

На другой RTMP сервер

- Описание
 - Поддерживаемые платформы и браузеры
 - Поддерживаемые кодеки
 - Аутентификация на RTMP-сервере
 - Схема работы
- REST-вызовы
 - REST-вызовы и статусы ответа
 - Параметры
 - Отправка REST-запроса к WCS-серверу
- JavaScript API
- Настройка сервера
- Передача параметров в URL сервера
 - Передача имени потока в URL
- Автоматическая ретрансляция на указанный сервер
- Последовательность выполнения операций (Call Flow)

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 11	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 publish.
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-вызовы

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	Пример тела REST-запроса	Пример ответа	Статусы ответа	Описание
/push /startup	<pre>{ "streamName": "name", "rtmpUrl": "rtmp://localhost:1935/live", "options": {} }</pre>	<pre>{ "mediaSessionId" : "eume87rjk3dfli9 u14elffga6t", "streamName": "rtmp_name", "rtmpUrl": "rtmp://localhost:1935/live", "width": 320, "height": 240, "muted": false, "soundEnabled": false, "options": {} }</pre>	409 - Conflict 500 - Internal error	Создает транспондер, который подписывается на указанный поток и отправляет медиа трафик на указанный rtmpUrl. Имя потока, указанное в запросе, может быть именем уже публикуемого потока или именем зарезервированным при создании SIP-звонка (для отправки медиа трафика, полученного со стороны SIP). Если транспондер для такого потока и rtmpUrl уже существует, возвращает 409 Conflict.
/push /find	<pre>{ "streamName": "name", "rtmpUrl": "rtmp://localhost:1935/live", } }</pre>	<pre>[{ "mediaSessionId" : "eume87rjk3dfli9 u14elffga6t", "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	Найти транспондеры по фильтру

/push /find_all		<pre>[{ "mediaSessionId" : "eume87rjk3dfli9 u14elffga6t", "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	Найти все транспондеры
/push /terminate	<pre>{ "mediaSessionId" : "eume87rjk3dfli9 u14elffga6t" }</pre>		404 - Not found transponder 500 - Internal error	Завершить работу транспондера
/push /mute	<pre>{ "mediaSessionId" : "eume87rjk3dfli9 u14elffga6t" }</pre>	void	404 - Not found transponder 500 - Internal error	Выключить аудио
/push /unmute	<pre>{ "mediaSessionId" : "eume87rjk3dfli9 u14elffga6t" }</pre>	void	404 - Not found transponder 500 - Internal error	Включить аудио
/push /sound_on	<pre>{ "mediaSessionId" : "eume87rjk3dfli9 u14elffga6t" "soundFile": "test.wav" "loop": true }</pre>	void	404 - Not found transponder 404 - No such file 500 - Internal error	Вставить аудио из RIFF WAV файла из директории /usr/local/FlashphonerWebCallServer/media/ на WCS-сервере

/push /sound_on	{ "mediaSessionId" : "eume87rjk3df1i9u14elffga6t" }	void	404 - Not found transponder 500 - Internal error	Завершить вставку аудио из файла
--------------------	---	------	---	----------------------------------

Параметры

Имя параметра	Описание	Пример
streamName	Имя ретранслируемого потока	streamName
rtmpUrl	URL сервера, на который производится ретрансляция	rtmp://localhost:1935/live
options	Опции транспондера	{"action": "mute"}
mediaSessionId	Уникальный идентификатор транспондера	eume87rjk3df1i9u14elffga6t
width	Ширина изображения	320
height	Высота изображения	240
muted	Звук приглушен	true
soundEnabled	Звук отключен	true
soundFile	Файл звуковой дорожки	test.wav
loop	Повтор	false

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}
```

Отправка REST-запроса к WCS-серверу

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

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](#)

1. 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.

код:

```
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

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

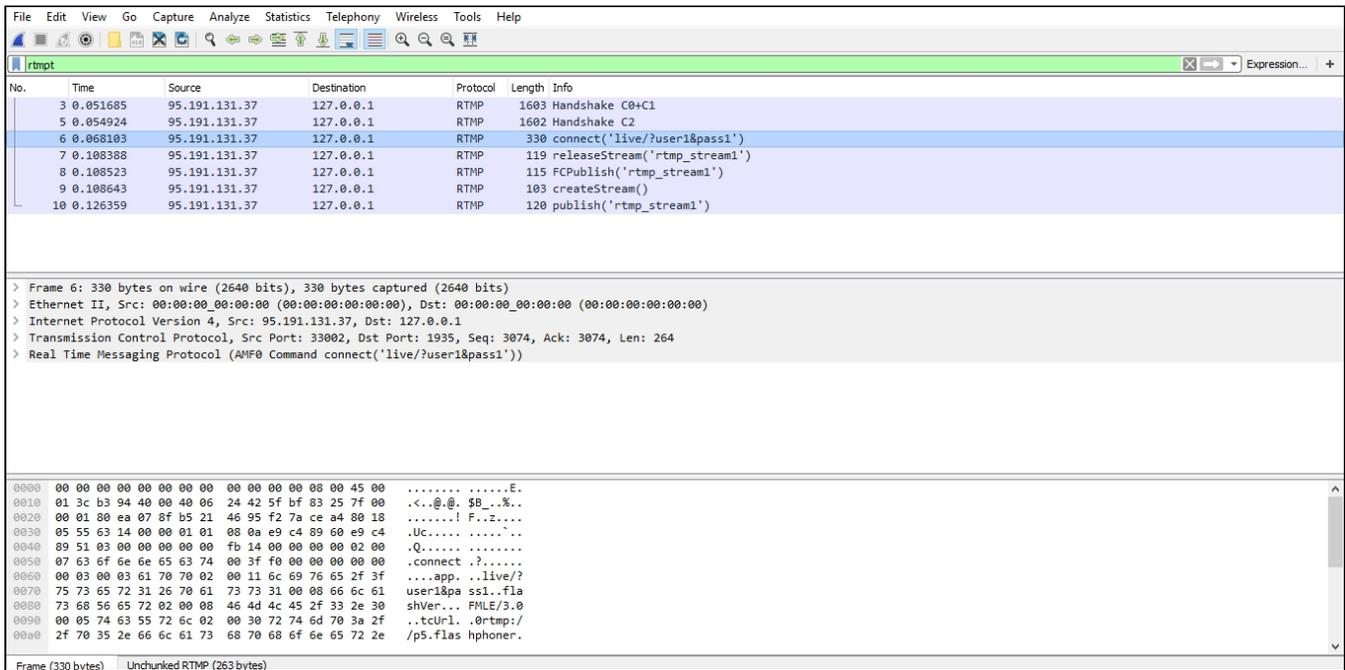
```
rtmp://myrtmpserver.com:1935/app_name/app_instance/?user=user1&pass=pass1
```

