

С помощью RTMP кодировщика (Live Encoder)

- Описание
 - Технические характеристики
 - Поддержка кодеков
 - Схема работы
- Краткое руководство по тестированию
- Последовательность выполнения операций (Call Flow)
- Обработка параметров, указанных в URL потока
 - Передача параметров соединения как параметров потока
- Указание серверного приложения при публикации RTMP-потока
- Публикация Sorenson Spark + Speex 16 kHz потока в контейнере FLV
 - Ограничения
- Контроль получения медиа данных по RTMP
 - Контроль наличия медиа данных в потоке
 - Таймаут на чтение данных
 - Таймаут на запись данных
 - Таймаут на чтение и запись данных
- Поворот изображения публикуемого RTMP потока
 - Настройка
 - Тестирование
 - Разработчику
 - Поворот изображения потока, опубликованного при помощи ffmpeg
- Управление размером буфера при декодировании
- Буферизация входящего RTMP потока
 - Прекращение буферизации потока при ухудшении его характеристик
 - Рекомендуемые настройки буферизации для RTMP потоков высокого качества
- Определение параметров публикуемого потока по метаданным или медиапакетам
- Известные проблемы

Для проведения онлайн-трансляций могут использоваться специальные аппаратные либо программные устройства видеозахвата (Live Encoder). Подобные устройства или программы захватывают видеопоток и отправляют его на сервер по протоколу RTMP.

Web Call Server может принимать RTMP видеопоток с такого устройства или ПО ([Wirecast](#), [ffmpeg](#), [OBS Studio](#), [FMLE](#) и т.п.) в кодеках H.264 + AAC или Sorenson Spark + Speex и раздавать этот видеопоток на браузеры и мобильные устройства.

Описание

Технические характеристики

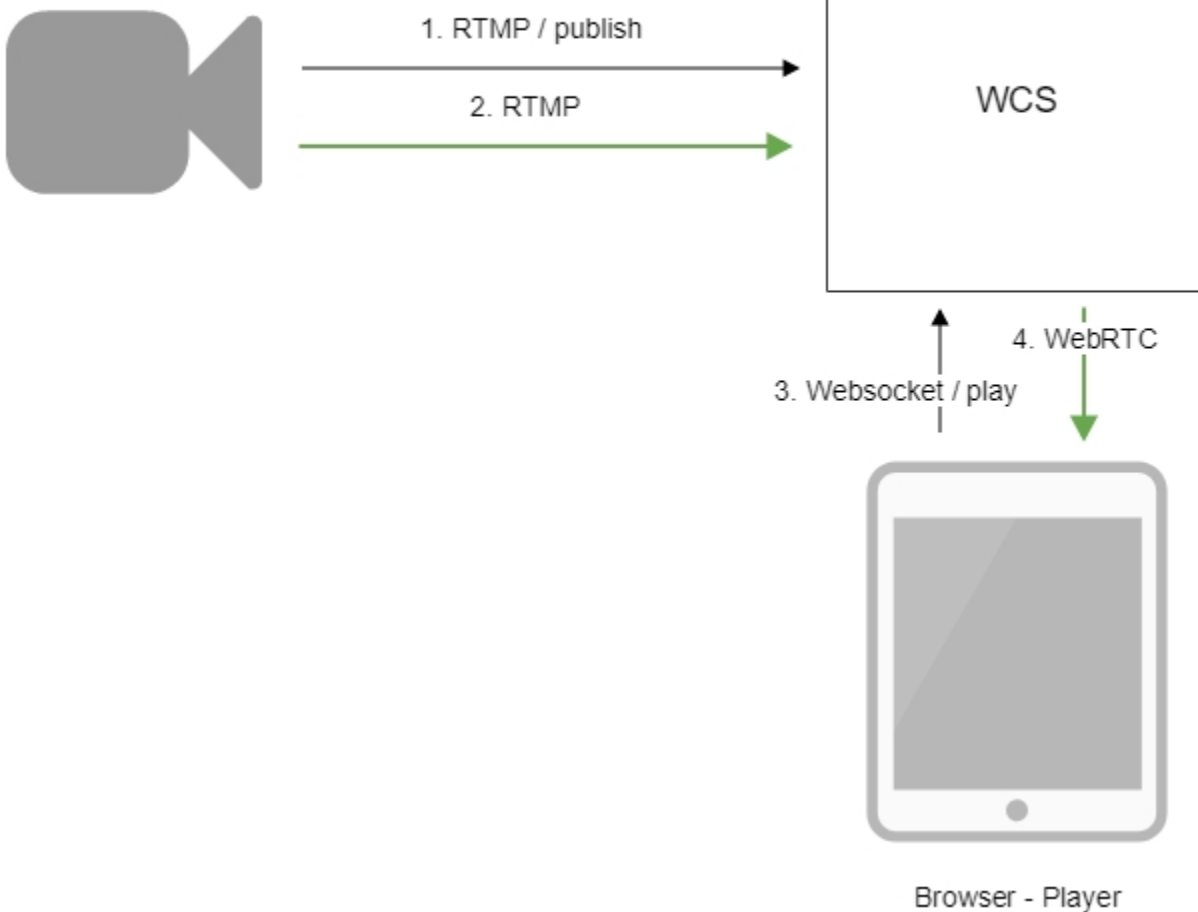
- Прием входящих аудио / видеопотоков по протоколу RTMP
- Раздача полученного видеопотока на браузеры и платформы: [любая из поддерживаемых WCS](#)
- Использование технологий воспроизведения видеопотока: [любая из поддерживаемых WCS](#)

