

В браузере с помощью Flash Player по RTMP

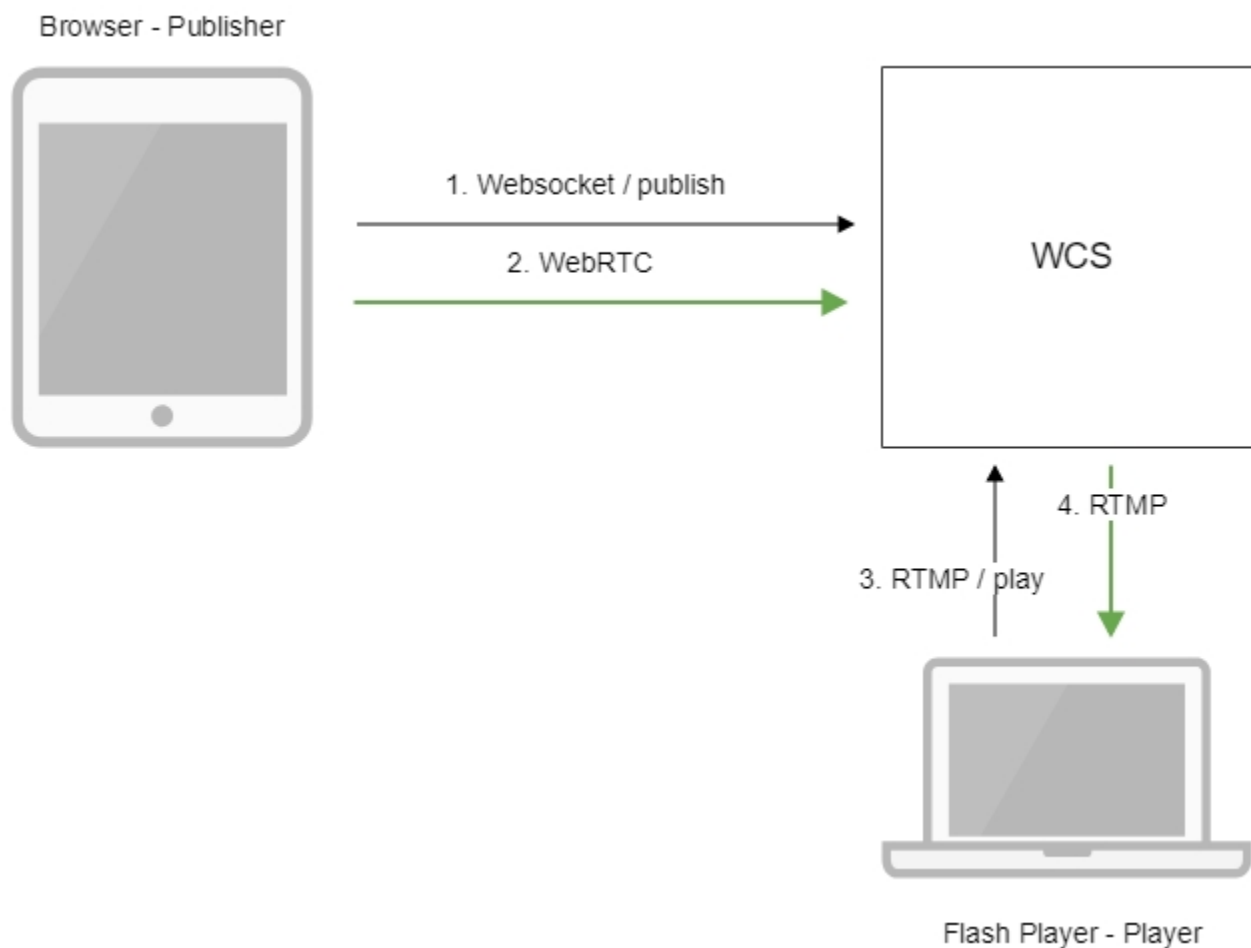
- [Описание](#)
 - [Поддерживаемые платформы](#)
 - [Схема работы](#)
- [Краткое руководство по тестированию](#)
 - [Трансляция потока с веб-камеры на сервер и воспроизведение при помощи Flash Player](#)
- [Последовательность выполнения операций \(Call Flow\)](#)

Описание

Поддерживаемые платформы

	Adobe Flash
Windows	+
Mac OS	+
Linux	+

Схема работы



1. Браузер устанавливает соединение по Websocket и отправляет команду publish.
2. Браузер отправляет WebRTC поток на сервер

3. Flash Player соединяется с сервером по протоколу RTMP и отправляет команду play.
4. Flash Player получает RTMP поток с сервера.

