AVLib SDK for Android v.1.0.1
Powerful RTMP-streaming solution

Features

* Ability to use with outdated and low-performance devices with Android 4.1+
* Auto-detection device performance and setting an optimal stream configuration
* Handling rotation and size changes with no stream re-initiation
* Ability to customize frame size, video codec quality, video and audio bitrate, and disable audio
* Activity and Fragment support
* Decreasing bitrate when connection is slow
* Switching the main/front cameras
* Scaling types, aspect ratio correction, and full handling lifecycle
* Handling errors
* Simple interface
* Fast performance
* No dependencies
* Armv7a and armv8-64 support
* Action cameras support
* "Tap to focus" feature
* "Record stream" feature

Introduction

This document briefly describes AVLib SDK for the development of broadcasting apps using RTMP protocol.

This guide will help developers to quickly integrate AVLib SDK into their native applications.

Setup

1. Add AVLib.aar to your project.
2. Make sure RECORD_AUDIO and CAMERA permissions are enabled before you start broadcasting.

Notes:

AVLib requires the following permissions (all of them have already been included into AVLib SDK):

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

If you use a record stream feature, add and handle `WRITE_EXTERNAL_STORAGE` permission.

Code obfuscation. Proguard

Add these lines:

```
-keep public interface com.onix.avlib.** { *; }
-keep public class com.onix.avlib.** { *; }
```

Usage

The SDK provides CameraStreamer and ActionCameraStreamer classes for broadcasting. The first one provides broadcasting from an Android native camera source. The second one performs streaming to server from action cameras, like GoPro.

All work with this SDK must be carried out in ‘Activity’ or ‘Fragment’ only.

Streaming from Android camera

The classes CameraStreamer and CameraStreamView are needed for streaming from Android camera:

- CameraStreamView is the view used for rendering video preview.
- CameraStreamer is the main class that provides the streaming functionality from Android camera to RTMP server.

There are some lifecycle methods in CameraStreamer class that should be called in ‘onCreate’, ‘onDestroy’, ‘onResume’, ‘onCreateView’, ‘onConfigurationChanged’, ‘onSaveInstanceState’ callbacks. CameraStreamView is the view for ‘stream preview’ and should be placed into your Layout.

All code snippets included below are correct for ‘Fragment’ implementation.

The object of CameraStreamer class can be instantiated with the initial camera type (front/main). By default it will be the MAIN camera.

Enum CamType: MAIN, FRONT

****

void onCreate(Activity activity, Bundle savedInstanceState)

activity - current 'Activity' instance
savedInstanceState - bundle instance for restoring states

```
@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setRetainInstance(true);
       mStreamer = new CameraStreamer(CamType.MAIN);
       mStreamer.onCreate(getActivity(), savedInstanceState);
}
```

****

void onCreateView(CameraStreamView cameraView, PreviewScaleType scaleType)

cameraView - rendering view for stream
scaleType - scale type for preview (described below)

```
@Override
public View onCreateView(...) {
       ….
       mStreamer.onCreateView(mBinding.cameraView, PreviewScaleType.CENTER_CROP);
       return view;
}
```

****

void onPause()

```
@Override
public void onPause() {
       super.onPause();
       mStreamer.onPause();
}
```

****

void onResume(IStreamerEvents eventListener)

eventListener - callback for all events (described below)

```
@Override
public void onResume() {
       super.onResume();
       mStreamer.onResume(this);
}
```

****

void onDestroy()

```
@Override
public void onPause() {
       super.onPause();
       mStreamer.onDestroy();
```

```
}
```

****

public void onConfigurationChanged(Configuration newConfig)

newConfig - new configuration

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
       super.onConfigurationChanged(newConfig);
       mStreamer.onConfigurationChanged(newConfig);
}
```

****

void onSaveInstanceState(Bundle outState)

outState - bundle instance for saving states

```
@Override
public void onSaveInstanceState(Bundle outState) {
       super.onSaveInstanceState(outState);
       mStreamer.onSaveInstanceState(outState);
}
```

****

Enum PreviewScaleType is used for scaling the bounds of stream preview to the CameraStreamView view.
CENTER_INSIDE - scale the preview uniformly while maintaining the aspect ratio
CENTER_CROP - cover all areas of CameraStreamView while maintaining the aspect ratio and crop it if needed.
*This option doesn't affect the actual picture size. It only configures the preview.

'Activity' implementation

If streaming is carried out within 'Activity,' there is only one difference - you need to place *onCreateView* at once after `onCreate` method, for example:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
 ...
   mStreamer = new CameraStreamer();
   mStreamer.onCreate(this, savedInstanceState);
   mStreamer.onCreateView(mBinding.cameraView,PreviewScaleType.CENTER_CROP);
...
}
```

****

Interface *IStreamerEvents* must be set with *onResume method.* This is how it is used for handling streamer events:

`void onInitStarted` - start streamer initialization.

