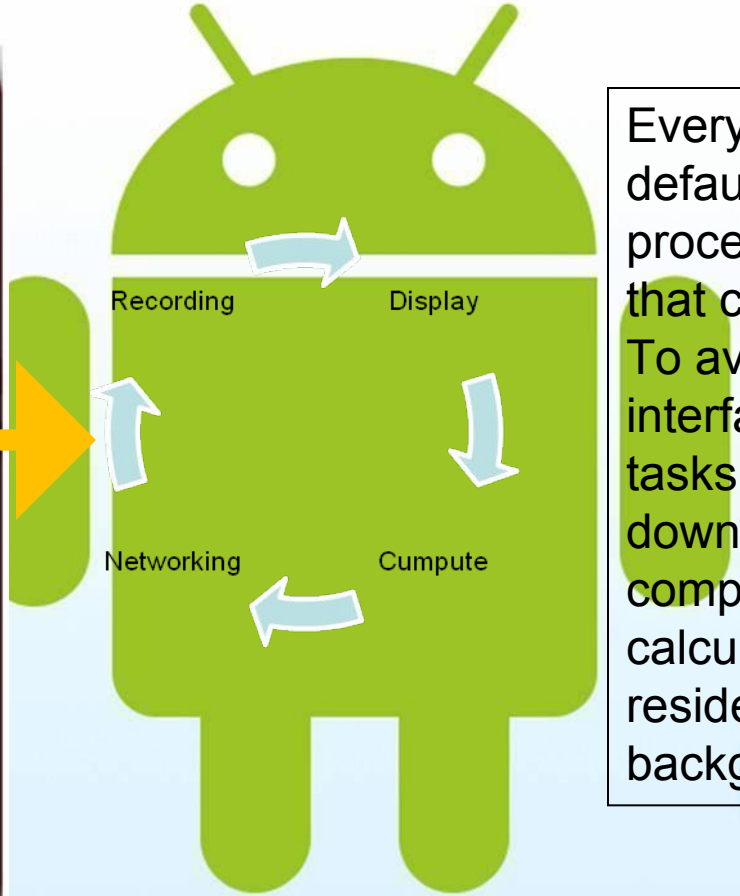
```
AVD_for_Nexus_s_by_Google
Nexus 480 x 800
Android 4.1.2 - API Level 16
ARM armeabi- v7a
(toutes options par défaut)
```

ajouter les apk suivants :

```
[fcamps@discovery Downloads]$ adb -s emulator-5554  install com.android.vending-108b71f4793186b8fd1768a9cd19bbd5.apk
1866 KB/s (6158861 bytes in 3.221s)
        pkg: /data/local/tmp/com.android.vending-108b71f4793186b8fd1768a9cd19bbd5.apk
Success
[fcamps@discovery Downloads]$ adb -s emulator-5554  install com.google.android.gms-6413100ccfca460007fdc988b1363db8.apk
1061 KB/s (4075228 bytes in 3.748s)
        pkg: /data/local/tmp/com.google.android.gms-6413100ccfca460007fdc988b1363db8.apk
Success
```

