Android AMP SDK **v6**

Migration Guide:
From v5 to v6

Updated: 18-Apr-16

amp-sdk-support@akamai.com

TABLE OF CONTENTS

1) PLAYING A STREAM

Previously, the developer had to know all about the different decoding modes included in the SDK, to choose an appropriate one for the stream to be played:

```
// Old
String mUrl = ...;
if (mUrl.endsWith(".mp4"))
{
    mVideoContainer.setMode(VideoPlayerContainer.MODE_NATIVE_BASIC);
}
else
{
    mVideoContainer.setMode(VideoPlayerContainer.MODE_HARDWARE_ADVANCED);
}
mVideoView.playUrl(mUrl);
```

Now on v6, the SDK automatically selects the best decoding mode, according to the specific device and stream (by analyzing its metadata and MIME type).

To play the video and do this automatic analysis, there is a new async callback:

```
// New
mVideoContainer.addVideoPlayerContainerCallback(new
VideoPlayerContainer.VideoPlayerContainerCallback() {
    @Override
    public void onVideoPlayerCreated() { Log.i(TAG, "onVideoPlayerCreated"); }

    @Override
    public void onResourceReady(MediaResource resource) {
        mVideoView = mVideoContainer.getVideoPlayer();//mode set automatically
        mVideoView.setLicense(LICENSE);
        mVideoView.play(resource);
    }

    @Override
    public void onResourceError() { Log.e(TAG, "onResourceError"); }
});
mVideoContainer.prepareResource(VIDEO_URL);
```

Notice the `com.akamai.media.elements.`MediaResource class provides information about the stream:

```
String url = resource.getResourceUrl();
String mimeType = resource.getMimeType();
```

2) PLAY A STREAM AT A SPECIFIED POSITION (IN SECONDS)

Apply a similar change, using the `play(url, position)` method of the async callback:

```
// Old, using a String
mVideoView.playUrl(mUrl, mCurrentPositionInSeconds);


// New, using the MediaResource (received as parameter), on the
onResourceReady() method, from the VideoPlayerContainerCallback
mVideoView.play(resource, mCurrentPositionInSeconds);
```

3) GETTING THE VERSION OF THE AMP LIBRARY

```
// Old
String ampVersion = VideoPlayerView.VERSION;


// New
String ampVersion = AMPLibraryInfo.VERSION;
```

4) EVENTS

```
// Old
mVideoPlayerView.setEventsListener(listener);
// "listener" is of type com.akamai.media.IPlayerEventsListener


// New
mVideoPlayerView.addEventsListener(listener);
//...do some more work...
mVideoPlayerView.removeEventsListener(listener);
```

5) DECODING MODES

The SDK automatically determines the best decoding mode available.
However, if you want to change it for testing, compatibility (or for any other reason), here's how:

```
// Old
mVideoContainer.setMode(iPlayerMode);


// New (not needed)
mVideoContainer.setDefaultMode(iPlayerMode);
```

The mVideoContainer.setDefaultMode() method should be invoked before the mVideoContainer.getVideoPlayer() method, in the first lines of the onResourceReady() method of the callback.

The available decoding modes for v6 are defined in `com.akamai.media.VideoPlayerContainer`:

- *MODE_AUTOMATIC* (default, **RECOMMENDED**)
- *MODE_EXO* (for MPEG-Dash, Smooth Streaming or HLS using ExoPlayer)
- *MODE_HARDWARE_ADVANCED* (AMP's HLS engine)
- *MODE_NATIVE_BASIC* (For PMD playback, using Android's own MediaPlayer)
- *MODE_HARDWARE* (Deprecated)
- *MODE_SOFTWARE* (Deprecated)
- *MODE_NONE*

The best method is selected by the SDK (in the `VideoPlayerContainerCallback`), according to the stream and device combination, so usually you don't have to worry about invoking the `setDefaultMode()` method.

The correct moment to call `setDefaultMode()` is before the `VideoPlayerView` is requested from the `VideoPlayerContainer`:

```
@Override
public void onResourceReady(MediaResource resource)
{
    mVideoContainer.setDefaultMode(VideoPlayerContainer.MODE_EXO);

    mVideoView = mVideoContainer.getVideoPlayer();
    mVideoView.setLicense(LICENSE);
    mVideoView.setFullScreen(true);
    mVideoView.play(resource);
}
```

