Android Hardware

Rémi Forax

Communication Réseau

Android implante java.net/java.nio.channels

donc réutilisation du cours Archi Appli Reseau !

Nouvelle API (plus bas niveau) android.net

- Information sur la connectivité réseau
- Gestionnaire WIFI
- Téléphonie (SMS, MMS, appel cellulaire)
- Client HTTP

et aussi VPN, SIP, RTP, Bluetooth, NFC, ...

ConnectivityManager

Récupération du ConnectivityManager :
Context.getSystemService(Context.CONNECTIVITY_SERVICE)

Permissions nécessaires :
android.permission.[ACCESS_NETWORK_STATE|ACCESS_WIFI_STATE]

Informations réseau :
NetworkInfo[] getAllNetworkInfo()

Réseau de la route par défaut :
NetworkInfo getActiveNetworkInfo()

NetworkInfo

- indique le type de réseau (mobile, Wi-Fi, Wimax, Ethernet, Bluetooth), son état (demande DHCP, connecté, roaming...)
- Le terminal est "en ligne" si getActiveNetworkInfo().isConnected() est vrai

Pour être informé des changements : réception d'Intent
ConnectionManager.CONNECTIVITY_ACTION
par un BroadcastReceiver (arguments EXTRA_[IS_FAILOVER| NETWORK_INFO| NETWORK_TYPE|NO_CONNECTIVITY| OTHER_NETWORK_INFO|REASON]

Gestion du WIFI

Récupération du WifiManager:
Context.getSystemService(Context.WIFI_SERVICE)

Verrous Wi-Fi :

- verrou pour recevoir les datagrammes multicast
  - createMulticastLock(String tag)
- verrou pour ne pas mettre en sommeil la pile Wi-Fi
  - WifiLock createWifiLock(String tag)
  - wifiLock.acquire() / wifiLock.release()
- informations sur la connexion courante (BSSID, SSID, RSSI, adresse IP, bande passante...)
  - WifiInfo getConnectionInfo()

Scan connections WIFI

```
WifiManager manager =
    (WifiManager)getSystemService(Context.WIFI_SERVICE);

BroadcastReceiver receiver = new BroadcastReceiver() {
  @Override
  public void onReceive(Context context, Intent intent) {
    unregisterReceiver(this); // sur l'activity

    List<ScanResult> scanResult  = manager.getScanResults();
    ...
  }
};

registerReceiver(receiver, new IntentFilter(
    WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
manager.startScan();
```