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 REST-запроса `/push/startup 'streamName'` 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 REST-запроса `/push/startup 'rtmpUrl'` или в соответствующей опции при создании потока указывается URL вида

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

или, для публикации в другой экземпляр приложения

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

В этом случае параметр REST-запроса `/push/startup 'streamName'` 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:

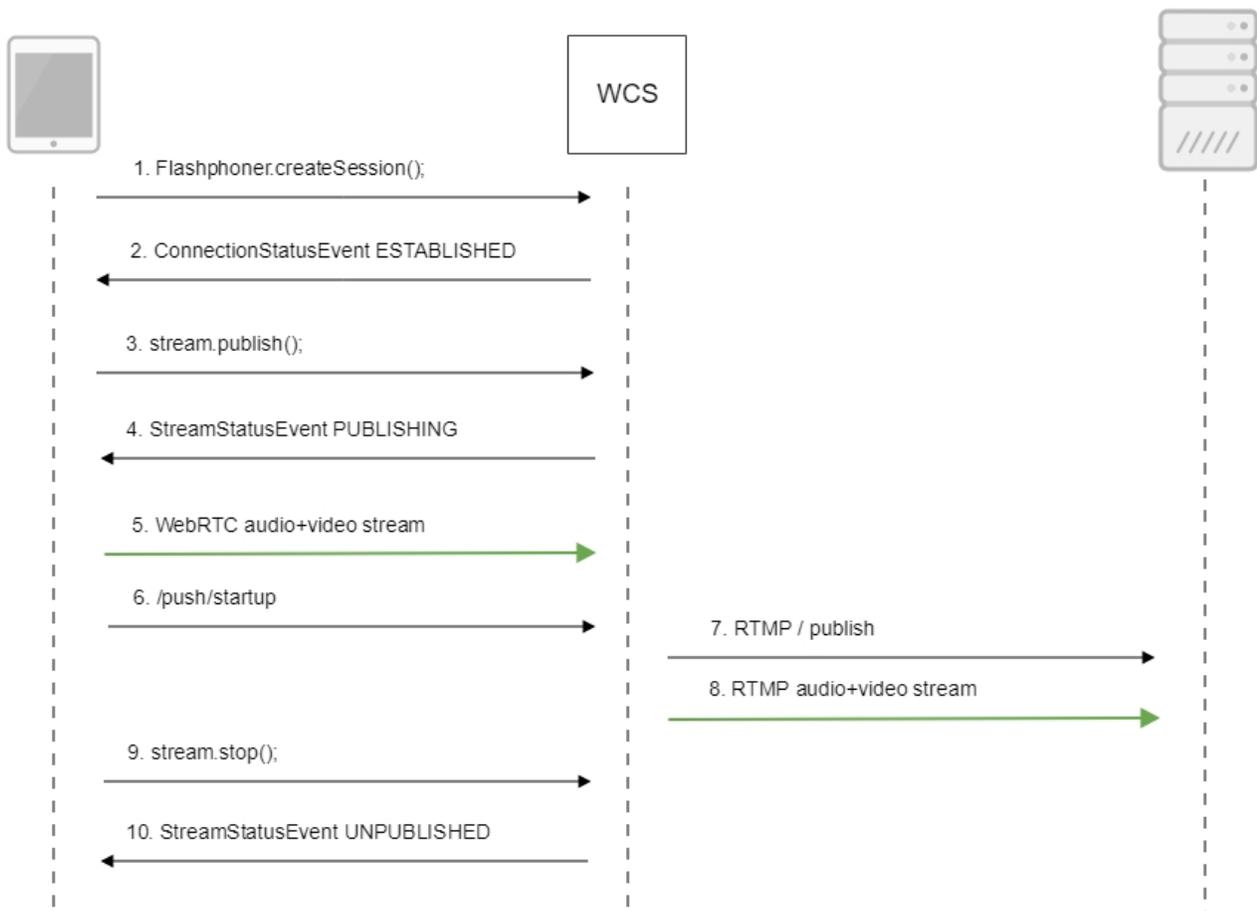
```
rtmp_push_auto_start=true
rtmp_push_auto_start_url=rtmp://rtmp.server.com:1935/
```

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

Последовательность выполнения операций (Call Flow)

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](#)



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.

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. 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 (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:9091/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.

StreamStatusEvent, статус 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();
```