Поддержка кодеков

- Видео H.264 + аудио AAC
- Видео Sorenson Spark + аудио Speex 16 kHz

Схема работы

RTMP Live Encoder - Publisher



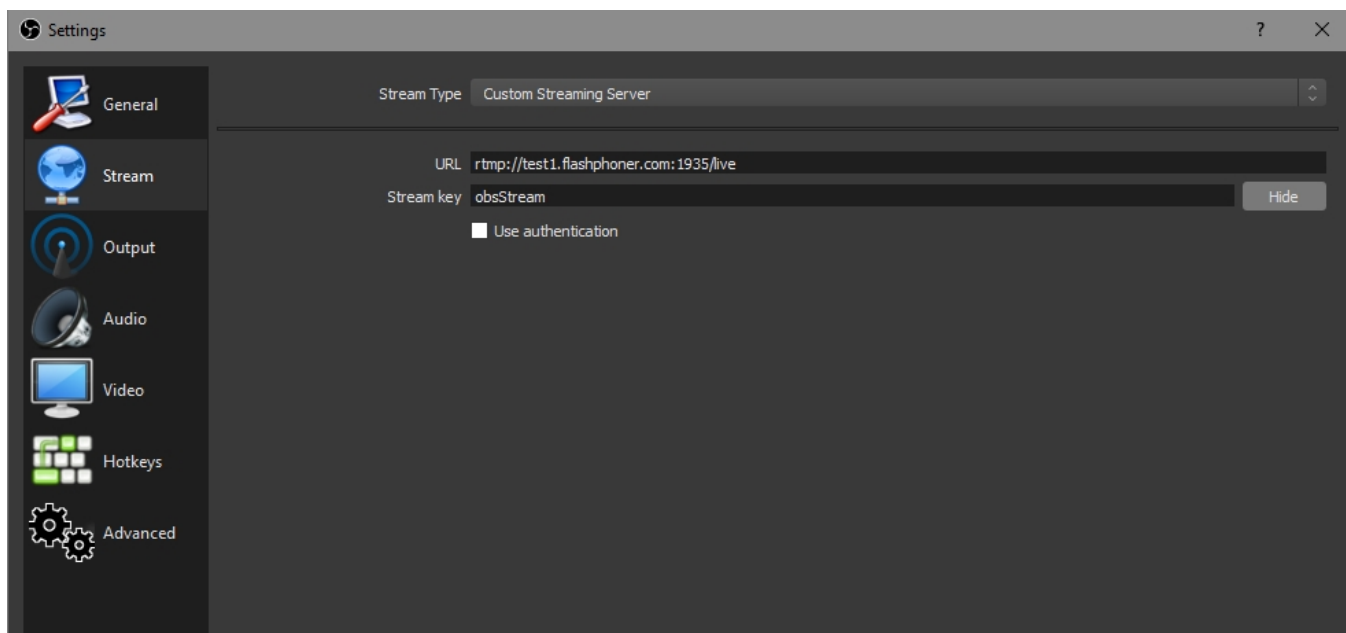
1. Live Encoder соединяется с сервером по протоколу RTMP и отправляет команду publish.
2. Live Encoder отправляет RTMP поток на сервер.
3. Браузер устанавливает соединение по Websocket и отправляет команду play.
4. Браузер получает WebRTC поток и воспроизводит этот поток на странице.

