

Захват VOD из файла

- [Описание](#)
 - [Поддерживаемые форматы и кодеки](#)
 - [Схема работы](#)
- [Краткое руководство по тестированию](#)
- [Последовательность выполнения операций](#)
- [Циклический захват потока из файла](#)
- [Захват файла, размещенного на AWS или другом S3 хранилище](#)
 - [Схема работы](#)
 - [Настройка](#)
 - [Доступ к хранилищу S3](#)
 - [AWS](#)
 - [Digital Ocean Spaces](#)
 - [Selectel](#)
 - [Захват VOD из файла во время загрузки](#)
- [Требования к формату файлов](#)
- [Требования к именам файлов](#)
- [Управление VOD при помощи REST API](#)
 - [REST-методы и статусы ответа](#)
 - [Параметры](#)
 - [Циклический захват потока из файла по запросу](#)
 - [Ограничения](#)
- [Настройка продолжительности публикации VOD потока после отключения подписчиков](#)
- [Известные проблемы](#)

WCS предоставляет возможность захвата медиапотока из файла MP4, расположенного на локальном диске сервера (Video on Demand, VOD). Полученный поток можно [воспроизвести](#), [ретранслировать](#), [управлять им](#), как любым потоком на WCS-сервере. Прежде всего, данная возможность предназначена для воспроизведения [записанных ранее трансляций](#) в браузере или мобильном приложении клиента.

Описание

Для захвата VOD из файла в качестве имени потока при вызове функции `session.createStream()` должна быть указана ссылка на файл в виде:

```
vod://sample.mp4
```

где `sample.mp4` - имя файла, который должен находиться в каталоге `/usr/local/FlashphonerWebCallServer/media`. Начиная со сборки [5.2.687](#), каталог для размещения файлов, может быть указан при помощи настройки в файле [flashphoner.properties](#)

```
media_dir=/usr/local/FlashphonerWebCallServer/media
```

В случае, если файл с таким именем отсутствует, сервер вернет сообщение `StreamStatusEvent FAILED`, в поле "info" которого будет указан диагноз "File not found".

Поток, созданный таким образом, предназначен для трансляции одному пользователю (персональный VOD). На этот поток нельзя подписаться, его нельзя транскодировать, добавить в микшер или проиграть по HLS

В случае, если необходимо организовать полноценную онлайн-трансляцию, следует указать ссылку на файл в виде:

