

Снятие превью трансляции в виде PNG

- Описание
 - Поддерживаемые протоколы
 - Поддерживаемые форматы превью
 - Схема работы
- REST-вызовы
 - Параметры
 - Отправка REST-запроса к WCS-серверу
- JavaScript API
- Краткое руководство по тестированию
- Последовательность выполнения операций (Call flow)

Описание

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

Поддерживаемые протоколы

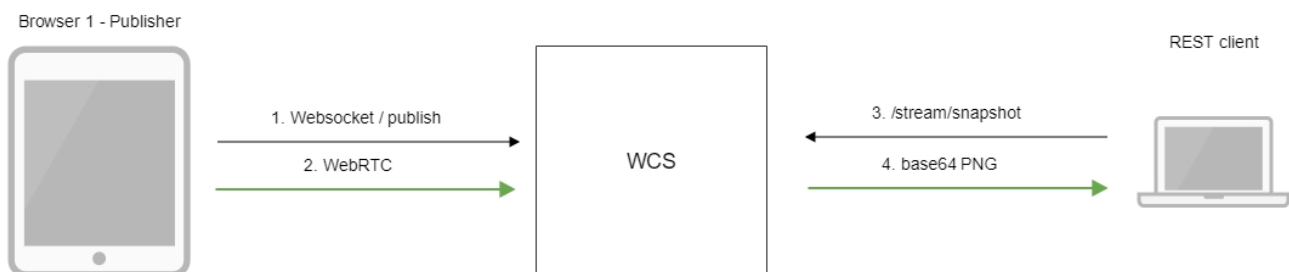
- WebRTC
- RTMP
- RTSP

Поддерживаемые форматы превью

- PNG

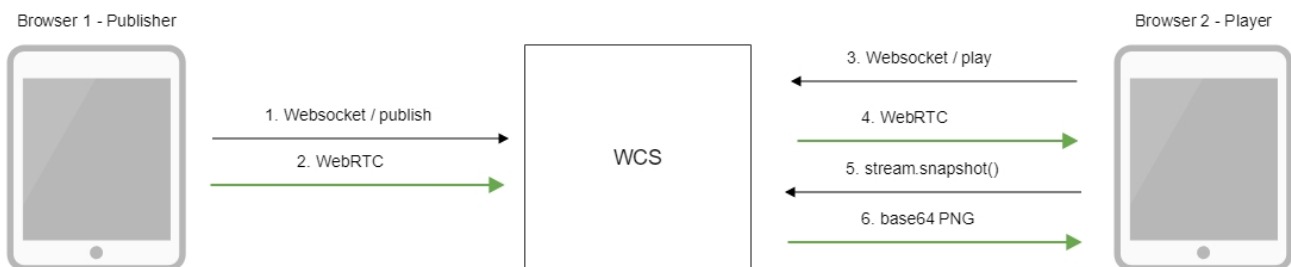
Схема работы

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	<pre>{ "streamName": "64966f33" }</pre>	<pre>{ "data": "iVBORw0KGgoAAAANSUhEUGAAAUAAADwCAYAAABxLb1rAAAACXBIXWMAAAAAAAAAAAQCErDzAAQAE1EQVR4nOzd9SPcd37feVjhrFKwXT6fr+r+gPvh6nx1V5bvLMt3liXbkiWtrFlv1O5qVxu4icucwUyABAMIAgSIQOQcBzlnzAADDIDJuadzjzmH173f7+/30216g04GiCHIZtWjeqZnejAAu5/9+XzD57tgwYIF6Ojo6PiSmvdfoK0jo20+ZPsv0NHR0TFf5v0X6Ojo6Jgv8/4LdHR0dMyXef8FOjo6OubLvP8CHR0dHfN13n+Bjo6Ojvky779AR0dHx3yZ91+go6OjY77M+y/Q0dHRMV/m/Rfo6OjomC/z/gt0dHR0zJd5/wU6Ojo65su8/wIdHR0d82Xef4GOjo6O+TLvv0BHR0fHfJn3X6Cjo6Njvsz7L9DR0dExX+b9F+jo6OiiYL/P+C3R0dHTM13n/BTo6OjrmY7z/Ah0dHR1t+Ue636rzGzr19Tv8jPn/S3R0dHS04z4E8HfR0dHR8XD5DaGC94/rdALY0dHxBXafAlj/wI6Ojo6Pb7foXh//edcJYEfh14gaCd0tDPXf0+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/md3xELV16PoaOj4"</pre>	200 - Превью зафиксирован 404 - Поток не найден