Краткое руководство по тестированию

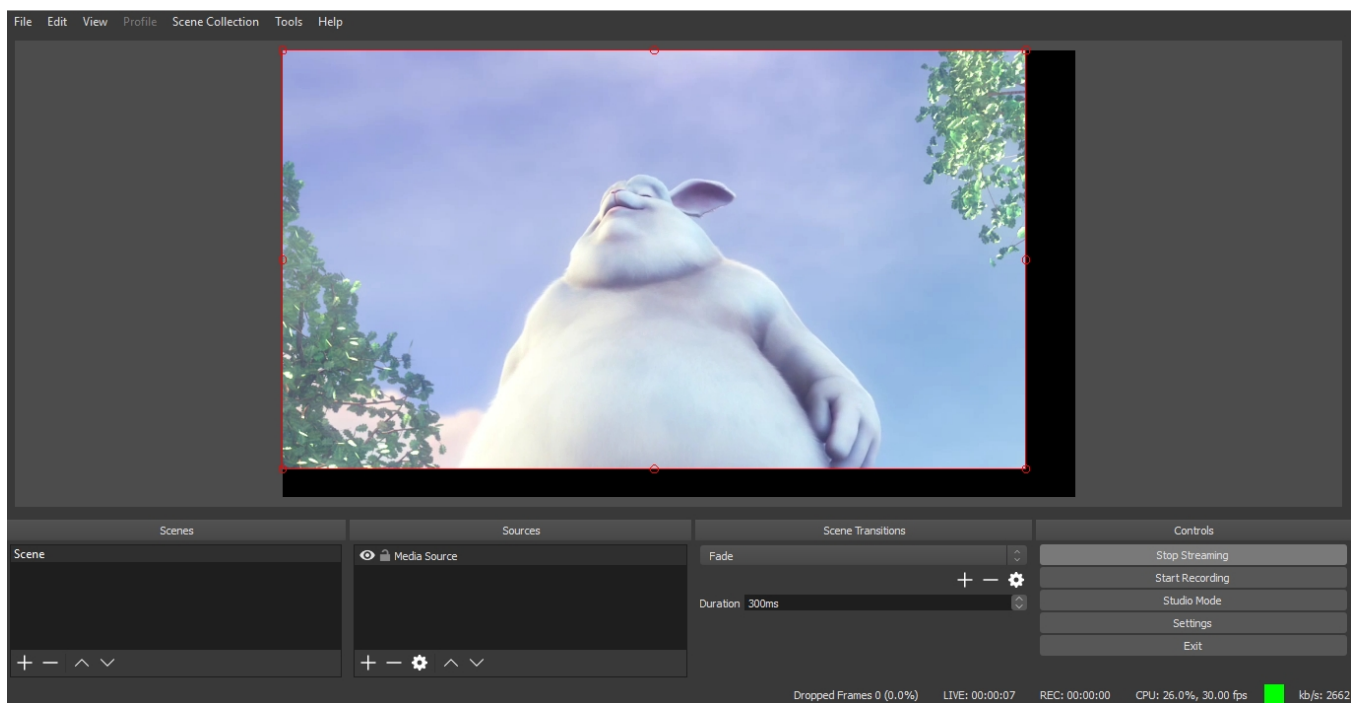
1. Для теста используем:

- WCS сервер
- OBS Studio
- веб-приложение [Player](#) в браузере Chrome для воспроизведения потока

2. Настройте вещание RTMP-потока на адрес сервера, например, `rtmp://test1.flashphoner.com:1935/live/`, ключ потока `obsStream`:



3. Запустите вещание в OBS Studio:



4. Откройте веб-приложение Player. Укажите в поле "Stream" ключ потока и нажмите кнопку "Start". Начнется трансляция захваченного потока.