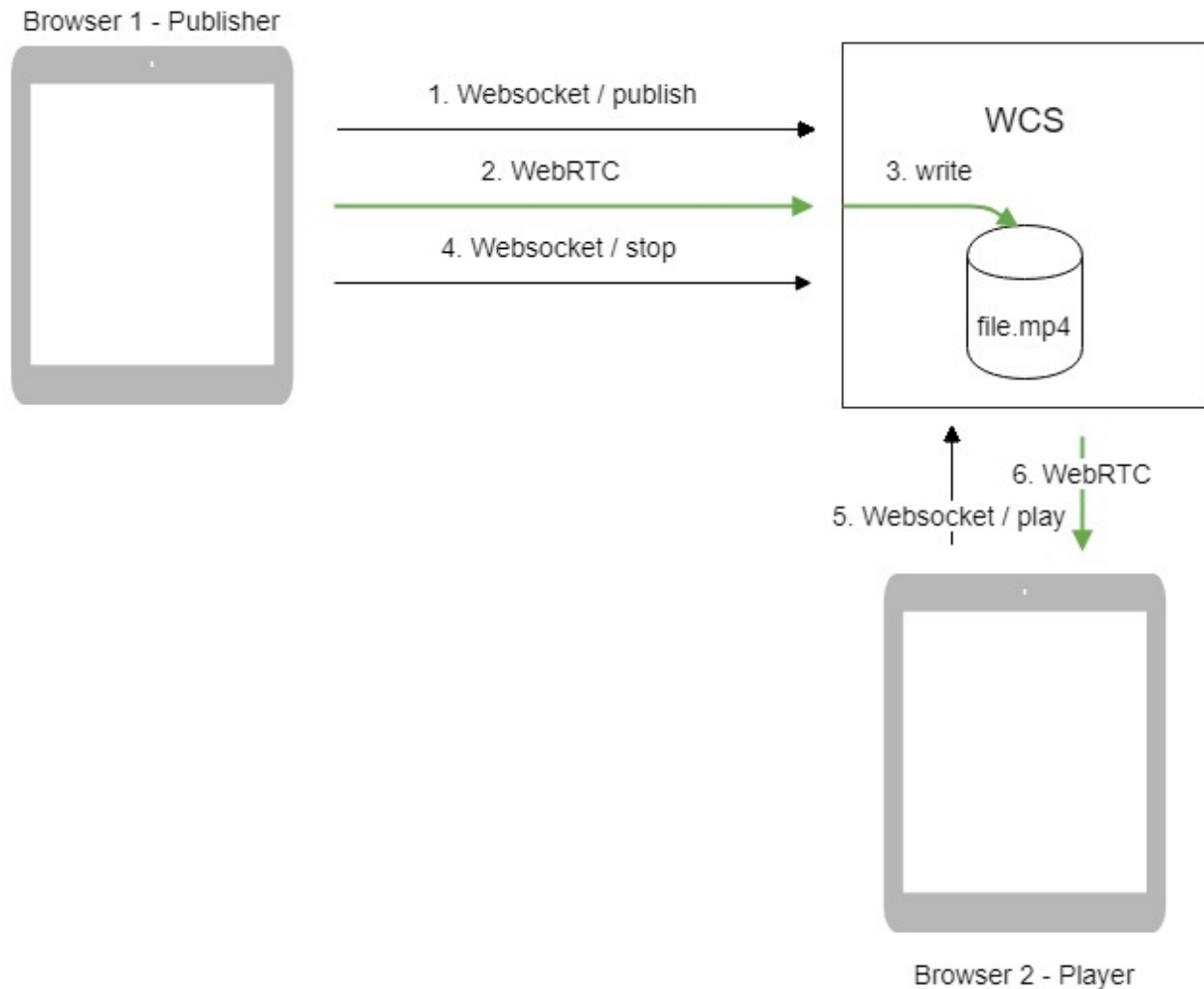
```
vod-live://sample.mp4
```

К такому потоку могут подключиться одновременно несколько пользователей в реальном времени, он может быть транскодирован, добавлен в микшер или воспроизведен по HLS.

Поддерживаемые форматы и кодеки

- Контейнер: MP4
- Видео: H.264
- Аудио: AAC

Схема работы



1. Браузер соединяется с сервером по протоколу Websocket и отправляет команду publish.
2. Браузер захватывает микрофон и камеру и отправляет WebRTC поток H.264 + AAC на сервер с параметром record: true.
3. WCS-сервер записывает поток в файл.
4. Браузер останавливает публикацию.
5. Второй браузер устанавливает соединение по Websocket, создает поток с указанием имени файла и отправляет команду play.
6. Второй браузер получает WebRTC поток и воспроизводит этот поток на странице.

Краткое руководство по тестированию

1. Для теста используем веб-приложение [Player](#) для воспроизведения файла.
2. Загрузите файл в каталог `/usr/local/FlashphonerWebCallServer/media/`
3. Откройте веб-приложение `Player`, укажите в поле `Stream` имя файла:

WCS URL

Stream

Volume

Full Screen

4. Нажмите Start. Начнется воспроизведение файла:

Player



WCS URL

Stream

5. Нажмите Stop для остановки воспроизведения.

6. Удалите файл из каталог /usr/local/FlashphonerWebCallServer/media/

7. Нажмите Start. Отобразится статус FAILED и сообщение "File not found":

WCS URL

wss://p11.flashphoner.com:8443

Stream

vod://sample.mp4

Volume

Full Screen

FAILED
File not found

Start

Последовательность выполнения операций

Ниже описана последовательность вызовов при использовании:

