

# Снятие превью трансляции в виде PNG

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

## Описание

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

## Поддерживаемые протоколы

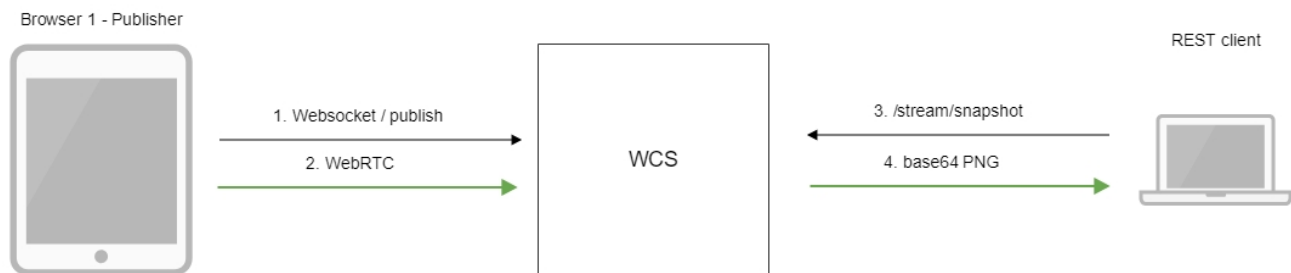
- WebRTC
- RTMP
- RTSP

## Поддерживаемые форматы превью

- PNG

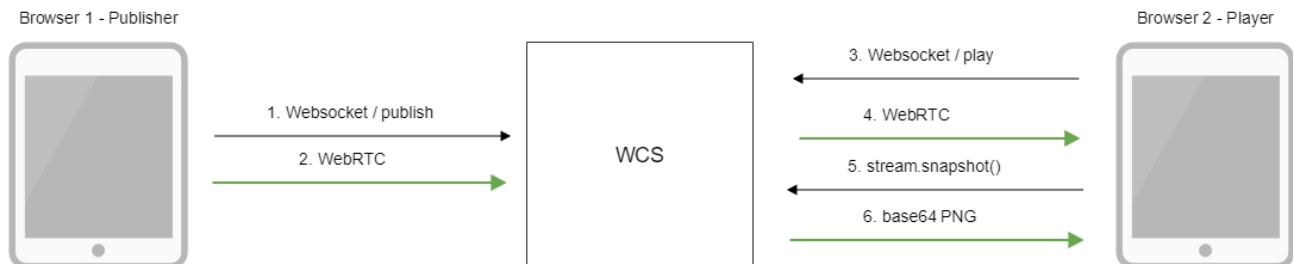
## Схема работы

### 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":  "iVBORw0KGgoAAAANSUHEUgAAAUAAADwCAYAAABxLb1rAAAACXBIWXMAAAAAAAAAAQCEeRdzAAAQAELEQVR4nOzd95Pcd37feVjhrFKwXT6fr+r+gPvh6nx1V5bvLMt3liXbkiWtrF1v105qVxu4icucwUyABAMIAgSIQOQcBzlnzAADDIDJvadzzjmH173f7+/302l6g04GiCHIZtwjeqZnejAAu5/9+XzD57tgwYIF60jo6PiSmvdfok0jo20+zPsv0NHR0TFf5v0X60jo6Jgv8/4LdHR0dMyXef8F0jo6OubLvP8CHR0dHfNl3n+Bjo60jvky779AR0dHx3yZ91+go60jY77M+y/Q0dHRMV/m/Rfo60jomC/z/gt0dHR0zJd5/wU60jo65su8/wIdHR0d82Xef4G0jo60+TLvv0BHR0fHfJn3X6Cjo6Njvsz7L9DR0dExX+b9F+jo60iYL/P+C3R0dHTMl3n/BTo60jrmy7z/Ah0dHR1t+Ue636rzGzr19Tv8jPn/S3R0dHS04z4E8HfR0dHR8XD5DaGC94/rdALY0dHxBXafAlj/wI60jofPb7foXh//edcJYEfHl4gaCd0tDPXf0+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/z5+7/d+D7/7u7+L3/md3xELVl6Poa0j4"</pre> | 200 - Превью зафиксирован<br><br>404 - Поток не найден<br><br>500 - Ошибка фиксации |

