Kurento Room Documentation

Release 6.6.1-dev

kurento.org

Mar 27, 2020

Contents


KURENTO

CHAPTER 1

Kurento Tree Description

Kurento Tree is a project that allows developers to build video broadcasting web applications over internet. It is developed using WebRtc technology and Kurento Media Server.

Kurento Tree project is formed by a server and two clients, a Java client and a JavaScript client. There are also two demonstration applications are available that makes use of this project to enable users to see the webcam of other user.

1.1 Kurento Tree Server

The Tree Server (**kurento-tree-server**) is a project designed to be deployed and controlled by clients. We provide a Java client and a JavaScript client specifically designed to browsers (it uses several browser-only APIs). In any case, other clients can be implemented if follow the Json-Rpc over websocket protocol. Tree Server uses a Kurento Media Server to provide WebRtc media handling to clients. For this reason, a Kurento Media Server have to be available and configured in Tree Server config file.

1.2 Kurento Tree Java Client

The Java Tree Client (**kurento-tree-client**) implements a Tree Client to be used from Java applications. It is designed to be used in Java web applications or Android applications. The client only contains an implementation of Json-Rpc websocket protocol implementated by Tree server. It does not contain any functionality to control video players or video capturing from webcams.

1.3 Kurento Tree JavaScript Client

The JavaScript Tree Client (**kurento-tree-client-js**) implements a Tree Client to be used in Single Page Applications (SPA). It contains the main file `KurentoTree.js` to connect to Tree Server. Besides the Json-Rpc websocket protocol, it uses several libraries to control WebRtc and player HTML5 APIs in the browsers. This allows to developer to focus in app functionality hidding low level details.

1.4 Kurento Tree Demo applications

We provide two demos in this repository to show how to use the two provided clients.

1.4.1 Demo using JavaScript client

The demo using Kurento Tree JavaScript client is located in the project **kurento-tree-demo-spa**. It allows a user to broadcast his webcam and any other user can see it.

From a technical point of view, this demo is Single Page Application (SPA) implemented as a bunch of HTML, CSS and JS files. This files are served from a SpringBoot web server, but can be served with any http server. The JavaScript code uses Kurento Tree JavaScript client to communicate directly with Kurento Tree Server.

1.4.2 Demo using Java client

The demo using Kurento Tree Java client is located in the project **kurento-tree-demo**. It allows a user to broadcast his webcam and any other user can see it (in the same way as the another demo).

From a technical point of view, this demo is a SpringBoot web application that serves HTML, CSS and JS files. JavaScript logic communicates with SpringBoot server by means of websocket protocol. The SpringBoot server handle websocket messages and uses Java Kurento Tree Client to communicate with Kurento Tree Server. That is, there is no direct communication between JavaScript code and Kurento Tree Server. All communications is through SpringBoot server app.

CHAPTER 2

Kurento Tree Source Code

Kurento Tree is hosted on github:

https://github.com/Kurento/kurento-tree

The git repository contains a Maven project with the following modules:

- kurento-tree - Reactor project that contains all modules.
- kurento-tree/kurento-tree-server - Server implementation of webcam broadcasting service. It provides the a custom protocol based on Json-Rpc over WebSockets for the communications between tree clients and the server.
- kurento-tree/kurento-tree-client - Java library that uses WebSockets and Json-Rpc to interact with the server. Can be used to implement the client-side of a tree application in a Java Application Server or in an Android device.
- kurento-tree/kurento-tree-client-js - Javascript library that uses WebSockets and Json-Rpc to interact with the server. It is also a wrapper for several JS APIs to make easier to develop a broadcast application from a browser.
- kurento-tree/kurento-tree-demo - Demonstration project of kurento-tree. It is implemented with a Java web application that uses Java client to communicate with tree server. The browser communicates with Java web application using a custom websocket protocol.
- kurento-tree/kurento-tree-demo-spa - Demonstration project of kurento-tree. It is implemented as a full Single Page Application. This project serves static assets with a SpringBoot server, but can be changed to any other server. Tree server is controlled from browser with JavaScript tree client.
- kurento-tree/kurento-tree-test - Includes integration tests for the tree server.