примера Stream Recording для публикации потока и записи файла

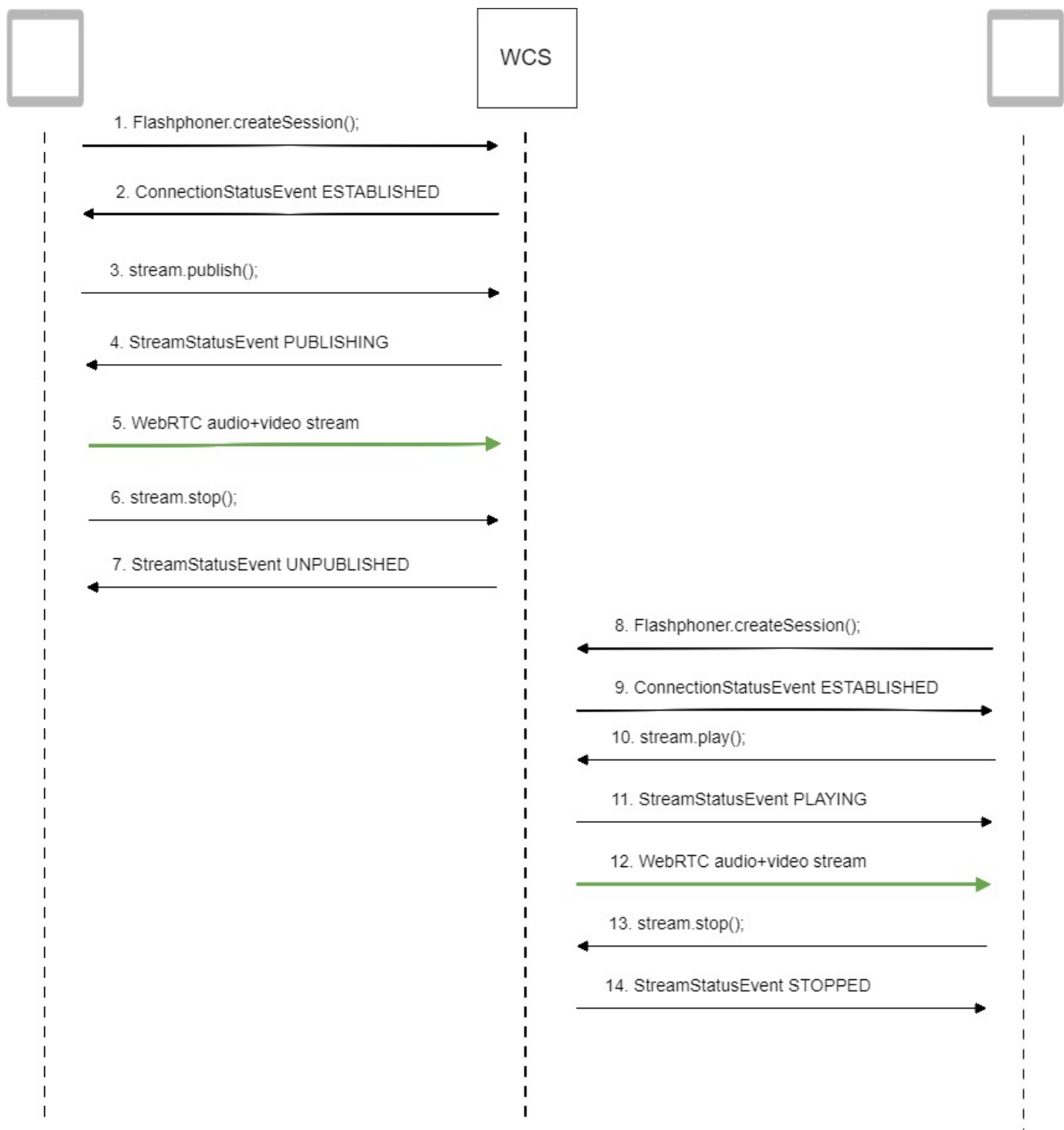
[recording.html](#)

[recording.js](#)

примера Player для воспроизведения VOD-потока

[player.html](#)

[player.js](#)



1. Установка соединения с сервером для публикации и записи потока.

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

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    ...
});
```

2. Получение от сервера события, подтверждающего успешное соединение.

`ConnectionStatusEvent ESTABLISHED`[code](#)

```
Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    setStatus(session.status());
    //session connected, start playback
    publishStream(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
    ...
}).on(SESSION_STATUS.FAILED, function(){
    ...
});
```

3. Публикация потока с указанием признака записи:

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

