

To another RTMP server

- [Overview](#)
 - [Supported platforms and browsers](#)
 - [Supported codecs](#)
 - [RTMP server authentication](#)
 - [Operation flowchart](#)
- [REST queries](#)
 - [REST-methods and response statuses](#)
 - [Parameters](#)
 - [Sending the REST query to the WCS server](#)
- [JavaScript API](#)
- [Server configuration](#)
- [Parameters passing in server URL](#)
 - [Stream name passing in server URL](#)
- [Call flow](#)

Overview

Upon request, Web Call Server converts a WebRTC audio and video stream to RTMP and sends it to the specified RTMP server. This way you can run a broadcasting from a web page to [Facebook](#), [YouTube Live](#), [Wowza](#), [Azure Media Services](#) and other live video services.

Republishing of an RTMP stream can be made using [REST queries](#) or [JavaScript API](#).

Supported platforms and browsers

	Chrome	Firefox	Safari 11	Edge
Windows	+	+		+
Mac OS	+	+	+	
Android	+	+		
iOS	-	-	+	

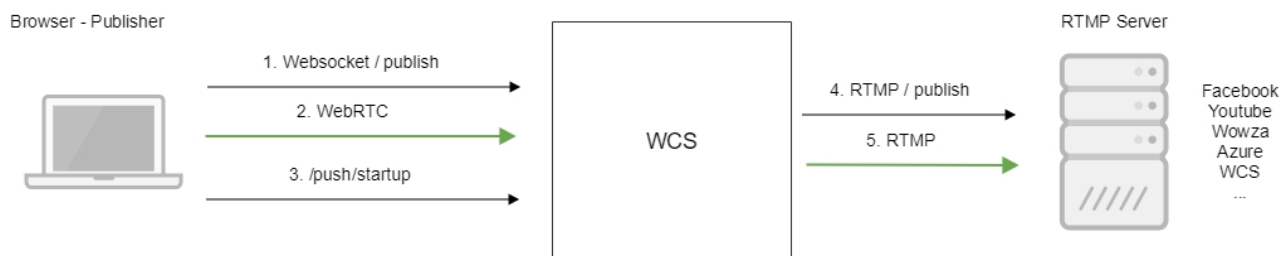
Supported codecs

- Video: H.264
- Audio: AAC, G.711, Speex 16

RTMP server authentication

Supported. Specify the name and password in the URL of the server, for example [rtmp://name:password@server:1935/live](#)

Operation flowchart



1. The browser connects to the server via the Websocket protocol and sends the publish command.
2. The browser captures the microphone and the camera and sends the WebRTC stream to the server.
3. The REST client sends the /push/startup query from the browser.
4. The WCS server publishes the RTMP stream on the RTMP server at the URL specified in the query.
5. The WCS server sends the RTMP stream.

REST queries

Republishing a video stream to another server can be performed using REST queries.

A REST query must be an HTTP/HTTPS POST query in the following form:

- HTTP: <http://streaming.flashphoner.com:8081/rest-api/push/startup>
- HTTPS: <https://streaming.flashphoner.com:8444/rest-api/push/startup>

Where:

- streaming.flashphoner.com- is the address of the WCS server
- 8081 - is the standard REST / HTTP port of the WCS server
- 8444- is the standard HTTPS port
- rest-api- is the required prefix
- /push/startup- is the REST-method used

REST-methods and response statuses

REST-method	Example of REST query body	Example of response	Response statuses	Description
/push /startup	<pre>{ "streamName": "name", "rtmpUrl": "rtmp://localhost :1935/live", "options": {} }</pre>	<pre>{ "mediaSessionId": "eume87rjk3df1i9u 14elffga6t", "streamName": "rtmp_name", "rtmpUrl": "rtmp://localhost :1935/live", "width": 320, "height": 240, "muted": false, "soundEnabled": false, "options": {} }</pre>	409 - Conflict 500 - Internal error	Creates a transponder that subscribes to the given stream and sends media traffic to the specified rtmpUrl. The name of the stream specified in the query can be the name of an already published stream or the name reserved when the SIP call was created (to send media traffic received from SIP). If a transponder for the given stream and rtmpUrl already exists, 409 Conflict is returned.
/push/find	<pre>{ "streamName": "name", "rtmpUrl": "rtmp://localhost :1935/live", }</pre>	<pre>[{ "mediaSessionId": "eume87rjk3df1i9u 14elffga6t", "streamName": "rtmp_name", "rtmpUrl": "rtmp://localhost :1935/live", "width": 320, "height": 240, "muted": false, "soundEnabled": false, "options": {} }]</pre>	404 - Transponder not found 500 - Internal error	Find transponders by a filter

