Снятие превью трансляции в виде PNG

Описание

WCS предоставляет возможность снятия превью публикуемого потока при помощи REST-вызовов, а также при помощиJavaScript API.

Поддерживаемые протоколы

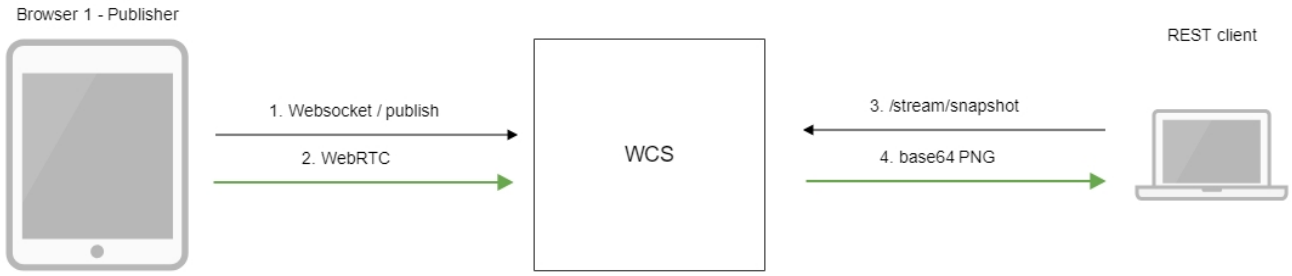
- WebRTC
- RTMP
- RTSP

Поддерживаемые форматы превью

- PNG

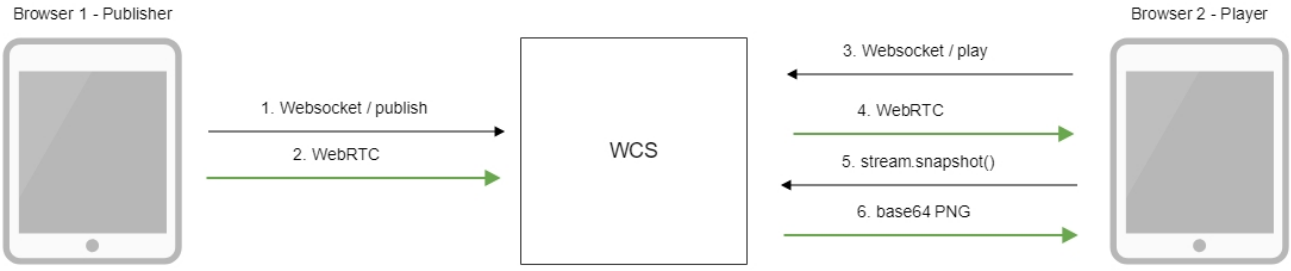
Схема работы

1: С использованием REST-запроса



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду publish.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток на сервер.
3. REST-клиент отправляет WCS REST-запрос /stream/snapshot
4. REST-клиент получает ответ с превью потока, закодированным в base64

2: С использованием JavaScript API



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду publish.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток на сервер.
3. Второй браузер устанавливает соединение также по Websocket и отправляет команду play.
4. Второй браузер получает WebRTC поток и воспроизводит этот поток на странице.
5. Второй браузер вызывает stream.snapshot() для снятия превью
6. Второй браузер получает ответ с превью потока, закодированным в base64

REST-вызовы

WCS-сервер поддерживает REST-метод /stream/snapshot для снятия превью:

REST-запрос должен быть HTTP/HTTPS POST запросом в таком виде:

- HTTP:http://streaming.flashphoner.com:8081/rest-api/stream/snapshot
- HTTPS:https://streaming.flashphoner.com:8444/rest-api/stream/snapshot

Здесь:

- streaming.flashphoner.com - адрес WCS-сервера
- 8081 - стандартный REST / HTTP порт WCS-сервера
- 8444 - стандартный HTTPS порт
- rest-api - обязательная часть URL
- /stream/snapshot - используемый REST-метод