Параметры

Имя параметра	Описание	Пример
streamName	Уникальное имя потока	64966f33
data	Файл превью в base64-кодировке	iVBORw0KGgoAAAANSUhEUGAAAUAAADwCAYAAABxLb1rAAAACXBIXWMAAAAAAAAAAAQCErDzAAQA

Отправка REST-запроса к WCS-серверу

Для отправки REST-запроса к WCS-серверу необходимо использовать [REST-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 = "data:image/png;base64,"+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();
}
```

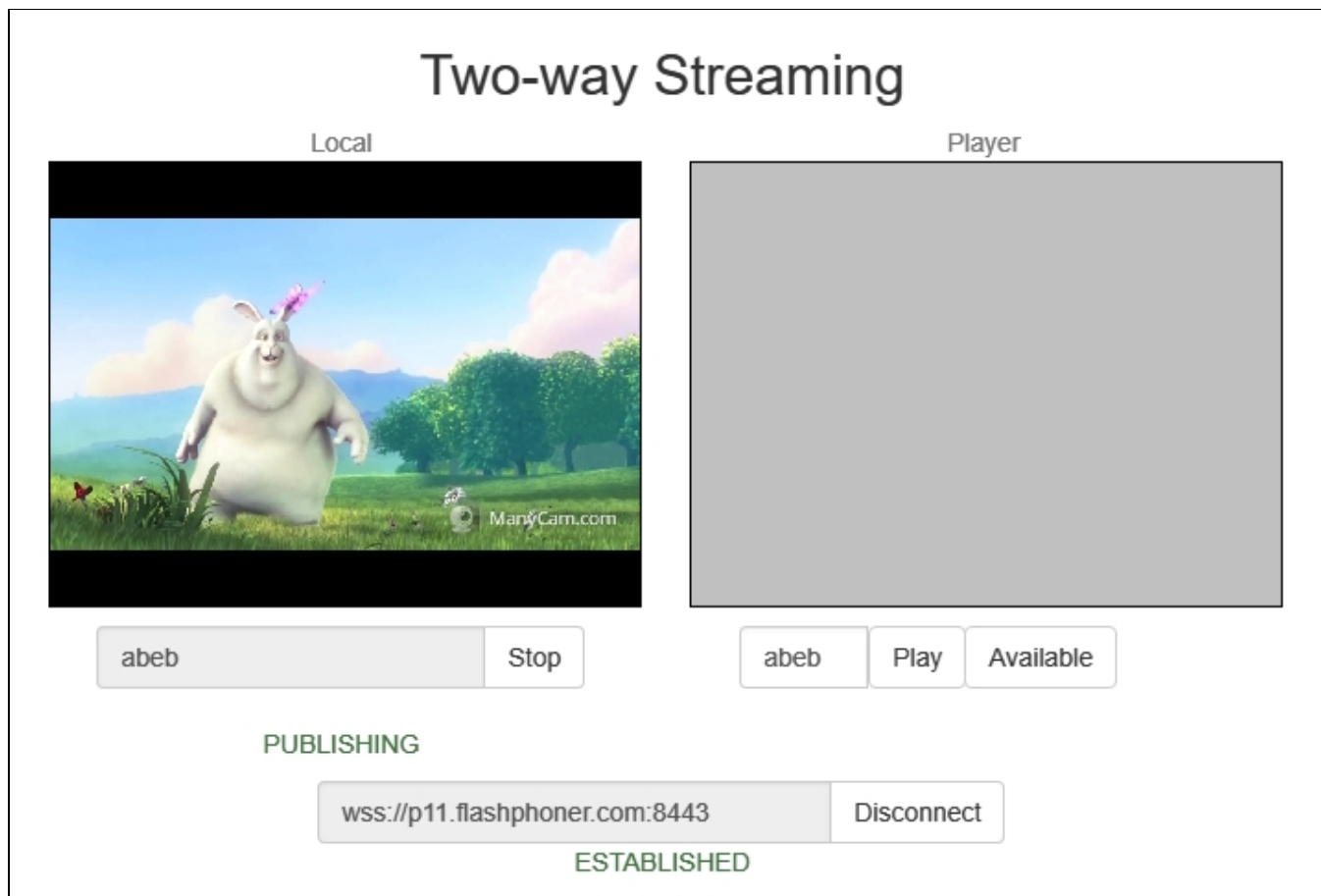
Краткое руководство по тестированию

1. Для теста используем:

- демо-сервер demo.flashphoner.com;

- браузер Chrome и [REST-клиент](#) для отправки запросов на сервер;
- веб-приложение [Two Way Streaming](#) для публикации потока;
- сервис <https://www.motobit.com/util/base64-decoder-encoder.asp> для 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:

MethodPOSTRequest URLhttp://p11.flashphoner.com:9091/rest-api/stream/snapshotSEND

Parameters

Headers

Body

Variables

Body content typeapplication/jsonEditor viewRaw input

FORMAT JSONMINIFY JSON