`void onInitCompleted` - completed streamer initialization.

`void onStreamingFpsChanged(int fps)` - FPS updating.

`void onPreviewSizesAvailable(List<PreviewSize> sizes)` - fired once all camera resolutions accessible for streaming are detected. `PreviewSize` - simple class with width and height fields.

`void onPreviewSizeSelected(PreviewSize size)`- fired when camera resolution is selected/changed.

`void onStreamStateChanged(StreamerState event)` - callback for handling the streaming state.

****

Enum `StreamerState`:

`ERROR_CONNECTION` - connection error (RTMP server does not respond, no Internet connection)
`ERROR_CAMERA_BUSY` - cannot start camera preview. Camera is not supported, busy or missing
`STREAM_STARTED` - streaming started successfully
`STREAM_STOPPED` - streaming stopped successfully
`ERROR_SERVER` - wrong server url scheme or empty
`ERROR_PERMISSIONS` - missing permissions (permission.CAMERA, permission.RECORD_AUDIO or permission.INTERNET)

****

public void setServerUrl(String serverUrl)

- set rtmp server url. Needs to be called before `startStreaming()`.

public void setRecordFile(String fileName)

- used if there is need to record stream to file.

public void switchCamera()

- switch cameras. Main cam is by default.

public void stopStreaming()

- stop streaming

public void startStreaming()

- start streaming

public List<PreviewSize> getAvailablePreviewSizes()

- get supported camera preview sizes

public PreviewSize getCurrentPreviewSize()

- get the current selected preview size

public void enableLogs()

- enable logs for development

public boolean isFrontCamera()

- check if the front camera is selected

public boolean isStarted()

- check if the stream is started

public void setVideoBitrate(VideoBitrate bitrate)

- set video bitrate

Enum VideoBitrate: B128K, B256K, B512K, B1024K, B2048K, B3072K, B4096K, B6144K

public void setAudioBitrate(AudioBitrate bitrate)

- set audio bitrate

Enum AudioBitrate: B32K, B64K, B128K, B256K

public boolean setPreviewSize(PreviewSize size)

- select preview size

public void enableAudio(boolean enable)

- enable/disable audio

public AudioBitrate getAudioBitrate()

- get the current audio bitrate

public VideoBitrate getVideoBitrate()

- get the current video bitrate

public boolean isAudioEnabled()

- check if audio is enabled

Streaming from Action cameras

The classes ActionCameraStreamer and ActionCameraStreamView are needed for streaming from action cameras.

- ActionCameraStreamView is the view used for rendering video preview. ActionCameraStreamer is the class that provides streaming functionality from an action camera to RTMP server.

Streaming from action cameras is available starting with Android LOLLIPOP. Also, mobile Internet connection should be enabled on a device.

ActionCameraStreamer must be initialized with `IActionCamera` implementation. `IActionCamera` has the following methods:

`void onResume()` - will be called when the app turns to the foreground.
`void onPause()` - will be called when the app turns to the background.
`String getPreviewUrl()` - returns direct uri to the preview stream of an action cam. This method will be carried out by ActionCameraStreamer in the background thread.

All integration steps are the same as for streaming from Android camera. The only difference is that `IActionCamStreamerEvents` is used as the interface for handling streamer states.

```
onResume(IActionCamStreamerEvents listener)
```

Enum ActionStreamerState:
ERROR_NETWORKS, ***ERROR_INTERNET_CONNECTION***, ***STREAM_STARTED***, ***STREAM_STOPPED***, ***ERROR_SERVER***, ***PREVIEW_STARTED***, ***ERROR_CAMERA_PREVIEW***, ***PROCESSING_NETWORKS***, ***PREVIEW_PREPARING***