Decoded frames interception and hangling with OpenCV

- Overview
- Building OpenCV
- Interceptor implementation using OpenCV
- Testing

Overview

The decoded frames interception and handling feature allows to use a powerful images handling library OpenCV. The feature may be useful, for example, in case of AR implementatioin at server side and other similar cases, and is available since WCS build 5.2.1914.

OpenCV library should be built before implementing a custom Java class to integrate it.

Building OpenCV

Let's describe OpenCV 4.9.0 building on Centos 7, to provide glibc 2.17 compatibility.

1. Install JDK 8 and ANT

```
yum install openjdk-8-jdk ant
```

2. Detect JDK installation folder path and set the environment variable `JAVA_HOME`

```
export JAVA_HOME=$(echo $(readlink -f $(which javac)) | sed 's/\/bin\/javac//g')
```

3. Install CMake 3.6.2 or newer

```
wget https://cmake.org/files/v3.6/cmake-3.6.2.tar.gz
tar -zxf cmake-3.6.2.tar.gz
cd cmake-3.6.2
./bootstrap --prefix=/usr/local
make -j$(nproc)
make install
```

4. Install GCC 11 and switch to it to build OpenCV

```
yum install centos-release-scl
yum install devtoolset-11-gcc devtoolset-11-gcc-c++
scl enable devtoolset-11 -- bash
```

5. Download and unpack OpenCV source code

```
wget -O opencv-490.zip https://github.com/opencv/opencv/archive/4.9.0.zip
unzip opencv-490.zip
```

6. Set up the build

```
cd opencv-490
mkdir build
cd build
cmake ..
```

7. Build OpenCV

```
make -j $(nproc)
```

8. Copy native and Java modules to WCS installation folder

```
cp bin/opencv-490.jar /usr/local/FlashphonerWebCallServer/lib/custom
cp -r lib/* /usr/local/FlashphonerWebCallServer/lib/so
```

Interceptor implementation using OpenCV

A Java class implementing Java IDecodedFrameInterceptor interface should be developed. The function `frameDecoded()` of this class will receive decoded frames in YUV I420 format, convert them using OpenCV functions to RGB format, apply bluring effect, convert the result back to YUV I420 and rewrite the frame data as solid byte array, not by pixel, using `Frame.rewriteData()` method

TestInterceptor.java

```
// Package name should be strictly defined as com.flashphoner.frameInterceptor
package com.flashphoner.frameInterceptor;

// Import decoded frame interceptor interface
import com.flashphoner.sdk.media.IDecodedFrameInterceptor;
// Import YUV frame description
import com.flashphoner.sdk.media.YUVFrame;
// Import OpenCV classes
import org.opencv.core.Core;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;

public class TestInterceptor implements IDecodedFrameInterceptor {

    static {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    }

    @Override
    public void frameDecoded(String streamName, YUVFrame frame) {

        int width = frame.getWidth();
        int height = frame.getHeight();
        // Calculate chroma height of YUV frame
        int chromaHeight = height * 3 / 2;

        Mat srcYuvMatrix = new Mat(chromaHeight, width, CvType.CV_8UC1, frame.getData());

        // Create RGB matrix for manipulations
        Mat rgbMatrix = new Mat(height, width, CvType.CV_8UC3);
        Mat blurredMatrix = new Mat(height, width, CvType.CV_8UC3);

        // Convert YUV matrix to RGB matrix
        Imgproc.cvtColor(srcYuvMatrix, rgbMatrix, Imgproc.COLOR_YUV2RGB_I420);
        // Apply blur to matrix
        Imgproc.GaussianBlur(rgbMatrix, blurredMatrix, new Size(15, 15), 0);

        Mat dstYuvMatrix = new Mat(chromaHeight, width, CvType.CV_8UC1);
        // Convert RGB back to YUV
        Imgproc.cvtColor(blurredMatrix, dstYuvMatrix, Imgproc.COLOR_RGB2YUV_I420);
        byte[] dstData = new byte[chromaHeight * width];
        // get data from destination matrix
        dstYuvMatrix.get(0,0, dstData);

        // Current method rewrites full frame data with provided dstData
        // This method is recommended for a complete rewrite of the frame, rather than pixel-by-pixel rewriting
due to color mismatch
        frame.rewriteData(dstData);
    }

}
```

