Android SDK

Synopsis

This document describes how to develop an application with the Android *I6netPhone library*.

The *I6netPhone library* is a java and native stack that provides you all needs to manage any kind of voicecalls or videocalls over Android devices connected to a Vxi / RTMP Channel Server. We will describe more in detail, the libI6netPhone (version V1.15) and we will speak about telephony uses or VoIP applications over Android.

This document is entented for users knowing java development and knowing the base of android development.

Description

The *I6netPhone library* package contents is:

Java android library	libi6net_phone.jar	Provides all classes needed to manage video calls in android environment.
Native wrapper library	libi6net_wrapperjava.so	Wrapper between java and native environnement.
Native libray	libi6net_phone.so	Native library which manage the RTMP connection and media streaming.

NOTE:

You'll need to extend some class and use singleton.

Class Reference

The *I6netPhone library* provides classes to create a voicecall or videocall application extending Android basic classes:

I6netDefines	This class gives you somes constantes used for I6netPhone library.
I6netPhoneDefines	This class gives you somes constantes used for I6netPhone library.
Phone	This class is the main class for using Android I6netPhone library.
PhoneActivity	This class is the second main class of Android I6netPhone library.
PhoneConfiguration	This class manages the configuration of Android I6netPhone library.

The first main class to create an android application is the **Activity** class. The main class of *I6netPhone library* is the **PhoneActivity** which extend **Activity** class. You need to extend this class to be able to manage a phone in a user interface application.

The second significant class is the **Phone** class, which is the kernel of the *I6netPhone library*. The API

of Phone class give you all needs to manage a voice or video call, to initialize, start and stop the phone, and so the connection with Vxi / RTMP Channel Server.

The **PhoneActivity** class allows to update the user interface on event from the *I6netPhone library*. All event managed by *I6netPhone library* are describe in our Event Handler.

How to create your first Voicecall or Videocall application

You have to include the three previous libraries in your projet.

The two natives libraries need to be include in the directory **libs/armeabi**, and your android projet must be compiled with the **libI6netPhone.jar** library.

How to use this library

Firstly, you need to create an Android Activity which extends **PhoneActivity** class. To be able to receive kernel and network events you need to override parent's methods (onConnected(), onDisconnected(), onInvite(), onCancelled() ... described below). To create a Phone, you need to instanciate, initialize and start the Phone class.

This class is a singleton, so you need to get this instance to work with it, in this way :

```
Phone.getInstance().initialize (context, appname, PkgNname, MyId, MyPsswd,
trace_level) ;
Phone.getInstance().start();
Phone.getInstance().stop();
```

Calling start() method will make the connection with the Vxi / RTMP Channel Server. This method return an error if the library initialization failed or if no data network was found. But a normal return doesn't means that the Phone is connected, it seams that the phone is connecting.

The library will generate an event onConnected when the connection will be ok. So you need to override this method to make your own GUI treatment.

To prevent network failure, you should launch a timeout of connection by calling :

```
lauchConnectedTimer();
```

The timeout will expire by a event with the method

```
onConnectionTimeout();
```

You need to override this method to make your own GUI treatment.

To be able to display a local and remote video, you need to set in library the FrameLayout in which you want to display the local video (from camera) and the remote video. So you must register it by the follow method in the **onResume()** of your activity which display the video :

```
Phone.getInstance().setVideoLayout(<localVideo>, <remoteVideo>);
```

After calling these methods, your are able to make video call.

For a complete description of all Phone class method, refer to the javadoc
http://docs.i6net.com/libi6net_phone/android/V1-15/index.html

Configuration

The *I6netPhone library* has to be configured before initialized.

You can change the video format size, the bitrate, frame per second, timeout and the RTMP URL. Here is an example to change these values :

```
Phone.getInstance().getConfig().setDisplaySize(mDisplayWidth,
mDisplayHeight) ;

Phone.getInstance().getConfig().setPlayerSize(PhoneConfiguration.DEFAULT_QCI
F_WIDTH, PhoneConfiguration.DEFAULT_QCIF_HEIGTH );

Phone.getInstance().getConfig().setPictureSize(I6netPhoneDefines.QCIF_WIDTH,
I6netPhoneDefines.QCIF_HEIGTH );

Phone.getInstance().getConfig().setFps(fps);

Phone.getInstance().getConfig().setBitRate(bitrate);

Phone.getInstance().getConfig().setTimeoutConnection(timeoutCnx*1000);

Phone.getInstance().getConfig().setTimeoutCall(timeoutCall*1000);

Phone.getInstance().getConfig().setRtmpServer(mSharedPrefs.getString(getStri
ng(R.string.prefKeyRtmpServer), AppliDefines.DEFAULT_RTMP_SERVER ));
```

You can disable audio or video feature in your application by the following commands :

```
Phone.getInstance().getConfig().setVideoUse(true/false);

Phone.getInstance().getConfig().setAudioUse(true/false);

Phone.getInstance().getConfig().setDefMicroUse(true/false );

Phone.getInstance().getConfig().setDefCamUse(true/false);
```

You can dynamically activate or deactivate the camera and the microphone by these commands :

```
Phone.getInstance().activateCamera()

Phone.getInstance().desactivateCamera(true)
```

```
Phone.getInstance().activateMicrophone()

Phone.getInstance().desactivateicrophone()
```

For a complete API description see http://docs.i6net.com/libi6net_phone/android/V1-15/index.html

Event Handler

The event managment from the *I6netPhone library* is managed in **PhoneActivity** class by predefined methods (onHangup(), onInvite() ...). If you want to update your user interface on specific event, you need to override one or severals following methods.

onAbort (int code,java.lang.String cause)	Event to inform that the connection with the server is aborted.
onAccept ()	Event received when outgoing call is accepted by remote end user.
onCancelled ()	Event received when call is refused by server or remote end user.
onHangup ()	Event received when call is hanged up by remote end.
onInvite (java.lang.String from)	Event received when library receives an incoming call.
onConnected ()	Event received when library is connected to the server.
onConnectionTimeout ()	Event received if the connection to the server is too long.
onDisconnected ()	Event received if library is disconnected to the server. The library is always initialized. To reconnect, you should use Phone.getInstance().reconnect().
onResume ()	Parent of onResume() method of your activity. All activity which extend this call must call this super class.
onGetConfig (java.lang.String key)	Event received from server administration console to get a configuration value.
onSetConfig (java.lang.String key, java.lang.String value)	Event received from server administration console to set a value for one configuration parameter.
onFirstRemotePicture ()	Event received when library receives the first video picture.
onEventDTMF (java.lang.String dtmf)	Event received when library receives a DTMF from network. DTMF is in the parameter.
onEventTEXT (java.lang.String text)	Event received when library receives a TEXT command DTMF from network. Command is in the parameter.
onEventREGISTERED (java.lang.String userid)	Event received when library receives a response of the command Phone.getInstance().sendGetUserStatus()to infom that user is registered.
onEventUNREGISTERED (java.lang.String userid)	Event received when library receives a response of the command Phone.getInstance().sendGetUserStatus() to infom that user is not registered.
onEventNetworkBusy ()	Event received on network failure.
onEventNetworkNormalBack ()	Event received when network back to normal after a failure.

Permissions

You need to add somes permissions in your AndroidManifest.xml file in your application. Add these followed permissions :

```
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"
/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"
/>
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
    <uses-permission
android:name="android.permission.RAISED_THREAD_PRIORITY" />
    <uses-feature android:name="android.hardware.camera" />
```