CHAPTER 3

Kurento Tree Server Deployment

Currently, there are no binary releases of Kurento Tree Server. In order to deploy a new Tree Server it is needed to build it from sources.

3.1 Software Requirements

In order to execute Kurento Tree Server it is necessary the following software:

- Ubuntu 14.04
- JDK 7 or 8
- Kurento Media Server or connection with at least a running instance (to install follow the official guide)

3.2 Build from sources

To compile Kurento Tree Server from sources, there are more dependencies:

First of all, to build from sources it is necessary to download the sources from git repository. For it, you have to install git:

```
sudo apt-get install git
```

Then, you have to install maven:

```
sudo apt-get install maven
```

Finally, you need Bower, npm and nodejs:

```
curl -sL https://deb.nodesource.com/setup | sudo bash -
sudo apt-get install -y nodejs
sudo npm install -g bower
```

When all dependencies are installed, you can get the code from the official github repository and move into the code folder:

```
git clone https://github.com/Kurento/kurento-tree.git
cd kurento-tree
```

This will downloadd the latest code into your machine. *That code is normally for development, and depends on SNAPSHOT versions of artifacts that are not deployed in Maven Central*. You can either continue working with the nightly build (with caution, as the development version might be unstable), or get the latest release of the project. The steps that you have to follow for each one are different.

- Working with development versions: Please read the official project's documentation on how to work with development versions, specially this part about how to use our internal Archiva repo.
- Using a release version: This is the recommended approach, as release versions are more stable. You'll need to download the last release tag:

  ```
  git checkout -b release 6.6.1-SNAPSHOT
  ```

Once you got the desired version, you just need to execute:

```
mvn install -DskipTests=true
```

And that will install all generated artifacts in your local Maven repository.

To execute Kurento Tree server from the recent build, execute the following commands:

```
cd kurento-tree-server
mvn exec:java
```

Then a bunch of log messages will appear in the console. When the following message appears in the console:

```
Started KurentoTreeServerApp in 4.058 seconds (JVM running for 8.017)
```

You can use Kurento Tree Server in port 8890.

CHAPTER 4

Kurento Tree JavaScript Client

The developer of Kurento Tree applications can use this client when implementing the front-end part of a broadcasting application with Kurento Tree. It is not necessary to write any server-side logic.

4.1 Using this client

The library files needed to use this client are served by Kurento Tree Server in the path:

- `http://server-host:port/js/KurentoTree.js`

There are also several third party libraries that need to be "imported" in the HTML in order to user this Kurento Tree JavaScript client. This third party libraries are also served by Kurento Tree Server in the paths:

- `http://treeserver:port/bower_components/adapter.js/adapter.js`
- `http://treeserver:port/bower_components/eventEmitter/EventEmitter.js`
- `http://treeserver:port/js/kurento-utils.js`
- `http://treeserver:port/lib/sockjs.js`
- `http://treeserver:port/js/websocketwithreconnection.js`
- `http://treeserver:port/js/kurento-jsonrpc.js`
- `http://treeserver:port/js/jsonRpcClient.js`

For example, all necessary dependencies to create a SPA Kurento Tree application can be included in the HTML with the elements:

```
<script src="http://treeserver:port/bower_components/adapter.js/adapter.js"></script>
<script src="http://treeserver:port/bower_components/eventEmitter/EventEmitter.js"></
↪script>
<script src="http://treeserver:port/js/kurento-utils.js"></script>
<script src="http://treeserver:port/lib/sockjs.js"></script>
<script src="http://treeserver:port/js/websocketwithreconnection.js"></script>
```

(continues on next page)

(continued from previous page)

```
<script src="http://treeserver:port/js/kurento-jsonrpc.js"></script>
<script src="http://treeserver:port/js/jsonRpcClient.js"></script>
<script src="http://treeserver:port/js/KurentoTree.js"></script>
```

When these files are "included" in the HTML, the class `KurentoTree` is available to subsequent JavaScript files included in the page. `KurentoTree` class is the entry point of the Kurento Tree JavaScript Client.

