Flash Streaming

Example of a streaming player in a native Flash / Flex application

The following example shows how to play a video stream while simultaneously publishing another stream by using a client Flash application that can be run by a simple swf file. Streaming in this example can work via two protocols: `rtmp://` and `rtmfp://`

The screenshot displays an example Flash application that sends a video stream to the server and plays the video from the server via the RTMFP protocol.

The `Server` field contains the RTMFP address of the server to establish connection. The `Publish` field contains the name of the video stream to send to the server from the web camera. The `Play` field contains the name of the video stream to play from the server. Below you can specify additional parameters of capturing and sending a video stream:

- width and height of a frame
- fps
- quality
- key frames per second
- is audio or video present in the video stream being sent

Example files

This example is a compiled SWF file embedded to an HTML page using Flex / ActionScript3 and MXML and accessible at:

/usr/local/FlashphonerWebCallServer/client2/examples/demo/streaming/flash_client/streaming.html

- streaming.html - example page
- streaming/bin/streaming.swf - application file

Working with the source code of the example

To analyze the code, let's take this version of the `streaming.mxml` file with the hash `90eb5073687bbe63bbb7467de3f3be4f3fe33802` located here. The result of compiling the `streaming.mxml` file is the example application `streaming.swf`. The compiled swf and the source code are available for download in the corresponding build 0.5.3.1894.

1. Accessing camera and microphone

Right after the application is loaded, we get access to the web camera and the microphone

line 42

line 44

```
cam = Camera.getCamera();
videoMy.attachCamera(cam);
mic = Microphone.getEnhancedMicrophone();
```

2. Applying camera and microphone settings

line 76

line 84

Recommended setting for the web camera:

- A motion sensibility threshold to send video:cam.setMotionLevel(0,2000);

Recommended setting for the microphone:

- Speex audio codec:mic.codec = SoundCodec.SPEEX;
- The number of frames per packet:mic.framesPerPacket=1;
- A sound threshold to send audio:mic.setSilenceLevel(0,2000);

```
private function initCam():void{
    cam.setMode(int(camWidth.text),int(camHeight.text),int(camFPS.text),true);
    cam.setQuality(0,int(camQuality.text));
    cam.setKeyFrameInterval(int(camKeyFrame.text));
    cam.setMotionLevel(0,2000);
    Logger.info("Cam initizlized "+cam.width+"x"+cam.height);
}

private function initMic():void{
    var options:MicrophoneEnhancedOptions = new MicrophoneEnhancedOptions();
    options.mode = MicrophoneEnhancedMode.FULL_DUPLEX;
    options.echoPath = 128;
    options.nonLinearProcessing = true;
    mic.codec = SoundCodec.SPEEX;
    mic.encodeQuality = 5;
    mic.framesPerPacket=1;
    mic.gain=50;
    mic.setSilenceLevel(0,2000);
    mic.enhancedOptions = options;
    Logger.info("Mic initialized");
}
```

3. Connecting to the server

line 103

Here we establish connection to the server and send `obj.appKey = "flashStreamingApp"`; This appKey tells the server, that it should deal with a Flash application, not with a Websocket/WebRTC client

```
private function connect():void{
    trace("connect");
    var url:String = connectUrl.text;
    nc = new NetConnection();
    //if (url.indexOf("rtmp") == 0){
    nc.objectEncoding = ObjectEncoding.AMF0;
    //}
    nc.client = this;
    nc.addEventListener(NetStatusEvent.NET_STATUS, handleConnectionStatus);
    var obj:Object = new Object();
    obj.login = generateRandomString(20);
    obj.appKey  = "flashStreamingApp";
    nc.connect(url,obj);
}
```

4. Stream publishing

Sending the stream to the server is performed in the `publish()` method of the example

line 165

```
if (publishAudio.selected){
    initMic();
    publishStream.attachAudio(mic);
    Logger.info("Init audio stream")
}
if (publishVideo.selected){
    initCam();
    publishStream.attachCamera(cam);
    addH264();
    Logger.info("Init video stream");
}
addListenerAndPublish
```

Directly before sending the stream, we set additional buffering parameters and parameters of the H.264 codec in `addH264()` and `addListenerAndPublish()` methods

line 199

line 208

```
private function addListenerAndPublish():void{
    publishStream.videoReliable=true;
    publishStream.audioReliable=false;
    publishStream.useHardwareDecoder=true;
    publishStream.addEventListener(NetStatusEvent.NET_STATUS, handleStreamStatus);
    publishStream.bufferTime=0;
    publishStream.publish(publishStreamName.text);
}

public function addH264():void{
    var videoStreamSettings:H264VideoStreamSettings = new
H264VideoStreamSettings();
    videoStreamSettings.setProfileLevel(H264Profile.MAIN,H264Level.LEVEL_3_1);
    publishStream.videoStreamSettings = videoStreamSettings;
}
```

5. Stream playback

Playing the stream starts after the `play()` method is invoked

line 223

```
private function play():void{
    if (playStreamName.text == "") {
        playStatus.text = "Empty stream name";
        playStatus.setStyle("color","#ff0000");
        return;
    }
    playStatus.setStyle("color","#000000");
    Logger.info("play");
    subscribeStream = new NetStream(nc);
    addListenerAndPlay();
}
```

All parameters and buffer sizes are set directly before playing in the `addListenerAndPlay()` method

line 244

```
private function addListenerAndPlay():void{
    subscribeStream.videoReliable=true;
    subscribeStream.audioReliable=false;
    subscribeStream.useHardwareDecoder=true;
    subscribeStream.addEventListener(NetStatusEvent.NET_STATUS,
handleSubscribeStreamStatus);
    subscribeStream.bufferTime=0;
    var soundTransform:SoundTransform = new SoundTransform();
    soundTransform.volume=0.7;
    subscribeStream.soundTransform = soundTransform;
    subscribeStreamObject = createStreamObject();
    subscribeStream.play(playStreamName.text);
    videoFarEnd.attachNetStream(subscribeStream);
    videoFarEnd.width = 320;
    videoFarEnd.height = 240;
    videoFarEnd.visible = true;
}
```