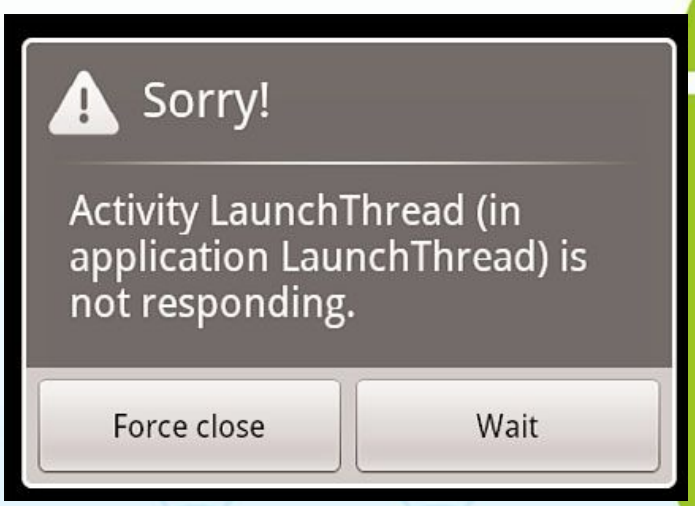
Multithreading

# Développement avancé : Thread




Every application by default runs a single process upon creation that contains all the tasks. To avoid hanging the user interface, time-consuming tasks, such as network downloads or computationally intensive calculations, should reside in a separate background

# Développement avancé : Thread




Most applications can benefit from the use of threads. If such occasions are not detected in the software design phase, they quickly display during testing because the Android system provides an alert to the user when the user interface (UI) hangs

## Launching a Secondary Thread

A ring-tone song is played when an onscreen button is pressed.This provides a simple illustration of how threads can be used with a time-consuming operation.

```
Button startButton = (Button) findViewById(R.id.trigger);
startButton.setOnClickListener(new View.OnClickListener()
{
public void onClick(View view)
{
// BAD USAGE: function call to time-consuming
// function causes main thread to hang
play_music();
}
});
```

This means any user request such as navigating back to the home screen or multiple pushes of an onscreen button are not registered until the music is completely finished playing.The unresponsive UI might even cause the Android system to show an error

## Launching a Secondary Thread

```
Button startButton = (Button) findViewById(R.id.trigger);
    startButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view){

            //standalone play_music() function call causes
            //main thread to hang.  Instead, create
            //separate thread for time-consuming task
            Thread initBkgdThread = new Thread(new Runnable() {
                public void run() {
                    play_music();
                }
            });
            initBkgdThread.start();
        }
    });
```

Thread

Loop

Start

## Launching a Secondary Thread

Method executed by thread

```
private void play_music()
{

for(int ii=0; ii<12; ii++) {
//check to ensure main activity not paused
if(!paused) {
if(m_mediaPlayer != null) {m_mediaPlayer.release();}
m_mediaPlayer = MediaPlayer.create(this, notes[ii%4]);
m_mediaPlayer.start();
try {
Thread.sleep(NOTE_DURATION);
} catch (InterruptedException e) {
e.printStackTrace();
}
}
}
}
```

```
boolean paused = false;
@Override
protected void onPause() {
paused = true;
super.onPause();
}
@Override
protected void onResume() {
super.onResume();
paused = false;
}
}
```

Check state of activity

Change state

## Computationally intensive function

If the main purpose of the activity is this time-consuming operation, it is natural to have the activity itself implement Runnable

```
public class EdgeDetection extends Activity implements Runnable
{
@Override
public void onCreate(Bundle savedInstanceState)
 {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
Thread thread = new Thread(EdgeDetection.this);
 thread.start();
}

@Override
public void run()
{
        detectEdges();
}
….
}
```

Create Thread

Start hread

Running code

## Canceling a Thread

The myThread.stop() method is deprecated : A typical example of a simple variable that is written to and read from different threads is a "stop request" flag allowing one thread to signal to another to finish:

```
public class StoppableTask extends Thread {
  private volatile boolean pleaseStop;

  public void run() {
    while (!pleaseStop) {
      // do some stuff...
    }
  }

  public void tellMeToStop() {
    pleaseStop = true;
  }
}
```