4.2 KurentoTree usage

To connect to a Kurento Tree Server it is necessary to create an instance of `KurentoTree` class indicating the URL of the server:

```
var tree = new KurentoTree('http://treeserver:port/kurento-tree');
```

In background, a websocket connection is made between browser and Kurento Tree server.

To broadcast the user webcam it is necessary to execute a code similar to the following:

```
var treeName = ...

var mediaOptions = {
   localVideo : video,
   mediaConstraints : {
      audio : true,
      video : {
         mandatory : {
            maxWidth : 640,
            maxFrameRate : 15,
            minFrameRate : 15
         }
      }
   }
};

tree.setTreeSource(treeName, mediaOptions);
```

Where `localVideo` refers to a HTML div element when local video being to be broadcasted will be shown. If this option is not specified, no local video will be shown in the page.

To include a player showing the video that it is broadcasting another user, it is necessary to include a code similar to:

```
var treeName = ...

var mediaOptions = {
   remoteVideo : video
}

tree.addTreeSink(treeName, options);
```

Where `remoteVideo` refers to a HTML div element when remote video will be shown.

To stop any transmission (from emitter or receiver), `close()` method can be invoked:

```
tree.close();
```

CHAPTER 5

Kurento Tree Java Client

The developer of Kurento Tree applications can use a Java client to control Kurento Tree Server. This library gives more control to developer and allows to include authentication and authorization to broadcast applications. To create a broadcast applications with this Java client, it is necessary to implement a frond-end logic in JavaScript that communicates with Java web applications using websockets or another technology. Then, Java back-end will communicate with Kurento Tree Server using this client.

5.1 Using this client

This client can be obtained as a maven dependency with the following coordinates:

```
<dependency>
   <groupId>org.kurento</groupId>
   <artifactId>kurento-tree-client</artifactId>
   <version>6.6.1-SNAPSHOT</version>
</dependency>
```

With this dependency, the developer can use the class `org.kurento.tree.client.KurentoTreeClient` to control Kurento Tree Server.

5.2 KurentoTreeClient usage

To connect to a Kurento Tree Server it is necessary to create an instance of `KurentoTreeClient` class indicating the URL of the server:

```
KurentoTreeClient tree = new KurentoTreeClient("http://treeserver:port/kurento-tree");
```

In background, a websocket connection is made between Java app and Kurento Tree server.

To broadcast the user webcam it is necessary to execute a code similar to the following:

```
String treeName = ... // Select a unique name for the broadcast

tree.createTree(treeName);

String sdpOffer = ... // Create sdp offer in browser

String sdpAnswer = tree.setTreeSource(treeName, sdpOffer);

// Send sdpAnswer to browser to complete media negotiation
```

In this code, `sdpOffer` have to be created in the browser and communicated to Java web app using websocket or other technology. On the other hand, `sdpAnswer` have to be send to browser.

```
String treeName = ... // The broadcast name

String sdpOffer = ... // Create sdp offer in browser

TreeEndpoint treeEndpoint = tree.addTreeSink(treeName, sdpOffer);

String viewerId = treeEndpoint.getId(); // Id of this viewer

String sdpAnswer = treeEndpoint.getSdp();

// Send sdpAnswer to browser to complete media negotiation
```

In same way as before, `sdpOffer` and `sdpAnswer` have to be interchanged with JavaScript code in the browser to perform the media negotiation.

