To another RTMP server

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 toFacebook,YouTube Live,Wowza,Azure Media Servicesand other live video services.

Republishing of an RTMP stream can be made usingREST queriesor 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 examplertmp://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. TheWCS 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", "rtmpTransponderFullUrl": false "options": {} }</pre>	<pre>{ "mediaSessionId": "eume87rjk3df1i9u14elffga6t", "streamName": "rtmp_name", "rtmpUrl": "rtmp://localhost:1935/live", "width": 320, "height": 240, "muted": false, "soundEnabled": false, "options": {} }</pre>	400 - Bad request 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. If rtmpUrl is not set, or is set incorrectly and cannot be resolved by DNS, 400 Bad request is returned
/push/find	<pre>{ "streamName": "name", "rtmpUrl": "rtmp://localhost:1935/live", }</pre>	<pre>[{ "mediaSessionId": "eume87rjk3df1i9u14elffga6t", "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": "eume87rjk3df1i9u14elffga6t", "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": "eume87rjk3df1i9u14elffga6t" }</pre>		404 - Not found transponder 500 - Internal error	Terminate operation of the transponder
/push /mute	<pre>{ "mediaSessionId": "eume87rjk3df1i9u14elffga6t" }</pre>	void	404 - Not found transponder 500 - Internal error	Turn off audio
/push /unmute	<pre>{ "mediaSessionId": "eume87rjk3df1i9u14elffga6t" }</pre>	void	404 - Not found transponder 500 - Internal error	Turn on audio
/push /sound_on	<pre>{ "mediaSessionId": "eume87rjk3df1i9u14elffga6t" "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": "eume87rjk3df1i9u14elffga6t" }</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
rtmpFlashVersion	RTMP subscriber Flash version	LNX 76.219.189.0
options	Transponder options	{"action": "mute"}
mediaSessionId	Unique identifier of the transponder	eume87rjk3df1i9u14elffga6t
width	Image width	320
height	Image height	240
bitrate	Video bitrate, kbps	500
keyFrameInterval	Video keyframe interval	60
fps	Video framerate	30
muted	Is sound muted	true
soundEnabled	Is sound enabled	true
soundFile	Sound file	test.wav
loop	Loop playback	false
rtmpTransponderFullUrl	Take stream name to publish to RTMP server from RTMP URL	false

Parameters added since build5.2.785: rtmpFlashVersion, keyFrameInterval and fps.

Since build 5.2.1043 bitrateparameter is added.

Theoptionsparameter 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}
```

Stream transcoding while republishing

Since build5.2.560, if picture width and height are not set in/push/startup query parameters

```
{
 "streamName": "name",
 "rtmpUrl": "rtmp://localhost:1935/live"
}
```

or they are set to 0

```
{
 "streamName": "name",
 "rtmpUrl": "rtmp://localhost:1935/live",
 "width": 0,
 "height": 0
}
```

transcoding will not be enabled for stream republishing.

If picture height is set explicitly (for example, if destination server does not accept streams below 720p)

```
{
 "streamName": "name",
 "rtmpUrl": "rtmp://localhost:1935/live",
 "width": 1280,
 "height": 720
}
```

the stream will be transcoded and pushed to destination server indefined resolution.

Specified width is applied only if picture aspect ratio preserving is disabled, and height is also specified. If only width parameter is passed - without height - it is not applied, and the stream is not transcoded.

Since build5.2.785, there are two more parameters enabling transcoding: keyFrameInterval and fps.Since build 5.2.1043 bitrate parameter is added which also enables stream transcoding while republishing.

Therefore, stream will be transcoded while republishing with any of the following parameters:

```
{
"streamName": "name",
"rtmpUrl": "rtmp://localhost:1935/live",
"height": 240,
"keyFrameInterval": 60,
"fps": 30,
"bitrate": 500
}
```

Set stream name to publish to RTMP server

By default, a stream will be published to RTMP server with the same name as it is publishing on WCS, and the prefix `rtmp_`, for example`rtmp_test`.This behaviour can be changed by the following parameters