```
session.createStream({
    name: streamName,
    display: localVideo,
    record: 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,
    record: true,
    receiveVideo: false,
    receiveAudio: false
}).on(STREAM_STATUS.PUBLISHING, function(stream) {
    setStatus(stream.status());
    onStart(stream);
}).on(STREAM_STATUS.UNPUBLISHED, function(stream) {
    ...
}).on(STREAM_STATUS.FAILED, function(stream) {
    ...
}).publish();
```

5. Отправка аудио-видео потока по WebRTC

6. Остановка публикации потока.

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

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

7. Получение от сервера события, подтверждающего остановку публикации потока.

StreamStatusEvent, статус UNPUBLISHED[code](#)

```

session.createStream({
    name: streamName,
    display: localVideo,
    record: true,
    receiveVideo: false,
    receiveAudio: false
}).on(STREAM_STATUS.PUBLISHING, function(stream) {
    ...
}).on(STREAM_STATUS.UNPUBLISHED, function(stream) {
    setStatus(stream.status());
    showDownloadLink(stream.getRecordInfo());
    onStoped();
}).on(STREAM_STATUS.FAILED, function(stream) {
    ...
}).publish();

```

8. Установка соединения с сервером для воспроизведения потока.

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

```

Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    ...
});

```

9. Получение от сервера события, подтверждающего успешное соединение.

ConnectionStatusEvent ESTABLISHED[code](#)

```

Flashphoner.createSession({urlServer: url}).on(SESSION_STATUS.ESTABLISHED, function(session){
    setStatus(session.status());
    //session connected, start playback
    playStream(session);
}).on(SESSION_STATUS.DISCONNECTED, function(){
    ...
}).on(SESSION_STATUS.FAILED, function(){
    ...
});

```

10. Воспроизведение потока.

stream.play():[code](#)

```

if (Flashphoner.getMediaProviders()[0] === "MSE" && mseCutByIFrameOnly) {
    options.mediaConnectionConstraints = {
        cutByIFrameOnly: mseCutByIFrameOnly
    }
}
if (resolution_for_wsplayer) {
    options.playWidth = resolution_for_wsplayer.playWidth;
    options.playHeight = resolution_for_wsplayer.playHeight;
} else if (resolution) {
    options.playWidth = resolution.split("x")[0];
    options.playHeight = resolution.split("x")[1];
}
stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
    ...
});
stream.play();

```

11. Получение от сервера события, подтверждающего успешное воспроизведение потока.

StreamStatusEvent, статус PLAYING[code](#)

```
stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
    ...
}).on(STREAM_STATUS.PLAYING, function(stream) {
    $("#preloader").show();
    setStatus(stream.status());
    onStarted(stream);
}).on(STREAM_STATUS.STOPPED, function() {
    ...
}).on(STREAM_STATUS.FAILED, function(stream) {
    ...
}).on(STREAM_STATUS.NOT_ENOUGH_BANDWIDTH, function(stream){
    ...
});
stream.play();
```

12. Прием аудио-видео потока по Websocket и воспроизведение по WebRTC

13. Остановка воспроизведения потока.

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

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

14. Получение от сервера события, подтверждающего остановку воспроизведения потока.

StreamStatusEvent, статус STOPPED[code](#)

```
stream = session.createStream(options).on(STREAM_STATUS.PENDING, function(stream) {
    ...
}).on(STREAM_STATUS.PLAYING, function(stream) {
    ...
}).on(STREAM_STATUS.STOPPED, function() {
    setStatus(STREAM_STATUS.STOPPED);
    onStopped();
}).on(STREAM_STATUS.FAILED, function(stream) {
    ...
}).on(STREAM_STATUS.NOT_ENOUGH_BANDWIDTH, function(stream){
    ...
});
stream.play();
```

Циклический захват потока из файла

Для трансляций vod-live поддерживается циклический захват потока, после окончания файла захват начинается сначала. Эта возможность включается настройкой в файле [flashphoner.properties](#)

```
vod_live_loop=true
```

Захват файла, размещенного на AWS или другом S3 хранилище

Поток может быть захвачен из файла, размещенного на AWS в хранилище S3. В отличие от VOD захвата файла с локального диска, файл, размещенный на внешнем хранилище, загружается и воспроизводится последовательно.

Для захвата VOD из файла на AWS в качестве имени потока при вызове функции `session.createStream()` должна быть указана ссылка на файл в виде:

```
vod://s3/bucket/sample.mp4
```

где

- bucket - имя корзины S3
- sample.mp4 - имя файла

В сборке [5.2.939](#) добавлена возможность указать полный URL файла в S3 хранилище, это позволяет захватывать файлы из других S3 хранилищ (Digital Ocean, Selectel и т.д.)

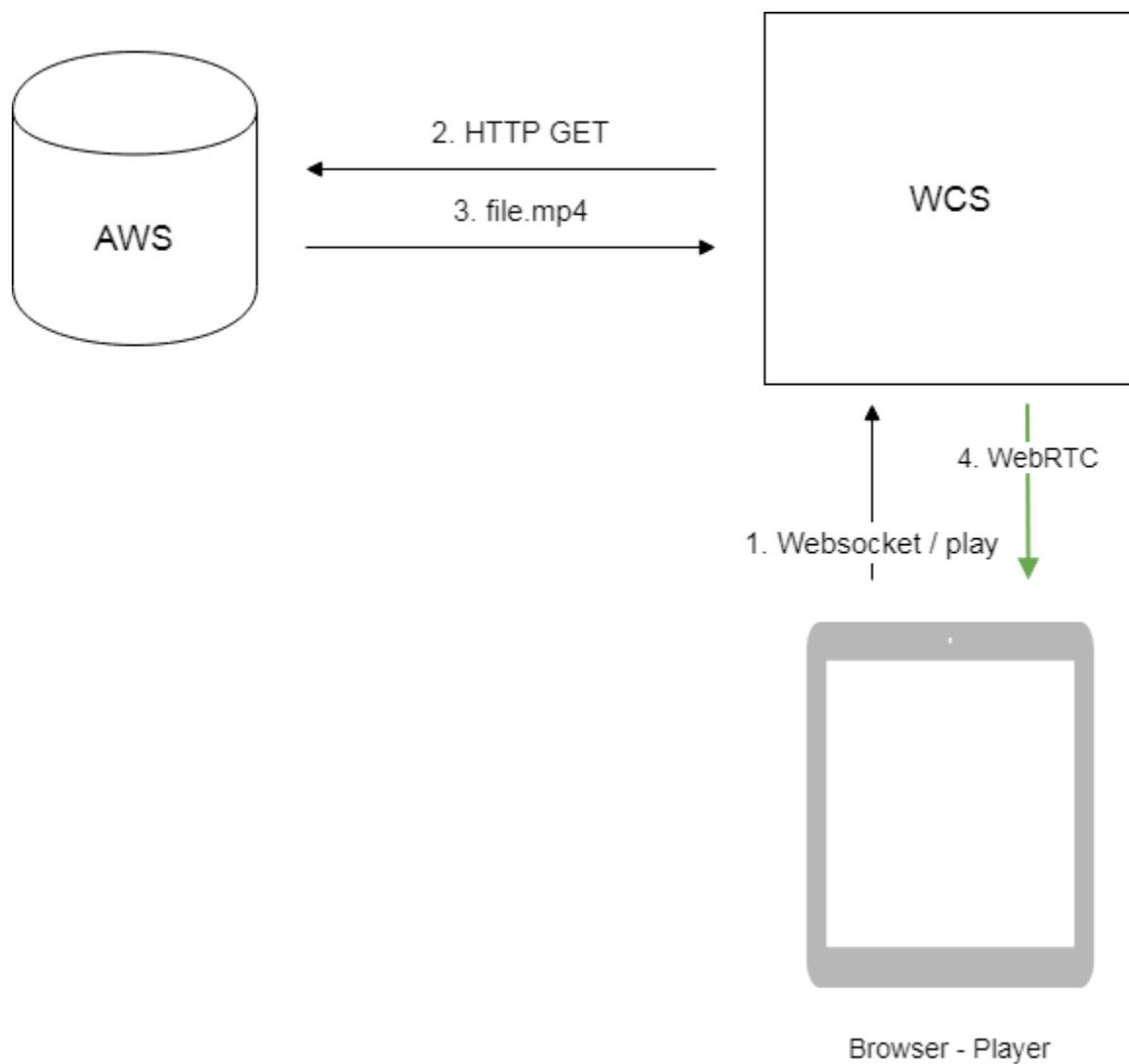
Пример для Digital Ocean Spaces

```
vod://s3/https://ams3.digitaloceanspaces.com/myspace/folder/file.mp4
```

Пример для Selectel

