Android Player

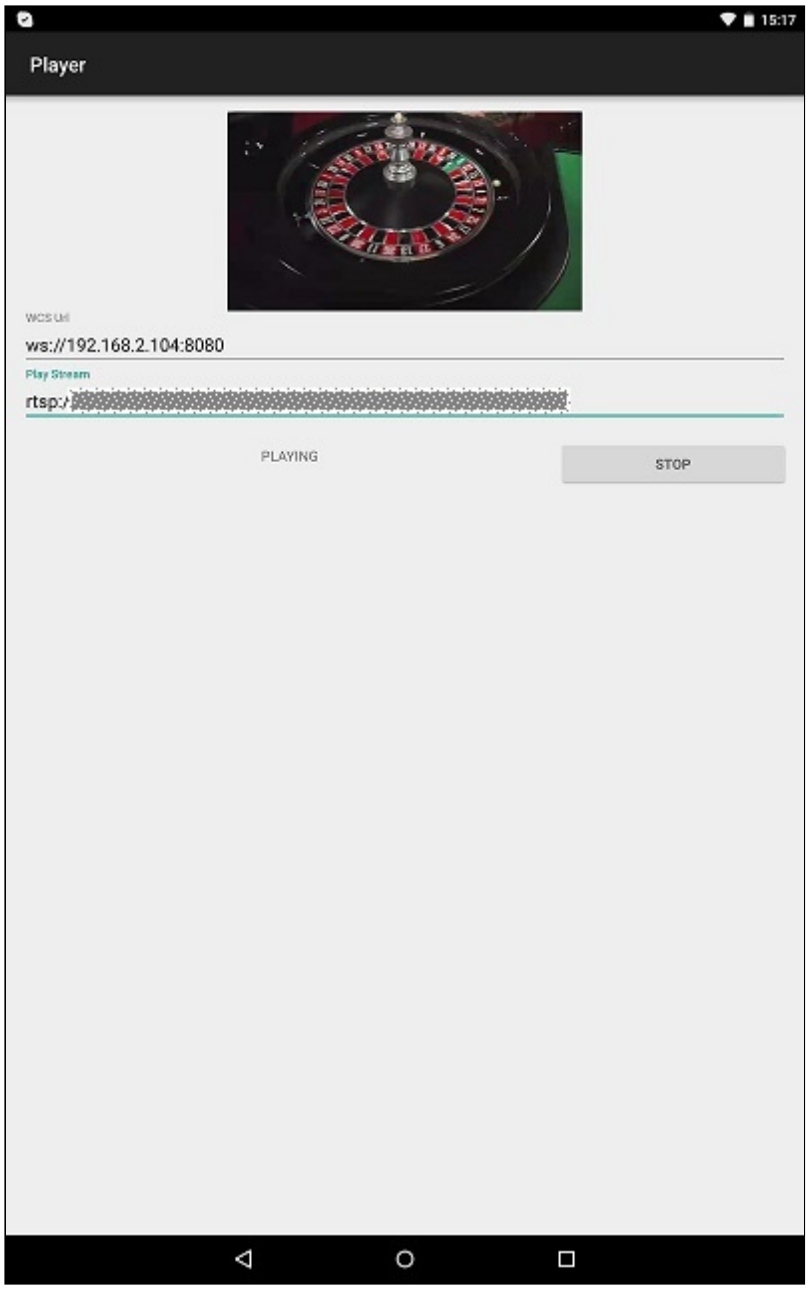
Пример плеера для Android

Данный плеер может использоваться для воспроизведения любого типа потока с Web Call Server:

- RTSP
- WebRTC
- RTMP

На скриншоте ниже представлен пример во время воспроизведения RTSP-потока.

В URL в поле ввода `WCS URL` `192.168.2.104` - адрес WCS-сервера.
В поле ввода `Play Stream` - имя потока, в данном случае RTSP URL.



Работа с кодом примера

Для разбора кода возьмем класс PlayerActivity.java примера `player`, который доступен для скачивания в соответствующей сборке 1.0.1.38.

1. Инициализация API

`Flashphoner.init()` code

При инициализации методу `init()` передается объект `Context`.

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

2. Создание сессии

`Flashphoner.createSession()` code

Методу передается объект `SessionOptions` со следующими параметрами

- URL WCS-сервера
- `SurfaceViewRenderer remoteRenderer`, который будет использоваться для воспроизведения видеопотока

```
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. Подключение к серверу

`Session.connect()` code

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

4. Получение от сервера события, подтверждающего успешное соединение

`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. Воспроизведение видеопотока

`Session.createStream()`, `Stream.play()` code

При создании потока методу `Session.createStream()` передается объект `StreamOptions` с именем видеопотока для воспроизведения

```
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. Закрытие соединения

`Session.disconnect()` code

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

7. Получение события, подтверждающего разъединение

`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());
        }
    });
}
```