kurento-tree-client-android Documentation

Release 1.0.4

Jukka Ahola

December 15, 2016

Contents

Contents:

CHAPTER 1

Readme

This project is part of NUBOMEDIA www.nubomedia.eu

1.1 kurento-tree-client-android

The repository contains documentation for installing and utilising the kurento Tree client for Android. The repository contains description of the architecture of the kurento-tree-client-android and also the source code of the kurento-tree-client-android implementation.

CHAPTER 2

Developers Guide

This documents provides information how to utilize the kurento-tree-client-android library for your project.

Setup the developing environment by importing the project to Android Studio. You can import this project to your own Android Studio project via Maven (jCenter or Maven Central) by adding the following line to module's build.gradle file:

```
compile 'fi.vtt.nubomedia:kurento-tree-client-android:(version-code)'
```

Where `(version-code)` is the latest version of the library. The version history can be found in https://mvnrepository.com/artifact/fi.vtt.nubomedia/kurento-tree-client-android

`KurentoTreeAPI` is used as follows. First import clauses

```
import fi.vtt.nubomedia.kurentotreeclientandroid.KurentoTreeAPI;
import fi.vtt.nubomedia.kurentotreeclientandroid.TreeListener;
import fi.vtt.nubomedia.kurentotreeclientandroid.TreeNotification;
import fi.vtt.nubomedia.kurentotreeclientandroid.TreeResponse;
import fi.vtt.nubomedia.kurentotreeclientandroid.TreeError;
import fi.vtt.nubomedia.utilitiesandroid.LooperExecutor;
import fi.vtt.nubotest.util.Constants;
```

Implement `TreeListener` either via inheritance or composition (we use inheritance in this example)

```
public class MyClass implements TreeListener {
        public void onTreeResponse(TreeResponse response){ ... }
        public void onTreeError(TreeError error){ ... }
        public void onTreeNotification(TreeNotification notification) { ... }
        public void onTreeConnected() { ... }
        public void onTreeDisconnected() { ... }
}
```

The callback functions are described in the javadoc. Next, implement `KurentoTreeAPI`:

```
public class MyClass implements TreeListener {

        private LooperExecutor executor;
        private static KurentoTreeAPI kurentoTreeAPI;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                executor = new LooperExecutor();
                executor.requestStart();
                String wsRoomUri = "ws://mykurentoserver:8080/room";
                kurentoTreeAPI = new KurentoTreeAPI(executor, wsUri, this);
```

```
        }
}
```

Your `KurentoTreeAPI` has been now created. To connect to the server, simple execute the following command:

```
kurentoTreeAPI.connectWebSocket();
```

After the connection has been established, you need to decide if the client is going to be master (the one which broadcasts the media) or one of viewers (which receives the media). As a master client, you will need to create a tree on the server by invoking

```
int createTreeRequestId = 123;
String treeId = "MyTree";
kurentoTreeAPI.sendCreateTree(treeId, createTreeRequestId);
```

Where `treeId` is a unique identifier for the tree object. `createTreeRequestId` is simply an id for tracking the responses to this request. On method `onTreeResponse` you may receive the response and for example create a media connection to the server by using `nbmWebRTCPeer`:

```
if (Integer.valueOf(response.getId()) == createTreeRequestId) {
    nbmWebRTCPeer.generateOffer("myoffer", true);
}
```

For more information on how to handle p2p media connectivity, please refer to https://readthedocs.org/projects/doc-webrtcpeer-android/

Now you have a master client created which has an active media connection to a tree. To disconnect from tree and destroy it, invoke the following:

```
int treeDisconnectId = 124;
int treeBurnId = 124;
kurentoTreeAPI.sendRemoveTreeSource(treeId, treeDisconnectId);
kurentoTreeAPI.sendRemoveTree(treeId, treeBurnId);
```

Creating a viewer client is a somewhat similar process. A viewer does not create a tree object but instead joins one. Start by creating an offer for receiving media

```
nbmWebRTCPeer.generateOffer("myoffer", false);
```

This time the second parameter, `includeLocalMedia`, should be set to `false`. Once you have the local offer generated, catch it in the `onLocalSdpOfferGenerated` and send `sendAddTreeSink` request:

```
public void onLocalSdpOfferGenerated(final SessionDescription sessionDescription, NBMPeerConnection r
    addSinkRequestId = 123;
    treeId = "MyTree";
    kurentoTreeAPI.sendAddTreeSink(treeId, sessionDescription.description, addSinkRequestId);
}
```

This tutorial only comprises the use of tree API. In order to have a complete mobile p2p application, all components `nbmWebRTCPeer`, `KurentoTreeAPI` and `KurentoRoomAPI` should be used in conjunction. You may find documentation on

https://readthedocs.org/projects/doc-webrtcpeer-android/

http://doc-kurento-room-client-android.readthedocs.io/en/latest/

Source code is available at https://github.com/nubomedia-vtt/kurento-tree-client-android

Please refer to the Java documentation to learn more about the available API functions. The Javadoc is included in the source code and can be downloaded from the link below: https://github.com/nubomedia-vtt/kurento-tree-client-android/tree/master/javadoc

Support is provided through the Nubomedia VTT Public Mailing List available at https://groups.google.com/forum/#!forum/nubomedia-vtt

2.1 Adding a trusted self-signed certificate

KurentoRoomAPI supports developers to add a trusted self-signed certificate. This allows testing without CA certificate, and moreover if the application uses only one Kurento server, no CA certificate is needed.

Here is an example on how to include a self-signed certificate from assets in Android Studio:

```
KurentoRoomAPI kurentoRoomAPI;
CertificateFactory cf = CertificateFactory.getInstance("X.509");
InputStream caInput = new BufferedInputStream(myActivity.context.getAssets().open("my_server_certifi
Certificate myCert = cf.generateCertificate(caInput);
kurentoRoomAPI.addTrustedCertificate("MyServersCertificate", myCert);
kurentoTreeAPI.useSelfSignedCertificate(true);
```

Now the application trusts a server which possesses private key of certificate "my_server_certificate.cer".

2.2 WSS support on Android 5.0.x (Lollipop) and up

kurento-room-client-android library uses Maven org.java_websocket: http://mvnrepository.com/artifact/org.java-websocket/Java-WebSocket/

However, org.java_websocket version 1.3.0 is not compatible with Android 5.0.x systems due to malfunction in wss protocol handshake. Until a newer version is uploaded to Maven, a workaround is to compile a newer version from git: https://github.com/TooTallNate/Java-WebSocket

This limitation is known to exist only in wss TSL handshake. Android 5.1.x and up should not have this issue.

CHAPTER 3

Installation Guide

This documents provides information how to compile the kurento-tree-client-android library from the sources for your project.

3.1 Clone repository:

```
sudo apt-get install git
git clone https://github.com/nubomedia-vtt/kurento-tree-client-android.git
```

Setup the developing environment by importing the project to Android Studio. Import the third-party libraries via Maven by adding the following lines to the module's build.gradle file

```
compile 'fi.vtt.nubomedia:utilities-android:1.0.0'
compile 'fi.vtt.nubomedia:jsonrpc-ws-android:1.0.4'
compile 'fi.vtt.nubomedia:webrtcpeer-android:1.0.0'
```

Support is provided through the Nubomedia VTT Public Mailing List available at https://groups.google.com/forum/#!forum/nubomedia-vtt

CHAPTER 4

License

Nubomedia Android client is distributed as Open Source Software basing BSD-license.