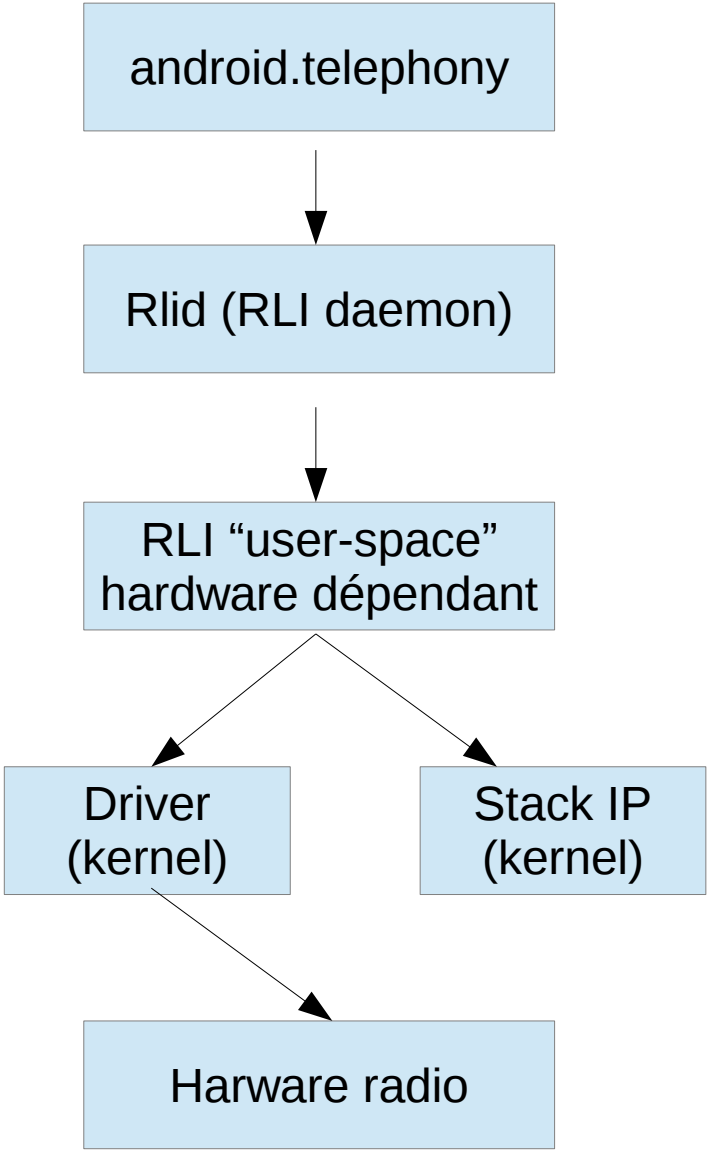
WIFI en mode ad-hoc

- Protocole facilitant l'échange point à point entre deux appareils
- Annonce et découverte de services avec Bonjour et UPnP
- Utilisation de WifiP2pManager
  - Initialisation
  - Enregistrement de services
  - Découverte de pairs et services
  - Connexion avec des pairs

exemples dans samples/WiFiDirectDemo

Téléphonie Cellulaire

- API android.telephony
- API read-only
  - Accès état communication
  - Changement d'état
- On ne peut pas accéder à la couche basse du Radio Layer Interface



Numéroter un appel

- On demande les permissions dans le manifeste
  <uses-permission android:name="android.permission.CALL_PHONE" />
  <uses-permission android:name="android.permission.READ_PHONE_STATE" />
- ou Numérotation indirecte sans permission avec Intent.ACTION_DIAL,
  on utilise le numéroteur installé en lui envoyant un Intent :

```
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:+33160957733"));    // tel: is a protocol specific URI !
startActivity(callIntent);
```

- On enregistre un PhoneCallListener pour être informé de l'état de l'appel

```
// create the listener
PhoneCallListener phoneListener = new PhoneCallListener() {
  private boolean calling = false ;
  @Override public void onCallStateChanged(int state, String incomingNumber) {
    switch (state) {
      case TelephonyManager.CALL_STATE_RINGING: // Phone is ringing
        break ;
      case TelephonyManager.CALL_STATE_OFFHOOK : // Phone call is starting
        break ;
      case TelephonyManager.CALL_STATE_IDLE : // no action
        // Either the phone has not dialed yet or the call is finished
        break ;
   } } };
// register the listener
TelephonyManager telephonyManager = (TelephonyManager)
this.getSystemService(Context.TELEPHONY_SERVICE);
telephonyManager.listen(phoneListener, PhoneStateListener.LISTEN_CALL_STATE);
```

TelephonyManager

- Etat de la connexion data (DATA_[DISCONNECTED|CONNECTING|CONNECTED|SUSPENDED])
  int getDataState()
- Activité de la connexion data (DATA_ACTIVITY_[NONE|IN|OUT|INOUT|DORMANT])
  int getDataActivity()
- retourne le numéro IMEI (requiert la permission READ_PHONE_STATE)
  String getDeviceId()
- indique si l'on est connecté sur un réseau en itinérance
  boolean isNetworkRoaming()
- information sur les cellules voisines (CID, RSSI)
  List<NeighboringCellInfo> getNeighboringCellInfo()
- retourne le code numérique MCC+MNC de l'opérateur
  String getNeworkOperator()
- retourne le type de réseau utilisé pour les transmissions data (GPRS, EDGE, UTMS, LTE...)
  int getNetworkType()

PhoneStateListener

Enregistrement du listener
TelephonyManager.listen(PhoneStateListener l, int events)

Evénements supportés :

- LISTEN_NONE: aucun évènement
- LISTEN_CALL_FORWARDING_INDICATOR changements de l'état de redirection des appels
- LISTEN_CALL_STATE: état de l'appel
- LISTEN_CELL_INFO: information de cellule
- LISTEN_CELL_LOCATION: changement concernant la localisation obtenue d'après les cellules
- LISTEN_DATA_ACTIVITY: activité data
- LISTEN_DATA_CONNECTIVITY_STATE: état data
- LISTEN_SERVICE_STATE: état de couverture (réseau dispnible, urgences seules...)
- LISTEN_SIGNAL_STRENGTHS: changement de puissance du réseau reçu

Appels entrants

- Reception d l'évènement broadcast TelephonyManager.ACTION_READ_PHONE_STATE_CHANGED (permission READ_PHONE_STATE)
  - EXTRA_STATE: état de l'appel
    - EXTRA_STATE_RINGING nous intéresse
  - EXTRA_INCOMING_NUMBER : numéro de l'appelant
- Réponse en envoyant l'intent ACTION_ANSWER

Envoi de SMS

Obtenir le SMSManager

SMSManager.getDefault()
permission android.permission.SEND_SMS

Envoie de messages:

- sendTextMessage(String dest, String center, String message, PendingIntent sendIntent, PendingIntent deliveryIntent)
- sendDataMessage(dest, center, short destinationPort, byte[] data, sendIntent, deliveryIntent)
- sendMultipartTextMessage(dest, center, ArrayList<String> parts, ArrayList<PrendingIntent> sendIntents, ArrayList<PendingIntent> deliveryIntent)
  - divideMessage() permet de découper en plusieurs message