Player



WCS URL

wss://test1.flashphoner.com:8443

Stream

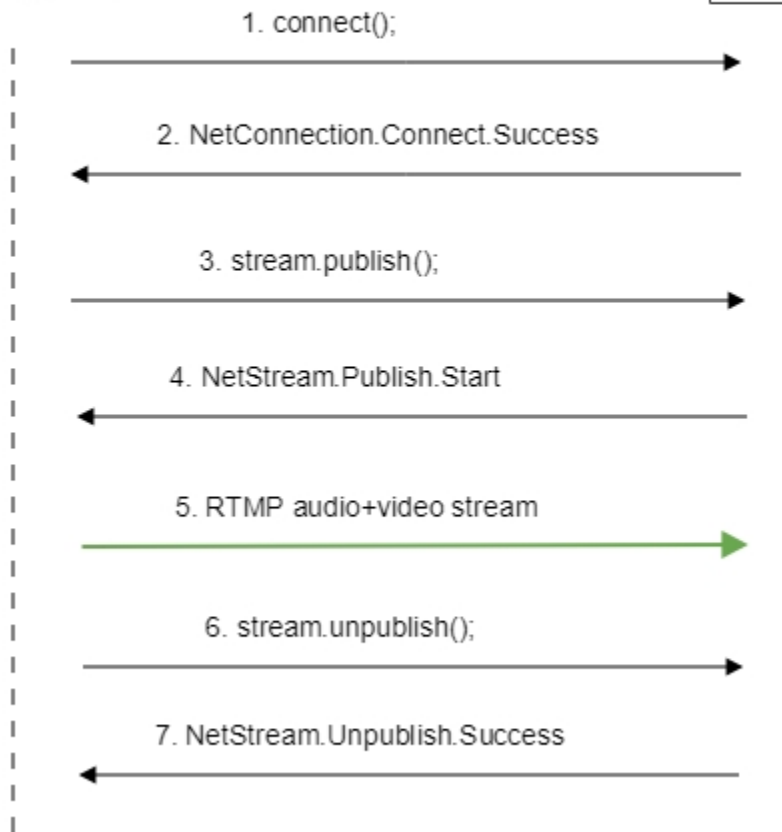
obsStream

Volume



Последовательность выполнения операций (Call Flow)

Ниже приводится последовательность выполнения операций при трансляции RTMP потока на WCS сервер с внешнего источника вещания (Live Encoder)



Обработка параметров, указанных в URL потока

При публикации или воспроизведении RTMP-потока на WCS, в URL потока могут быть указаны параметры RTMP-соединения и параметры потока:

```
rtmp://host:1935/live?connectParam1=val1&connectParam2=val2/streamName?streamParam1=val1&streamParam2=val2
```

Здесь

- host - WCS-сервер;
- connectParam1, connectParam2 - параметры RTMP-соединения;
- streamName - имя потока на сервере;
- streamParam1, streamParam2 - параметры потока.

WCS-сервер передает указанные параметры бэкенд-серверу в [REST hook](#), в поле custom, например:

Параметры соединения

```
URL:http://localhost:8081/apps/EchoApp/connect
OBJECT:
{
  "nodeId" : "Qb3rAjf3lzoy6PEl1WZkUhRG1DsTykgj@192.168.1.1",
  "appKey" : "flashStreamingApp",
  "sessionId" : "/127.0.0.1:5643/192.168.1.1:1935",
  "useWsTunnel" : false,
  "useWsTunnelPacketization2" : false,
  "useBase64BinaryEncoding" : false,
  "keepAlive" : false,
  "custom" : {
    "connectParam1" : "val1",
    "connectParam2" : "val2"
  },
  "login" : "rQq83sodiCPY0pJXCxGO"
}
```

Параметры публикации

```
URL:http://localhost:8081/apps/EchoApp/publishStream
OBJECT:
{
  "nodeId" : "Qb3rAjf3lzoy6PEl1WZkUhRG1DsTykgj@192.168.1.1",
  "appKey" : "flashStreamingApp",
  "sessionId" : "/127.0.0.1:5643/192.168.1.1:1935",
  "mediaSessionId" : "627990f9-8fe5-4e92-bb2a-863cc4eb43de",
  "name" : "stream1",
  "published" : true,
  "hasVideo" : false,
  "hasAudio" : true,
  "status" : "NEW",
  "record" : true,
  "width" : 0,
  "height" : 0,
  "bitrate" : 0,
  "minBitrate" : 0,
  "maxBitrate" : 0,
  "quality" : 0,
  "mediaProvider" : "Flash",
  "custom" : {
    "streamParam1" : "val1",
    "streamParam2" : "val2"
  }
}
```