## Параметры

| Имя параметра | Описание                       | Пример                                                                        |
|---------------|--------------------------------|-------------------------------------------------------------------------------|
| streamName    | Уникальное имя потока          | 64966f33                                                                      |
| data          | Файл превью в base64-кодировке | iVBORw0KGgoAAAANSUHEUgAAAUAAADwCAYAAABxLb1rAAAACXBIWXMAAAAAAAAAAQCEeRdzAAAQAE |

## Отправка REST-запроса к WCS-серверу

Для отправки REST-запроса к WCS-серверу необходимо использовать [REST-клиент](#).

## Настройка

Начиная со сборки [5.2.1116](#), при получении превью трансляции при помощи REST API можно настроить максимальную длительность фиксации превью, включая возможную задержку при записи на диск сервера. По умолчанию, максимальная длительность установлена в 3000 мс, за это время предпринимается 30 попыток проверить, готов ли файл превью

```
snapshot_taking_interval_ms=3000
snapshot_taking_attempts=30
```

Если файл превью не готов по истечении указанного интервала, запрос /stream/snapshot возвращает ошибку

```
{
  "exception": "com.flashphoner.rest.server.exception.InternalErrorException",
  "reason": "com.flashphoner.rest.server.exception.InternalErrorException, Internal Server Error, Snapshot
response timeout, ts: 1640836780816, path: /rest-api/stream/snapshot",
  "path": "/rest-api/stream/snapshot",
  "error": "Internal Server Error",
  "message": "Snapshot response timeout",
  "timestamp": 1640836780816,
  "status": 500
}
```

## 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](http://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:

# Two-way Streaming

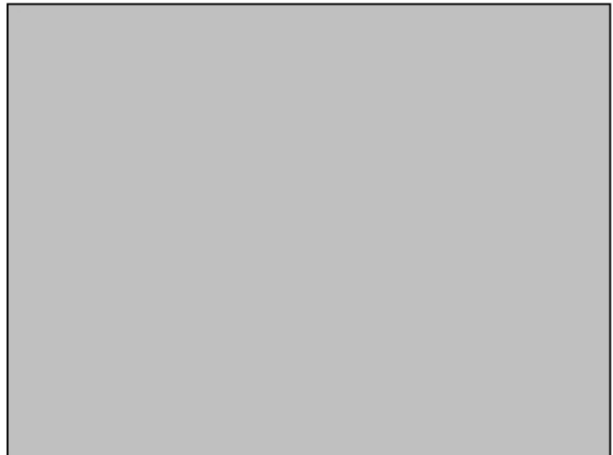
Local



abeb

Stop

Player



abeb

Play

Available

PUBLISHING

wss://p11.flashphoner.com:8443

Disconnect

ESTABLISHED

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

Method

Request URL

POST

http://p11.flashphoner.com:9091/rest-api/stream/snapshot

SEND

Parameters ^

Headers

Body

Variables

Body content type

Editor view

application/json

Raw input

FORMAT JSON MINIFY JSON

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

4. Убедитесь, что ответ получен:

200 OK458.60 ms

DETAILS

```
{
  "data": "iVBORw0KGgoAAAANSUHEUgAAAAAADwCAYAAABxLb1rAAACXBIWMAAAAAAAAAAAQCEeRdzAAAQAE1EQVR4nOzd95Pcd37feVjhrFKwXT6fr+r+g
Pvh6nx1V5bvLMt31iXbkiWtrF1v1O5qVxu4iCucwUyABAMIAgSIQ0QcBz1nzAADDIDJuaazzjmH173f7+/30216g046iCHIZtwjeqZnejaAu5/9+XzD57tgwYI
F60jo6PiSmvdfokOjo20+zPsv0NHR0TFf5v0X60jo6Jgv8/4LdHR0dMyXef8FOjo6OubLvP8CHR0dHfN13n+Bjo60jvky779AR0dHx3yZ91+go60jy77M+y/Q0
dHRMV/m/Rfo60jomC/z/gt0dHR0zJd5/wU60jo65su8/wIdHR0d82Xef4G0jo60+TLvv0BHR0fHfJn3X6Cjo6Njvsz7L9DR0dExX+b9F+jo60iYL/P+C3R0dHT
M13n/BTo60jrmY7z/Ah0dHR1t+Ue636rzGzr19Tvb8jPn/S3R0dHS04z4E8HFR0dHR8XD5DaGC94/rdALY0dHx8XafA1j/wI60jofPb7foXh//edcJYEfH14gaC
d0tDPXf0+rjHxbNT4F/6/fr0dHxkPsNnhb+DvnN07zYf1P7nt/8vTYf/3nzj8hvk/+B/GPT67DIFey3dHP8jN/8g/8JHR0dD7d/9Hv/Ixb87j/Hgt/5ZzSt+6d
Y8D/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/md3xELV16Poaj4
+G2+mYSa261sPZ2Guv6M/h0IIsNQ/kyvo+/xt/D3/tJX6K1x3+ebBwuY0dwCrtG0rg4YseIUQeujFrRM27HjTGj6B+dEAMjY2J8fAQTE60YnBzD1NQ4ZmamxAL
+S3c8vPgJ+zDjF9uXGYfMfuAobBopYpcB2D0D7DMD86wVfB9/jb9HhYQfP34PFZa5Hv950mUDTpBTDmDAHsegI4FhRxRj7gSmXSFhdnqExeESbrCpPp8boZaf4
XAAiURULPh4A0h48FbpVveXsIZu1wwa6wdyNe4Xy+U+6H6nbj+nbkZ/CJkm0dL5Y/V583aMgbRymM+L4/nx22dKikt41nNRBpbjJzNN2T6dw86pHHZN57FzNI1
dYxnsHc+JA1PF01Lfp6iR107RjNg3kb+r/ZMFwT+va7pUdpDi2azDuq0TJXF8otaJJSQrcNH8mRnPWcJwzVZwnFyJ3SeYer1pTuOVMYNCbgdnzh0Uhi/gRYB1F
wq7RTjEdGYD7G4H/FEAMlUiKbAg0DHg6cCuEZpM4Dz0Q2pjlkr6i0ibb1v3cP4+0304t4+BeyY5tuChqK2w58v2m5DQeyZoJBNVs30twDy91THTIWPY9hM/05
3AI9NaTh41Th+pw1a+FT8zpsr0H6XaDTI8btmS+02KykBtPgDEkB/0IcgjfSqw1cdv0QyqAVvkdCX1b3awT4ZQ/gw6g6gDumixqK2s6ZQtnUACV+U4WakN0tg
NWqR4PtBLBadQzvp1EAOXokX49HfuURHwXvorWC43eZpsGXHX1ccRzW05fF7UAetMAA9nAIAR71cfCiLk3cK+IjH8XPT/ELIJu0dGLYCEd9C2A7U8hZyfiS00I
H7DSUNBS1XcZi0/bMaPZPa+4WvrvFsNn4fRYBPGG0jV8zAZT4EY6fCmB/sECjvyCckXDDANbHL5MNdWLYCNAngJ0AthbA+h2+udyAA/pETxhqASQo6eo7X5qy