Envoie simple

```
public class SMSActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    final EditText phoneNumber = (EditText)findViewById(R.id.phoneNumber);
    final EditText message = (EditText)findViewById(R.id.message);
    Button send = (Button)findViewById(R.id.send);
    send.setOnClickListener(new Button.OnClickListener(){
      @Override
      public void onClick(View view) {
        String smsNumber = phoneNumber.getText().toString();
        String smsText = message.getText().toString();
        SmsManager smsManager = SmsManager.getDefault();
        smsManager.sendTextMessage(smsNumber, null, smsText, null, null);
      }
    });
  }
}
```

Envoie en plusieurs parties + reception confirmation

```
private static final String SENT_ACTION = SMSActivity.class.getName() + ".sent";
private static final String DELIVERY_ACTION = SMSActivity.class.getName() + ".delivery"
...
SmsManager smsManager = SmsManager.getDefault();
ArrayList<String> split = smsManager.divideMessage(message);

BroadcastReceiver br = new BroadcastReceiver() { /* cf page suivante */ };
IntentFilter filter = new IntentFilter();
filter.addAction(SENT_ACTION);
filter.addAction(DELIVERY_ACTION);
registerReceiver(br, filter);

ArrayList<PendingIntent> sentIntents = new ArrayList<PendingIntent>();
ArrayList<PendingIntent> deliveryIntents = new ArrayList<PendingIntent>();
for (int i = 0; i < split.size(); i++) {
  sentIntents.add(PendingIntent.getBroadcast(this, 2*i,
      new Intent(SENT_ACTION).putExtra("chunkID", i), 0));
  deliveryIntents.add(PendingIntent.getBroadcast(this, 2*i+1,
      new Intent(DELIVERY_ACTION).putExtra("chunkID", i), 0));
}

smsManager.sendMultipartTextMessage(recipient, null,
    split, sentIntents, deliveryIntents);
```

Reception confirmation / erreurs

```
new BroadcastReceiver() {
  @Override
  public void onReceive(Context context, Intent intent) {
    int chunkID = intent.getIntExtra("chunkID", -1);
    String action = intent.getAction();
    if (action.equals(SENT_ACTION)) {
      switch (getResultCode()) {
        case Activity.RESULT_OK:
          /* cool */ break;
        case SmsManager.RESULT_ERROR_RADIO_OFF:
          /* not GSM, 3G, etc signal */ break;
        case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
         /* other error: intent.getIntExtra("errorCode", -1) */ break;
        default:
          // oops ??
      }
    } else {  // action == DELIVERY_ACTION
      message = SmsMessage.createFromPdu(intent.getByteArrayExtra("pdu"))
          .getDisplayMessageBody();
    }
```

Reception de SMS

- Permission: android.permission.RECEIVE_SMS
- BroadcastReceiver que l'on enregistre sur l'Intent android.provider.Telephony.SMS_RECEIVED
  - L'extras contient des données PDUs (format binaire des SMS)
    - getExtras().get("pdus")
  - Conversion en SmsMessage
    - SmsMessage.createFromPdu(pdu)
  - Ne pas propager le broadcast
    - broadcastReceiver.abordBroadcast()

Exemple

```
new BroadcastReceiver() {
  @Override
  public void onReceive(Context context, Intent intent) {
    byte[][] pdus = (byte[][])intent.getExtras().get("pdus");
    for (byte[] pdu: pdus) {
      SmsMessage message =
          SmsMessage.createFromPdu(pdu);
      String text = message.getDisplayMessageBody();
      if (checkSpam(text)) {
        abortBroadcast();
      } } };
```

```
<receiver android:name="fr.umlv.android.SMSActivity">
  <intent-filter android:priority="1000">
    <action android:name="android.provider.Telephony.SMS_RECEIVED" />
  </intent-filter>
</receiver>
```

Fichier AndroidManifest.xml

Clients HTTP

Deux APIs

Apache HTTP client

- Grosse API
- pas compatible entre les versions d'android ??
  - pas évolué dans les versions récente >= API 9

java.net.URL/HttpURLConnection

- Même API que celle de Java
  - Assez simple d'utilisation
- Seule maintenue à partir de API >= 9
- Bugs dans les version < 9

java.net.HTTPURLConnection

Client HTTP simple instantiable depuis une URL :
URL url = new URL(address) ;
HttpURLConnection conn = (HttpURLConnection)url.openConnection() ;
InputStream is = conn.getInputStream() ;

Client HTTP bloquant : à employer dans une nouvelle thread (AsyncTask)