Параметры воспроизведения

```
URL:http://localhost:8081/apps/EchoApp/playStream
OBJECT:
{
  "nodeId" : "Qb3rAjf3lzoy6PEl1WZkUhRG1DsTykgj@192.168.1.1",
  "appKey" : "flashStreamingApp",
  "sessionId" : "/127.0.0.1:5643/192.168.1.1:1935",
  "mediaSessionId" : "stream1/127.0.0.1:5643/192.168.1.1:1935",
  "name" : "stream1",
  "published" : false,
  "hasVideo" : true,
  "hasAudio" : true,
  "status" : "NEW",
  "record" : false,
  "width" : 0,
  "height" : 0,
  "bitrate" : 0,
  "minBitrate" : 0,
  "maxBitrate" : 0,
  "quality" : 0,
  "mediaProvider" : "Flash",
  "custom" : {
    "streamParam1" : "val1",
    "streamParam2" : "val2"
  }
}
```

Эту возможность можно использовать, например, для авторизации клиента на бэкэнд-сервере при публикации или воспроизведения RTMP-потока на WCS.

Передача параметров соединения как параметров потока

В некоторых случаях возникает необходимость передать параметры соединения, например, параметры авторизации, как параметры потока, например

```
rtmp://test.flashphoner.com:1935/live/test?auth=key
```

Эта возможность включается настройкой

```
rtmp_use_stream_params_as_connection=true
```

В этом случае вышеприведенный пример RTMP URL будет аналогичен следующему

```
rtmp://test.flashphoner.com:1935/live?auth=key/test
```

Указание серверного приложения при публикации RTMP-потока

При публикации RTMP-потока на WCS сервере можно указать [приложение](#), которое будет использовано для взаимодействия с бэкэнд-сервером, при помощи параметра в URL потока:

```
rtmp://host:1935/live?appKey=key1/streamName
```

Здесь

- host - WCS-сервер;
- key1 - ключ приложения на WCS-сервере;
- streamName - имя потока на сервере

По умолчанию, если ключ приложения не указан, используется стандартное приложение `flashStreamingApp`.

Кроме того, приложение может быть указано явным образом как часть URL. Для этого необходимо в файле `flashphoner.properties` установить настройку

```
rtmp_appkey_source=app
```

Тогда приложение должно быть указано в URL потока как

```
rtmp://host:1935/key1/streamName
```

В этом случае значение live также рассматривается, как имя приложения, поэтому при публикации потока

```
rtmp://host:1935/live/streamName
```

на WCS сервере должно быть определено приложение live.

Публикация Sorenson Spark + Speex 16 kHz потока в контейнере FLV

WCS сервер принимает RTMP поток, закодированный в Sorenson Spark + Speex 16kHz в контейнере FLV. Такой поток можно опубликовать, например, при помощи ffmpeg следующим образом:

```
ffmpeg -re -i BigBuckBunny.flv -preset ultrafast -ar 16000 -ac 1 -acodec speex -vcodec flv -strict -2 -f flv  
rtmp://test1.flashphoner.com:1935/live/test
```

Ограничения

1. Для дальнейшей обработки на сервере, в том числе для записи, такой поток будет транскодирован в H.264 + AAC.
2. При публикации в SDP для видео и для аудио должны быть указаны payload type 127 и 97 соответственно, например

```
v=0  
o=- 1988962254 1988962254 IN IP4 0.0.0.0  
c=IN IP4 0.0.0.0  
t=0 0  
a=sdplang:en  
m=video 0 RTP/AVP 127  
a=rtpmap:127 FLV/90000  
a=sendonly  
m=audio 0 RTP/AVP 97 8 0  
a=rtpmap:97 SPEEX/16000  
a=rtpmap:8 PCMA/8000  
a=rtpmap:0 PCMU/8000  
a=sendonly
```

Контроль получения медиа данных по RTMP

В некоторых случаях, если RTMP-кодировщик не поддерживает отсылку Keep Alive пакетов, либо этот механизм отключен по другим причинам при помощи настройки

```
keep_alive.algorithm=NONE
```

возникает необходимость контролировать RTMP-соединения и закрывать их, если в течение длительного времени не передается никаких данных. Для этого предусмотрены следующие настройки.

Контроль наличия медиа данных в потоке

Начиная со сборки [5.2.533](#), контроль наличия медиа данных в RTMP потоке включается при помощи настройке файла `flashphoner.properties`:

```
flash_rtp_activity_enabled=true
```


Таймаут на чтение данных

Таймаут на чтение управляется при помощи следующих параметров в файле [flashphoner.properties](#):

```
rtmp.server_read_socket_timeout=120
```

В данном случае RTMP-соединение будет закрыто, если в течение 120 секунд из него не было принято никаких данных.

