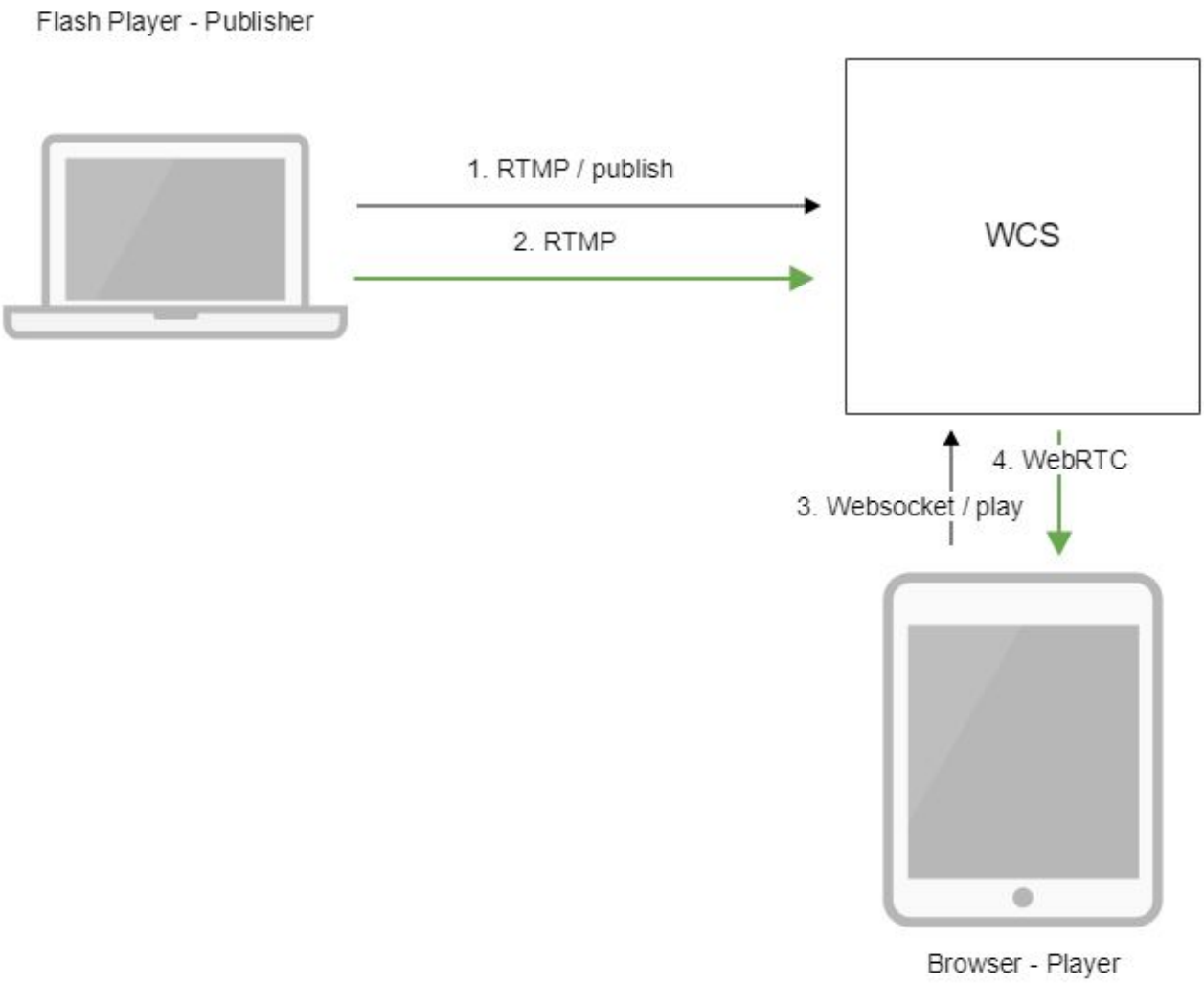
С помощью Flash Player по RTMP

Описание

Поддерживаемые платформы

	Adobe Flash
Windows	+
Mac OS	+
Linux	+

Схема работы



1. Flash Player соединяется с сервером по протоколу RTMP и отправляет команду publish.
2. Flash Player захватывает микрофон и камеру и отправляет RTMP поток на сервер.
3. Браузер устанавливает соединение по Websocket и отправляет команду play.
4. Браузер получает WebRTC поток и воспроизводит этот поток на странице.