```
vod://s3/https://s3.selcdn.ru/mystorage/file.mp4
```

Схема работы



1. Браузер запрашивает захват потока из файла на AWS
2. WCS сервер направляет запрос AWS
3. Файл загружается на WCS сервер
4. WebRTC поток из файла передается в браузер для воспроизведения

Настройка

Доступ к хранилищу S3

AWS

Для загрузки файлов из AWS необходимо указать в файле настроек `flashphoner.properties` данные для доступа к хранилищу S3

```
aws_s3_credentials=zone;login;hash
```

Здесь

- zone - регион, где размещено хранилище
- login - идентификатор ключа доступа (Access Key ID)
- hash - секретный ключ доступа (Secret Accesss Key)

Пример настройки доступа:

```
aws_s3_credentials=eu-central-1;AA22BB33CC44DE;Dh1AkpZ4adc1HhblWhTNL4hvwTo80Njo
```

Digital Ocean Spaces

Для загрузки файлов из DO Spaces необходимо указать настройку

```
aws_s3_credentials=ams3;access_key;secret
```

Здесь

- ams3 - поддомен digitaloceanspaces.com
- access_key - ключ доступа к хранилищу
- secret - секретный код доступа к хранилищу

Selectel

Для загрузки файлов из Selectel S3 необходимо указать настройку

```
aws_s3_credentials=ru-1a;login;password
```

Здесь

- ru-1a - регион хранилища
- login - имя пользователя
- password - пароль

Захват VOD из файла во время загрузки

Чтобы захватывать поток из файла во время его загрузки, необходимо указать следующую настройку

```
vod_mp4_container_new=true
```

При низкой пропускной способности канала между WCS и хранилищем S3, либо при недостаточной его стабильности, может быть включена буферизация файла при загрузке. Размер буфера задается в миллисекундах настройкой

```
vod_mp4_container_new_buffer_ms=10000
```

В данном случае, размер буфера составит 10 секунд.

Требования к формату файлов

Заголовок (moov) должен всегда располагаться перед данными (mdat). Примерная структура файла должна быть такой:

```
Atom ftyp @ 0 of size: 32, ends @ 32
Atom moov @ 32 of size: 357961, ends @ 357993
...
Atom free @ 357993 of size: 8, ends @ 358001
Atom mdat @ 358001 of size: 212741950, ends @ 213099951
```

Проверить структуру файла можно при помощи утилиты [AtomicParsley](#)

```
AtomicParsley file.mp4 -T 1
```

Если структура файла не соответствует требованиям, файл не будет воспроизводиться. При необходимости, структуру файла можно исправить при помощи ffmpeg без перекодирования

```
ffmpeg -i bad.mp4 -acodec copy -vcodec copy -movflags +faststart good.mp4
```

Требования к именам файлов

Официальная [документация](#) по AWS S3 не рекомендует использование пробелов наряду с другими специальными символами, но и не запрещает их. Если пробелы в именах файлов все же используются, их необходимо заменять на '%20', например

vod://s3/bucket/sample%20with%20spaces.mp4

Управление VOD при помощи REST API

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

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

Здесь:

- test.flashphoner.com - адрес WCS-сервера
- 8081 - стандартный REST / HTTP порт WCS-сервера
- 8444 - стандартный HTTPS порт
- rest-api - обязательная часть URL
- /vod/startup - используемый REST-метод

REST-методы и статусы ответа

REST-метод	Пример тела REST-запроса	Пример тела REST-ответа	Статусы ответа	Описание
/vod/startup	<pre>{ "uri": "vod-live://sample.mp4", "localStreamName": "test" }</pre>		409 - Conflict 500 - Internal error	Захватить поток из указанного файла
/vod/find	<pre>{ "localStreamName": "test" }</pre>	<pre>[{ "localMediaSessionId": "29ec3236-1093-42bb-88d6-d4ac37af3ac0", "localStreamName": "test", "uri": "vod-live://sample.mp4", "status": "PROCESSED_LOCAL", "hasAudio": true, "hasVideo": true, "record": false, "loop": false }]</pre>	200 – потоки найдены 404 – потоки не найдены	Найти VOD-потоки по указанному критерию
/vod/find_all		<pre>[{ "localMediaSessionId": "29ec3236-1093-42bb-88d6-d4ac37af3ac0", "localStreamName": "test", "uri": "vod-live://sample.mp4", "status": "PROCESSED_LOCAL", "hasAudio": true, "hasVideo": true, "record": false, "loop": false }]</pre>	200 – потоки найдены 404 – потоки не найдены	Найти все VOD-потоки

/vod /terminate	<pre>{ "uri": " vod://sample.mp4", "localStreamName": "test" }</pre>		200 - поток завершен 404 - поток не найден	Завершить VOD-поток
--------------------	---	--	---	---------------------

Параметры

Имя параметра	Описание	Пример
uri	Имя файла для захвата потока	vod://sample.mp4
localStreamName	Имя создаваемого потока	test
status	Текущий статус потока	PROCESSED_LOCAL
localMediaSessionId	Идентификатор медиасессии	29ec3236-1093-42bb-88d6-d4ac37af3ac0
hasAudio	В потоке есть аудио	true
hasVideo	В потоке есть видео	true
record	Поток записывается	false
loop	Файл захватывается циклически	false

Циклический захват потока из файла по запросу

В сборке [5.2.1528](#) добавлена возможность указать, должен ли файл захватываться циклически, при создании VOD live трансляции по REST API