```
rtmp_transponder_full_url=true
rtmp_transponder_stream_name_prefix=
```

But, these settings are applyed to all the republishings, and require server restart. That's why since build5.2.860the /push/startup query parameter is added to allow to define full RTMP URL, including stream name on RTMP server, regardless of server settings

```
POST /rest-api/push/startup HTTP/1.1
Host: localhost:8081
Content-Type: application/json

{
 "streamName":"stream1",
 "rtmpUrl":"rtmp://rtmp.flashphoner.com:1935/live/test",
 "rtmpTransponderFullUrl":true
}
```

In this case, the stream will be published to RTMP server with the name defined in RTMP URL even with default WCS settings.

Sending the REST query to the WCS server

To send the REST query to the WCS server, use aREST-client.

JavaScript API

Using Web SDK you can republish a stream to an RTMP server upon creation, similar to theSIP as streamfunction. 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 theflashphoner.propertiesfile:

```
rtmp_transponder_stream_name_prefix=rtmp_
```

If the server the stream is republished to has certain requirements to the name (Facebook,YouTube), this line must be commented out.

The option

```
rtmp_transponder_full_url=true
```

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

A network interface to bind RTMP client for republishing may be set with the following parameter

```
rtmp_publisher_ip=127.0.0.1
```

In this case, RTMP will be republished to localhost only.

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

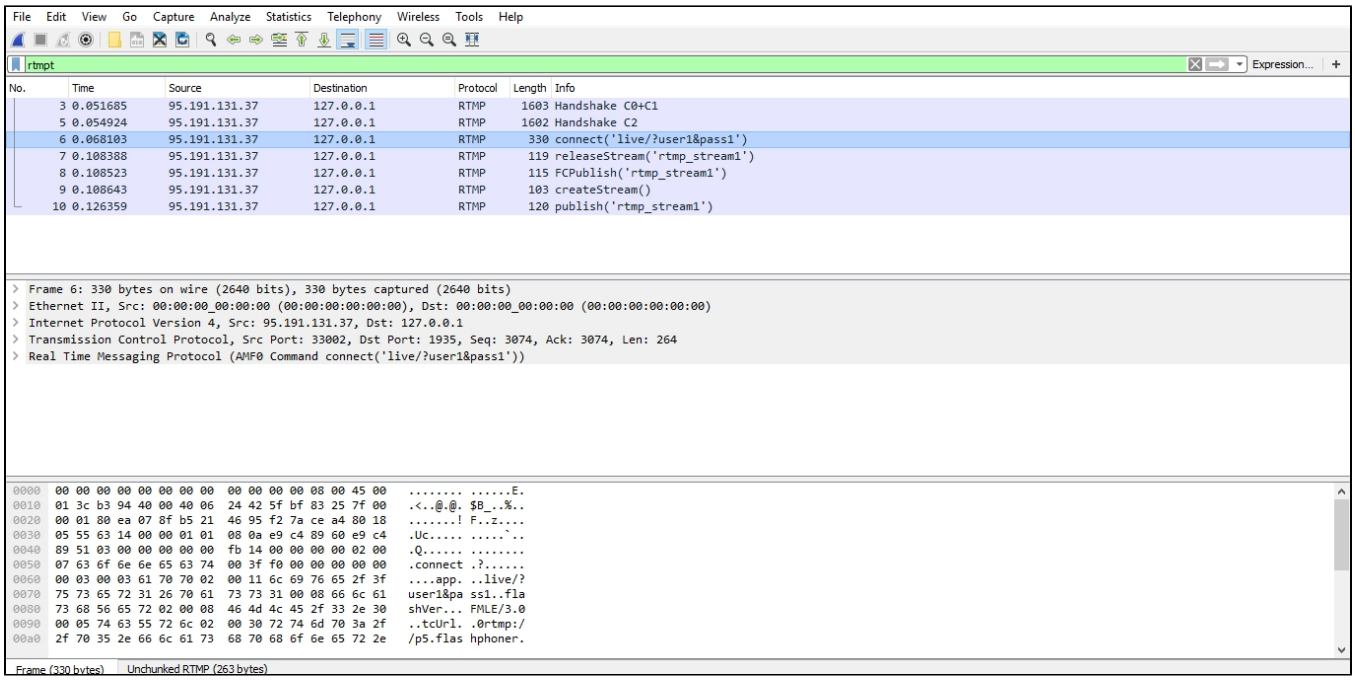
```
rtmp://myrtmpserver.com:1935/app_name/app_instance/?user=user1&pass=pass1
```

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



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 inflashphoner.propertiesfile

