Android Streamer

Пример стримера для Android

Данный стример может использоваться для публикации WebRTC-видеопотока на Web Call Server.

На скриншоте ниже представлен пример во время публикации потока.

В URL в поле ввода

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

Слева отображается видео с камеры, справа воспроизводится опубликованный поток

Работа с кодом примера

Для разбора кода возьмем класс StreamerActivity.java примера streamer, 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
- `SurfaceViewRenderer remoteRenderer`, который будет использоваться для воспроизведения опубликованного видеопотока

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

           /**
             * The options for the stream to publish are set.
             * The stream name is passed when StreamOptions object is
created.
             */
           StreamOptions streamOptions = new StreamOptions(streamName);
```

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

5. Создание видеопотока

`Session.createStream()`, `ActivityCompat.requestPermissions()` code

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

```
StreamOptions streamOptions = new StreamOptions(streamName);

/**
  * Stream is created with method Session.createStream().
  */
publishStream = session.createStream(streamOptions);
...
ActivityCompat.requestPermissions(StreamerActivity.this,
       new String[]{Manifest.permission.RECORD_AUDIO,
Manifest.permission.CAMERA},
       PUBLISH_REQUEST_CODE);
```

6. Публикация потока

`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

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

При получении данного события создается превью-видеопоток при помощи `Session.createStream()` и вызывается `Stream.play()` для его воспроизведения

```
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)) {

                    /**
                      * The options for the stream to play are set.
                      * The stream name is passed when StreamOptions object
is created.
                      */
                    StreamOptions streamOptions = new
StreamOptions(streamName);

                    /**
                      * Stream is created with method Session.createStream().
                      */
                    playStream = session.createStream(streamOptions);
                    ...
                    /**
                      * Method Stream.play() is called to start playback of
the stream.
                      */
                    playStream.play();
                } else {
                    Log.e(TAG, "Can not publish stream " + stream.getName() +
" " + streamStatus);
                }
                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

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