



# Android AMP SDK **v6**

## Migration Guide: From v5 to v6

Updated: 4-Feb-16

[amp-sdk-support@akamai.com](mailto:amp-sdk-support@akamai.com)



## TABLE OF CONTENTS

1) Playing the stream	3
2) Play a stream at a specified position (in seconds)	4
3) Getting the VERSION of the AMP Library	4
4) Events	4
5) Decoding modes	5
6) Playing an HLS in MODE_EXO	6
7) Enabling OctoShape	7

## 1) PLAYING THE 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:

```
// v5
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:

```
// v6
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:

```
// v5, using a String
mVideoView.playUrl(mUrl, mCurrentPositionInSeconds);
```

```
// v6, 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

```
// v5
String ampVersion = VideoPlayerView.VERSION;
```

```
// v6
String ampVersion = AMPLibraryInfo.VERSION;
```

## 4) EVENTS

```
// v5
mVideoPlayerView.setEventListener(listener);
// where "listener" is of type com.akamai.media.IPlayerEventsListener
```

```
// v6
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 (or for any other reason), here's how:

```
// v5
mVideoContainer.setMode(iPlayerMode);
```

```
// v6
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.

## 6) PLAYING AN HLS IN MODE\_EXO

By default, HLS streams are played in the `MODE_HARDWARE_ADVANCED`.

For testing or other purposes, they can also be decoded by the `MODE_EXO`.

To achieve this, the `MODE_EXO` should be set 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);
}
```

## 7) ENABLING OCTOSHAPE

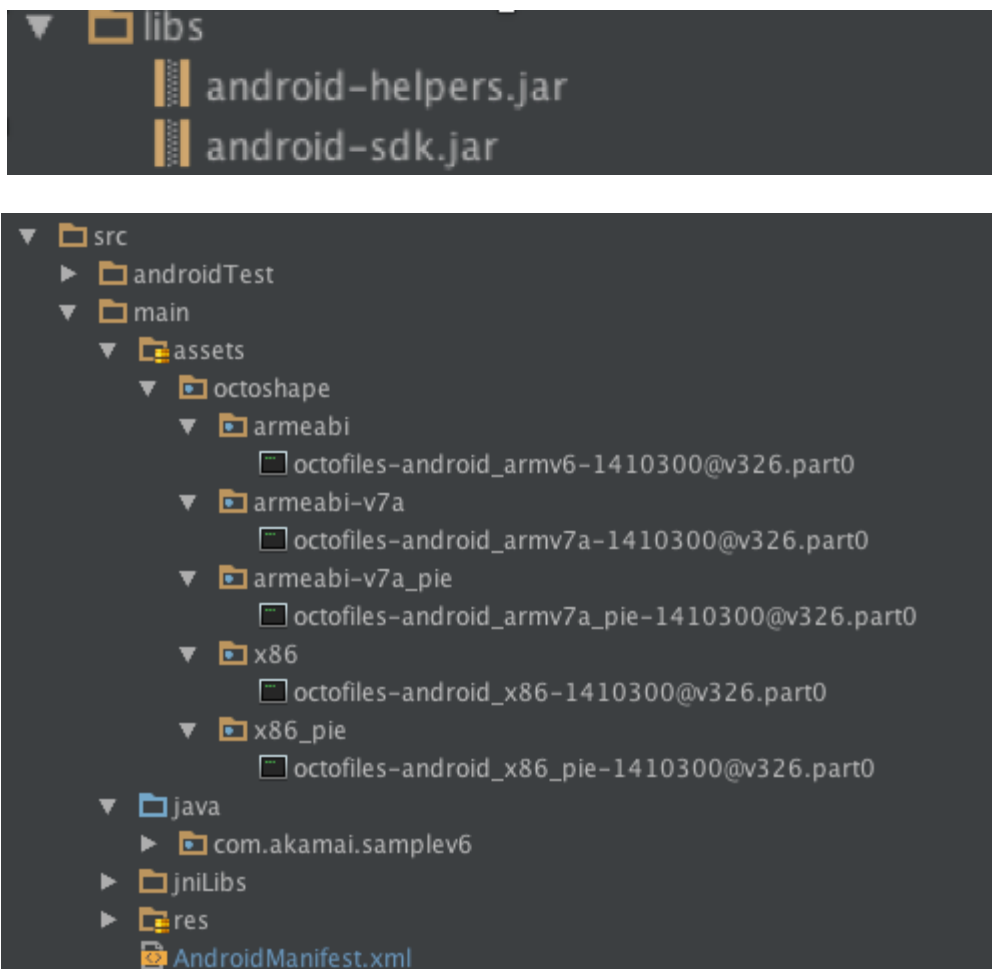
Important: If you don't need to play Octoshape stream videos, these steps are not required.

Octoshape offers services to deliver high quality video over the Internet, supporting the standard video formats for PC, AMC, Linux, Android and iOS Devices.

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

To be able to play them, two steps must be followed:

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>
```







## 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)

### [www.akamai.com](http://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](http://www.akamai.com) and follow @Akamai on Twitter.

### International Offices

Unterfoehring, Germany	Bangalore, India
Paris, France	Sydney, Australia
Milan, Italy	Beijing, China
London, England	Tokyo, Japan
Madrid, Spain	Seoul, Korea
Stockholm, Sweden	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.

Akamai Technologies, Inc.