REST-методы и статусы ответа

REST-метод	Пример тела REST-запроса	Пример тела REST-ответа	Статусы ответа

/stream/snapshot	{
	"streamName":"64966f33"
	}
		{
		"data":
		"iVBORw0KGgoAAAANSUhEUgAAAUAAAADwCAYAAABxLb1rAAAACXBIWXMAAAAAAAAAAQCEeRdzAAAQAElEQV
		R4nOzd95Pcd37feVjhrFKwXT6fr+r+gPvh6nx1V5bvLMt3liXbkiWtrF1v1O5qVxu4icucwUyABAMIAgSIQ
		OQcBzlnzAADDIDJuadzzjmH173f7+/302l6gO4GiCHIZtWjeqZnejAAu5
		/9+XzD57tgwYIF6Ojo6PiSmvdfoKOjo2O+zPsv0NHR0TFf5v0X6Ojo6Jgv8
		/4LdHR0dMyXef8FOjo6OubLvP8CHR0dHfNl3n+Bjo6Ojvky779AR0dHx3yZ91+go6OjY77M+y/Q0dHRMV/m
		/Rfo6OjomC/z/gt0dHR0zJd5/wU6Ojo65su8
		/wIdHR0d82Xef4GOjo6O+TLvv0BHR0fHfJn3X6Cjo6Njvsz7L9DR0dExX+b9F+jo6OiYL/P+C3R0dHTMl3n
		/BTo6Ojrmy7z/Ah0dHR1t+Ue636rzGzr19Tv8jPn/S3R0dHS04z4E8HfR0dHR8XD5DaGC94
		/rdALY0dHxBXafAlj/wI6OjofPb7foXh//edcJYEfHl4gaCd0tDPXf0+rjHxbNT4F/6
		/fR0dHxkPsNnhb+DvnNO7zYf1P7nt/8vTYf/3nzj8hvk/+B/GPt76DIfey3dHP8jN/8g/8JHR0dD7d/9Hv
		/Ixb87j/Hgt/5ZzSt+6dY8D/8Eyz47T+o4Pv4a/Q9v/H7/6L1x3+e8O+m/134d/+tf/Iv8dv/9H/G7/zz
		/6Xs9/7ZvxS//8/+hfiDP/gD/P7v/z5+7/d+D7/7u7+L3/md3xELVl6PoaOj4"
		}
			200 - Превью зафиксирован
			
			404 - Поток не найден

Параметры

Имя параметра	Описание	Пример
streamName	Уникальное имя потока	64966f33
data	Файл превью в base64-кодировке	iVBORw0KGgoAAAANSUhEUgAAAUAAAADwCAYAAABxLb1rAAAACXBIWXMAAAAAAAAAAQCEeRdzAAAQA

Отправка REST-запроса к WCS-серверу

Для отправки REST-запроса к WCS-серверу необходимо использоватьREST-клиент.

JavaScript API

Для снятия превью трансляции при помощи WebSDK предназначен метод snapshot объекта Stream. Пример использования метода приведен в веб-приложении Stream Snapshot для публикации потока и снятия превью.

stream-snapshot.html

stream-snapshot.js

1. Из опубликованного потока создается новый поток

код:

```
function snapshot(name) {
    setSnapshotStatus();
    var session = Flashphoner.getSessions()[0];
    session.createStream({name: name}).on(STREAM_STATUS.SNAPSHOT_COMPLETE, function(stream){
    ...
}
```

2. Вызывается метод snapshot()

код:

```
function snapshot(name) {
    setSnapshotStatus();
    var session = Flashphoner.getSessions()[0];
    session.createStream({name: name}).on(STREAM_STATUS.SNAPSHOT_COMPLETE, function(stream){
        ...
    }).snapshot();
}
```

3. При получении события SNAPSHOT_COMPLETE, функция stream.getInfo() возвращает превью, закодированный в base64

