Android Stream Recording

Example of Android application for stream recording

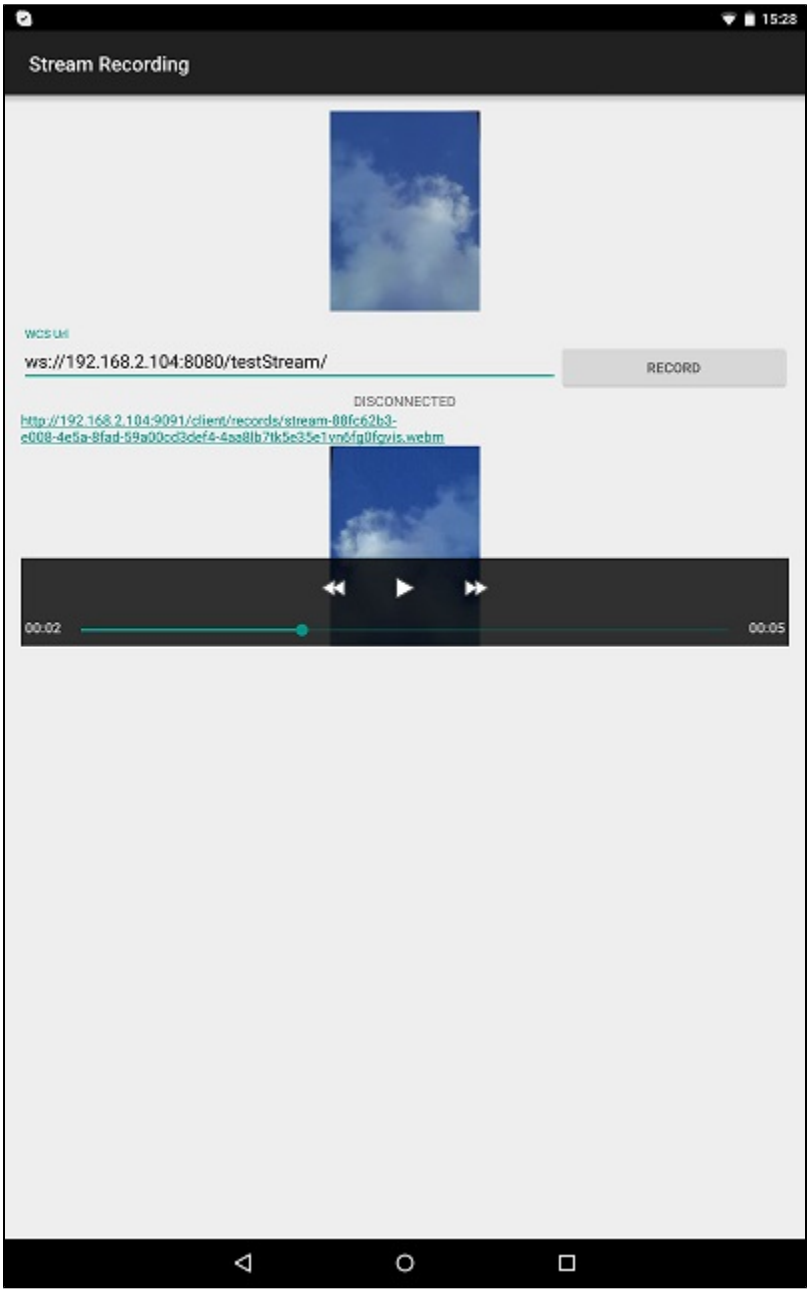
This streamer can be used to publish and record WebRTC video stream on Web Call Server.