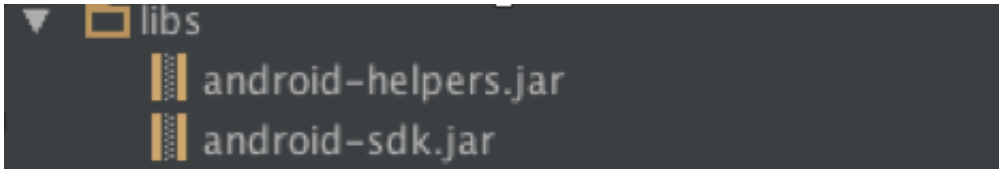
6) ENABLING OCTOSHAPE

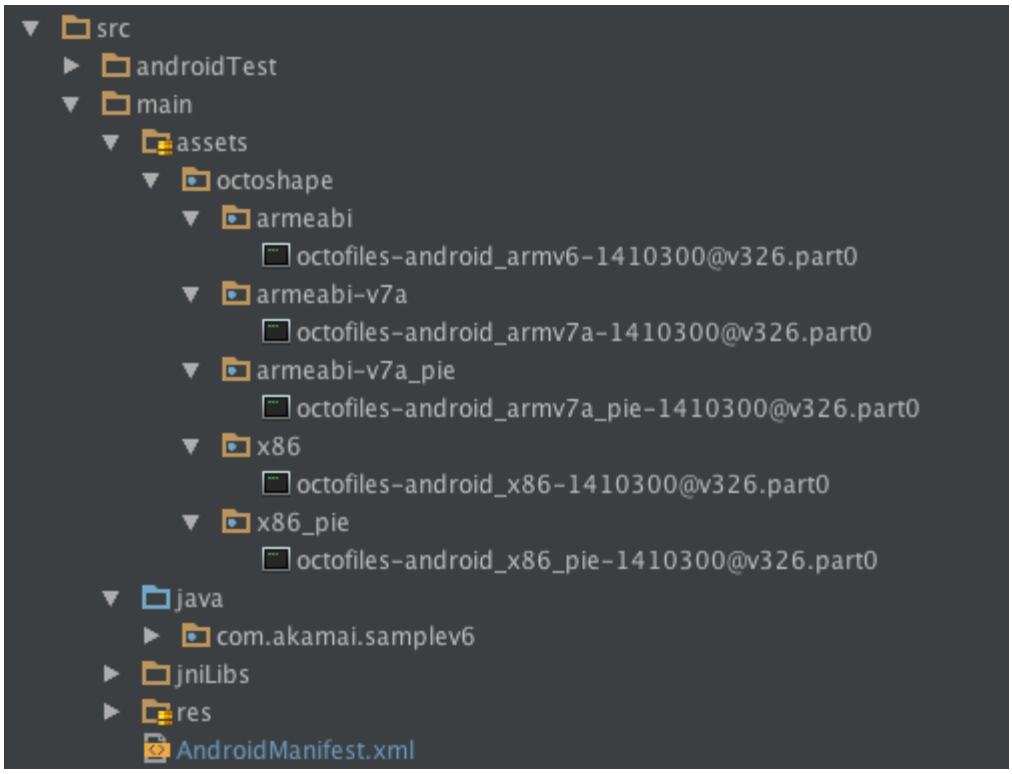
Octoshape provides the highest quality multiscreen TV-quality viewing experience for TV Everywhere and (OTT) Over-The-Top broadband content offerings. It offers services to deliver high quality video over the Internet, supporting the standard video formats for PC, Linux, Android and iOS Devices.

Octoshape streams use the “octoshape” protocol, resulting in an URL like octoshape://streams.octoshape.net/demo/live/trailers/abr

If you don't need to play Octoshape stream videos, the following steps are not required. Please follow the included samples, for more detail on every step:

6.1) Add the .JARs and native libraries as shown in the images below: (These files are included in the `/AndroidAMP-Standard-VERSION/modules/octo` folder, inside the "Modules" zip file)





6.2) Add the following permissions the AndroidManifest.xml:

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

6.3) Add the reference of the following Services in the AndroidManifest.xml:

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="OctoTest"
    android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen" >
    <activity
        android:name="com.akamai.media.octotest.MainActivity"
        android:screenOrientation="portrait"
        android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <service android:name="com.octoshape.android.service.StreamService" android:process=":OctoProc"/>
    <service android:name="com.octoshape.android.service.StreamServiceDebug" android:process=":OctoProc"/>
</application>
```

6.4) In your `android.app.Application`, create a `com.akamai.media.octoshape.OctoSystemBuilder` as a Singleton; expose it to the activities with a method like getOctoSystemBuilder().

6.5) On the activity that creates the `VideoPlayerContainer`, set the `OctoSystemBuilder`:

```
mVideoPlayerContainer.setOctoSystemBuilder(app.getOctoSystemBuilder());
```

6.6) And that's it! You can now play any Octoshape stream.

The Akamai Difference

U.S. Headquarters
8 Cambridge Center
Cambridge, MA 02142
Tel 617.444.3000
Fax 617.444.3001
U.S. toll-free 877.4AKAMAI
(877.425.2624

International Offices
Unterfoehring, Germany
Paris, France
Milan, Italy
London, England
Madrid, Spain
Stockholm, Sweden
Bangalore, India
Sydney, Australia
Beijing, China
Tokyo, Japan
Seoul, Korea
Singapore
San José, Costa Rica

©2010 Akamai Technologies, Inc. All Rights Reserved. Reproduction in whole or in part in any form or medium without express written permission is prohibited. Akamai and the Akamai wave logo are registered trademarks. Other trademarks contained herein are the property of their respective owners. Akamai believes that the information in this publication is accurate as of its publication date; such information is subject to change without notice.

www.akamai.com

Akamai® provides market-leading, cloud-based services for optimizing Web and mobile content and applications, online HD video, and secure e-commerce. Combining highly-distributed, energy-efficient computing with intelligent software, Akamai's global platform is transforming the cloud into a more viable place to inform, entertain, advertise, transact and collaborate. To learn how the world's leading enterprises are optimizing their business in the cloud, please visit www.akamai.com and follow @Akamai on Twitter.

Akamai Technologies, Inc.