```
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 thepublished streams to a specified RTMP server. To activate this feature, set the following options inflashphoner.propertiesfile:

```
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.

This feature is supposed to be used for debug, not in production.

Since build 5.2.1110 it is possible to set authentication parameters

```
rtmp_push_auto_start_url=rtmp://user:password@rtmp.server.com:1935/live
```

or

```
rtmp_push_auto_start_url=rtmp://rtmp.server.com:1935/live?username=user&password=pwd
```

Parameters will be passed in RTMP connect command.

Known limits

Only one RTMP URL can be used for automatic republishing.

Automatic reconnection when channel is closed

When RTMP stream is published to another RTMP server, connection to this server may be interrupted and channel may be closed for some reasons (destination server restart, network problems etc). In this case automatic reconnection and RTMP stream republishing can be enabled with the following parameter inflashphoner.propertiesfile:

```
rtmp_push_restore=true
```

Reconnection attempts maxumum count and interval between attempts in milliseconds should also be set

```
rtmp_push_restore_attempts=3
rtmp_push_restore_interval_ms=5000
```

In this case, 3 attempts will be made to reconnect to RTMP server with 5 seconds interval. After that, reconnection stops.

RTMP outgoing stream buffering

Since build5.2.700outgoing RTMP stream can be buffered. This icreases translation latency, but allows to play the stream more smooth from destination RTMP server. Bufferization is enabled with the following parameter