Timeout en lecture
setReadTimeout(int millis) ; lève SocketTimeoutException

Pour envoyer un fichier
conn.setDoOutput(true) ;
conn.setChunkedStreamingMode(0) ;
OutputStream out = conn.getOutputStream();

Ne pas oublier de fermer la connexion:
conn.disconnect()

Gestion automatique d'un pool de connexions ouvertes (réutilisation de connexions)
désactivation avec
System.setProperty("http.keepAlive", "false")

android.net.http.HttpResponseCache fournit un cache depuis ICS

En-têtes requête/réponse

Requête

- Choix de la méthode (GET, POST, PUT, OPTION...)
  setRequestMethod(String method)
- addRequestProperty(String field, String value)

Réponse

- Champs de la réponse
  Map<String, List<String>> getHeaderFields()
- Date de dernière modification
  long getLastModified() (ms depuis epoch)
- Date de la réponse
  Date getDate()

Gestion des cookies

On installe un CookieHandler globale à l'application

```
CookieManager cm = new CookieManager();
CookieHandler.setDefault(cm);
```

Les cookies sont stockées en mémoire par défaut, pour un stockage persistent, il faut founir un CookieStore au CookieManager

Ajouter un cookie

- Manuellement

```
HttpCookie cookie = new HttpCookie("key", "value");
cookie.setDomain("univ-mlv.fr");
cm.getCookieStore().add(
    new URI("http://www.univ-mlv.fr"), cookie);
```

- Automatiquement par HttpURLConnection

Autres APIs de communication

Bluetooth (android.bluetooth) :

- permet de rendre découvrable, de découvrir d'autres périphériques, de les associer
- propose des sockets client et serveur pour la communication en flux
- supporte les profiles headset et A2DP pour la communication audio

NFC (android.nfc) :

- propose de scanner des tags NFC et de lancer l'activité la plus appropriée pour les traiter
- permet l'échange de petits volumes de données à faible distance entre téléphones

USB (android.usb) :

- permet d'échanger des flux de données en tant qu'hôte ou accessoire

Capteurs

- Localisation
- Vibreur
- Senseurs
- Caméra, MediaRecorder
- MediaPlayer/AudioManager

Localisation: LocationManager

- Obtenir le LocationManager
  - getSystemService(Context.LOCATION_SERVICE)
- Permissions
  - android.permission.ACCESS_COARSE_LOCATION (borne céllulaire ou wifi)
  - android.permission.ACCESS_FINE_LOCATION (GPS)
- La localisation est en faite par des providers
  - List<String> getAllProviders()
    LocationProvider getProvider(String name)
  - providers par défaut
    - GPS_PROVIDER, NETWORK_PROVIDER, PASSIVE_PROVIDER

Obtenir des localisations

Dernière localisation connue

locationManager.getLastKnownLocation(String provider)

- Retourne null si le provider est pas disponible
- sinon retourn un objet Location
  - getLatitude/getLongitude en degrés
  - getAltitude() en metre
  - getTime() en ms depuis epoch

Mise à jour périodique

Demande de mise à jour

locationManager.requestLocationUpdates(String provider, long minTimeMillis, float minDistance, LocationListener listener)

- définie une Période temporelle minTimeMillis et spatiale minDistance
- LocationListener doit implanter la méthode onLocationChanged(Location) appelée à chaque mise à jour
- Desenregistrement du listener locationManager.removeUpdates(LocationListener)

Alerte de proximité

- Ajouter une alerte de proximité addProximityAlert(double lat, double lon, float radius, long expirationInMillis, PendingIntent intent)
  - L'intent est envoyé lorsque l'on passe dans la zone spécifiée
  - L'alerte expire après expirationInMillis
    - Infinie si -1
  - Suppression de l'alerte removeProximityAlert(PrendingIntent intent)

Service qui log la localisation

```
public class LocationLogger extends Service {
  private LocationManager locationManager;
  private LocationListener locationListener;
  @Override public IBinder onBind(Intent intent) { return null; }
  @Override public void onCreate() {
    locationManager = (LocationManager)getSystemService(LOCATION_SERVICE);
    locationListener = new LocationListener() {
      @Override public void onStatusChanged(String provider, int status, Bundle extras) {
        log(String.format("Change of status of provider %s: %d", provider, status));
      }
      @Override public void onProviderEnabled(String provider) { }
      @Override public void onProviderDisabled(String provider) { }
      @Override public void onLocationChanged(Location location) {
        log(String.format("latitude=%f, longitude=%f, altitude=%f",
          location.getLatitude(), location.getLongitude(), location.getAltitude()));
      }
    };
  }
  @Override public int onStartCommand(Intent intent, int flags, int startId) {
     locationManager.requestLocationUpdates(
       intent.getStringExtra("provider"),      // must be set in the Android Manifest
       intent.getLongExtra("minTime", 10000),
       intent.getFloatExtra("minDistance", 100.0f),
       locationListener);
     return Service.START_REDELIVER_INTENT; // Restart the service with the intent if it is destroyed
  }
  @Override public void onDestroy() {
    locationManager.removeUpdates(locationListener);
    locationManager = locationListener = null;
  }
```