Kurento Tree support media negotiation with Trickle ICE . In this sense, besides SDP offer and answer interchange between browser and media server, it is necessary to interchange Ice candidates between peers.

```
while (true) {

   // Retrieve new ice candidate from server
   IceCandidateInfo cand = tree.getServerCandidate();

   if (candidateInfo == null) {
      // No more ice candidates. Connection closed
      break;
   }

   // Tree to which belongs this candidate
   String treeName = cand.getTreeId();

   // Viewer to which belongs this candidate (if null, it is tree source)
   String viewerId = cand.getSinkId();

   // Ice candidate info
   String candidate = cand.getIceCandidate().getCandidate());
   int sdpMLineIndex = cand.getIceCandidate().getSdpMLineIndex());
   String sdpMid = cand.getIceCandidate().getSdpMid());

   // Send candidate info to browser to complete media negotiation

}
```

When a new ice candidate is received from browser it is necessary to process it properly to achieve a successful media negotiation. This is done using the following code:

```
String treeName = ...
String viewerId = ... // null if is tree source

String candidate = ...
int sdpMLineIndex = ...
String sdpMid = ...

tree.addIceCandidate(treeName, viewerId,
    new IceCandidate(candidate, sdpMid, sdpMLineIndex));
```

5.3 Reference documentation

You can take a look to the *JavaDoc* of this client.

CHAPTER 6

Kurento Tree Demos

We provide two demos in this repository to show how to use the two provided clients.

6.1 Demo using JavaScript client

The demo using Kurento Tree JavaScript client is located in the project **kurento-tree-demo-spa**. It allows a user to broadcast his webcam and any other user can see it.

From a technical point of view, this demo is Single Page Application (SPA) implemented as a bunch of HTML, CSS and JS files. This files are served from a SpringBoot web server, but can be served with any http server. The JavaScript code uses Kurento Tree JavaScript client to communicate directly with Kurento Tree Server.

6.2 Demo using Java client

The demo using Kurento Tree Java client is located in the project **kurento-tree-demo**. It allows a user to broadcast his webcam and any other user can see it (in the same way as the another demo).

From a technical point of view, this demo is a SpringBoot web application that serves HTML, CSS and JS files. JavaScript logic communicates with SpringBoot server by means of websocket protocol. The SpringBoot server handle websocket messages and uses Java Kurento Tree Client to communicate with Kurento Tree Server. That is, there is no direct communication between JavaScript code and Kurento Tree Server. All communications is through SpringBoot server app.

6.2.1 Installation

To start this demo application you first need to download the source code as specified in *Kurento Tree Server Deployment* section.

First, you have to have a Kurento Media Server available. Please refer to official Kurento documentation for information on how to get it.

Then, you have to start Kurento Tree Server using the following commands:

```
mvn install -DskipTests=true
cd kurento-tree-server
mvn exec:java
```

These commands will start Kurento Tree Server assuming that Kurento Media Server is located in the same server. That is, Kurento Tree Server will try to connect to a KMS in `ws://localhost:8888/kurento`. If you need Kurento Tree Server connects to KMS with another URI, you have to exec it using the following command:

```
mvn exec:java -Dkms.url=ws://<kms_host>:8888/kurento
```

These commands start kurento-tree-server in `8890` port. You can change listening port using the parameter `-Dserver.port=<port>`.

The following step is start the Kurento Tree Demo app that uses Kurento Tree Server. You have to open another terminal and locate in the root of the source code repository, then you have to execute the following commands:

```
cd kurento-tree-demo
mvn install -DskipTests=true
mvn exec:java
```

These commands execute Kurento Tree Demo assuming that Kurento Tree Server is located in localhost. The demo app tries to connect to `wss://localhost:8890/kurento-tree`. If you want to connect to a Kurento Tree Server located in another host, you have to start it using the following command:

```
mvn exec:java -Dkts.ws.uri=wss://<kts_host>:8890/kurento-tree
```

Kurento Tree Demo app is started by default in port `8443`. If you want to change the port you can use the parameter `-Dserver.port=<port>`.

