Android Stream Recording

Пример Android-приложения для записи видеопотока

Данный пример может использоваться с Web Call Server для публикации и записи WebRTC-видеопотока.

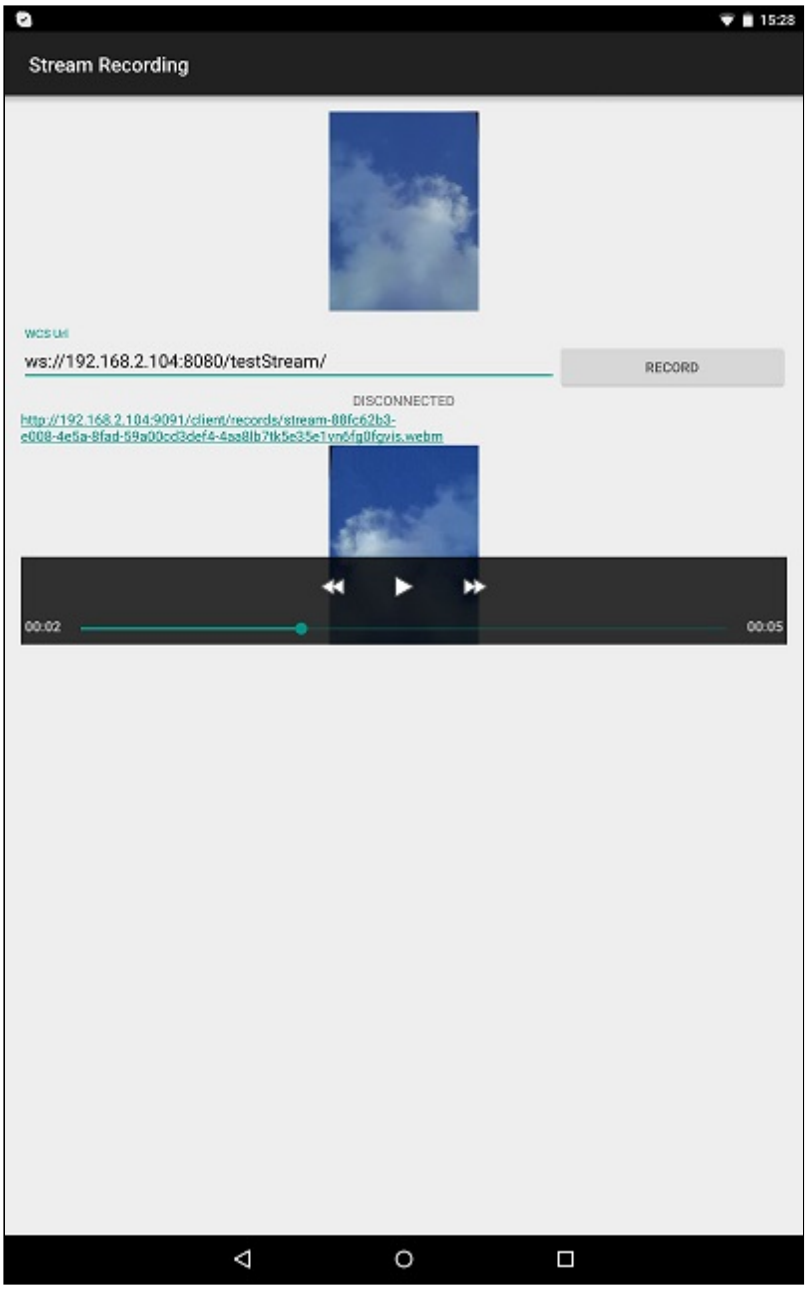
На скриншоте ниже представлен пример после завершения публикации потока.

В URL в поле ввода

- `192.168.2.104` - адрес WCS-сервера
- `testStream` - имя потока

Над полем ввода отображается видео с камеры.
Под полем ввода расположены

- ссылка для скачивания записи потока
- медиа-плеер, в котором можно воспроизвести запись



Работа с кодом примера

Для разбора кода возьмем класс StreamRecordingActivity.java примера `stream-recording`, 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 localRenderer`, который будет использоваться для отображения видео с камеры

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

/**
  * 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. Создание видеопотока и подготовка к публикации

`Session.createStream()` code

Методу `Session.createStream()` передается объект `StreamOptions` с параметрами:

- имя видеопотока
- `setRecord(true)` для записи потока

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

ActivityCompat.requestPermissions(StreamRecordingActivity.this,
      new String[]{Manifest.permission.RECORD_AUDIO,
Manifest.permission.CAMERA},
      PUBLISH_REQUEST_CODE);
```

6. Публикация потока после предоставления соответствующих прав

`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. Получение от сервера события, подтверждающего успешную публикацию потока

`StreamStatusEvent.PUBLISHING` code

При получении данного события определяется имя файла записи потока с помощью метода `Stream.getRecordName()`.

```
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. Закрытие соединения

`Session.disconnect()` code

```
mStartButton.setEnabled(false);

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

9. Получение события, подтверждающего разъединение

`Session.onDisconnection()` code

При получении данного события формируется ссылка на скачивание файла записи потока и запускается локальный медиаплеер для воспроизведения файла

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