Краткое руководство по тестированию

Захват видеопотока с веб-камеры и подготовка к его трансляции

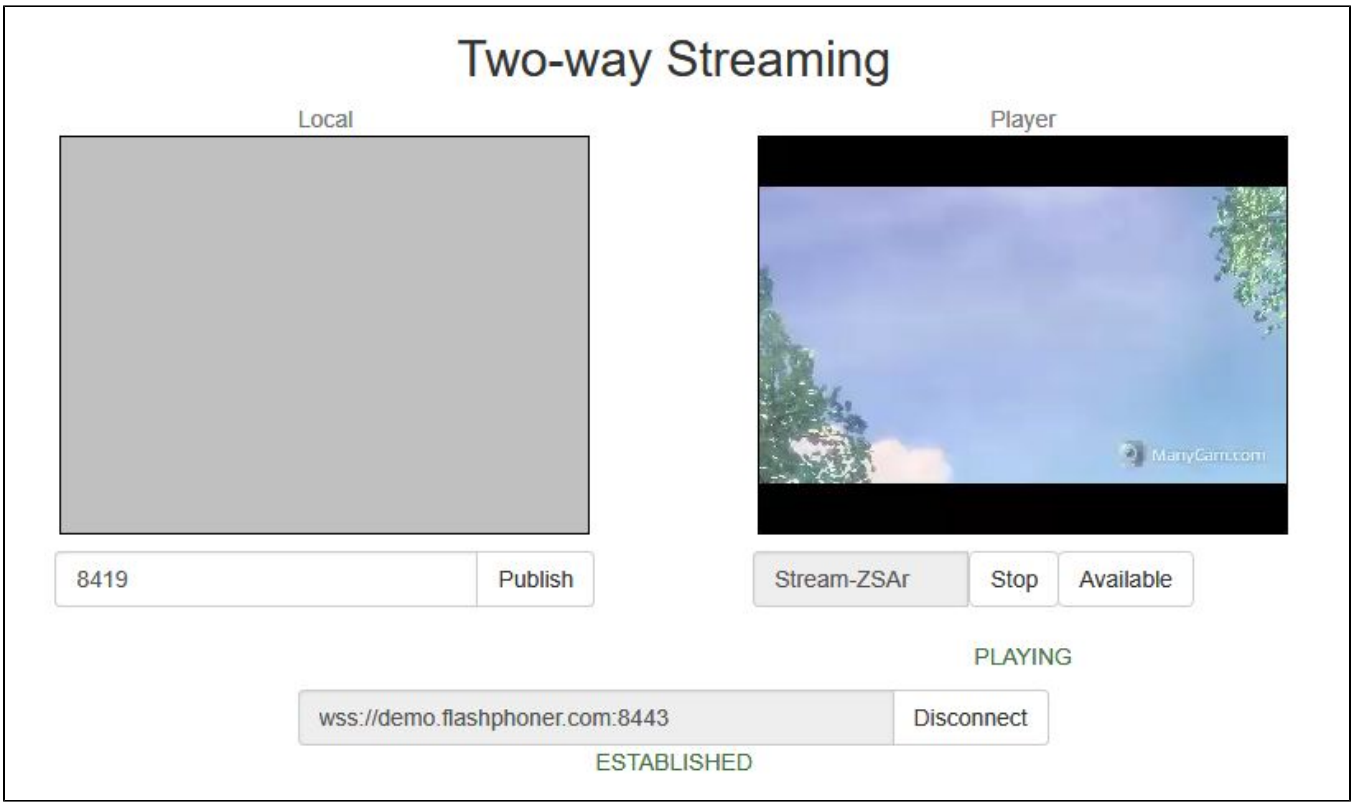
1. Для теста используем демо-сервер demo.flashphoner.com и веб-приложение Flash Streaming в браузере Internet Explorer

https://demo.flashphoner.com/client2/examples/demo/streaming/flash_client/streaming.html

Установите Flash Player. Откройте страницу веб-приложения и разрешите запуск Flash в браузере:

2. Нажмите кнопку "Login". При появлении надписи "Connected" нажмите кнопку Start в поле Publish:

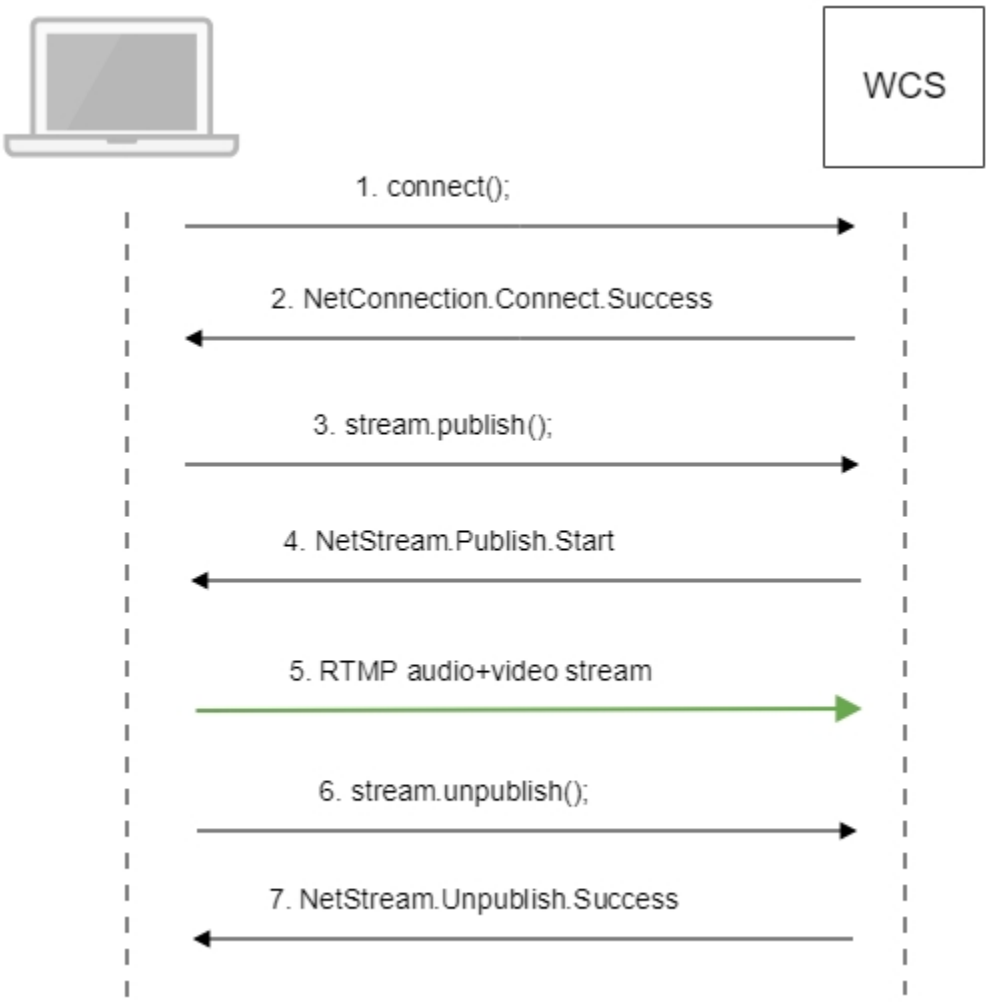
3. Чтобы убедиться, что трансляция идет успешно, откройте веб-приложениеTwo Way Streamingв отдельном окне, нажмите Connect и укажите идентификатор потока, затем нажмите Play



Последовательность выполнения операций (Call Flow)

Ниже описана последовательность вызовов при использовании примера Flash Streaming

streaming.mxml



1. Установка соединения с сервером.

connect();code

```
        private function connect():void{
                var url:String = StringUtil.trim(connectUrl.text);
                Logger.info("connect " + url);
                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);
        }
```

2. Получение от сервера события, подтверждающего успешное соединение.

NetConnection.Connect.Successcode

```
        private function handleConnectionStatus(event:NetStatusEvent):void{
                Logger.info("handleConnectionStatus: "+event.info.code);
                if (event.info.code=="NetConnection.Connect.Success"){
                        Logger.info("near id: "+nc.nearID);
                        Logger.info("far id: "+nc.farID);
                        Logger.info("Connection opened");
                        disconnectBtn.visible = true;
                        connectBtn.visible = false;
                        playBtn.enabled = true;
                        publishBtn.enabled = true;
                        setConnectionStatus("CONNECTED");
                } else if (event.info.code=="NetConnection.Connect.Closed" || event.info.code=="NetConnection.
Connect.Failed"){
            ...
                }
        }
```

3. Публикация потока.

stream.publish();code

```
        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);
        }
```

4. Получение от сервера события, подтверждающего успешную публикацию потока.

