

Ретрансляция на другой RTMP сервер

Описание

Web Call Server по запросу конвертирует WebRTC аудио и видео поток в RTMP и отправляет на указанный RTMP-сервер. Таким образом может быть создана трансляция с веб-страницы на [Facebook](#), [YouTube Live](#), [Wowza](#), [Azure Media Services](#) и другие сервисы, транслирующие живое видео.

Ретрансляция RTMP-потока может быть организована как при помощи REST-вызовов, так и средствами JavaScript API.

Поддерживаемые платформы и браузеры

	Chrome	Firefox	Safari	Edge
Windows	✓	✓	✗	✓
Mac OS	✓	✓	✓	✓
Android	✓	✓	✗	✓
iOS	✓	✓	✓	✓

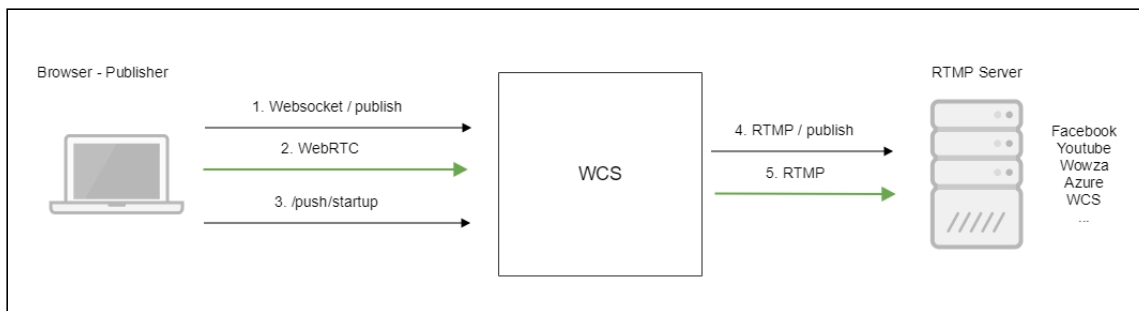
Поддерживаемые кодеки

- Видео: H.264
- Аудио: AAC, G.711, Speex 16

Аутентификация на RTMP-сервере

Поддерживается, имя и пароль необходимо указывать в URL сервера, например `rtmp://name:password@server:1935/live`

Схема работы



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду `publishStream`.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток на сервер.
3. REST-клиент из браузера отправляет запрос `/push/startup`.
4. WCS-сервер публикует RTMP поток на указанный в запросе URL RTMP-сервера.
5. WCS-сервер передает RTMP поток.

REST API

Ретрансляция видеопотока на другой RTMP-сервер производится при помощи REST-вызовов

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

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

Здесь:

- `streaming.flashphoner.com` - адрес WCS-сервера
- `8081` - стандартный REST / HTTP порт WCS-сервера
- `8444` - стандартный HTTPS порт
- `rest-api` - обязательный префикс
- `/push/startup` - используемый REST-вызов

REST-вызовы и статусы ответа

REST метод	Тело запроса	Тело ответа	Статусы ответа	Описание

REST метод	Тело запроса	Тело ответа	Статусы ответа	Описание
<code>/push/startup</code>	<pre>{ "streamName": "name", "rtmpUrl": "rtmp://localhost:1935/live", "rtmpTransponderFullUrl": false, "rtmpFlashVersion": "LNX 76.219.189.0", "options": {} }</pre>	<pre>{ "mediaSessionId": "eume87rjk3df1i9u14elffga6t", "streamName": "rtmp_name", "rtmpUrl": "rtmp://localhost:1935/live", "width": 320, "height": 240, "muted": false, "soundEnabled": false, "options": {} }</pre>	<p>400 Bad request 409 Conflict 500 Internal error</p>	<p>Создать транспондер, который подписывается на указанный поток и отправляет медиатрафик на указанный <code>rtmpUrl</code>. Имя потока, указанное в запросе, может быть именем уже публикуемого потока или именем зарезервированным при создании SIP-звонка (для отправки медиатрафика, полученного со стороны SIP). Если транспондер для такого потока и <code>rtmpUrl</code> уже существует, возвращается <code>409 Conflict</code>. Если <code>rtmpUrl</code> не указан, или указан неверно и не найден по DNS, возвращается <code>400 Bad request</code></p>