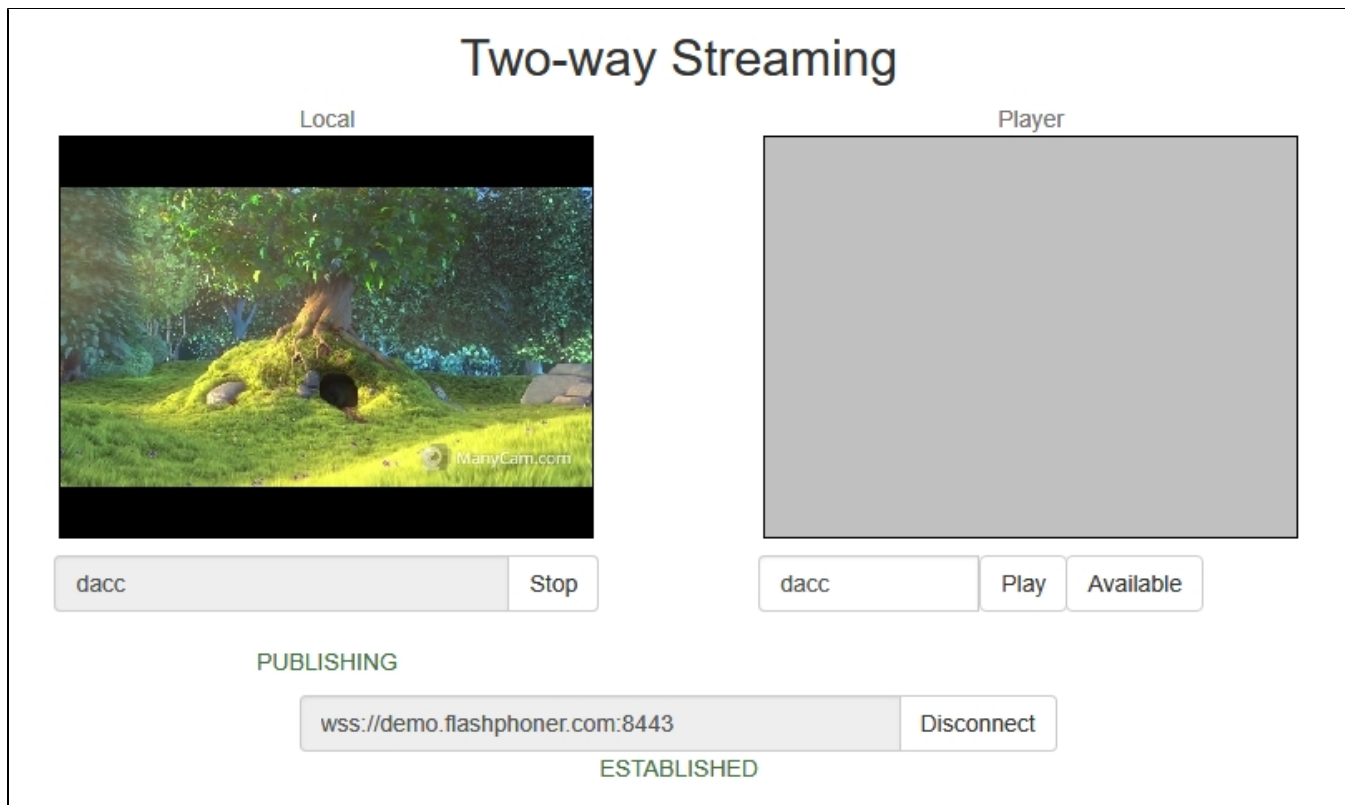
Краткое руководство по тестированию

Трансляция потока с веб-камеры на сервер и воспроизведение при помощи Flash Player

1. Для теста используем:

- демо-сервер demo.flashphoner.com:
- веб-приложение [Two Way Streaming](#) в браузере Chrome для публикации потока
- веб-приложение [Flash Streaming](#) в браузере Internet Explorer для воспроизведения потока

2. Откройте веб-приложение Two Way Streaming. Нажмите Connect, затем Publish. Скопируйте идентификатор потока:



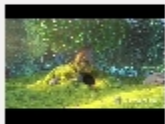
3. Установите Flash Player. Откройте страницу веб-приложения Flash Streaming и разрешите запуск Flash в браузере:

Flash Streaming

Server:

Publish

Play



☒ audio ☒ video

320	240	15	80	15
width	height	fps	quality	keyframe

4. Нажмите кнопку Login. При появлении надписи "Connected" укажите в поле Play идентификатор транслируемого потока:

Flash Streaming

Server:

CONNECTED

Publish

Play



☒ audio ☒ video

320	240	15	80	15
width	height	fps	quality	keyframe

5. Нажмите кнопку Start в поле Play. Начнется воспроизведение потока:

Flash Streaming

Server:
CONNECTED

Publish:

Play:
PLAYING



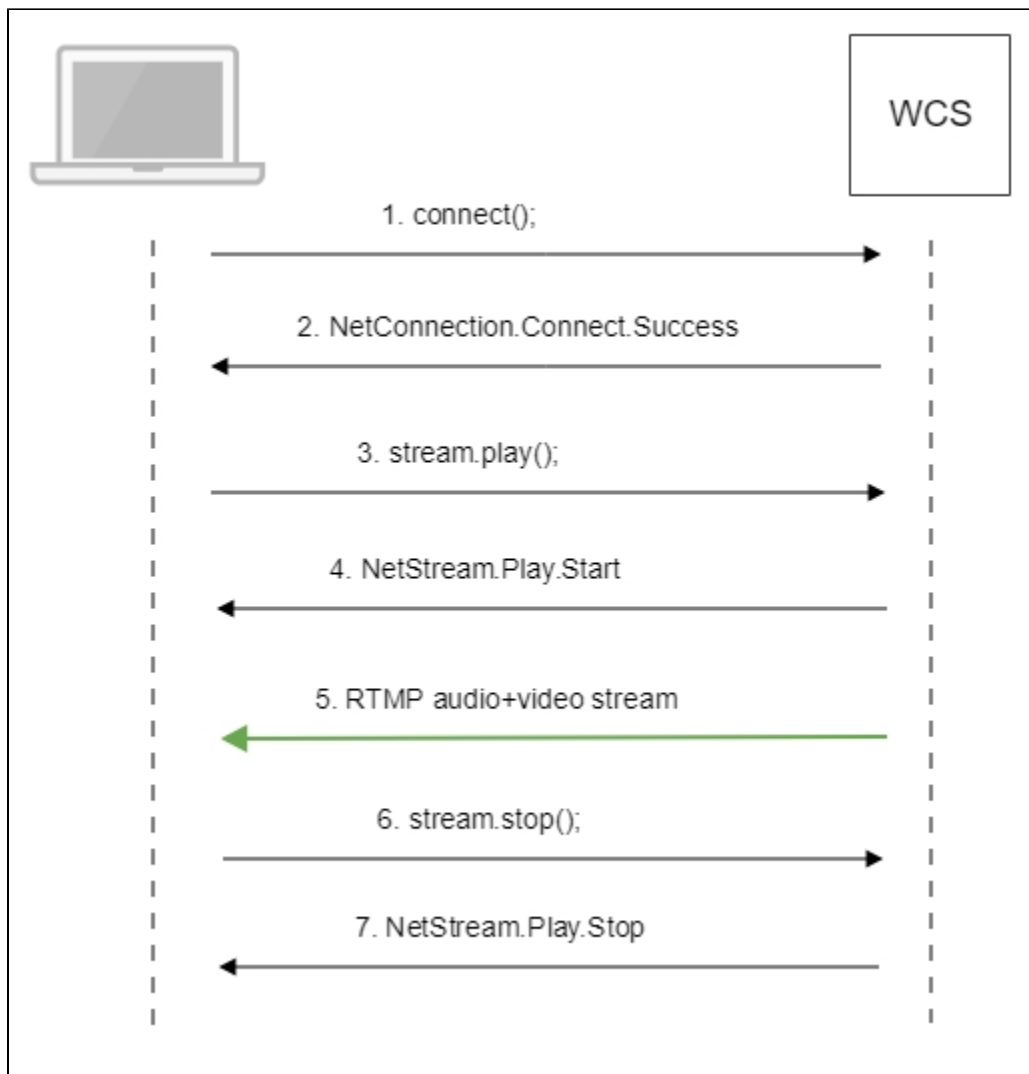
☒ audio ☒ video

320	240	15	80	15
width	height	fps	quality	keyframe

Последовательность выполнения операций (Call Flow)

Ниже описана последовательность вызовов при использовании примера Flash Streaming для воспроизведения потока

[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.Success`[code](#)

```

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. Воспроизведение потока.

stream.play();[code](#)

```

private function addListenerAndPlay():void{
    ...
    subscribeStreamObject = createStreamObject();
    subscribeStream.play(playStreamName.text);
    videoFarEnd.attachNetStream(subscribeStream);
    videoFarEnd.width = 320;
    videoFarEnd.height = 240;
    videoFarEnd.visible = true;
}

```

4. Получение от сервера события, подтверждающего успешное воспроизведение потока

NetStream.Play.Start[code](#)

```

private function handleSubscribeStreamStatus(event:NetStatusEvent):void{
    Logger.info("handleSubscribeStreamStatus: "+event.info.code);
    switch (event.info.code) {
        case "NetStream.Play.PublishNotify":
        case "NetStream.Play.Start":
            setPlayStatus("PLAYING");
            playBtn.visible = false;
            stopBtn.enabled = true;
            stopBtn.visible = true;
            break;
        ...
    }
}

```

5. Прием аудио-видео потока по RTMP

6. Остановка воспроизведения потока.

stream.close();[code](#)

```

private function stop():void{
    if (subscribeStream != null) {
        stopBtn.enabled = false;
        subscribeStream.close();
        subscribeStream = null;
    }
    subscribeStreamObject = null;
    videoFarEnd.visible = false;
}

```

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

NetStream.Play.Stopcode

```

private function handleSubscribeStreamStatus(event:NetStatusEvent):void{
    Logger.info("handleSubscribeStreamStatus: "+event.info.code);
    switch (event.info.code) {
        ...
        case "NetStream.Play.UnpublishNotify":
        case "NetStream.Play.Stop":
            setPlayStatus("STOPPED");
            playBtn.enabled = true;
            playBtn.visible = true;
            stopBtn.visible = false;
            break;
        ...
    }
}

```