Then the class should be complied into byte code. To do this, create folder tree accordind to TestInterceptor class package name

```
mkdir -p com/flashphoner/frameInterceptor
```

and execute the command

```
javac -cp /usr/local/FlashphonerWebCallServer/lib/wcs-core.jar:/usr/local/FlashphonerWebCallServer/lib/custom
/opencv-490.jar ./com/flashphoner/frameInterceptor/TestInterceptor.java
```

Now, pack the code compiled to jar file

```
jar -cf testinterceptor.jar ./com/flashphoner/frameInterceptor/TestInterceptor.class
```

and copy this file to WCS custom libraries folder

```
cp testinterceptor.jar /usr/local/FlashphonerWebCallServer/lib/custom
```

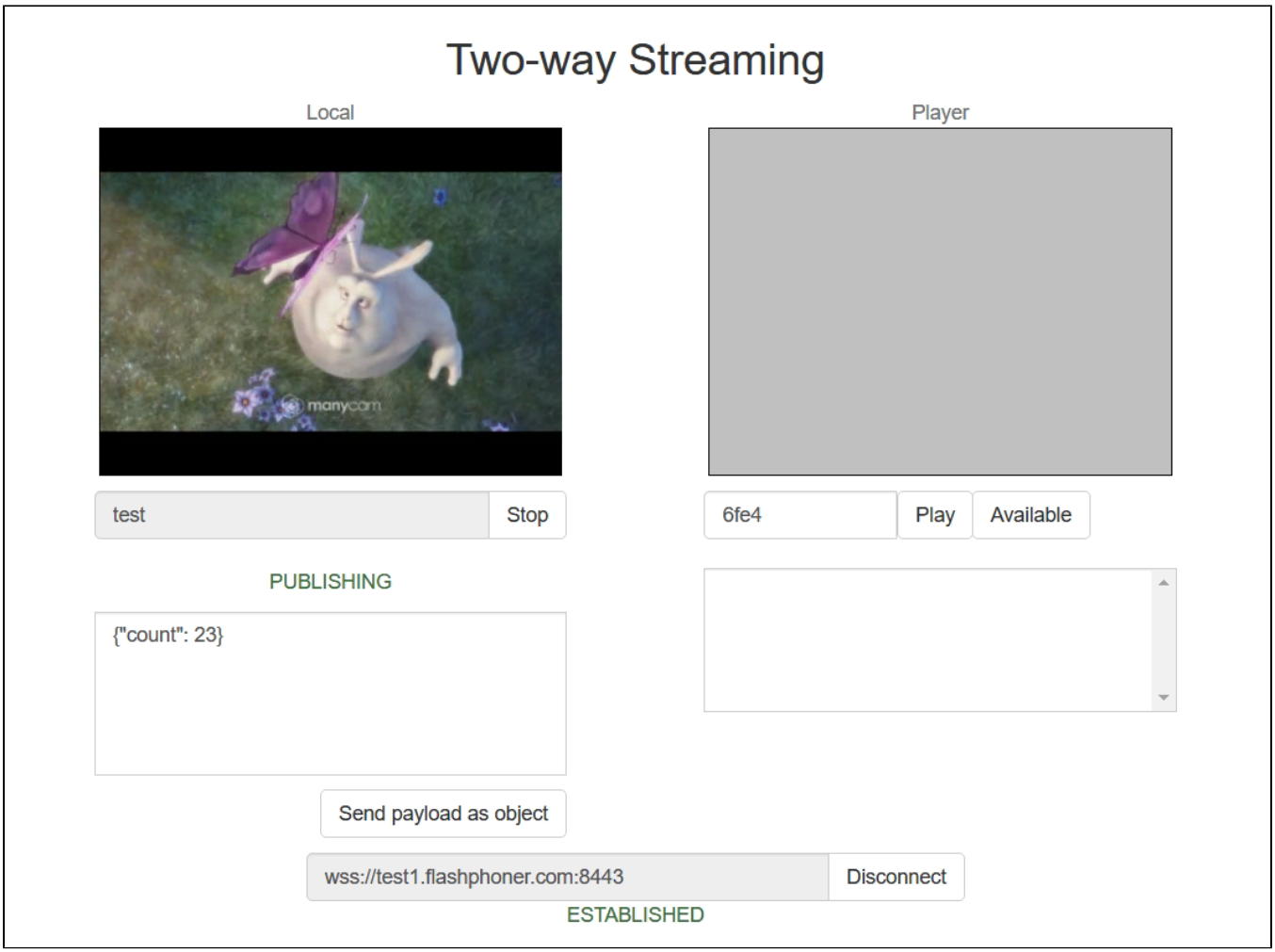
To use custom frames interceptor class, set its package name to the following parameter in flashphoner.properties

```
decoded_frame_interceptor=com.flashphoner.frameInterceptor.TestInterceptor
```