REST метод	Тело запроса	Тело ответа	Статусы ответа	Описание
<code>`/push/find`</code>	<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 Not found 500 Internal error	Найти транспондеры по фильтру

REST метод	Тело запроса	Тело ответа	Статусы ответа	Описание
/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 500 Internal error	Найти все транспондеры

REST метод	Тело запроса	Тело ответа	Статусы ответа	Описание
<code>`/push/terminate`</code>	<pre>{ "mediaSessionId": "eume87rjk3df1i9u14elffga6t" }</pre>		404 Not found 500 Internal error	Завершить работу транспондера
<code>`/push/mute`</code>	<pre>{ "mediaSessionId": "eume87rjk3df1i9u14elffga6t" }</pre>		404 Not found 500 Internal error	Выключить аудио
<code>`/push/unmute`</code>	<pre>{ "mediaSessionId": "eume87rjk3df1i9u14elffga6t" }</pre>		404 Not found 500 Internal error	Включить аудио

REST метод	Тело запроса	Тело ответа	Статусы ответа	Описание
<code>`/push/sound_on`</code>	<pre>{ "mediaSessionId": "eume87rjk3df1i9u14e1ffga6t" "soundFile": "test.wav" "loop": true }</pre>		404 Not found 500 Internal error	Вставить аудио из RIFF WAV файла из директории <code>`/usr/local/FlashphonerWebCallServer/media/`</code> на WCS-сервере
<code>`/push/sound_off`</code>	<pre>{ "mediaSessionId": "eume87rjk3df1i9u14e1ffga6t" }</pre>		404 Not found 500 Internal error	Завершить вставку аудио из файла/td>

Параметры

Параметр	Описание	Пример
streamName	Имя ретранслируемого потока	<code>`streamName`</code>
rtmpUrl	URL сервера, на который производится ретрансляция	<code>`rtmp://localhost:1935/live`</code>
rtmpFlashVersion	Версия RTMP протокола	<code>`LNX 76.219.189.0`</code>

Параметр	Описание	Пример
options	Опции транспондера	<code>{"action": "mute"}</code>
mediaSessionId	Уникальный идентификатор транспондера	<code>eume87rjk3df1i9u14elffga6t</code>
width	Ширина изображения	<code>320</code>
height	Высота изображения	<code>240</code>
bitrate	Битрейт видео, кбит/с	<code>500</code>
keyFrameInterval	Интервал ключевых кадров видео	<code>60</code>
fps	Частота кадров видео	<code>30</code>
muted	Звук отключен	<code>true</code>
soundEnabled	Вставка звука активна	<code>true</code>
soundFile	Файл аудио для вставки	<code>test.wav</code>
loop	Повтор файла	<code>false</code>
rtmpTransponderFullUrl	Брать имя для публикации на RTMP сервере и из RTMP URL	<code>false</code>

Параметр `options` может быть использован, чтобы выключить аудио или вставить аудио из файла при создании транспондера.

Например,

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

Транскодинг при ретрансляции потока

Начиная со сборки [5.2.560](#), если в параметрах запроса `/push/startup` не указывать ширину и высоту картинки

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

либо указать их равными 0

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

транскодинг для ретранслируемого потока не включается.

Если задать высоту картинки явным образом (например, если сервер не принимает потоки меньше чем 720p)

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

поток при ретрансляции будет транскодирован и отправлен на целевой сервер в указанном разрешении.

Заданная ширина картинки применяется, только если отключено [сохранение соотношения сторон](#), и задана и высота картинки. Если передать только параметр `width` без `height`, то он не применяется, и транскодинг для ретранслируемого потока не включается.

Начиная со сборки [5.2.785](#), добавлены другие два параметра, с которыми включается транскодинг: `keyFrameInterval` и `fps`. В сборке [5.2.1043](#) добавлен параметр `bitrate`, который также включает транскодинг при ретрансляции.

