Player SDK for Android 1.1

Programming Guide

UPDATED TO JUNE 2ND 2011

Introduction

RayV Player SDK for Android is a set of native and Java libraries for developing Android players that can stream and display RayV based live TV channels. The SDK supports Android 2.2 or higher. It is recommended to use the SDK Android devices only since the Android emulators are usually too slow for the real time nature of the RayV SDK.

For other operating systems, RayV provides a C/C++ SDK with which provides low level streaming APIs and requires developer to provide their own media rendering code. However since Android does not provide any API for decoding H.264 video, implementing an efficient, hardware accelerated, renderer is not a simple task. Therefore RayV provides an Android specific layer that translates, locally on the device, the RayV proprietary protocol into standard RTSP protocol. This layer provides the following benefits:

- A simplified Java API for the RayV Player SDK.
- Standard RTSP protocol output allows using standard media player with all the standard user interface and features.
- Efficient hardware accelerated media decoding and rendering allowing live channels with high quality.

A RayV specific video view is also part of the SDK, hiding all the RayV specific code and making it even easier to develop RayV based applications. Note that in the future the RayV SDK might switch from RTSP to another protocol or decoding/rendering APIs so its best to use the provided video view instead of using the RTSP stream directly.

This document provide high level guide for using the SDK. More information can be found in the SDK automatically generated documentation.

RayV Player SDK for Android files

The SDK is packaged in a zip file and should be extracted to a folder on the development machine.

The SDK include the following folders and files:

RayV SDK

The SDK is an Android Library Project. Application should include a reference to this project, causing the SDK classes, libraries and resources to be automatically included in the application. If for any reason a library project cannot be used (e.g. application is not built with Eclipse) then the SDK classes, libraries and resources can be copied manually into the application folder. Using the manual method means that the copy should be repeated with every new version of the SDK.

See http://developer.android.com/guide/developing/projects/projects-eclipse.html for more information about library projects.

The SDK contains the following some Android Library Project files and the following folders:

- **libs**: SDK native libraries.
- **src**: SDK Java classes
- **res**: SDK resources.

RayVSDKforAndroidTest

A sample application that demonstrates playing a RayV channel using the RVViewView class. The application is referencing the SDK Android Library Project.

Using the SDK

To use the SDK, unzip the SDK zip file. Then in Eclipse use Import… to import the RayVSDK folder as an Eclipse project. For running the sample application, use Import… to import the RayVSDKforAndroidTest as an Eclipse project. Then simply build and run the test project.

To reference the SDK from your application, just follow the procedure to reference an Android Library Project here:
http://developer.android.com/guide/developing/projects/projects-eclipse.html

Updating Application Manifest

When using the SDK it is important to update AndroidManifest.xml with the following items.

Enabling SDK Permissions

For allowing the SDK to access the network and network state the following lines should be added:

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

Handling Video Orientation Changes

By default activities are destroyed and recreated when the device orientation changes. This will cause the video playback to be restarted. In order to prevent this, add the following attribute to the activity that contains the RVVideoView object.

```
android:configChanges="orientation"
```

Including SDK Activities

In order to include the SDK activities the manifest should include:

```
<activity android:name="com.rayv.androidsdk.SupportLogActivity"/>
```

```
<activity android:name="com.rayv.androidsdk.P2PInfoActivity"
          android:theme="@android:style/Theme.Translucent"/>
<activity android:name="com.rayv.androidsdk.SupportInfoActivity"/>
```

AndroidPlayerCore

AndroidPlayerCore is the object that creates and manages the RayV Player SDK. It is in charge of the following:

- Connecting to the RayV Grid Server, passing the information about the application and SDK and keeping the connection alive.
- Creating, configuring and managing the P2P engine.
- Creating viewing sessions for playing RayV channels.
- Providing internal, support/debugging related, information to the application.

The AndroidPlayerCore is a singleton and only one instance of it can be created. It is created with the following code:

```
private AndroidPlayerCore playerCore;

playerCore = AndroidPlayerCore.initWithDistirbutor(this, "distributor", 2, 0, 0, 0);
```

Each company using RayV has its own distributor ID and should use it to identify its players. The distributor ID and application version are used for statistics and service rules.

It is important that the AndroidPlayerCore is stored in a member of the main application class so it won't be destroyed by the garbage collector or by an activity that is closed.

The application should then start the SDK using the following code:

```
playerCore.start();
```

This will connect to the Grid and then start the P2P engine.

When the application terminates it should stop the SDK using the following code:

```
playerCore.stop();
```

This will disconnect from the Grid and stop the P2P engine.

RVViewView

RVViewView is an object that internally uses RayV RTSP server and Android VideoView to play RayV channels and takes care of the following:

- Create and manage an RTSPServer object.
- Handle session state and display session progress user interface.

The RVVideoView implements a subset of VideoView API so the original VideoView documentation can be used as a reference. Here is the API reference with a description of issues related to VideoView only:

Instance Methods

```
public RVVideoView(Context context)
```

```
public void setVideoURI(Uri uri)
```

The URL should have the format “rayv:channelkey”. For example for channel iphone1demo the URL should be “rayv:iphone1demo”.

```
public void start()
public void pause()
public void stopPlayback()
public void resume()
```

It is important to call stopPlayback() when playback is not required anymore so the RayV SDK will stop the RTSP server and the RayV streaming protocol.

```
public void setMediaController(MediaController controller)
```

For live channels it is recommended to create MediaController with useFastForward parameter set to false so only play/pause button will be displayed.

Application and Activity Life Cycle

The test program provides an example to how application and activity life cycle can be handled correctly when playing RayV channels:

- Call RVVideoView::pause() when activity receives onPause(). This will pause the video (and RayV stream) when another activity covers the video playing activity. When the activity is back on front the video playback can be resumed when the activity receives onResume().
- Call RVVideoView::stopPlayback() and destroy the RVVideoView when activity receives onStop(). This will stop the video (and RayV stream) and reduce resource consumption when switching to another application or to the home screen. When the activity is restarted the RVVideoView can be recreated when the activity receives onStart() and playback can be resumed.
- Call RVVideoView::stopPlayback() and destroy the RVVideoView when activity receives onDestroy(). This will stop the video (and RayV stream) and free all resources.

Troubleshooting Tools

The SDK include several tools for troubleshooting connectivity and streaming problems:

- Double tap on the progress bar will display current channel playback state.
- Tapping 5 times quickly on the progress bar will add a P2P Info button on the video. Clicking the button will display transparent P2P information on the video.
- A support information activity is included with the SDK and can be used to display internal SDK information and logs.