код:

```
function snapshot(name) {
    setSnapshotStatus();
    var session = Flashphoner.getSessions()[0];
    session.createStream({name: name}).on(STREAM_STATUS.SNAPSHOT_COMPLETE, function(stream){
        console.log("Snapshot complete");
        setSnapshotStatus(STREAM_STATUS.SNAPSHOT_COMPLETE);
        snapshotImg.src = ""+stream.getInfo();
        ...
}
```

4. Поток останавливается

код:

```
function snapshot(name) {
    setSnapshotStatus();
    var session = Flashphoner.getSessions()[0];
    session.createStream({name: name}).on(STREAM_STATUS.SNAPSHOT_COMPLETE, function(stream){
        ...
        stream.stop();
    }).on(STREAM_STATUS.FAILED, function(stream){
        setSnapshotStatus(STREAM_STATUS.FAILED);
        console.log("Snapshot failed, info: " + stream.getInfo());
    }).snapshot();
}
```

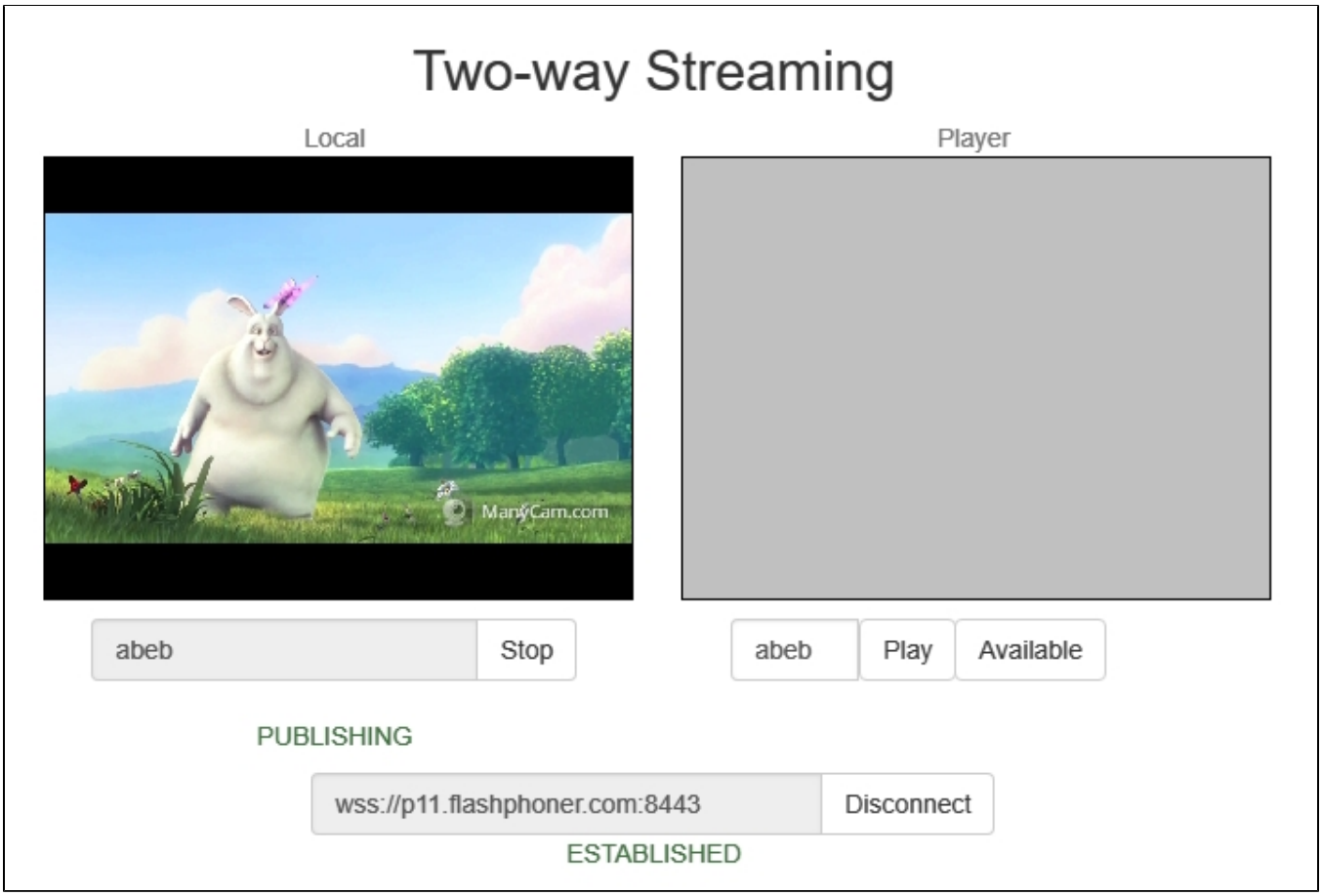
Краткое руководство по тестированию

1. Для теста используем:

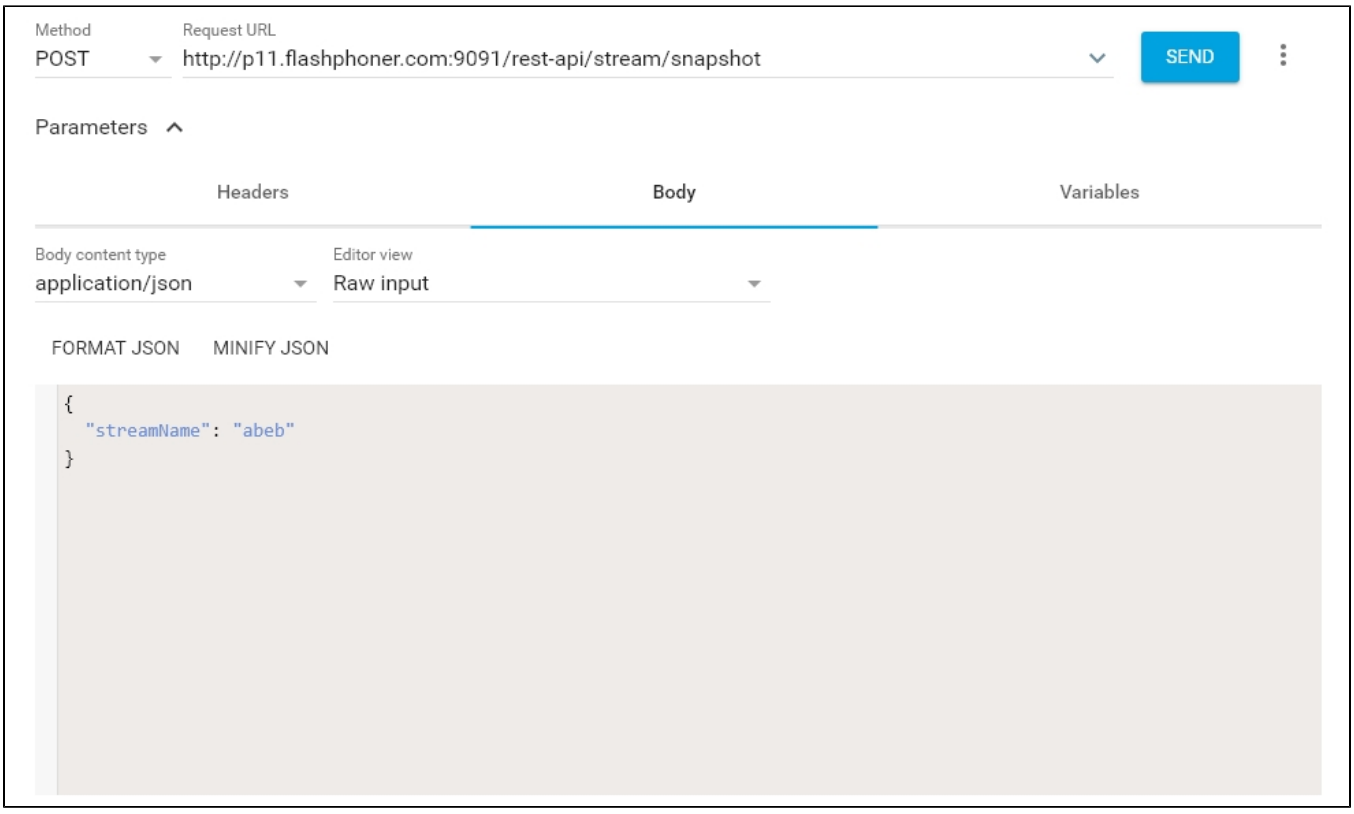
- демо-сервер demo.flashphoner.com;