The other thread which is using the same variable, can see the value

Stop

# Développement avancé : Thread

## AsyncTask

http://developer.android.com/reference/android/os/AsyncTask.html
http://www.vogella.com/tutorials/AndroidBackgroundProcessing/article.html

```
public class
AsyncTaskActivity extends Activity implements
OnClickListener {

    Button btn;

    @Override
    public void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btn = (Button) findViewById(R.id.button1);
            btn.setOnClickListener(this);
    }

    public void onClick(View view) {
        // detect the view that was "clicked"
        switch (view.getId()) {
        case R.id.button1:
            new LongOperation().execute("");
            break;
        }
    }
```

```
private class LongOperation extends AsyncTask<String, Void, String> {

        @Override
        protected String doInBackground(String... params) {
            for (int i = 0; i < 5; i++) {
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    Thread.interrupted();
                }
            }
            return "Executed";
        }

        @Override
        protected void onPostExecute(String result) {
            TextView txt = (TextView) findViewById(R.id.output);
            txt.setText("Executed"); // txt.setText(result);
            // might want to change "executed" for the returned string
passed
            // into onPostExecute() but that is upto you
        }

        @Override
        protected void onPreExecute() {}

        @Override
        protected void onProgressUpdate(Void... values) {}
    }
}
```

http://developer.android.com/guide/topics/wireless/bluetooth.html

# Développement avancé : Bluetooth

Bluetooth from the IEEE standard 802.15.1 is an open, wireless protocol for exchanging data between devices over short distances.A common example is from a phone to a headset, but other applications can include proximity tracking.To communicate between devices using Bluetooth, four steps need to be performed:

1. Turn on Bluetooth for the device.
2. Find paired or available devices in a valid range.
3. Connect to devices.
4. Transfer data between devices.

http://developer.android.com/guide/topics/wireless/bluetooth.html

# Développement avancé : Bluetooth

All the Bluetooth API functionality resides in the android.bluetooth package.There are five main classes that provide the features:

| |
|---|
| **BluetoothAdapter** — Represents the Bluetooth radio interface that is used to discover devices and instantiate Bluetooth connections |
| **BluetoothClass** — Describes the general characteristics of the Bluetooth device |
| **BluetoothDevice** — Represents a remote Bluetooth device |
| **BluetoothSocket** — Represents the socket or connection point for data exchange with another Bluetooth device |
| **BluetoothServerSocket** — Represents an open socket listening for incoming requests |

# Développement avancé : Bluetooth

To use the Bluetooth Service, the application needs to have BLUETOOTH permission to receive and transmit and BLUETOOTH_ADMIN permission to manipulate Bluetooth settings or initiate device discovery.These require the following lines in the AndroidManifest XML file.

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

## Turning on Bluetooth

Bluetooth is initialized using the BluetoothAdapter class.The getDefaultAdapter() method retrieves information about the Bluetooth radio interface. If null is returned, it means the device does not support Bluetooth:

```
BluetoothAdapter myBluetooth = BluetoothAdapter.getDefaultAdapter();
```

Activate Bluetooth using this BluetoothAdapter instance to query the status. If not enabled, the Android built-in activity ACTION_REQUEST_ENABLE can be used to ask the user to start Bluetooth:

```
if(!myBluetooth.isEnabled())
{
  Intent enableIntent = new Intent(BluetoothAdapter .ACTION_REQUEST_ENABLE);
  startActivity(enableIntent);
}
```

Turning on Bluetooth

**Finding Devices**
find remote Bluetooth devices

**Querying paired devices**
the desired device is already known

For security reasons the pairing process is typically triggered automatically the first time a device receives a connection request from a device with which it is not yet paired

**Discovering devices**
The discovery process usually involves an inquiry scan of about 12 seconds

**Enabling discoverability**

# Gestion de la batterie

# Développement avancé : Monitoring the Battery

- Mobile devices operate with the use of the battery
- For instance, a monitoring application can reduce the monitoring frequency when the battery is low and can increase it if the handset is powered by an external power source
- Then the application needs to register for a particular BroadcastIntent.
- Next, the application needs to provide an implementation of the BroadcastReceiver

http://developer.android.com/reference/android/os/BatteryManager.html

# Développement avancé : Monitoring the Battery

The BatteryManager class contains strings and constants used for values in the ACTION_BATTERY_CHANGED Intent :

```
int BATTERY_HEALTH_COLD
int BATTERY_HEALTH_DEAD
int BATTERY_HEALTH_GOOD
int BATTERY_HEALTH_OVERHEAT
int BATTERY_HEALTH_OVER_VOLTAGE
int BATTERY_HEALTH_UNKNOWN
int BATTERY_HEALTH_UNSPECIFIED_FAILURE
int BATTERY_PLUGGED_AC   Power source is an AC charger.
int BATTERY_PLUGGED_USB  Power source is a USB port.
..
```

http://developer.android.com/training/monitoring-device-state/battery-monitoring.html