Vibreur

Récupérer le service de vibration
Vibrator v = (Vibrator)getSystemService(Context.VIBRATOR_SERVICE)

- Permission android.permission.VIBRATE

méthodes utiles:

- y-a-t-il un vibreur disponible ?
  boolean hasVibrator()
- Vibre pendant le durée indiqué
  vibrate(int durationInMillis)

vibre selon la séquence indiquée
vibrate(int[] array, int index)

- Array[0] indique un temps de non-vibration, array[1] un temps de vibration, etc...
  index spécifie un indice pour commencer la répétition

annule une vibration demandée
cancel():

Senseurs

- Obtenir le SensorManager
  Context.getSystemService(Context.SENSOR_SERVICE)
- Liste des senseurs :
  SensorManager.getSensorList(int typeOfSensor)
- Types de senseurs actuellement supportés :
  - Accéléromètre (gyroscope)
  - Thermomètre, hygromètre, baromètre
  - Magnétomètre
  - Senseur de proximité
  - Luxmètre
- Certains senseurs sont accessibles de plusieurs manières (données brutes ou données analysées) :
  - Accéléromètre : Sensor.TYPE_ACCELEROMETER, Sensor.TYPE_GRAVITY, Sensor.TYPE_LINEAR_ACCELERATION

Utilisation

On enregistre un SensorEventListener (par ex. dans onResume)

sensorManager.registerListener(listener, sensor, delay)

Un SensorEventListener à deux méthodes à implanter :

- appelée lorsque la précision change
  void onAccuracyChanged(Sensor s, int accuracy)
  SENSOR_STATUS_ACCURACY_[UNRELIABLE|LOW|MEDIUM, HIGH]
- Appelé lors de l'aquisition de donnée
  void onSensorChanged(SensorEvent event)
- event contient les champs accuracy, sensor,; timestamp et values (tableau de float décrivant les données)

Lorsque l'on ne souhaite plus recevoir des événements des senseurs (par ex. Dans onPause()), on enlève le listener :

sensorManager.unregisterListener(listener, sensor) ;

Exemple

- public class SensorDisplayer extends Activity {
  private SensorManager sensorManager;
  private List<Sensor> sensors;
  private SensorEventListener sensorListener;
- @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
    sensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
    this.sensorListener = new SensorEventListener() {
      @Override public void onSensorChanged(SensorEvent event) {
        // update sensor value list
      }
      @Override public void onAccuracyChanged(Sensor sensor, int accuracy) {}
    };
    ...
    setContentView(...);
  }
- @Override public void onResume() {
  super.onResume();
  for (Sensor s: sensors)
    sensorManager.registerListener(sensorListener, s, SensorManager.SENSOR_DELAY_UI);
}
- @Override public void onPause() {
  super.onPause();
  for (Sensor s: sensors)
    sensorManager.unregisterListener(sensorListener, s);
}

Exemple (suite)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
  sensors = sensorManager.getSensorList(Sensor.TYPE_ALL);
  final List<float[]> sensorValues = new ArrayList<float[]>();
  for (Sensor sensor: sensors) sensorValues.add(null);

  final ArrayAdapter<float[]> arrayAdapter = new ArrayAdapter<float[]>(this, R.layout.simpletextview, sensorValues) {
    @Override public View getView(int position, View convertView, ViewGroup parent) {
      TextView tv = (TextView)convertView;
      if (tv == null) tv = new TextView(SensorDisplayer.this);
      float[] values = getItem(position);
      String message = sensors.get(position).getName() + '\n' +
          ((values == null)?"no data available":Arrays.toString(values));
      tv.setText(message);
      return tv;
    }
  };
  this.sensorListener = new SensorEventListener() {
    @Override public void onSensorChanged(SensorEvent event) {
      sensorValues.set(sensors.indexOf(event.sensor), Arrays.copyOf(event.values, event.values.length));
      arrayAdapter.notifyDataSetChanged();
    }
    @Override public void onAccuracyChanged(Sensor sensor, int accuracy) {}
  };