To use the demo application you have to open a browser pointing to `https://localhost:8443`. The main use case of Kurento Tree Demo is broadcast a webcam to other users. To test this scenario, open two browsers, first configure one browser as presenter and then configure the other browser as viewer and you should be the webcam from the presenter in the viewer.

CHAPTER 7

Kurento Tree WebSocket Protocol

The Kurento Tree server exposes a SockJS websocket at http://treeserver:port/kurento-tree, where the hostname and port depend on the current setup.

The exchanged messages between server and clients are JSON-RPC 2.0 requests and responses. The events are sent from the server to client as notifications (they don't require a response and they don't include an identifier).

7.1 WebSocket messages

7.1.1 1 - Create tree

Request to create a new tree in Tree server. It is send by clients to server.

- **Method**: createTree
- **Parameters**:
  - **treeId** - new tree Id (optional)
- **Example request**:

```
{
   "jsonrpc":"2.0",
   "id": 1,
   "method":"createTree",
   "params": {
       "treeId":"Channel 1"
    }
}
```

- **Server response (result)**:
  - **value** - new tree id
  - **sessionId** - id of the WebSocket session between the client and the server

- **Example response**:

```
{
   "jsonrpc":"2.0",
   "id": 1,
   "result": {
       "value": "Channel 1",
       "sessionId":"dv41ks9hj761rndhcc8nd8cj8q"
   }
}
```

7.1.2 2 - Set tree source

Request to configure the emitter (source) in a broadcast session (tree). It is send by clients to server.

- **Method**: setTreeSource
- **Parameters**:
  - **treeId:** tree id to configure the source
  - **offerSdp:** Offer SDP created in the WebRtc peer that wants to broadcast media to viewers
- **Example request**:

```
{
   "jsonrpc":"2.0",
   "id": 2,
   "method":"setTreeSource",
   "params": {
       "treeId":"Channel 1",
       "offerSdp":"v=0....apt=100\r\n"
    }
}
```

- **Server response (result)**
  - **sdpAnswer:** SDP answer build by the the user's server WebRTC endpoint
  - **sessionId:** id of the WebSocket session
- **Example response**:

```
{
   "jsonrpc":"2.0",
   "id": 2,
   "result": {
       "answerSdp": "v=0....apt=100\r\n",
       "sessionId":"dv41ks9hj761rndhcc8nd8cj8q"
   }
}
```

7.1.3 3 - Remove tree source

Request to remove the current emitter of a tree. It is send by clients to server.

- **Method**: remoteTreeSource
- **Parameters**:

  - **treeId:** tree id to remove the source

- **Example request**:

```
{
   "jsonrpc":"2.0",
   "id": 3,
   "method":"remoteTreeSource",
   "params": {
       "treeId":"Channel 1"
    }
}
```

- **Server response (result)**
  - **sessionId:** id of the WebSocket session
- **Example response**:

```
{
   "jsonrpc":"2.0",
   "id": 3,
   "result": {
       "sessionId":"dv41ks9hj761rndhcc8nd8cj8q"
   }
}
```

7.1.4 4 - Add tree sink

Request to add a new viewer (sink) to the tree. It is send by clients to server.

- **Method**: addTreeSink
- **Parameters**:
  - **treeId:** tree id to add a new viewer
  - **offerSdp:** Offer SDP created in the WebRtc peer that wants to receive media from tree
- **Example request**:

```
{
   "jsonrpc":"2.0",
   "id": 4,
   "method":"addTreeSink",
   "params": {
       "treeId":"Channel 1",
       "offerSdp":"v=0....apt=100\r\n"
    }
}
```

- **Server response (result)**
  - **sdpAnswer:** SDP answer build by the the user's server WebRTC endpoint
  - **sinkId:** New sink id. This id will be used to remove the sink and to exchange ice candidates.
  - **sessionId:** id of the WebSocket session
- **Example response**:

```
{
    "jsonrpc":"2.0",
    "id": 4,
    "result": {
        "answerSdp": "v=0....apt=100\r\n",
        "sinkId": "dab37f17-be82-4cd3-af20-edea13548254",
        "sessionId":"dv41ks9hj761rndhcc8nd8cj8q"
    }
 }
```

7.1.5 5 - Remove tree sink

Request to remove a previously connected sink (viewer). It is send by clients to server.