/push /find_all		<pre>[{ "mediaSessionId": "eume87rjk3df1i9u 14elffga6t", "streamName": "rtmp_name", "rtmpUrl": "rtmp://localhost :1935/live", "width": 320, "height": 240, "muted": false, "soundEnabled": false, "options": {} }]</pre>	404 - Not found any transponder 500 - Internal error	Find all transponders
/push /terminate	<pre>{ "mediaSessionId": "eume87rjk3df1i9u 14elffga6t" }</pre>		404 - Not found transponder 500 - Internal error	Terminate operation of the transponder
/push /mute	<pre>{ "mediaSessionId": "eume87rjk3df1i9u 14elffga6t" }</pre>	void	404 - Not found transponder 500 - Internal error	Turn off audio
/push /unmute	<pre>{ "mediaSessionId": "eume87rjk3df1i9u 14elffga6t" }</pre>	void	404 - Not found transponder 500 - Internal error	Turn on audio
/push /sound_on	<pre>{ "mediaSessionId": "eume87rjk3df1i9u 14elffga6t" "soundFile": "test.wav" "loop": true }</pre>	void	404 - Not found transponder 404 - No such file 500 - Internal error	Insert audio from a RIFF WAV file located in the /usr/local/FlashphonerWebCallServer/media/ directory on the WCS server
/push /sound_off	<pre>{ "mediaSessionId": "eume87rjk3df1i9u 14elffga6t" }</pre>	void	404 - Not found transponder 500 - Internal error	Stop inserting audio from the file

Parameters

Parameter name	Description	Example
streamName	Name of the republished stream	streamName

rtmpUrl	URL of the server the stream is republished to	rtmp://localhost:1935/live
options	Transponder options	{"action": "mute"}
mediaSessionId	Unique identifier of the transponder	eume87rjk3df1i9u14elffga6t
width	Image width	320
height	Image height	240
muted	Is sound muted	true
soundEnabled	Is sound enabled	true
soundFile	Sound file	test.wav
loop	Loop playback	false

The `options` parameter can be used to turn off audio or insert audio from a file when creating a transponder.

Example,

```
"options": {"action": "mute"}
"options": {"action": "sound_on", "soundFile": "sound.wav", "loop": true}
```

Sending the REST query to the WCS server

To send the REST query to the WCS server, use a [REST-client](#).

JavaScript API

Using WebSDK you can republish a stream to an RTMP server upon creation, similar to the [SIP as stream](#) function. Usage example for this method is available in the WebRTC as RTMP web application.

[webrtc-as-rtmp-republishing.html](#)

[webrtc-as-rtmp-republishing.js](#)

1. When a stream is created, the method `session.createStream()` receives the parameter `rtmpUrl` that specifies the URL of the RTMP server that accepts the broadcast. The name of the stream is specified in compliance with rules of the RTMP server.

code:

```
function startStreaming(session) {
  var streamName = field("streamName");
  var rtmpUrl = field("rtmpUrl");
  session.createStream({
    name: streamName,
    display: localVideo,
    cacheLocalResources: true,
    receiveVideo: false,
    receiveAudio: false,
    rtmpUrl: rtmpUrl
    ...
  }).publish();
}
```

Republishing of the stream starts directly after it is successfully published on the WCS server.

Server configuration

When WCS creates an RTMP transponder it automatically adds a prefix to the republished stream as set in the [flashphoner.properties](#) file:

```
rtmp_transponder_stream_name_prefix=rtmp_
```

The option

turns on a possibility to pass some request parameters to RTMP server.

Parameters passing in server URL

It is possible to pass some parameters to server, to which a stream should be republished. Parameters to pass are specified in server URL, e.g.

```
rtmp://myrtmpserver.com:1935/app_name/?user=user1&pass=pass1
```

or, if a stream supposed to be published to a specified instance of RTMP server application

```
rtmp://myrtmpserver.com:1935/app_name/app_instance/?user=user1&pass=pass1
```

Where

- myrtmpserver.com is the RTMP server name
- app_name is the application on the RTMP server name
- app_instance is the instance name of the RTMP server application

Stream name is set in REST query /push/startup parameter 'streamName' or in corresponding stream creation option.