```
{
  "streamName": "abeb"
}
```

4. Убедитесь, что ответ получен:

200 OK458.60 msDETAILS

```
{
  "data": "iVBORw0KGgoAAAANSUHEUgAAAAAADwCAYAAABxLb1rAAACXBIXMMAAAAAAAAAAAQCEeRdzAAAE1EQVR4nOzd95Pcd37feVjhrFKwXT6fr+r+gPvh6nx1V5bvLMt31iXbkiWtrF1v105qVxu4iCucwUyABAMIAgSIQ0QcBz1nzAADDIDJuaazzjmH173f7+/302l6g046iCHIZtwjeqZnejaAu5/9+XzD57tgwYIF60jo6PiSmvdfok0jo20+zPsv0NHR0TFf5v0X60jo6Jgv8/4LdHR0dMyXef8FOjo60UblvP8CHR0dHfN13n+Bjo60jvky779AR0dHx3yZ91+go60jY77M+y/Q0dHRMV/m/Rfo60jomC/z/gt0dHR0zJd5/wU60jo65su8/wIdHR0d82Xef460jo60+TLvv0BHR0dHfJn3X6Cjo6Njvsz7L9DR0dExx+b9f+jo60iYL/P+C3R0dHTM13n/BT060jrmjy7z/Ah0dHR1t+Ue636rzGzr19Tvr8jPn/S3R0dHS04z4E8HFR0dHR8XD5DaGC94/rdALY0dHx8XafA1j/wI60jofPb7foXh//edcJYEFH14gaCd0tDPXf0+rjHxbNT4F/6/fr0dHxkPsNnhb+DvnN07zYf1P7nt/8vTYf/3nzj8hvk/+B/GPT76DIfeY3dHP8jN/8g/8JHR0dD7d/9Hv/Ixb87j/Hgt/5ZzSt+6dY8D/8Eyz47T+o4Pv4a/Q9v/H7/6L1x3+e80+m/134d/+tf/Iv8dv/9H/G7/zz/6Xs9/7ZvxS//8/+hfiDP/gD/P7v/z57/d+D7/7u7+L3/md3xELV16Poa0j4+G2+mY5a261sPZ2Guv6M/h0IIsNQ/kyvo+/xt/D3/tJX6K1x3+ebBwuY0dwCrtG0rg4YselUQeujFRM27HjTGj6B+dEAMjY2J8FAQE60YnBzD1NQ4ZmamxAL+S3c8vPgJ+zDjF9uXGYfmfuaObBopYpcB2D0D7DMDB6wVfB9/jb9HhYQfP34PFZa5Hv950mUDTPBTDMAHsegI4FhRxRj7gSmXSfHdnqExeESbrcdPp8boZaf4XAAiURULPh4A0h48fbpVveXsIZu1wWla6wdyNe4Xy+U+6H6nbj+nbkZ/Cjk0dL5Y/V583aMgbRymM+L4/nx22dKIkt41nNRBpbjzNN2T6dw86pHHZN57FzNI1dYxnsHc+JA1PF01Lfp6iR107RjNg3kb+r/ZMFwT+va7pUdpD12azDuqOTJXF8otaJ5QrcNHBMrnPwCJwzVZwnFyjS3eYerlpTuOVMYNCbgdnnhz0Uhi/gRYB1Fwq7RTjiEdGYD7G4H/FEAMlUiKbAg0DHg6cCuEZpM4Dz0Q2pj1kr6i0ib81v3cP4+0304t4+BeyY5tuChqK2w5Bv2m5DQeyZoJBVNsJ0twdY91THTIWPY9hM/053AI9NaTh41Th+pw1a+FT8zpsrOH6XaDTI8btmS+02KykbTpgDEkB/0IcgjfsQw1cdv0QyQAVvvdCX1b3awT4ZQ/gw6g6gDumixqK2s6ZQtNUACV+U4WakN0tgNWqR4PtBLBadQzvp1EAOXoKx49HfUURHwXvorWC43eZpsGXHX1ccrZw05FF7UAetmAAjNaiAR71cfCilk3cK+IJH8XPT/ELIJU0dglYCEd9C2A7U8hZYfiS00IH7DSUNBS1XcZi0/bMaPZpa+4WvrvFsnN4fRYBPGGojV8zAZT4EY6fCmB/sECjvyCckXDDANbHL5MNdWLYCwAngJ0AthbA+hi2+udyAA/pETxhqASQo6eo7X5qyiu/jPluFxm80dLUk4pY/h4FQUUZ/rmgEQZr2Rni6G3Nr9PipAHL8sr1IJ4CdAHYC2AngvQWw2en3XAE80VOJXv10j/KUL6Nnr5D40bQA9rhLMv2tDiCp/qK8rS/u0VTFT43+OghH8ARw1UAJnwxilrsFsRzGoeIddQL4+fr5CeBc2wabdwA6q0ujy5BpysGZLA7P5HUVUVRJYwGn6PYM/Z30mkp153jkZ6Hw2TWXHTzdtregmPS6g30Ph8xYwGcjBEC7CHU7AF0sjFgvLYS7peFBkYv6ybdWgconONsB0ADsB/FIHUG2Ta+fxMpoz5HRA2Jpx2KTF75i5JPE7ba5ET+H4XbRp4VPx63ZVCPyuuofBXxFD/pLEzxifxm8fz0j8kskYMomQUNGr1gngPKo+HOazDOC92jCHjcM1swkEZZtHm/e1D+CUZqdBKWEXH6zcpD0Gzf4pzYHJ5nVNaQ50Vz5WnzfrsFFHMDNkm3BEnMMxSwbHrVmKX0GcNRUoFmWy85bSnAG85gVukwE/MEEBnA4Cvpr+2iNAMJZDOFFAMP5AKpFENhYR+Yhf5MK+Mv68E8B0ADsB/BiHsF2NAsasha82RS17id8KlWwx1LUajokRw/nvqWp7w86nlnren009SndAWCSR38hwBYuRWHxC+SLEr80skUcvGoKEQDNQHhrxy2wngPADQIJg0292CqKwbxh11Avj59HkM40GZ5h0xaY5SyI5RyI7ZUzjuSDft1DMjzt01520ZXHTkyi4587jsKqDbUxI9NOK75tdcD2qjvyEe+fnzsAeLCNIU0BIDohTBWAI0xc0hnyyglMgIROibw1TMoE+UA1657QSwE8B0AL/EAayJmrFSR82a4/Y8TjgKOMfBc+eadkZ3wZUXHLxqHL8r7qKET8wvN6C5EdJGfyNhSPycdBu0VukXSEhIv0gVgWRWUxVAj1/R7+kEsBPATgA7Aww9gEdps5HfNQvGzQuJ3y1XcAu++Jwd19xpAR6gEN8wPR38qfsk0JH7fDAlI5TSxsBbBkL92BhIv57T0d0geVp0AdgL4sAWQwyfxH+vxswFnaCB11lpy11fEOX+paRd0170aNdWtn/LydfFh8LGBES1+YxQ+Hv159Kmv1l8qA4nfrAAyHgXqEeTbBY02wLeicjHreYjWwvS6wtYQ7grjbtng5XKketGXunNQ01Xtz7626u98hY8VhU0V9/I6bnSdo5HeS4nfGqVgX6qUAXqegzS1QqNGnu+Up1t2mEFZTOzoU3ubHRIAMuXRCa936KIIBil+cpCiAaQpgplEA41FNJKQJB+W2E8B51wng1zmArUTvfgeQR32tBFDf7xSN/M66NB1/npa5vuAd0FS12i1dv75H1/GOjWrv0EMRn8Ijv+lw4wCqCPIUuGYbIMcvGp4dwPX0JG4Xv0DUck58S1c1Psc32pzhGyroc18qq4aLYs43f/q3ZXY5Ru6e8Ty99364ULZB15gU7dxtNS0TRS/TfM2wCm1pCu0ZNdUuEyhX739k61RU9ey6WJLDtCumbWYBFV5Kaq6HWZNAccF2WChXCYzo+WPkM3X+ez8y8N2kqe9tItY4jdQm6HbGm4M5ULyDfJkLWIAtUIKdGmTxFXS1G1Y/hC1Qwd/PHNQVH90GeFYbKVDBikHcsjGCiKfKKGQyIL5XF0kOTIrRGiIH47K7YIN9H7d0hXbdTRAeYv17XSzB0oMR1lCD8G04fvXz0wAchmuMoKnj0aKDVXHCvEbVeFz1UA4Vfwe1gDv811cP5m2VvVwVtBaURW17nTB6T3Vb61720bwAb4lrcMdvTAC3Hx4Tz76
```

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

You can use this base64 sample decoder and encoder to:

- Decode base64 strings (base64 string looks like YTM0NZomIzI2OTsmlzM0NTueYQ==)
- Decode a base64 encoded file (for example ICO files or files from MIME message)
- Convert text data from several code pages and encode them to a base64 string or a file
- **New: Try [CSS/base64 analyzer](#) and simple [Base64 decoder and encoder](#).**

The Form.SizeLimit is 10000000bytes. Please, do not post more data using this form.

Type (or copy-paste) some text to a textbox bellow. The text can be a Base64 string to decode or any string to encode to a Base64.