  ListView sensorListView = new ListView(this);
  sensorListView.setAdapter(arrayAdapter);
  setContentView(sensorListView);
}
```

Caméra

Deux façons accéder à la Caméra

- Soit on contrôle la caméra directment
  - Contrôle plus fin mais plein de paramètre
- Soit on appel l'application caméra avec un Intent
  - Utilise l'application par défaut

API Caméra

- Dans le manifeste

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

- Ouvrir une Camera
  Camera.open(int numero_camera)
  - Le numero de la camera entre 0 et Camera.getNumberOfCamera() -1
- Paramètres
  getParameters()/setParameters(Camera.Parameters)
- Prendre un photo
  Camera.takePicture(callbacks ...)
- Fermer la caméa
  camera.release()

Prévisualisation de la caméra

Où envoyer les données de prévisualisation ?

- sur une texture OpenGL :
  setPreviewTexture(SurfaceTexture st)
- sur un SurfaceHolder (typiquement obtenu avec SurfaceView.getHolder()
  setPreviewDisplay(SurfaceHolder h)
- sur une méthode callback :
  setPreviewCallback(Camera.PreviewCallback cb)
  - setPreviewFormat(int)
    définit le format binaire des previews (par défaut NV21)
  - previewCallback.onPreviewFrame(byte[] data, Camera c) doit être implantée

Contrôle de la prévisualisation

- pour démarrer
  startPreview()
- pour arrêter
  stopPreview()

Zoom et Capture

Contrôle du zoom

- zoom maximal (grand angle=0)
  getMaxZoom()
- fixation du zoom
  setZoom(int value)
- zoom progressif
  startSmoothZoom(int value)/stopSmoothZoom()
  (possibilité d'utiliser un listener)

Capture
takePicture(Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback postview, Camera.PictureCallback jpeg)

- Jouer un son lors de la prise de photo
  shutter.onShutter()
- Chaque PictureCallback est optionnel (peut être null) selon les données souhaitées, brutes, post-traitées ou compressées en JPEG.
  La méthode onPictureTaken(byte[] data, Camera camera) doit être implantée.

Appel Application caméra

- ```
  Intent intent = new Intent((video)?MediaStore.ACTION_VIDEO_CAPTURE:MediaStore.ACTION_IMAGE_CAPTURE);
  // We can also use MediaStore.ACTION_IMAGE_CAPTURE_SECURE to capture an image in lock mode
  // To capture a video, we use MediaStore.ACTION_VIDEO_CAPTURE
  File pictDir = new File(Environment.getExternalStoragePublicDirectory(
      Environment.DIRECTORY_PICTURES), this.getClass().getSimpleName());
  String timestamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
  File dest = new File(pictDir, timestamp); // Destination file
  intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(dest)); // Specify the destination file