Таймаут на запись данных

Таймаут на запись управляется при помощи следующего параметра

```
rtmp.server_write_socket_timeout=120
```

В данном случае RTMP-соединение будет закрыто, если в течение 120 секунд в него не было отправлено никаких данных.

Таймаут на чтение и запись данных

Таймаут на чтение и запись управляется при помощи следующего параметра

```
rtmp.server_socket_timeout=120
```

В данном случае RTMP-соединение будет закрыто, если в течение 120 секунд из него не было принято и в него не было отправлено никаких данных.

Поворот изображения публикуемого RTMP потока

При публикации RTMP потока на WCS, можно повернуть изображение, отправив необходимые RTMP-метаданные. Это может быть полезным для изменения ориентации картинки на лету при публикации потока с мобильного источника.

Для того, чтобы повернуть изображение на указанный угол, клиент должен прислать серверу RTMP-метаданные с полем 'orientation'. Поле может принимать следующие значения:

Значение поля	Угол поворота в градусах
0	0
1	90
2	180
3	270

Изображение поворачивается по часовой стрелке.

Настройка

Поворот изображения по метаданным включается при помощи следующей настройки в файле [flashphoner.properties](#):

```
video_filter_enable_rotate=true
```

Отметим, что поворот изображения работает только при использовании транскодирования.

Тестирование

1. Для теста используем:

- WCS сервер с включенной поддержкой поворота изображения по метаданным
- Приложение [Flash Streaming](#) для публикации и вращения изображения
- Приложение [Player](#) для воспроизведения потока

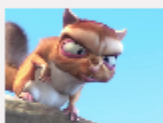
2. Откройте приложение Flash Streaming. Введите имя потока test, укажите желаемые параметры публикации потока

Flash Streaming

Server:

Publish

Play



☒ audio ☒ video

Rotate camera

width height fps quality keyframe

codec

3. Нажмите Login, затем Start. Начнется публикация потока

Flash Streaming

Server:

CONNECTED

Publish:

PUBLISHING

Play:



☒ audio ☒ video

Rotate camera

width height fps quality keyframe

codec

4. Откройте в другой вкладке или в другом браузере приложение Player, воспроизведите поток test

Player



WCS URL

wss://test2.flashphoner.com:8443

Stream

test

Volume



Full Screen



PLAYING

Stop

5. В приложении Flash Player нажмите кнопку 180 в разделе Rotate camera. В приложении Player отобразится изображение, повернутое на 180 градусов по часовой стрелке

Player



WCS URL

wss://test2.flashphoner.com:8443

Stream

test

Volume



Full Screen



PLAYING

Stop

Разработчику

Отправка метаданных для поворота изображения реализована в приложении Flash Player следующим образом:

[code](#)

```

        private function rotate(degree:Number):void {
            var metaDataObj:Object = new Object();
            switch(degree) {
            case 0:
                Logger.info("rotate camera to 0");
                metaDataObj.orientation = 0;
                break;
            case 90:
                Logger.info("rotate camera to 90");
                metaDataObj.orientation = 1;
                break;
            case 180:
                Logger.info("rotate camera to 180");
                metaDataObj.orientation = 2;
                break;
            case 270:
                Logger.info("rotate camera to 270");
                metaDataObj.orientation = 3;
                break;
            default:
                metaDataObj.orientation = 0;
                break;
            }
            sendMetaData(metaDataObj);
        }

        private function sendMetaData(data:Object):void{
            if (publishStream != null) {
                publishStream.send("@setDataFrame", "onMetaData", data);
            }
        }

```

Обратите внимание, что отправляется не угол, а соответствующее значение поля orientation.

Поворот изображения потока, опубликованного при помощи ffmpeg

RTMP кодировщик ffmpeg дает возможность отправить метаданные ориентации изображения серверу при помощи ключей командной строки:

```
ffmpeg -i input.mp4 -metadata:s:v rotate=90 -vcodec copy -acodec copy -strict -2 -f flv rtmp://test1.flashphoner.com:1935/live/stream_ffmpeg
```

Отметим, что настройка поворота для ffmpeg указывается в градусах, при этом на сервер передается соответствующее значение поля orientation.