Таким образом, поток при ретрансляции бode транскодирован с любыми из следующих параметров:

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

Указание имени потока для публикации на RTMP сервере

По умолчанию, поток будет опубликован на RTMP сервере с тем же именем, под которым он опубликован на WCS, и префиксом `rtmp_`, например `rtmp_test`. Это поведение меняется настройками

```
rtmp_transponder_full_url=true  
rtmp_transponder_stream_name_prefix=
```

Однако, эти настройки применяются ко всем ретрансляциям, и требуют перезапуска сервера. Поэтому в сборке 5.2.860 добавлен параметр запроса `/push/startup`, позволяющий указать полный RTMP URL, включая имя потока на RTMP сервере

```
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  
}
```

В этом случае поток будет опубликован на RTMP сервере под именем, указанным в RTMP URL, даже при настройках WCS по умолчанию

JavaScript API

При помощи WebSDK поток может быть ретранслирован на RTMP-сервер при создании, по аналогии с функцией [SIP as stream](#). Пример использования данного метода приведен в веб-приложении WebRTC as RTMP.

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

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

При создании потока методу `session.createStream()` передается параметр `rtmpUrl` с указанием URL RTMP-сервера, принимающего трансляцию. Имя потока указывается в соответствии с правилами RTMP-сервера.

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();  
}
```

Ретрансляция потока начинается сразу после его успешной публикации на WCS-сервере

Настройка сервера

При создании RTMP-транспондера, WCS автоматически добавляет к имени ретранслируемого потока префикс в соответствии с настройкой в файле [flashphoner.properties](#):

```
rtmp_transponder_stream_name_prefix=rtmp_
```

Если сервер, на который ретранслируется поток, предъявляет определенные требования к имени ([Facebook](#), [YouTube](#)), данная строка должна быть закомментирована.

Настройка

```
rtmp_transponder_full_url=true
```

включает возможность передачи параметров запроса серверу, на который ретранслируется поток.

Указать сетевой интерфейс для трансляции RTMP можно при помощи параметра

```
rtmp_publisher_ip=127.0.0.1
```

В данном случае RTMP будет публиковаться только на localhost

Передача параметров в URL сервера

Существует возможность передать параметры запроса серверу, на который ретранслируется поток, указав их в URL сервера, например

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

или, если публикация предполагается в отдельный экземпляр приложения на RTMP-сервере

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

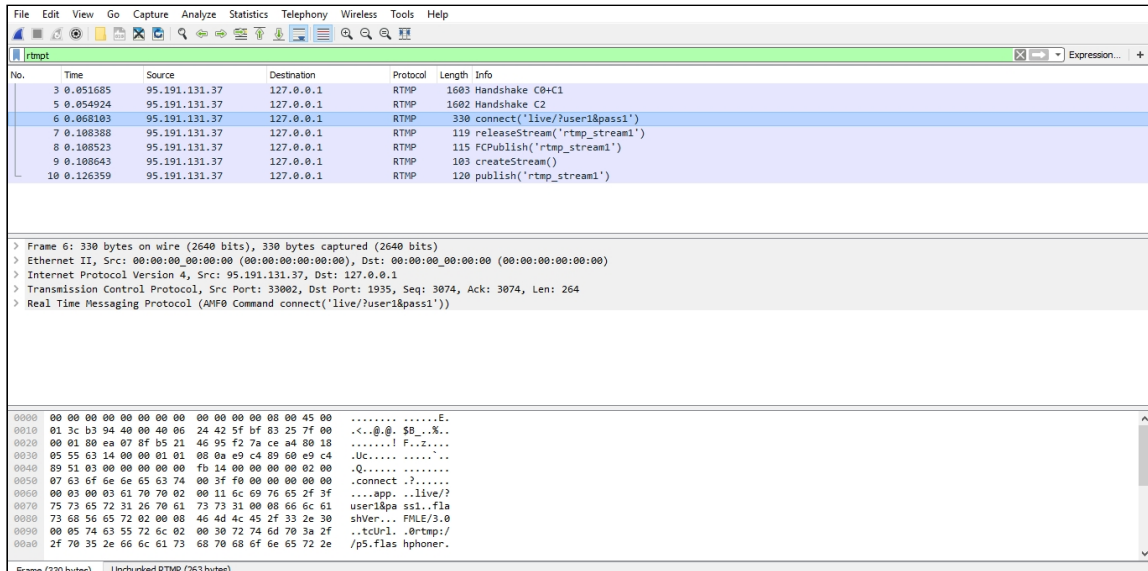
Здесь

- `myrtmpserver.com` - имя RTMP-сервера
- `app_name` - имя приложения на RTMP-сервере

- `app_instance` - имя экземпляра приложения на RTMP-сервере

Имя потока указывается в параметре `streamName` REST-запроса `/push/startup` или в соответствующей опции при создании потока.

Пример установки RTMP-соединения с передачей параметров запроса



Передача имени потока в URL

В некоторых случаях имя потока при публикации необходимо передать в URL. Для этого необходимо указать настройку в файле `flashphoner.properties`