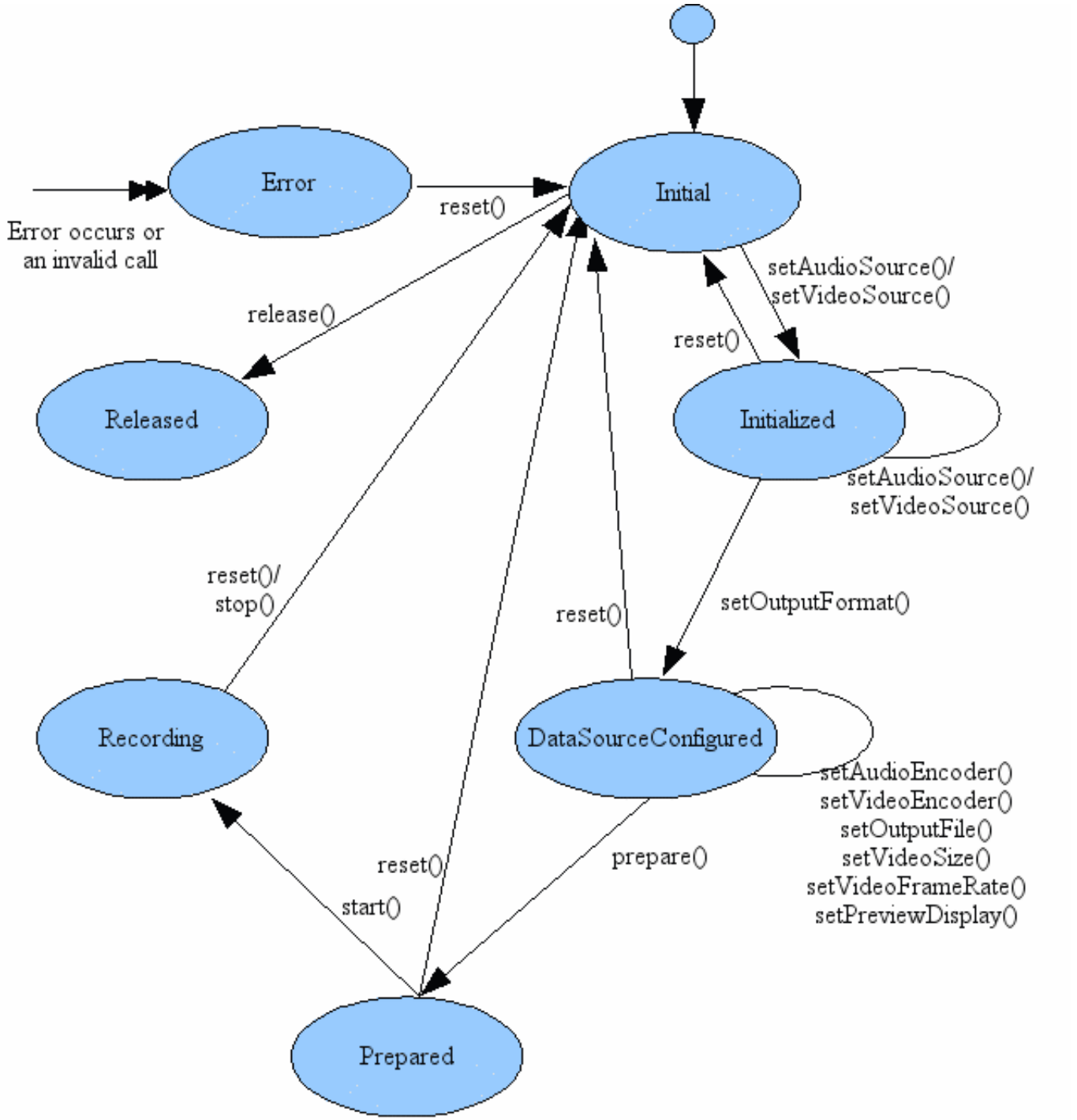
  // Now, we start the picture capture activity
  startActivityForResult(intent, (video)?VIDEO_CAPTURE_REQUEST_CODE:IMAGE_CAPTURE_REQUEST_CODE);
  ...
  @Override
  public void onActivityResult(int requestCode, int resultCode, Intent data) {
    switch (requestCode){
      case IMAGE_CAPTURE_REQUEST_CODE:
        switch (resultCode) {
          case RESULT_OK:
            Toast.makeText(this, "Image saved to " + data.getData(), Toast.LENGTH_LONG).show();
            ((ImageView)findViewById(R.id.imageCaptureView)).setImageURI(data.getData());
            break;
           default: // case RESULT_CANCELED:
            Toast.makeText(this, "Capture of image failed", Toast.LENGTH_SHORT).show();
            break;
          }
          break;
        case VIDEO_CAPTURE_REQUEST_CODE:
          ...
  ```
  ```

MediaRecorder

Requiert android.permission.{RECORD_AUDIO, RECORD_VIDEO}

Pour enregistrer du son/vidéo:

```
MediaRecorder mr = new MediaRecorder();
mr.setAudioSource(MediaRecorder.AudioSource.{DEFAULT, MIC, VOICE_CALL,
VOICE_COMMUNICATION, VOICE_RECOGNITION, VOICE_DOWNLINK,
VOICE_UPLINK});
mr.setVideoSource(MediaRecorder.VideoSource.{DEFAULT, CAMERA});
mr.setOutputFormat(MediaRecorder.OutputFormat.{DEFAULT, THREE_GP, MPEG_4,
AMR_NB, AMR_WB, AAC_ADTS, ...});
mr.setAudioEncoder(MediaRecoarder.AudioEncoder.{DEFAULT, AAC, AMR_NB,
AMR_WB, ...})
mr.setVideoEncoder(MediaRecorder.VideoEncoder.{DEFAULT, H263, H264});
mr.setOutputFile(path);
mr.prepare();
mr.start();
...
mr.stop();
mr.release();
```