Управление размером буфера при декодировании

Если RTMP поток публикуется программным кодировщиком с поддержкой аппаратного ускорения на GPU NVIDIA, и содержит B-фреймы, при проигрывании такого потока по RTMP или HLS в некоторых плеерах картинка подергивается. В связи с этим, в сборке [5.2.863](#) была добавлена настройка, которая задает максимальный размер буфера декодирования, передаваемый в SPS

```
h264_sps_max_dec_frame_buffering=-1
```

По умолчанию, размер буфера не ограничен. Это убирает подергивания картинки, но может приводить к задержкам при излишней буферизации. В этом случае, можно ограничить буфер двумя кадрами (значение по умолчанию до сборки [5.2.863](#))

```
h264_sps_max_dec_frame_buffering=2
```

или большим числом, чтобы убрать подергивания картинки и не допустить задержки.

Буферизация входящего RTMP потока

При публикации RTMP потока в большом разрешении, с высоким битрейтом при нестабильном канале поток может играть по WebRTC не плавно, с фризами или снижением FPS. Чтобы предотвратить такое поведение, необходимо буферизовать входящий поток

```
rtmp_in_buffer_enabled=true
```

Адаптивный буфер для входящего RTMP потока имеет следующие тонкие настройки:

Параметр	Описание	Значение по умолчанию
rtmp_in_buffer_start_size	Исходный объем буфера, мс	300
rtmp_in_buffer_initial_size	Максимальный объем буфера, мс	2000
rtmp_in_buffer_max_bufferings_allowed	Максимальное количество увеличений буфера	-1 (не ограничено)
rtmp_in_buffer_polling_time	Периодичность проверки наличия данных в буфере, мс	100
rtmp_in_buffer_overflow_allowe d_deviation	Максимально допустимая разность между минимальным и максимальным объемами буфера, мс	1000
rtmp_in_buffer_overflow_deviati on_window	Размер окна, в течение которого отслеживается разность, мс	30000
rtmp_in_buffer_overflow_rate	Максимально допустимая частота переполнений буфера	0.15
rtmp_in_buffer_clear_threshold	При наполнении буфера до указанной величины, сбросить все данные, объем которых превышает максимальный, мс	30000

Прекращение буферизации потока при ухудшении его характеристик

Если программному RTMP кодировщику не хватает производительности системы, на которой он запущен, или не хватает пропускной способности канала, метки времени в пакетах могут давать задержку относительно времени сервера. Буферизация такого потока будет давать периодические фризы при проигрывании. Поэтому в сборке [5.2.1311](#) добавлена настройка для отключения буферизации и пропуска полученного трафика напрямую в движок сервера

```
rtmp_in_buffer_input_delay_threshold=0
```

По умолчанию, при включенной буферизации RTMP трафик будет всегда помещаться в буфер. Буферизация может быть отключена при достижении определенного значения задержки в миллисекундах

```
rtmp_in_buffer_input_delay_threshold=3000
```

При этом буфер освобождается и переходит в статус PASSTHROUGH . Даже если задержка затем снизится, буфер останется в таком статусе. и поток не будет буферизоваться до окончания публикации.

Рекомендуемые настройки буферизации для RTMP потоков высокого качества

При публикации RTMP потоков высокого качества рекомендуются следующие настройки буферизации

```
rtmp_in_buffer_enabled=true
rtmp_in_buffer_start_size=1000
rtmp_in_buffer_initial_size=3000
rtmp_in_buffer_polling_time=30
rtmp_in_buffer_max_bufferings_allowed=-1
```

В этом случае публикуемый поток будет играть плавно у зрителей и в микшере.

Определение параметров публикуемого потока по метаданным или медиапакетам

По умолчанию, возможные параметры публикуемого RTMP потока определяются [файлом настройки SDP](#). В сборке [5.2.1862](#) добавлена настройка, которая включает автоматическое определение параметров публикуемого потока по метаданным или по информации в медиапакетах

```
flash_detect_metadata_by_traffic=true
```

Настройка включена по умолчанию. В этом случае WCS корректирует SDP в соответствии с полученными от публикующего клиента метаданными или по информации из полученных медиапакетов.

В сборке [5.2.1935](#) добавлена настройка интервала, в течение которого WCS пытается определить метаданные потока по полученным медиапакетам

```
flash_detect_metadata_by_traffic_timeout=1000
```

По умолчанию интервал составляет 1000 мс.