NetStream.Publish.Startcode

```
        private function handleStreamStatus(event:NetStatusEvent):void{
                Logger.info("handleStreamStatus: "+event.info.code);
                switch (event.info.code) {
            ...
                        case "NetStream.Publish.Start":
                                setPublishStatus("PUBLISHING");
                                publishBtn.visible = false;
                                unpublishBtn.visible = true;
                                break;
                }
        }
```

5. Отправка аудио-видео потока по RTMP

6. Остановка публикации потока.

stream.unpublish();code

```
        private function unpublish():void{
                Logger.info("unpublish");
                if (publishStream!=null){
                        publishStream.close();
                }
                videoFarEnd.clear();
        }
```

7. Получение от сервера события, подтверждающего остановку публикации потока.

NetStream.Unpublish.Successcode

```
        private function handleStreamStatus(event:NetStatusEvent):void{
                Logger.info("handleStreamStatus: "+event.info.code);
                switch (event.info.code) {
            ...
                        case "NetStream.Unpublish.Success":
                                publishStream.removeEventListener(NetStatusEvent.NET_STATUS,
handleStreamStatus);
                                publishStream=null;
                                setPublishStatus("UNPUBLISHED");
                                publishBtn.visible = true;
                                unpublishBtn.visible = false;
                                break;
            ...
                }
        }
```

Указание серверного приложения при публикации RTMP-потока

При публикации RTMP-потока на WCS сервере можно указать приложение, которое будет использовано для взаимодействия с бэкенд-сервером, при помощи параметра в URL потока:

```
rtmp://host:1935/live?appKey=key1/streamName
```

Здесь

- host - WCS-сервер;
- key1 - ключ приложения на WCS-сервере;
- streamName - имя потока на сервере

По умолчанию, если ключ приложения не указан, используется стандартное приложение `flashStreamingApp`.

Кроме того, приложение может быть указано явным образом как часть URL. Для этого необходимо в файлеflashphoner.properties установить настройку

```
rtmp_appkey_source=app
```

Тогда приложение должно быть указано в URL потока как

```
rtmp://host:1935/key1/streamName
```

В этом случае значение `live` также рассматривается, как имя приложения, поэтому при публикации потока

```
rtmp://host:1935/live/streamName
```

на WCS сервере должно быть определено приложение `live`.

Известные проблемы

1. При публикации потока, содержащего только звук, и воспроизведении этого потока по WebRTC в браузере, звук не проигрывается.

Симптомы: нет звука при воспроизведении audio-only потока, опубликованного Flash клиентом

Решение: изменить настройку SDP для потоков, публикуемых с Flash клиентов,flash_handler_publish.sdpна сервере, оставив только аудио

```
v=0
o=- 1988962254 1988962254 IN IP4 0.0.0.0
c=IN IP4 0.0.0.0
t=0 0
a=sdplang:en
m=audio 0 RTP/AVP 97 8 0
a=rtpmap:97 SPEEX/16000
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=sendonly
```

2.При публикации потока при помощи Flash Streaming, воспроизведении этого потока в iOS Safari по WebRTC и одновременной публикации потока по WebRTC из Safari перестает воспроизводиться звук.

Симптомы:

а) Публикация потока stream1 из приложения Flash Streaming в браузере Chrome под Windows
б) Воспроизведение потока stream1 на iOS Safari в приложении Two Way Streaming. Звук и видео воспроизводятся нормально.
в) Публикация потока из iOS Safari в приложении Two Way Streaming. Воспроизведение звука пропадает.
г) Остановка публикации в iOS Safari. Воспроизведение звука восстанавливается.

Решение: отключить алгоритм избегания транскодинга (Avoid Transcoding Alhorithm) на сервере при помощи опции в файле flashphoner.properties

```
disable_rtc_avoid_transcoding_alg=true
```

3. Обработка параметров, указанных в URL потока, не поддерживается при публикации с помощью Flash клиента.