MediaRecorder state diagram

MediaPlayer

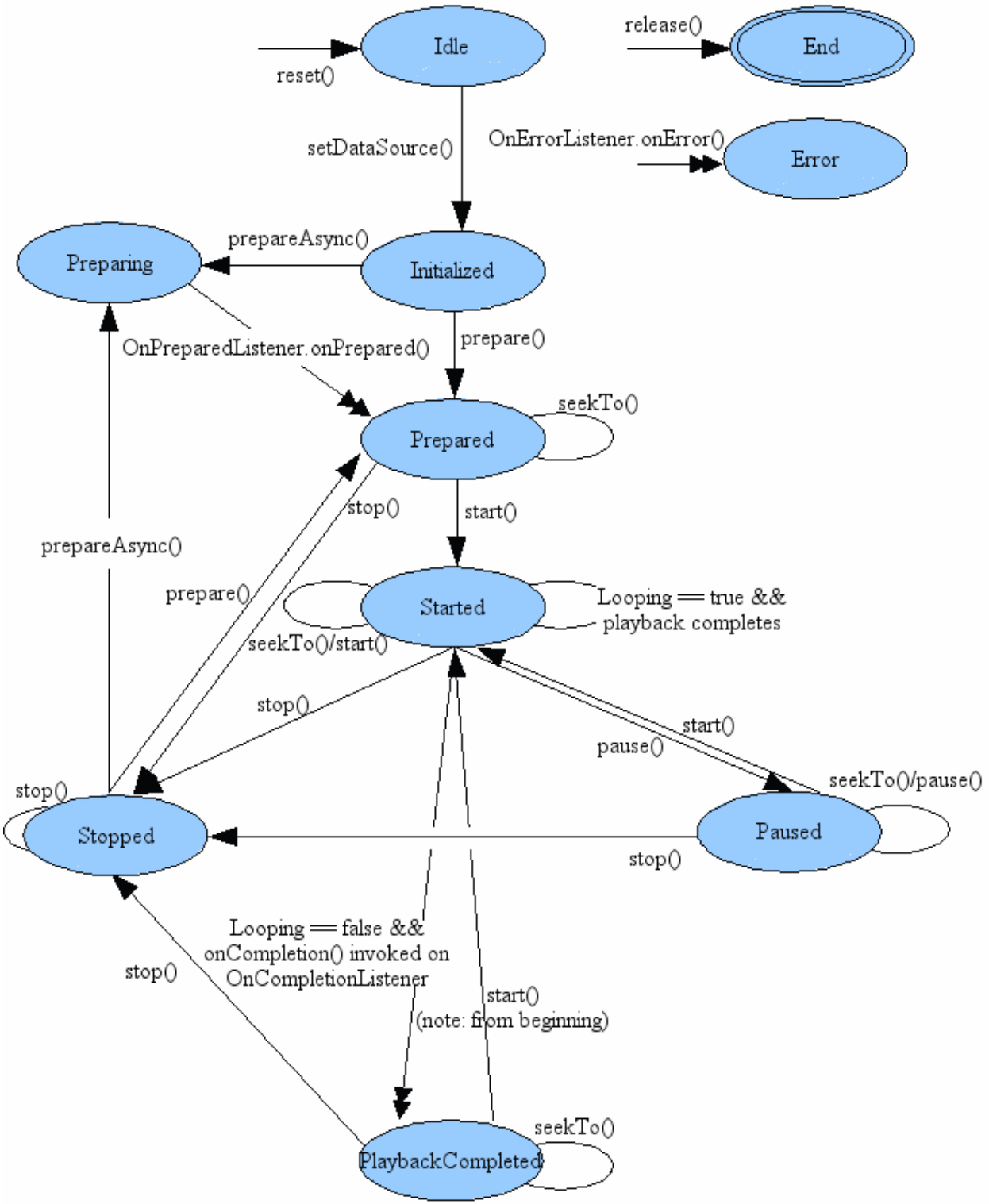
MediaPlayer offre une API pour lire les formats usuels

- vidéo : h264, audio : AAC, FLAC, MP3, midi, Vorbis...
- en local ou avec les protocoles réseau RTP et HTTP

L'utilisation de verrous peut être utile lors de la lecture : wake lock, wifi lock...

Initialisation

```
MediaPlayer mp = new MediaPlayer() ;
mp.setAudioStreamType(AudioManager.STREAM_MUSIC
) ;
mp.setDataSource(myURL) ;
mp.setOnPreparedListener(new OnPreparedListener() {
  public void onPrepared(MediaPlayer mp) { ... }
});
mp.setWakeMode(getApplicationContext(),
    PowerManager.PARTIAL_WAKE_LOCK);
mp.prepareAsync();
...
mp.relase();    // free resources
```


Idle
reset()
release()
End
setDataSource()
OnErrorListener.onError()
Error
Preparing
prepareAsync()
Initialized
OnPreparedListener.onPrepared()
prepare()
seekTo()
Prepared
stop()
start()
prepareAsync()
prepare()
Started
Looping == true &&
playback completes
seekTo()/start()
stop()
start()
pause()
seekTo()/pause()
stop()
Stopped
Paused
stop()
Looping == false &&
onCompletion() invoked on
OnCompletionListener
stop()
start()
(note: from beginning)
seekTo()
PlaybackCompleted

AudioManager

Coopération pour la gestion du focus audio

- Focus audio possédé par un seul composant
- Focus audio transférable à un autre composant :
  - FocusManager.requestAudioFocus(OnAudioFocusChangeListener listener, int streamType, int durationHint)
    - OnAudioFocusChangeListener. onAudioFocusChange(int focusChanger)
      - AUDIOFOCUS_GAIN
      - AUDIOFOCUS_LOSS
      - AUDIOFOCUS_LOSS_TRANSIENT
      - AUDIO_FOCUS_LOSS_TRANSIENT_**CAN_DUCK**

Information du passage sur HP : broadcast de android.media.AUDIO_BECOMING_NOISY