

Decoded frames interception and handling

Overview When [stream transcoding](./Stream_transcoding.en.md) is enabled, a stream published decoded picture frames in YUV format can be intercepted and changed pixel by pixel on server side. The frame changed will be then encoded and transferred to transcoder output stream as usual frame.

Interceptor implementation To intercept the decoded frames, Java class implementing `IDecodedFrameInterceptor` interface should be developed. The function `frameDecoded()` of this class will receive decoded frames in YUV format, for example!!! example "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;

/**
 * Custom decoded frames interceptor implementation example
 * The example draws a cross over the picture
 */
public class TestInterceptor implements IDecodedFrameInterceptor {

    // Constants to parse pixel
    private final int Y = 0;
    private final int U = 1;
    private final int V = 2;

    // Dark colored pixel
    private final byte[] DarkPixel = new byte []{42, -128, -128};

    /**
     * Function to handle decoded frame
     * @param streamName - stream name
     * @param frame - decoded YUV frame
     */
    @Override
    public void frameDecoded(String streamName, YUVFrame frame) {
        // Get frame height
        int frameHeight = frame.getHeight();
        // Get frame width
        int frameWidth = frame.getWidth();
        // Declare cross lines padding
        int PADDING = 4;
        // Define frame center
        int frameCenterX = frameWidth / 2;
        int frameCenterY = frameHeight / 2;
        // Define vertical line bounds
        int leftBound = frameCenterX - PADDING;
        int rightBound = frameCenterX + PADDING;
        // Define horizontal line bounds
        int topBound = frameCenterY - PADDING;
        int bottomBound = frameCenterY + PADDING;

        // Walk through the frame pixels and draw a cross
        for (int x = 0; x < frameWidth; x++) {
            for (int y = 0; y < frameHeight; y++) {
                if (validateCoord(x, leftBound, rightBound) || validateCoord(y, topBound, bottomBound)) {
                    // Read the pixel
                    byte[] pixel = frame.readPixel(x, y);
                    // Modify the pixel
                    pixel[Y] = DarkPixel[Y];
                    pixel[U] = DarkPixel[U];
                    pixel[V] = DarkPixel[V];
                    // Write the pixel back
                    frame.writePixel(x, y, pixel);
                }
            }
        }
    }

    /**
     * Helper function to validate pixel drawing
     * @param coord - pixel coordinate
     */
}
```

```

* @param low - low coordinate bound
* @param high - high coordinate bound
* @return true if coordinate is valid
*/
private boolean validateCoord(int coord, int low, int high) {
    return (coord > low && coord < high);
}
}

```

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

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

and execute the command

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

Now, pack the code compiled to jar file

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

and copy this file to WCS libraries folder

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

To use custom frames interceptor class, set its package name to the following parameter in [flashphoner.properties] (`../Working_with_the_server/Core_settings/Settings_file_flashphoner.properties.en.md`) file

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

and restart WCS.

A separate folder for custom Java libraries Since build [5.2.1512]

(<https://flashphoner.com/downloads/builds/WCS/5.2/FlashphonerWebCallServer-5.2.1512.tar.gz>), custom layout Java libraries (jar files) should be placed to the folder `/usr/local/FlashphonerWebCallServer/lib/custom`

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

This folder is kept while updating WCS to a newer builds. A jar files do not need to be copied again after updating. ## Testing 1. Publish a test stream in Two Way Streaming

example `https://test1.flashphoner.com:8444/client2/examples/demo/streaming/two_way_streaming/two_way_streaming.html`, where `test1.flashphoner.com` is WCS server address {width="800"} 2. Play the stream in Player example with explicit resolution setting to enable transcoding, for example `https://test1.flashphoner.com:8444/client2/examples/demo/streaming/player/player.html?resolution=320x240`, where `test1.flashphoner.com` is WCS server address {width="800"} The pixels changed will be in the stream picture.