```
rtmp_transponder_full_url=true
```

Тогда, для публикации в параметре `rtmpUrl` REST-запроса `/push/startup` или в соответствующей опции при создании потока указывается URL вида

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

или, для публикации в другой экземпляре приложения

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

В этом случае параметр `streamName` REST-запроса `/push/startup` или соответствующая опция при создании потока игнорируется.

Автоматическая ретрансляция на указанный сервер

WCS-сервер может автоматически ретранслировать все публикуемые на нем потоки на заданный RTMP-сервер. Для того, чтобы активировать эту возможность,

необходимо в файле [flashphoner.properties](#) указать следующие настройки:

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

Здесь `rtmp.server.com` - имя RTMP-сервера, на который должны ретранслироваться все потоки с WCS.

Warning

Эта возможность предназначена только для отладки, но не для промышленной эксплуатации.

В сборке [5.2.1110](#) добавлена возможность указать параметры аутентификации

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

или

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

Параметры будут переданы в команде RTMP `connect`.

Ограничения

Для автоматической ретрансляции может быть указан только один RTMP URL.

Автоматическое восстановление соединения при закрытии канала

Во время публикации RTMP-потока на другой RTMP-сервер, соединение может быть прервано и канал закрыт по различным причинам (перезапуск принимающего сервера, сетевые проблемы и т.п.). В этом случае может быть настроено автоматическое восстановление соединения и повторная публикация RTMP-потока при помощи параметра в файле [flashphoner.properties](#)

```
rtmp_push_restore=true
```

Необходимо также настроить количество попыток восстановить соединение и интервал между попытками:

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

В данном случае будет предпринято 3 попытки повторного подключения к RTMP серверу с интервалом 5 секунд. После этого попытки восстановить соединение прекращаются.

Буферизация исходящего RTMP-потока

В сборке 5.2.700 добавлена возможность буферизации исходящего RTMP потока. Это увеличивает задержку трансляции, но позволяет получить более плавное воспроизведение потока на сервере, куда поток ретранслируется. Буферизация включается при помощи параметра

```
rtmp_out_buffer_enabled=true
```

Могут быть настроены следующие параметры буферизации

Параметр	Значение по умолчанию	Описание
<code>`rtmp_out_buffer_start_size`</code>	300	Размер буфера при запуске буферизации для потока, мс
<code>`rtmp_out_buffer_initial_size`</code>	2000	Первоначальный размер буфера, распределяемый при инициализации, мс
<code>`rtmp_out_buffer_polling_time`</code>	50	Периодичность проверки буфера, мс
<code>`rtmp_out_buffer_max_bufferings_allowed`</code>	-1	Количество буферизаций для потока, по умолчанию не ограничено

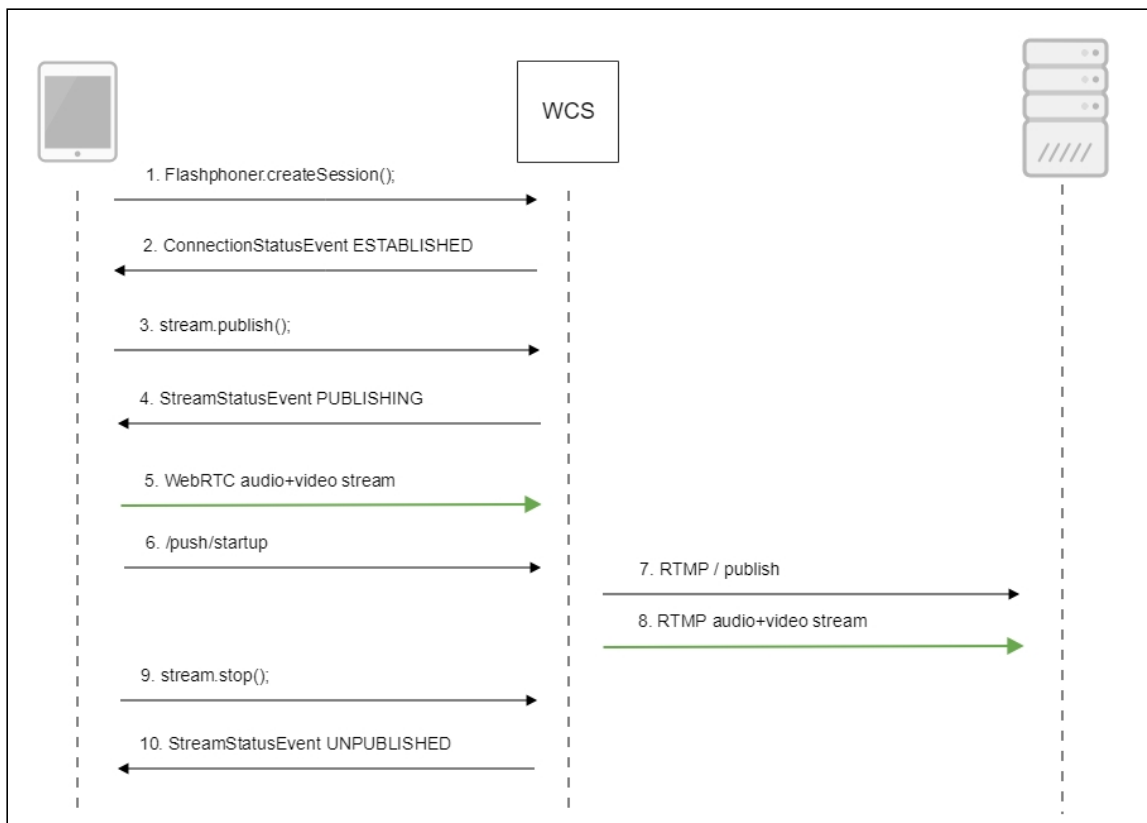
Последовательность выполнения операций

Ниже описана последовательность вызовов при использовании примера Two Way Streaming для публикации потока и REST-клиента для отправки запроса

