

Android Player

Example of player for Android

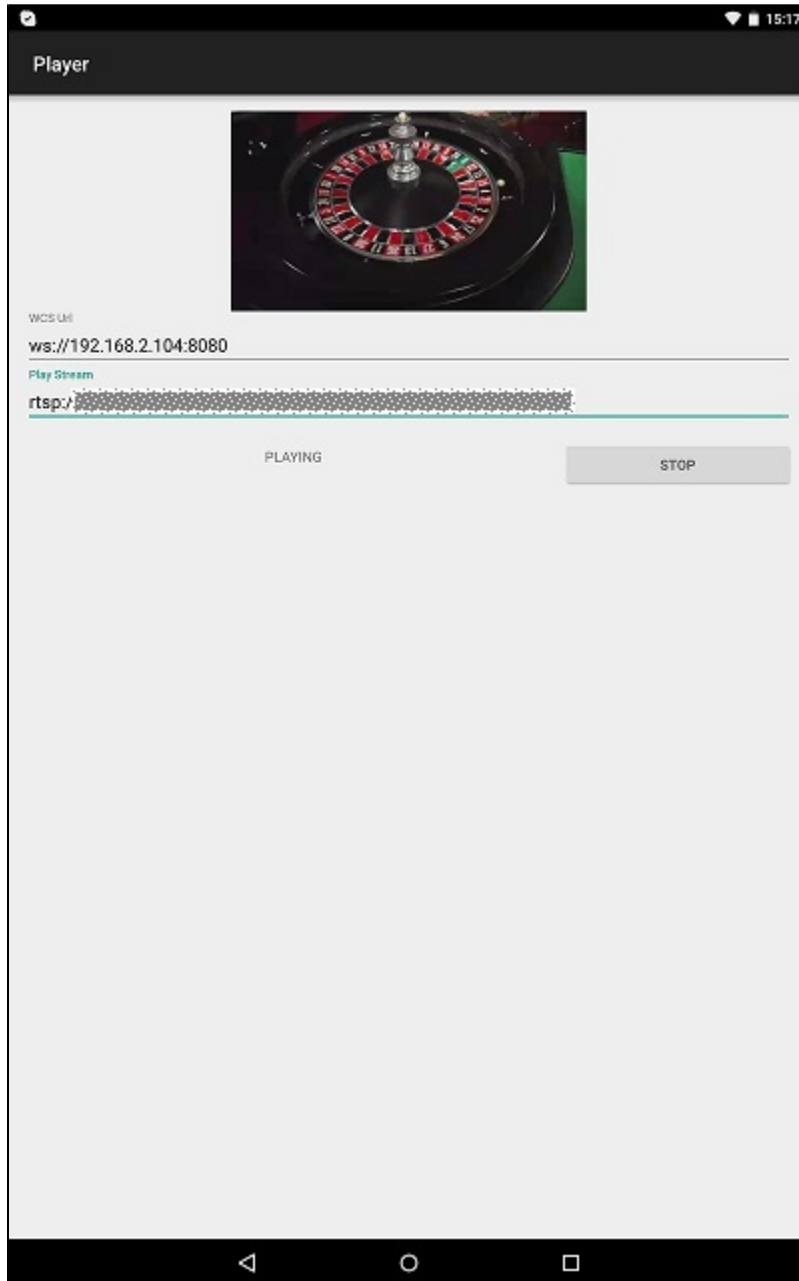
This player can be used to play any type of stream on Web Call Server

- RTSP
- WebRTC
- RTMP
- RTMFP

On the screenshot below an RTSP stream is being playing.

In the input fields

- 192.168.2.104 in the URL is the address of the WCS server
- stream name is entered in to the 'Play Stream' field (RTSP URL in this case)



Work with code of the example

To analyze the code, let's take class [PlayerActivity.java](#) of the player example, which can be downloaded with corresponding build [1.0.1.38](#).

1. Initialization of the API

Flashphoner.init()[code](#)

For initialization, object Context is passed to the init() method.

```
Flashphoner.init(this);
```

2. Session creation.

Flashphoner.createSession()[code](#)

Object SessionOptions with the following parameters is passed to createSession() method()

- URL of WCS server
- SurfaceViewRenderer, which will be used to play video stream

```
SessionOptions sessionOptions = new SessionOptions(mWcsUrlView.getText().toString());
sessionOptions.setRemoteRenderer(remoteRender);

/**
 * Session for connection to WCS server is created with method createSession().
 */
session = Flashphoner.createSession(sessionOptions);
```

3. Connection to the server.

Session.connect()[code](#)

```
session.connect(new Connection());
```

4. Receiving the event confirming successful connection

session.onConnected()[code](#)

```
@Override
public void onConnected(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            mStartButton.setText(R.string.action_stop);
            mStartButton.setTag(R.string.action_stop);
            mStartButton.setEnabled(true);
            mStatusView.setText(connection.getStatus());
            ...
        }
    });
}
```

5. Playback of video stream.

Session.createStream(), Stream.play()[code](#)

Object StreamOptions ([line 114](#)) with name of the stream is passed to the createStream() method.

```

StreamOptions streamOptions = new StreamOptions(mPlayStreamView.getText().toString());

/**
 * Stream is created with method Session.createStream().
 */
playStream = session.createStream(streamOptions);

/**
 * Callback function for stream status change is added to display the status.
 */
playStream.on(new StreamStatusEvent() {
    @Override
    public void onStreamStatus(final Stream stream, final StreamStatus streamStatus) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (!StreamStatus.PLAYING.equals(streamStatus)) {
                    Log.e(TAG, "Can not play stream " + stream.getName() + " " + streamStatus);
                } else if (StreamStatus.NOT_ENOUGH_BANDWIDTH.equals(streamStatus)) {
                    Log.w(TAG, "Not enough bandwidth stream " + stream.getName() + ", consider using lower
video resolution or bitrate. " +
                        "Bandwidth " + (Math.round(stream.getNetworkBandwidth() / 1000)) + " " +
                        "bitrate " + (Math.round(stream.getRemoteBitrate() / 1000)));
                } else {
                    mStatusView.setText(streamStatus.toString());
                }
            }
        });
    }
});

/**
 * Method Stream.play() is called to start playback of the stream.
 */
playStream.play();

```

6. Session disconnection.

Session.disconnect()[code](#)

```
session.disconnect();
```

7. Receiving the event confirming successful disconnection

session.onDisconnection()[code](#)

```

@Override
public void onDisconnection(final Connection connection) {
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            mStartButton.setText(R.string.action_start);
            mStartButton.setTag(R.string.action_start);
            mStartButton.setEnabled(true);
            mStatusView.setText(connection.getStatus());
        }
    });
}

```