iujPluFxm80dLuk4pY/h4FQUUZ/rmgEQZr2Rni6G3Nr9PipAHL8sr1Ij4CdAHYC2AngvQWw2en3XAE80VOJXv10j/KUL6Nnr5D40bQA9rhLmv2tDiCP/qk8rS/
u0VTFT43+OgH8HARw1UAJnwx1lrsFsRzGoeIddQL4+fr5CeBc2wabdwA6q0ujy5BpysGZLA7P5HHUVMRjYwGn6PYM/Z30mkp153jkZ6Hw2TWXHTzdregmP56g3
0Ph8xYwGcjBEC7CHU7AF0sJfGvLYS7peFBkYv6ybdWgconONsB0ADsB/FIHUG2Ta+fxMpoz5HRA2Jpx2KT7F5i5JPE7ba5ET+H4XbRp4VPx63ZVcPyyuoFBXXf
D/plEzxiFxm8fz0j8kskYMoMQUNGr1gngPKo+HOazD0C92jCHjcM1swkEZZtHm/elD+CUZqdBKWEXH6zcpD0Gzf4pzYHJ5nVNaQ50Vz5WnzfrsFFHMdNkm3bEn
MMxSwbHrVmKX0GcNRUofMwy85bSnAG85gVukwE/MEEBNA4CVpr+2iNAMJZDOFFAMp5AKpFENhYR+Yhf5MK+Mv68E8B0ADsB/BiHsF2Nashha8ZRS17id8KwWx1
LUaJoKRw/nvqWp7w86Nren09S5DAWCSR38hwBYuRWXhC+SLEr80skUcvGoKEQDNQHMhrxy2wngPADQIjg0292CqKwbxh11Avj59HkM40GZ5h0xaY5Si5Ry
I7ZUzjuSDft1DMjzto1520ZXHTkyi4587jsKqDbUxI9NOK75tdC2DqjvyEe+fnzsAeLCNIU0BIDohTBWAI0xc0hnyyglMgIROibw1TMOE+UA1657QSwE8B0AL/
EAayJmrF5R82a4/Y8TjgKOMFBc+eadkZ3wZUXHLxqHL8r7qKET8WwN6C5EdJGfyNhSPycdBu0VvKXSEHiV0gVgWRWUxVAj1/R7+kEsBPATgA7Aw9gEdp5HfMQ
vGzQuJ3y1XCaU++Jwd19xpAR6gEN8WPR38qfsk0JH7FdAl15TSxsBbBkL92BHiV57T0d0geVp0AdgL4sAwQwyfxM+vxswFnaCB11py11fEOX+paRd0170aNdW
tn/LydfFh8LGBES1+YxQ+Hv159Kmv18qA4nfrAAyHgXqEeTbBY02wLeicjhHreYjWvqS6wTwYQ7grjbtng5XKketGXunNQd01Xtz76Z6u98hY8VhU0V9/I6bn
SdoHeS4nfGqV6x6qUAXqegz5lQqNGnu+UplT2mEFZT0zoU3ubHRIaUxRCA936KIIBi1+cpCiAaQpgplEA41FNJKQJB+W2E8B51wnglzmArUTvfgeQR32tBFD
F7xSN/M66NBI/npaSuAd0FS12i1dv75H1/G0jWrv0eMRn8Ijv+lw4wCqCPIUuGyBIMcvGp4dwPX0JG4Xv0DUck58S1c1Pc32pzhGyrocl8qq4aLYs43F/q3Z
ZXYSRu6e8Ty99364ULZB15gU7dxtNS0TRS/TfMZWm1pCu0ZNDUueyhx7J9k61RU9ey6JlDTCumbWYBFV5Kaqg6HwZNacocF2WChXCYzo+WPkM3X+ez8yg8N2
kqe9tItY4jdQm6HbGm4M5ULYDfJk1WIAtUIKdGmTxFXS1G1Y/hC1Qwd/PHNQVh90GeFYbKVDbikHcscjGciKfKKGQyI1SXFokOTIRGIH47K7YIN9M7dHXBd
TBaYeV17X5zRooNBilCD8G04fyXzQwAcbmuMoKa30aKDVYHcvFhVeFz1UA4Vfwe1qDv811cP5m2VqVwWtBa1UW17nTB6T3Vh61770hw4b41cMduTACJHx+Tz7G
```