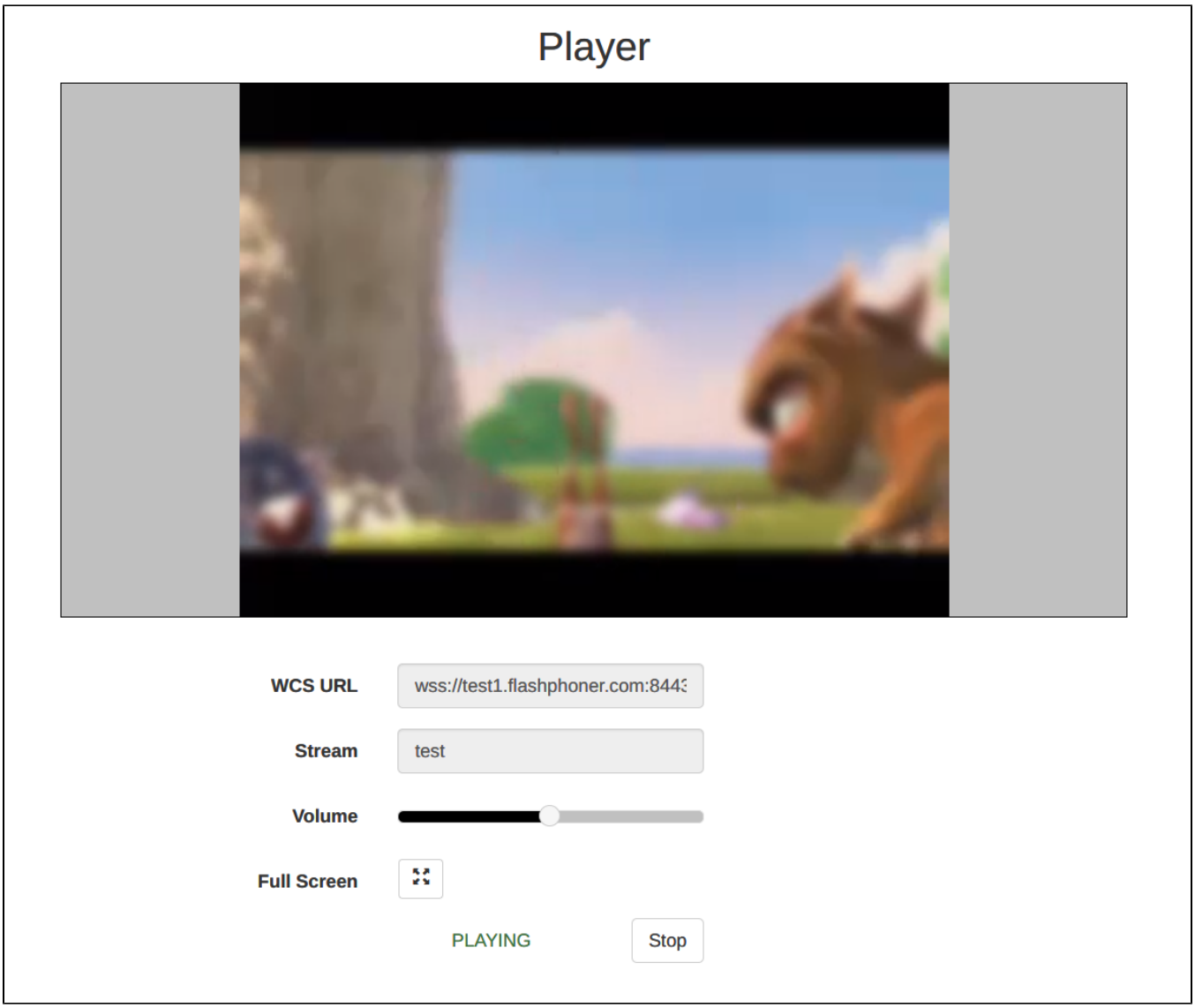
and restart WCS.

Testing

1. Publish a test stream in Two Way Streaming example



2. Play the stream in Player example with explicit resolution setting to enable transcoding, for examplehttps://test1.flashphoner.com:8444/client2/examples/demo/streaming/player/player.html?resolution=320x240,where test1.flashphoner.com is WCS server address



The picture will be blurred.