# Développement avancé : Monitoring the Battery

```
public class MainActivity extends Activity {

private BroadcastReceiver mBatInfoReceiver = new BroadcastReceiver(){

@Override
public void onReceive(Context arg0, Intent intent) {
int level = intent.getIntExtra("level", 0);
Log.v(TAG, String.valueOf(level) + "%");
}
};
@Override
public void onCreate(Bundle savedInstanceState) {
this.registerReceiver(this.mBatInfoReceiver,
new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
}

@Override
protected void onDestroy()
{
…
this.unregisterReceiver(this.mBatInfoReceiver);
…
}
```

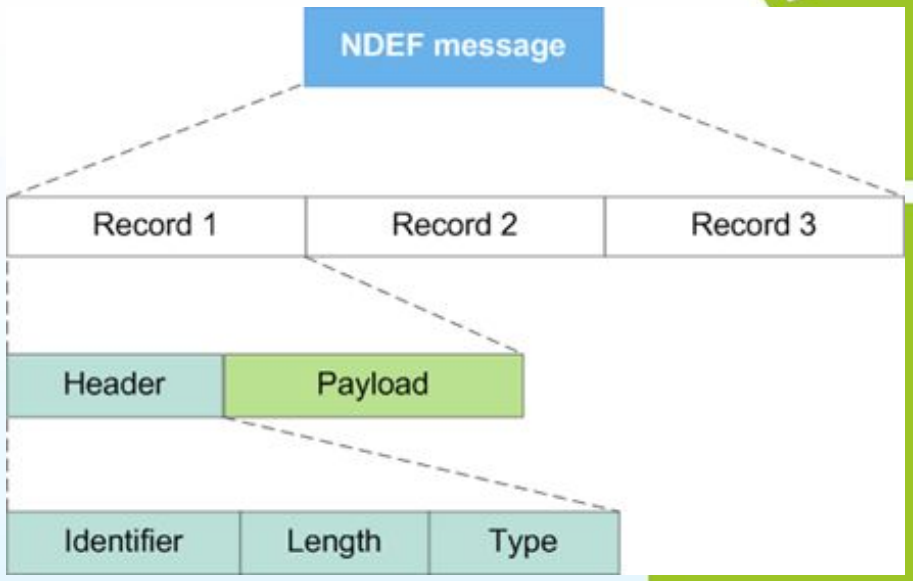
# Développement avancé : Monitoring the Battery