```
rtmp_out_buffer_enabled=true
```

The following bufferization parameters can be tuned

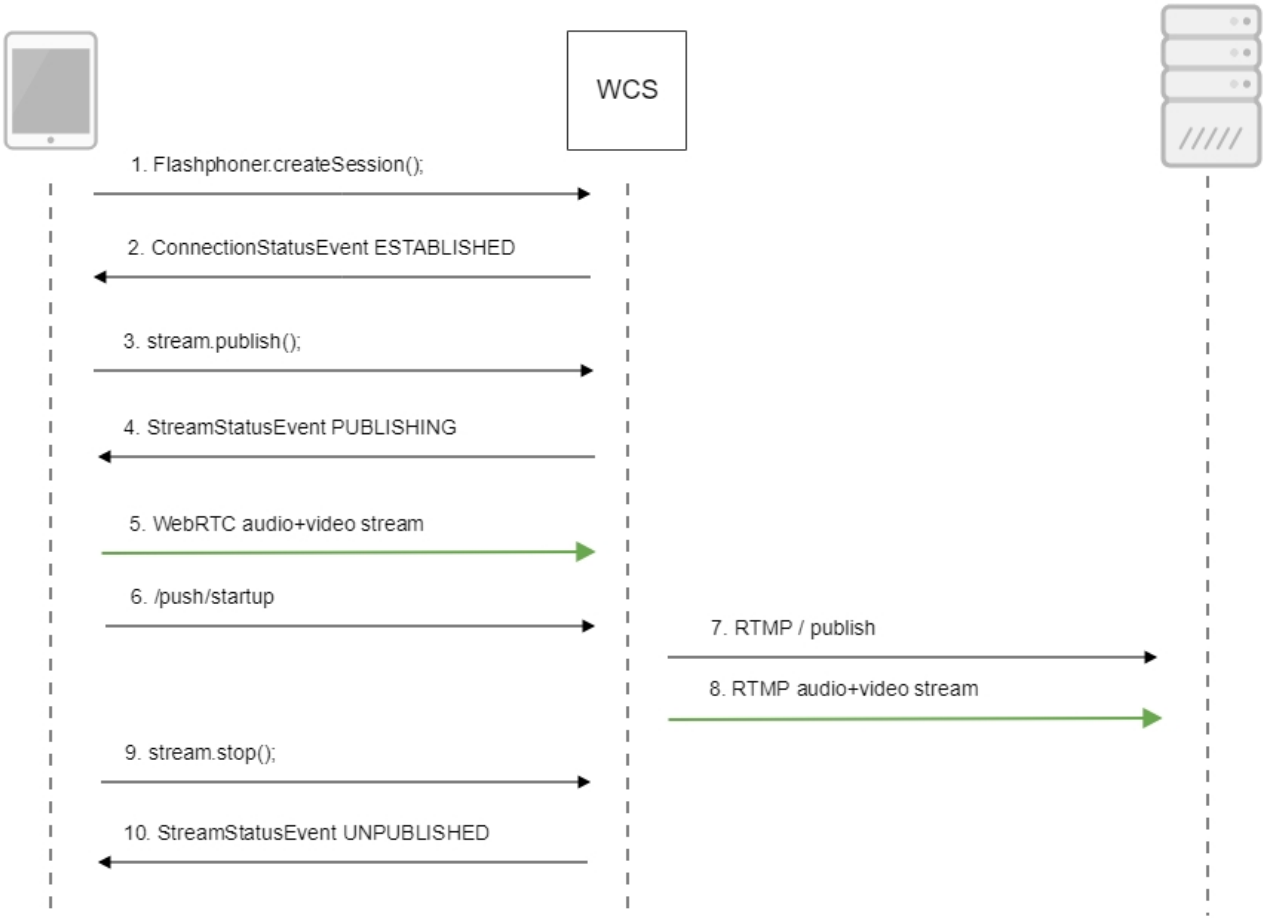
Parameter	Default value	Description
rtmp_out_buffer_start_size	300	Stream buffer start size, мс
rtmp_out_buffer_initial_size	2000	Stream buffer initial size, мс
rtmp_out_buffer_polling_time	50	Buffer polling timeout, мс
rtmp_out_buffer_max_bufferings_allowed	-1	Maximum stream bufferings allowed, unlimited by default

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 ESTABLISHEDcode

```
    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 PUBLISHINGcode

```
   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 UNPUBLISHEDcode

```
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();
```

Known issues

1. When stream is republished to RTMP server and is played from this server in JWPlayer, stream picture aspect ration can be distorted

Symptoms: playing stream aspect ratio in JWPlayer differs from published one

Solution: enable metadata sending while stream republishing as RTMP

```
rtmp_transponder_send_metadata=true
```

2. Republishing may fail if RTMP destination server requires specific Flash version

Symptoms: RTMP handshake fails, the channel is closed with RTMP error in WCS server log

Solution: specify RTMP subscriber Flash version, either using rtmp_flash_ver_subscriber setting in flashphoner.properties, or rtmpFlashVersion parameter in republishing REST request

For example, for republishing to Periscope:

```
rtmp_flash_ver_subscriber = LNX 76.219.189.0
```

3. RTMP destination server may require specific stream parameters: bitrate, keyframe interval, or framerate

Symptoms: e.g., Periscope displays warnings about not corresponding to the recommended settings

Solution: set specific constraints to the source stream (e.g., for audio bitrate) and specify required parametersin republishing REST request (keyFrameInterval and fps)

4. When republishing FullHD, 2K, 4K streams with big frame size, data packets to send may not fit to socket buffer, this leads to artifacts in some players

Symptoms: artifacts occur while playing republished RTMP stream via good channel

Solution: enable RTMP packets buffering with the parameter

```
rtmp.server_buffer_enabled=true
```