```
/push/startup
```

[two_way_streaming.html](#)

[two_way_streaming.js](#)



1. Установка соединения с сервером

`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. Получение от сервера события, подтверждающего успешное соединение

`SESSION_STATUS.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. 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

`STREAM_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. Отправка аудио-видео потока по WebRTC

6. Отправка запроса `/push/startup`

```
http://demo.flashphoner.com:8081/rest-api/push/startup
{
  "streamName": "testStream",
  "rtmpUrl": "rtmp://demo.flashphoner.com:1935/live/testStream"
}
```

7. Установка соединения по RTMP с указанным сервером, публикация потока

8. Отправка аудио-видео потока по RTMP

9. 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);
  $("#publishInfo").text("");
}
```

10. Получение от сервера события, подтверждающего остановку публикации потока

`STREAM_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();
```

Известные проблемы

1. При републикации потока на другой RTMP-сервер и проигрывании потока с этого сервера в плеере JWPlayer. соотношение сторон картинки может быть искажено



Симптомы

Соотношение сторон при проигрывании потока с RTMP-сервера отличается от опубликованного потока



Решение

Включить отправку метаданных при републикации

```
rtmp_transponder_send_metadata=true
```

2. Републикация может не работать, если RTMP-сервер, на который ретранслируется поток, требует определенную версию Flash



Симптомы

Не проходит RTMP handshake, канал закрывается с ошибкой (RTMP error) в логе WCS-сервера



Решение

Задать версию Flash RTMP-клиента, используя либо настройку

`rtmp_flash_ver_subscriber` в [flashphoner.properties](#), либо параметр `rtmpFlashVersion` в REST-вызове для ретрансляции

Например, для републикации на [Periscope](#):

```
rtmp_flash_ver_subscriber = LNX 76.219.189.0
```

3. Сервер, на который ретранслируется поток, может требовать определенные параметры потока: битрейт, частоту кадров и интервал ключевых кадров видео



Симптомы

Сервер показывает предупреждения о несоответствии рекомендуемым параметрами



Решение

Задать требуемые ограничения для исходного потока (например, для битрейта аудио) и параметры в REST-вызове для ретрансляции (`keyFrameInterval` и `fps`)

4. При ретрансляции потоков с большими размерами кадров, пакеты могут не помещаться в буфер сокета на отправку



Симптомы

При проигрывании ретранслированного потока на хорошем канале эпизодически появляются артефакты

✓ **Решение**

Включить буферизацию RTMP пакетов при отправке настройкой

```
rtmp.server_buffer_enabled=true
```