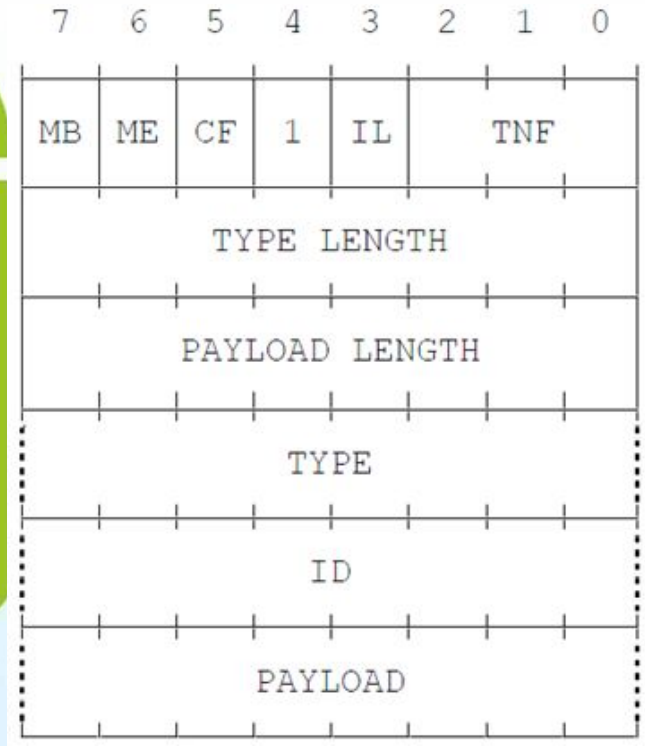
**TP**

ANDROID_BATTERY_MANAGER

http://developer.android.com/guide/topics/connectivity/nfc/nfc.html

**NFC : NFC Data Exchange Format**




NDEF data is encapsulated inside a message (NdefMessage) that contains one or more records (NdefRecord).

http://developer.android.com/guide/topics/connectivity/nfc/index.html

# Développement avancé : NFC

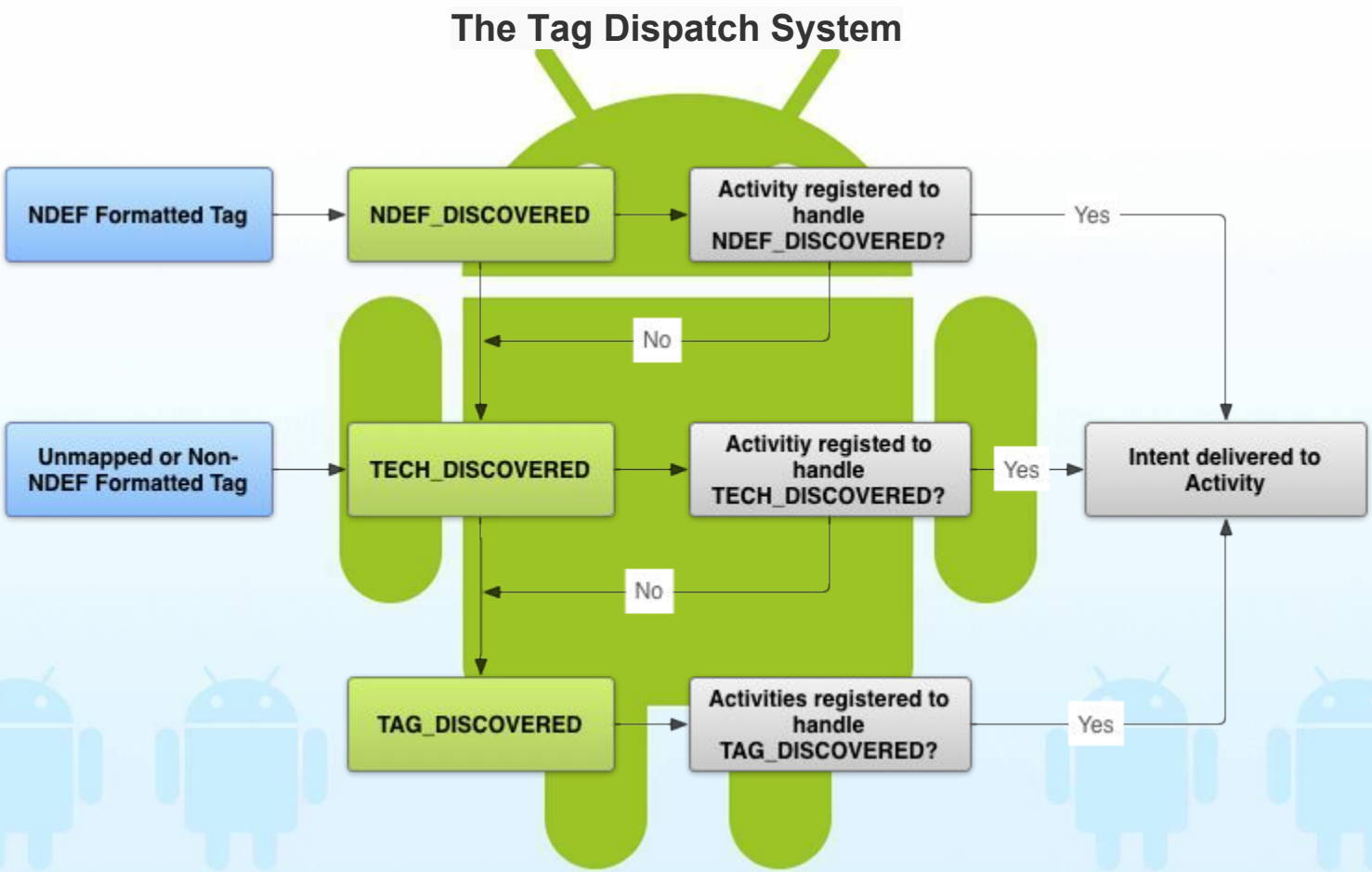
Android fait un support des technologies suivantes