- браузер Chrome иREST-клиентдля отправки запросов на сервер;
- веб-приложениеTwo Way Streamingдля публикации потока;
- сервисhttps://www.motobit.com/util/base64-decoder-encoder.aspдля декодирования превью.

2. Откройте страницу веб-приложения Two Way Streaming. Нажмите "Connect", затем нажмите "Publish" для публикации потока:



3. Откройте REST-клиент. Отправьте запрос /stream/snapshot, указав в параметрах идентификатор опубликованного потока:



4. Убедитесь, что ответ получен:

5. Откройте страницу онлайн-декодера, скопируйте в форму содержание ответа и нажмите "Convert the source data":

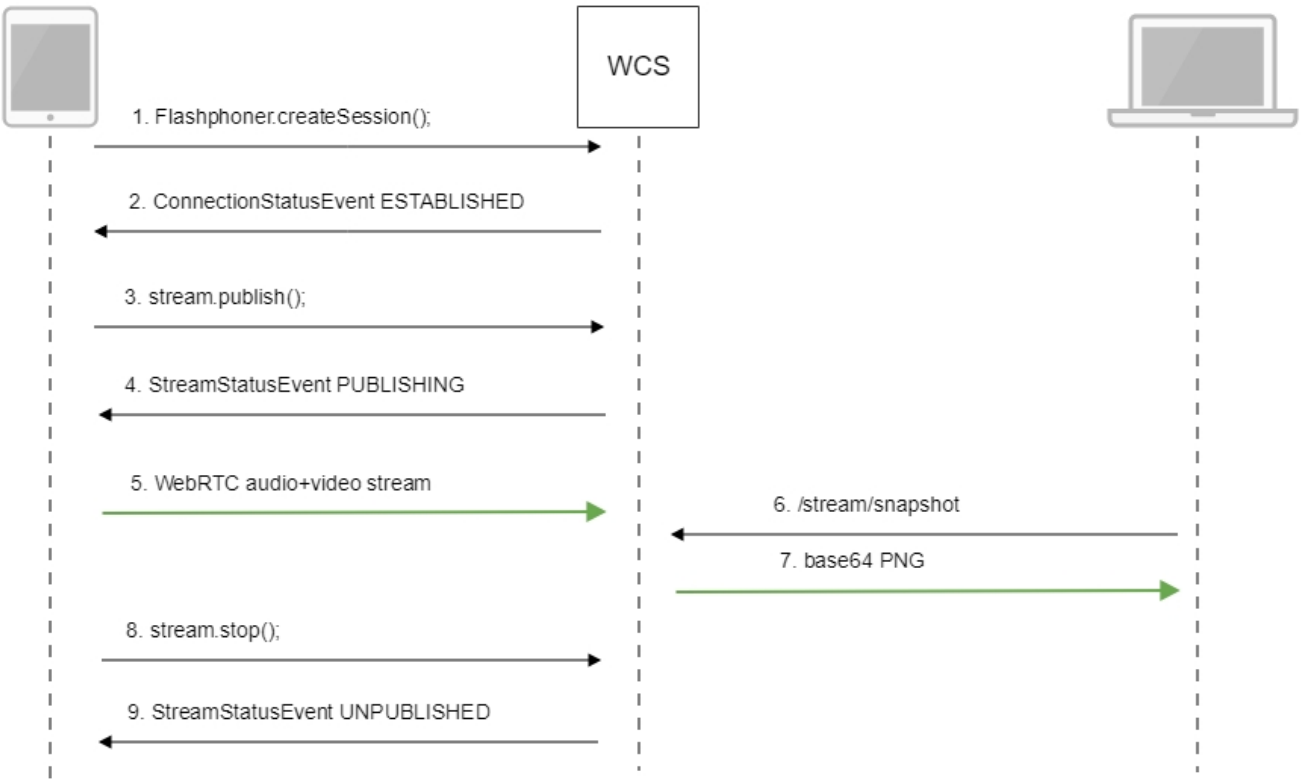
6. Полученный файл превью:

Последовательность выполнения операций (Call flow)

Ниже описана последовательность вызовов при использовании примера Stream Snapshot для публикации потока и снятия превью

stream-snapshot.html

stream-snapshot.js



1. Установка соединения с сервером.

Flashphoner.createSession();code

```
        Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
            ...
        });
```

2. Получение от сервера события, подтверждающего успешное соединение.

ConnectionStatusEvent ESTABLISHEDcode

```
        Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
            //session connected, start streaming
            startStreaming(session);
        }).on(SESSION_STATUS.DISCONNECTED, function(){
            ...
        }).on(SESSION_STATUS.FAILED, function(){
            ...
        });
```

3. Публикация потока.

stream.publish();code

```
session.createStream({
       name: streamName,
       display: localVideo,
       cacheLocalResources: true,
       receiveVideo: false,
       receiveAudio: false
       ...
   }).publish();
```

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

StreamStatusEvent, статус PUBLISHINGcode

```
    session.createStream({
       name: streamName,
       display: localVideo,
       cacheLocalResources: true,
       receiveVideo: false,
       receiveAudio: false
   }).on(STREAM_STATUS.PUBLISHING, function(publishStream){
       setStatus(STREAM_STATUS.PUBLISHING);
       onPublishing(publishStream);
   }).on(STREAM_STATUS.UNPUBLISHED, function(){
       ...
   }).on(STREAM_STATUS.FAILED, function(stream){
       ...
   }).publish();
```

5. Отправка аудио-видео потока по WebRTC

6. Снятие превью трансляции. Создается новый поток из опубликованного, специально для снятия превью.

stream.snapshot();code

```
function snapshot(name) {
    setSnapshotStatus();
    var session = Flashphoner.getSessions()[0];
    session.createStream({name: name}).on(STREAM_STATUS.SNAPSHOT_COMPLETE, function(stream){
        console.log("Snapshot complete");
        setSnapshotStatus(STREAM_STATUS.SNAPSHOT_COMPLETE);
        snapshotImg.src = ""+stream.getInfo();
        //remove failed callback
        stream.on(STREAM_STATUS.FAILED, function(){});
        //release stream object
        stream.stop();
    }).on(STREAM_STATUS.FAILED, function(stream){
        setSnapshotStatus(STREAM_STATUS.FAILED);
        console.log("Snapshot failed, info: " + stream.getInfo());
    }).snapshot();
}
```

8. Остановка публикации потока.

stream.stop();code

```
function onPublishing(stream) {
    $("#publishBtn").text("Stop").off('click').click(function(){
        $(this).prop('disabled', true);
        stream.stop();
    }).prop('disabled', false);
    ...
}
```

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

StreamStatusEvent, статус UNPUBLISHEDcode

```
 session.createStream({
        name: streamName,
        display: localVideo,
        cacheLocalResources: true,
        receiveVideo: false,
        receiveAudio: false
    }).on(STREAM_STATUS.PUBLISHING, function(publishStream){
        ...
    }).on(STREAM_STATUS.UNPUBLISHED, function(){
        setStatus(STREAM_STATUS.UNPUBLISHED);
        //enable start button
        onUnpublished();
    }).on(STREAM_STATUS.FAILED, function(stream){
        ...
    }).publish();
```