On the screenshot below the example is displayed when connection to server is closed and stream publication is stopped.
In the URL specified in the input field

- 192.168.2.104 is the address of the WCS server
- testStream is the stream name

Above the input field video from the camera is displayed.
Under the input field are located

- download link for the recording of published stream
- media player, which can be used to play the recording



Work with code of the example

To analyze the code, let's take class StreamRecordingActivity.javaof the stream-recording 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 SessionOptionswith the following parameters is passed to createSession() method:

- URL of WCS server
- SurfaceViewRenderer, which will be used to display video from the camera

```
SessionOptions sessionOptions = new SessionOptions(url);
sessionOptions.setLocalRenderer(localRender);

/**
  * 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. Video stream creation

Session.createStream(), ActivityCompat.requestPermissions()code
Object StreamOptions (line 135) with is passed to the sreateStream() method

- name of the stream
- true for parameter 'record' - to enable stream recording

```
StreamOptions streamOptions = new StreamOptions(streamName);
streamOptions.setRecord(true);

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

/**
  * Callback function for stream status change is added to display the status.
  */
publishStream.on(new StreamStatusEvent() {
    @Override
    public void onStreamStatus(final Stream stream, final StreamStatus streamStatus) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
               if (StreamStatus.PUBLISHING.equals(streamStatus)) {
                   mStatusView.setText("RECORDING");

                   /**
                     * Filename of the recording is determined.
                     */
                   recordFilename = stream.getRecordName();
                   return;
               } else if (StreamStatus.FAILED.equals(streamStatus)) {
                   Log.e(TAG, "Can not publish stream " + stream.getName() + " " + streamStatus);
                   recordFilename = null;
               }
               mStatusView.setText(streamStatus.toString());
            }
       });
    }
});
```

6. Video stream publishing when permissions is granted

Stream.publish()code

```
case PUBLISH_REQUEST_CODE: {
    if (grantResults.length == 0 ||
           grantResults[0] != PackageManager.PERMISSION_GRANTED ||
           grantResults[1] != PackageManager.PERMISSION_GRANTED) {
        mStartButton.setEnabled(false);
        session.disconnect();
        Log.i(TAG, "Permission has been denied by user");
    } else {
        /**
          * Method Stream.publish() is called to publish stream.
          */
        publishStream.publish();
        Log.i(TAG, "Permission has been granted by user");
    }
}
```

7.Receiving the event confirming successful stream publishing

StreamStatusEvent PUBLISHINGcode

On this event, stream record filemane is defined with Stream.getRecordName() method.

```
publishStream.on(new StreamStatusEvent() {
    @Override
    public void onStreamStatus(final Stream stream, final StreamStatus streamStatus) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
               if (StreamStatus.PUBLISHING.equals(streamStatus)) {
                   mStatusView.setText("RECORDING");

                   /**
                     * Filename of the recording is determined.
                     */
                   recordFilename = stream.getRecordName();
                   return;
               } else if (StreamStatus.FAILED.equals(streamStatus)) {
                   Log.e(TAG, "Can not publish stream " + stream.getName() + " " + streamStatus);
                   recordFilename = null;
               }
               mStatusView.setText(streamStatus.toString());
           }
       });
    }
});
```

8. Session disconnection.

Session.disconnect()code

```
mStartButton.setEnabled(false);

/**
  * Connection to WCS server is closed with method Session.disconnect().
  */
session.disconnect();
```

9.Receiving the event confirming successful disconnection

session.onDisconnection()code

On this event, the record file download link is formed, and local mediaplayer is launched to play the file

```
@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());

            /**
              * After disconnection, download link for the recording of the published stream is displayed, and
the recording can be played in the media player of the application.
              */
            if (recordFilename != null) {
                /**
                  * Download link is formed.
                  * Stream recordings are saved to directory WCS_HOME/client/records on the server.
                  */
                String url = "http://" + uri.getHost() +":9091/client/records/" + recordFilename;
                mRecordedLink.setText(url);
                Linkify.addLinks(mRecordedLink, Linkify.WEB_URLS);

                MediaController mediaController = new MediaController(StreamRecordingActivity.this);
                mediaController.setAnchorView(mRecordedVideoView);
                mRecordedVideoView.setMediaController(mediaController);
                mRecordedVideoView.setVideoURI(Uri.parse(url));

                /**
                  * Playback of the recording in the media player is started.
                  */
                mRecordedVideoView.start();
            }
        }
    });
```