| | |
|---|---|
| TagTechnology | The interface that all tag technology classes must implement. |
| NfcA | Provides access to NFC-A (ISO 14443-3A) properties and I/O operations. |
| NfcB | Provides access to NFC-B (ISO 14443-3B) properties and I/O operations. |
| NfcF | Provides access to NFC-F (JIS 6319-4) properties and I/O operations. |
| NfcV | Provides access to NFC-V (ISO 15693) properties and I/O operations. |
| IsoDep | Provides access to ISO-DEP (ISO 14443-4) properties and I/O operations. |
| Ndef | Provides access to NDEF data and operations on NFC tags that have been formatted as NDEF. |
| NdefFormatable | Provides a format operations for tags that may be NDEF formattable. |

| Class | Description |
|---|---|
| MifareClassic | Provides access to MIFARE Classic properties and I/O operations, if this Android device supports MIFARE. |
| MifareUltralight | Provides access to MIFARE Ultralight properties and I/O operations, if this Android device supports MIFARE. |

optionnel

# Développement avancé : NFC




http://developer.android.com/guide/topics/connectivity/nfc/index.html

## Filtering for NFC Intents

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" />
</intent-filter>
```

```
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED" />
    <category android:name="android.intent.category.DEFAULT" />
   <data android:scheme="http"
              android:host="developer.android.com"
              android:pathPrefix="/index.html" />
</intent-filter>
```

http://developer.android.com/guide/topics/connectivity/nfc/index.html

# Développement avancé : NFC

**ACTION_TECH_DISCOVERED**

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2" >
    <tech-list>
        <tech>android.nfc.tech.IsoDep </tech>
        <tech>android.nfc.tech.NfcA </tech>
        <tech>android.nfc.tech.NfcB </tech>
        <tech>android.nfc.tech.NfcF </tech>
        <tech>android.nfc.tech.NfcV </tech>
        <tech>android.nfc.tech.Ndef </tech>
        <tech>android.nfc.tech.NdefFormatable </tech>
        <tech>android.nfc.tech.MifareClassic </tech>
        <tech>android.nfc.tech.MifareUltralight </tech>
    </tech-list>
</resources>
```

http://developer.android.com/guide/topics/connectivity/nfc/index.html

## Beaming NDEF Messages to Other Devices

Android Beam allows simple peer-to-peer data exchange between two Android-powered devices. The application that wants to beam data to another device must be in the foreground and the device receiving the data must not be locked. When the beaming device comes in close enough contact with a receiving device, the beaming device displays the "Touch to Beam" UI. The user can then choose whether or not to beam the message to the receiving device.

voir example dans les samples du JDK

http://developer.android.com/guide/topics/connectivity/nfc/index.html