- **Method**: removeTreeSink
- **Parameters**:
  - **treeId:** tree id to remove the sink
  - **sinkId:** sink id to be removed
- **Example request**:

```
{
    "jsonrpc":"2.0",
    "id": 5,
    "method":"removeTreeSink",
    "params": {
        "treeId":"Channel 1",
        "sinkId": "dab37f17-be82-4cd3-af20-edea13548254"
     }
 }
```

- **Server response (result)**
  - **sessionId:** id of the WebSocket session
- **Example response**:

```
{
   "jsonrpc":"2.0",
   "id": 5,
   "result": {
       "sessionId":"dv41ks9hj761rndhcc8nd8cj8q"
   }
}
```

7.1.6 6 - Ice candidate

Notification sent form server to client when a new Ice candidate is received from Kurento Media Server. It is send by server to clients.

- **Method**: iceCandidate
- **Parameters**:
  - **treeId** - Tree id to which belongs this candidate

  - **sinkId** - Sink id to which belongs this candidate (if not present, this candidate is referred to the tree source)
  - **sdpMid** - Ice candidate sdpMid
  - **sdpMLineIndex** - Ice candidate sdpMLineIndex
  - **candidate** - Ice candidate string

- **Example message**:

```
{
   "jsonrpc":"2.0",
   "method":"iceCandidate",
   "params": {
       "treeId":"Channel 1",
       "sdpMid": "audio",
       "sdpMLineIndex": 0,
       "candidate": "candidate:2 2 UDP 2013266430 10.1.34.190 54211 typ host"
   }
}
```

7.1.7 7 - Add ice candidate

Request used to add a new ice candidate generated in the browser. It is send by clients to server.

- **Method**: addIceCandidate
- **Parameters**:
  - **treeId**: Tree id to which belongs this candidate
  - **sinkId**: Sink id to which belongs this candidate (if not present, this candidate is referred to the tree source)
  - **sdpMid**: Ice candidate sdpMid
  - **sdpMLineIndex** Ice candidate sdpMLineIndex
  - **candidate**: Ice candidate string
- **Example request**:

```
{
   "jsonrpc":"2.0",
   "id": 7,
   "method":"addIceCandidate",
   "params": {
      "treeId": "Channel 1",
      "sinkId": "dab37f17-be82-4cd3-af20-edea13548254",
      "sdpMid":"video",
      "sdpMLineIndex": 1,
      "candidate": "candidate:952163293 2 .... port 55404 generation 0",
   }
}
```

- **Server response (result)**
  - **sessionId:** id of the WebSocket session
- **Example response**:

```
{
   "jsonrpc":"2.0",
   "id": 7,
   "result": {
       "sessionId":"dv41ks9hj761rndhcc8nd8cj8q"
   }
}
```

7.1.8 8 - Remove tree

Request used to remove a tree. It is send by clients to server.

- **Method**: removeTree
- **Parameters**:
  - **treeId**: Tree id to be removed
- **Example request**:

```
{
   "jsonrpc":"2.0",
   "id": 8,
   "method":"removeTree",
   "params": {
      "treeId": "Channel 1"
   }
}
```

- **Server response (result)**
  - **sessionId:** id of the WebSocket session
- **Example response**:

```
{
   "jsonrpc":"2.0",
   "id": 8,
   "result": {
       "sessionId":"dv41ks9hj761rndhcc8nd8cj8q"
   }
}
```

CHAPTER 8

Kurento JavaDoc

8.1 Kurento Tree Client JavaDoc

- kurento-room-sdk