This is the example on RTMP connection establishing with query parameters passing

File
Edit
View
Go
Capture
Analyze
Statistics
Telephony
Wireless
Tools
Help

rtmp
Expression... +

No.	Time	Source	Destination	Protocol	Length	Info
3	0.051685	95.191.131.37	127.0.0.1	RTMP	1603	Handshake C0+C1
5	0.054924	95.191.131.37	127.0.0.1	RTMP	1602	Handshake C2
6	0.068103	95.191.131.37	127.0.0.1	RTMP	330	connect('live/?user1&pass1')
7	0.108388	95.191.131.37	127.0.0.1	RTMP	119	releaseStream('rtmp_stream1')
8	0.108523	95.191.131.37	127.0.0.1	RTMP	115	FCPublish('rtmp_stream1')
9	0.108643	95.191.131.37	127.0.0.1	RTMP	103	createStream()
10	0.126359	95.191.131.37	127.0.0.1	RTMP	120	publish('rtmp_stream1')

> Frame 6: 330 bytes on wire (2640 bits), 330 bytes captured (2640 bits)

> Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)

> Internet Protocol Version 4, Src: 95.191.131.37, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 33002, Dst Port: 1935, Seq: 3074, Ack: 3074, Len: 264

> Real Time Messaging Protocol (AMF0 Command connect('live/?user1&pass1'))

```

0000  00 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 .....E.
0010  01 3c b3 94 40 00 40 06 24 42 5f bf 83 25 7f 00 <...@. $0_..%
0020  00 01 80 ea 07 8f b5 21 46 95 f2 7a ce a4 80 18 .....! F..z...
0030  05 55 63 14 00 00 01 01 08 0a e9 c4 89 60 e9 c4 .Uc.....'.
0040  89 51 03 00 00 00 00 00 fb 14 00 00 00 00 02 00 .Q.....
0050  07 63 6f 6e 6e 65 63 74 00 3f f0 00 00 00 00 00 .connect.?.
0060  00 03 00 03 61 70 70 02 00 11 6c 69 76 65 2f 3f ....app..live/?
0070  75 73 65 72 31 26 70 61 73 73 31 00 08 66 6c 61 user1&pa ss1..fla
0080  73 68 56 65 72 02 00 08 46 4d 4c 45 2f 33 2e 30 shVer... FILE/3.0
0090  00 95 74 63 55 72 6c 02 00 30 72 74 6d 70 3a 2f ..tcUrl..0rtmp:/
00a0  2f 70 35 2e 66 6c 61 73 68 70 68 6f 6e 65 72 2e /p5.flas hphoner.

```

Frame (330 bytes)
Unchunked RTMP (263 bytes)

Stream name passing in server URL

In some cases, a stream publishing name should be passed in the server URL. To do this, the following option must be set in `flashphoner.properties` file

```
rtmp_transponder_full_url=true
```

Then, the URL to publish should be set in REST query `/push/startup 'rtmpUrl'` parameter or in corresponding stream creation option like this:

```
rtmp://myrtmpserver.com:1935/app_name/stream_name
```

or, to publish to another application instance

```
rtmp://myrtmpserver.com:1935/app_name/app_instance/stream_name
```

In this case, 'streamName' parameter or REST query /push/startup or corresponding stream creation option is ignored.

Automatic republishing to a specified RTMP server

WCS server can automatically republish all the streams published to a specified RTMP server. To activate this feature, set the next options in [flashphoner.properties](#) file:

```
rtmp_push_auto_start=true  
rtmp_push_auto_start_url=rtmp://rtmp.server.com:1935/
```