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/33K0dH30424E0nR00dHk0X0D0B0C0347F0AET0dHXB0X0A1J/W100J0TF0D7F0Xn7/ed037E7H14g0C00C0FX1
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+va7pUdpDi2azDuqOTJXF8otaJ5QrcNHBmRnPWJCJwzVZwnFyJ53eYE
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:

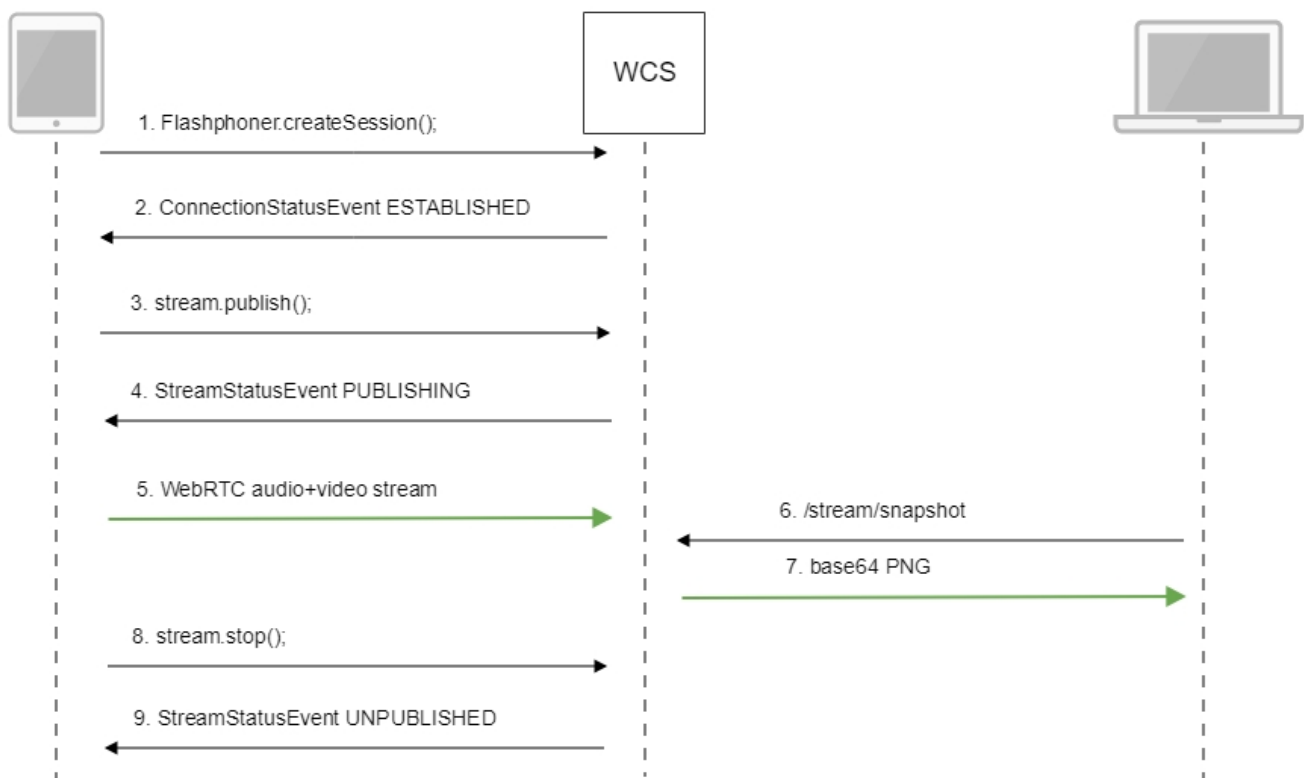


## Последовательность выполнения операций (Call flow)

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();
```

## Автоматическое создание превью опубликованного потока

При необходимости, превью каждого потока, поддерживаемого формата, опубликованного на сервере, могут создаваться автоматически. Эта возможность включается при помощи настройки в файле [flashphoner.properties](#)

```
snapshot_auto_enabled=true
```

Расположение кадров превью задается настройкой

```
snapshot_auto_dir=/usr/local/FlashphonerWebCallServer/snapshots
```

В указанном каталоге для опубликованного потока создается подкаталог с именем, соответствующим идентификатору медиасессии (по умолчанию)

```
snapshot_auto_naming=mediaSessionId
```

или имени потока

```
snapshot_auto_naming=streamName
```

Кадры превью в каталоге нумеруются последовательно и создаются с периодичностью, заданной при помощи настройки

```
snapshot_auto_rate=30
```

В этом случае будет создано превью каждого 30 кадра.

Для экономии дискового пространства, может быть задано ограничение на количество хранимых кадров превью при помощи настройки

```
snapshot_auto_retention=20
```

В этом случае в каталоге для потока будут сохранены последние 20 кадров превью.

Если поток с таким же именем публикуется повторно, нумерация кадров превью будет продолжена.