```
PH/33K0dH30424E0nK0dHk0X0D0B0C0347F0AET0dHx0X0A7A1j/W100j0TF0D7F0Xn7/ed037E7H14g0C00C0F0X1
0+rjHxbNT4F/6/fR0dHxkPsNnhb+DvnN07zYf1P7nt/8vTYf/3nzj8hvk/+B/GPt76DIfeY3dHP8jN/8g/8JHR0
dD7d/9Hv/Ixb87j/Hgt/5ZzSt+6dY8D/8Eyz47T+o4Pv4a/Q9v/H7/6L1x3+e80+m/134d/+tf/Iv8dv/9H/G7/
zz/6Xs9/7ZvxS//8/+hfiDP/gD/P7v/z5+7/d+D7/7u7+L3/md3xELV16Poa0j4+G2+mYsa261sPZ2Guv6M/h0I
IsNQ/kyvo+/xt/D3/tjX6K1x3+ebBwuY0dwCrtG0rg4Yse1UQeujFrRM27HjTGj6B+dEAMjY2J8fAQTE6OYnBzD
1NQ4ZmamxAL+S3c8vPgJ+zDjF9uXGYfmfuAobBopYpcB2D0D7DMDB6wVfB9/jb9HhYQfp34PFZa5Hv950mUDTpB
TDmDAHsegI4FhRxRj7gSmXSFhdnqExeESbrCDPP8boZaf4XAAiURULPh4AOH48FbpVveXsIZu1wwWa6wdyNe4Xy
+U+6H6nbj+nbkZ/CJkm0dL5Y/V583aMgbRymM+L4/nx22dKIkt41nNRBpbJzNN2T6dw86pHHZN57FzNI1dYxnsH
c+JA1PF01Lfp6iR107RjNg3kb+r/ZMFwT+va7pUdpDi2azDuqOTJXF8otaJ5QrcNHBmRnPWcJwzVZwnFyJ53eYE
r1pTuOVMYNCbgdnnhz0Uhi/gRYBiFwq7RTjiEdGYD7G4H/FEAM1UiKbAg0DHg6cCuEZpM4Dz0Q2pj1kr6i0ibB1
v3cP4+0304t4+8eyY5tuChqK2w5Bv2m5DQeyZoJBNVsJ0twDy91THTIWPY9hM/053AI9NaTh41Th+pw1a+FT8zp
srOH6XaDTI8btmS+02KyKbTPgDEkB/0IcgjJSqw1cdv0QyqAVwvkdCX1b3awT4ZQ/gw6g6gDumixqK2s6ZQtNUA