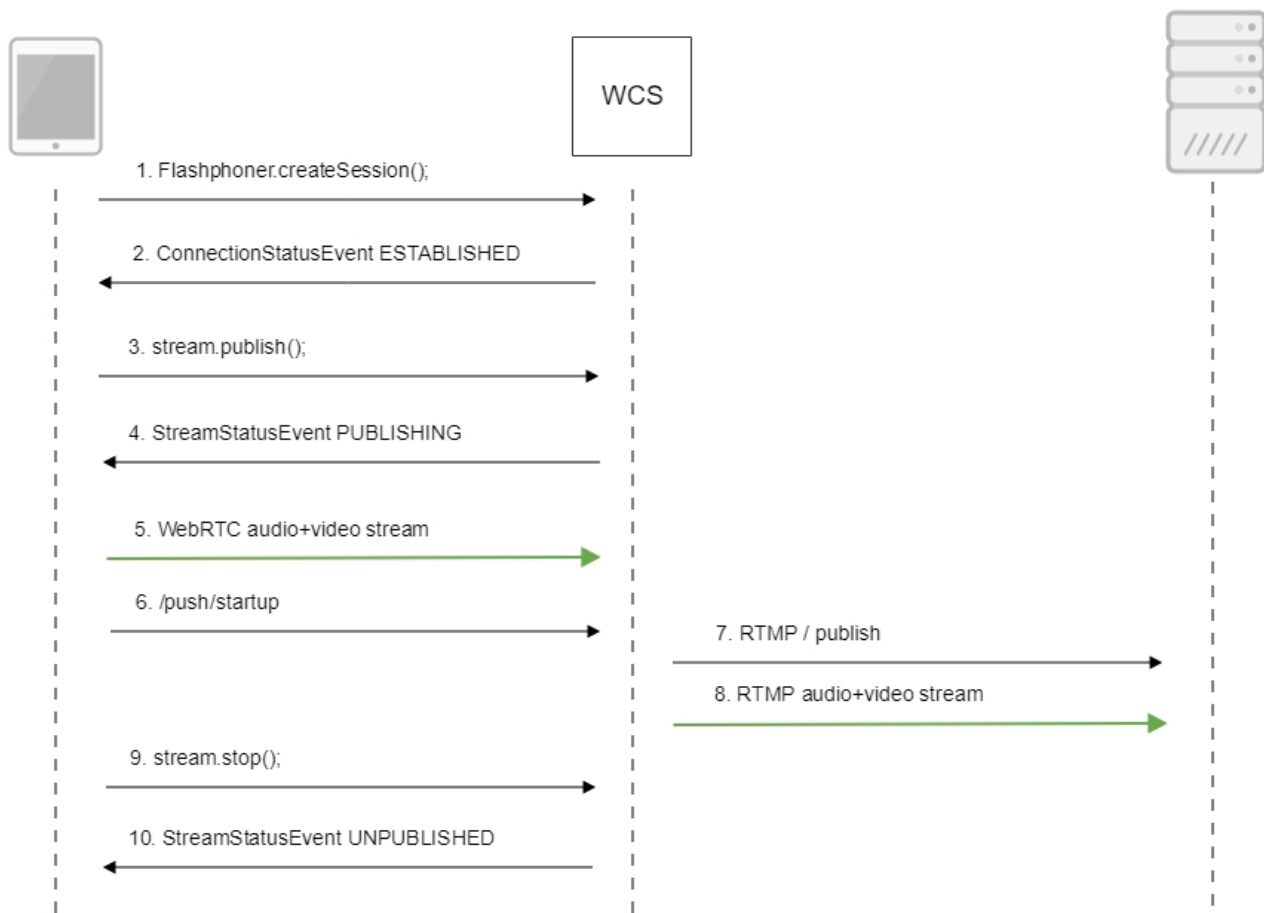
wherertmp.server.com is RTMP server name to republish all streams from WCS

Call flow

Below is the call flow when using the Two Way Streaming example to publish a stream and the REST client to send the /push/startup query:

[two_way_streaming.html](#)

[two_way_streaming.js](#)



1. Establishing a connection to the server.

Flashphoner.createSession();[code](#)

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function (session) {
    setStatus("#connectStatus", session.status());
    onConnected(session);
}).on(SESSION_STATUS.DISCONNECTED, function () {
    setStatus("#connectStatus", SESSION_STATUS.DISCONNECTED);
    onDisconnected();
}).on(SESSION_STATUS.FAILED, function () {
    setStatus("#connectStatus", SESSION_STATUS.FAILED);
    onDisconnected();
});
```

2. Receiving from the server an event confirming successful connection.

ConnectionStatusEvent ESTABLISHED[code](#)

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function (session) {
    setStatus("#connectStatus", session.status());
    onConnected(session);
}).on(SESSION_STATUS.DISCONNECTED, function () {
    ...
}).on(SESSION_STATUS.FAILED, function () {
    ...
});
```

3. Publishing the stream.

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

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

4. Receiving from the server and event confirming successful publishing of the stream.

StreamStatusEvent, status PUBLISHING[code](#)

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

5. Sending the audio-video stream via WebRTC

6. Sending the /push/startup query

```
http://demo.flashphoner.com:9091/rest-api/push/startup
{
  "streamName": "testStream",
  "rtmpUrl": "rtmp://demo.flashphoner.com:1935/live/testStream"
}
```

7. Establishing a connection via RTMP with the specified server, publishing the stream

8. Sending the audio-video stream via RTMP

9. Stopping publishing the stream.

`stream.stop();`[code](#)

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

10. Receiving from the server an event confirming unpublishing of the stream.

`StreamStatusEvent`, status `UNPUBLISHED`[code](#)

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