Известные проблемы

1. Поток, содержащий В-фреймы, не воспроизводится либо воспроизводится с артефактами (задержки, подергивания)

Симптомы:

- поток не проигрывается, дает задержки видео или подергивания
- предупреждения в [клиентском логе](#):

```
09:32:31,238 WARN 4BitstreamNormalizer - RTMP-pool-10-thread-5 It is B-frame!
```

Решение:

- изменить настройки кодировщика таким образом, чтобы исключить использование В-фреймов (понижить профиль кодирования, указать в командной строке и т.п.).
- [транскодировать](#) поток, в этом случае в выходном потоке транскодера В-фреймов не будет

2. AAC фреймы типа 0 не поддерживаются декодером FFmpeg и будут игнорироваться при воспроизведении захваченного потока

При этом в [клиентском логе](#) будут выведены предупреждения:

```
10:13:06,815 WARN AAC - AudioProcessor-c6c22de8-a129-43b2-bf67-1f433a814ba9 Dropping AAC frame that starts with 0, 119056e500
```

Решение: использовать кодек Fraunhofer при помощи настройки в файле [flashphoner.properties](#)

```
use_fdk_aac=true
```

3. При публикации и последующем воспроизведении и записи H264 + AAC потока возможна рассинхронизация видео и звука, либо полное отсутствие звука.

Симптомы: при воспроизведении H264 + AAC потока, опубликованного на сервере, а также в записи потока, звук не синхронизирован с видео или отсутствует

Решение:

а) установить настройку в файле [flashphoner.properties](#)

```
disable_drop_aac_frame=true
```

Эта настройка, в том числе, отключает игнорирование AAC фреймов.

б) использовать кодек Fraunhofer при помощи настройки

```
use_fdk_aac=true
```

4. При преобразовании звуковой дорожки AAC к частоте дискретизации 11025 Гц звук искажен или отсутствует

Симптомы: при публикации H264 + AAC потока на WCS сервере и воспроизведении его как H264 + AAC с частотой дискретизации звука 11025 Гц звук искажен или отсутствует

Решение: не использовать частоту дискретизации звука 11025 Гц, либо избегать преобразования звука к данной частоте, например, не указывать данную частоту в [файлах настроек SDP](#).

5. Некоторые функции RTMP не поддерживаются и будут игнорированы:

- FCSubscribe
- FCPublish
- FCUnpublish
- onStatus
- onUpstreamBase
- releaseStream

6. Не все RTMP-кодировщики поддерживают KeepAlive.

Симптомы: частые разрывы соединения при публикации потока с RTMP-кодировщика.

Решение: отключить KeepAlive для RTMP на сервере при помощи настройки в файле [flashphoner.properties](#)

```
keep_alive.enabled=websocket,rtmfp
```

7. При воспроизведении потока, публикуемого из RTMP-кодировщика, как HLS, могут наблюдаться фриззы, если GOP не кратен частоте кадров публикуемого файла

Симптомы: при воспроизведении RTMP-потока как HLS наблюдаются фриззы

Решение: установить в параметрах кодировщика GOP равный или кратный частоте кадров публикуемого файла. Например, если публикуется файл с fps 25, необходимо указать GOP 50.

8. При воспроизведении потока, опубликованного из RTMP кодировщика, как WebRTC, звук в потоке низкого качества

Симптомы: при воспроизведении RTMP потока как WebRTC звук подходит для передачи речи, но не музыки

Решение: установить настройку битрейта кодирования в Opus в соответствии с битрейтом публикации звука, например

```
opus.encoder.bitrate=160000
```

если звук в RTMP потоке публикуется с битрейтом 160 кбит/с.

9. При публикации RTMP потока H264+speex (например, при помощи Flash) большая нагрузка на процессор при транскодинге звука

Симптомы: большая нагрузка на процессор при транскодинге звука из speex в AAC или Opus

Решение: использовать нативную реализацию декодера speex

```
use_speex_java_impl=false
```

10. Поток с неподдерживаемым аудио или видео кодеком не может быть опубликован

Симптомы: RTMP поток с MP3 или AC3 аудио не публикуется, в логе сервера сообщения

```
11:01:00,921 WARN      ServerHandler - RTMP-pool-15-thread-1 Codecs not supported! audio: NoCodec, video: NoCodec
```

Решение: перекодировать поток к поддерживаемым кодекам при публикации при помощи соответствующих настроек кодировщика