CV+U4WaKN0tgNWqR4PtBLBadQzvp1EA0XoKx49HfuURHwXvorWC43eZpsGXHX1ccRZw05fF7UAetMAAjnAIAR71
cfCi1k3cK+TJH8XPT/ELTJU0deIYCeD9C2A7U8hZYfiS00TH7DSUNBS1Xc7i0/bMaP7Pa+4WwrvFsNp4fRYBPGG
```

or select a file to convert to a Base64 string.

Выберите файл Файл не выбран

Convert the source data

What to do with the source data:

- ☐ **encode** the source data to a **Base64** string (base64 encoding)
Maximum characters per line:
- ☒ **decode** the data from a **Base64** string (base64 decoding)

Output data:

- ☐ output to a textbox (as a string)
- ☒ export to a binary file, filename:

6. Полученный файл превью:

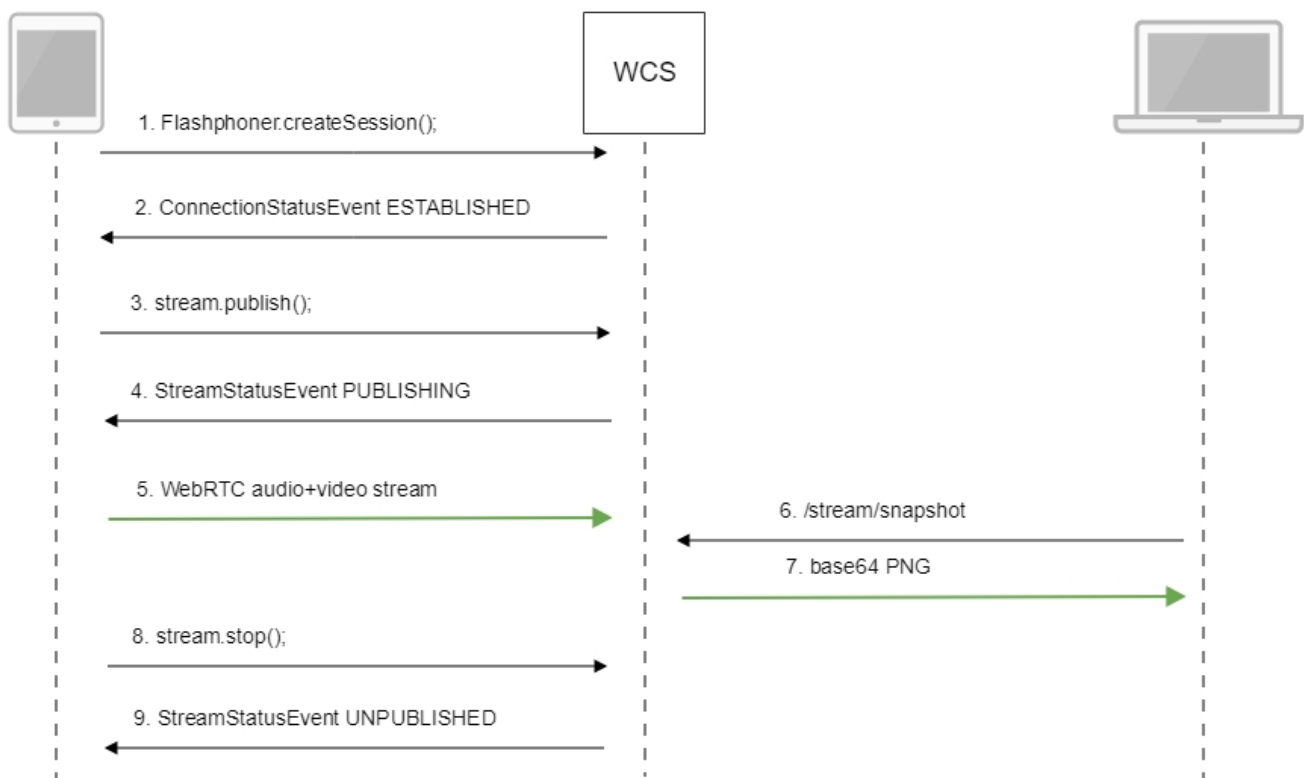


Последовательность выполнения операций (Call flow)

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 ESTABLISHED[code](#)

```
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 PUBLISHING[code](#)

```
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 = "data:image/png;base64,"+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 UNPUBLISHED[code](#)

```

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();

```