```
{  
  "uri": "vod-live://sample.mp4",  
  "localStreamName": "test",  
  "loop": true  
}
```

По умолчанию, если параметр loop не указан, применяется настройка vod_live_loop . Если параметр указан, то, в зависимости от его значения

- true - файл будет захватываться циклически
- false - файл будет проигран однократно, после чего VOD live трансляция остановится

Значение параметра loop имеет приоритет над значением настройки vod_live_loop .

Ограничения

Запрос /rest-api/vod/startup может применяться только для создания VOD live трансляций. При этом, запросы find, find_all и terminate могут быть применены как к VOD, так и к VOD live трансляциям.

Настройка продолжительности публикации VOD потока после отключения подписчиков

По умолчанию, VOD поток остается опубликованным на сервере в течение 30 секунд после отключения последнего подписчика, при условии, что продолжительность файла превышает этот интервал. Данное время может быть изменено при помощи настройки

```
vod_stream_timeout=60000
```

В этом случае, VOD поток останется опубликованным в течение 60 секунд.

Известные проблемы

1. AAC фреймы типа 0 не поддерживаются декодером на базе ffmpeg и будут игнорироваться при воспроизведении захваченного потока

Симптомы: предупреждения в [клиентском лог](#)е:

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

Решение: переключиться на использование FDK AAC декодера

```
use_fdk_aac=true
```

2. Файлы, содержащие В-фреймы, могут проигрываться неплavno, с фризами или артефактами

Симптомы: периодические фризы, артефакты при проигрывании файла через VOD, предупреждения в клиентском лог

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

Решение: перекодировать файл таким образом, чтобы исключить В-фреймы, например

```
ffmpeg -i bad.mp4 -preset ultrafast -acodec copy -vcodec h264 -g 24 -bf 0 good.mp4
```

3. При захвате VOD из продолжительного файла или при одновременном захвате нескольких VOD потоков процесс сервера может завершиться с Out of memory

Симптомы: процесс сервера завершается; "Map failed" в [серверном лог](#)еи в error*.log

```
19:30:53,277 ERROR DefaultMp4SampleList - Thread-34 java.io.IOException: Map failed
    at sun.nio.ch.FileChannelImpl.map(FileChannelImpl.java:940)
    at com.googlecode.mp4parser.FileDataSourceImpl.map(FileDataSourceImpl.java:62)
    at com.googlecode.mp4parser.BasicContainer.getBytesBuffer(BasicContainer.java:223)
    at com.googlecode.mp4parser.authoring.samples.DefaultMp4SampleList$SampleImpl.asByteBuffer
(DefaultMp4SampleList.java:204)
    at com.flashphoner.media.F.A.A.A$1.A(Unknown Source)
    at com.flashphoner.media.M.B.C.D(Unknown Source)
    at com.flashphoner.server.C.A.B.A(Unknown Source)
    at com.flashphoner.server.C.A.B.C(Unknown Source)
    at java.lang.Thread.run(Thread.java:748)
Caused by: java.lang.OutOfMemoryError: Map failed
    at sun.nio.ch.FileChannelImpl.map0(Native Method)
    at sun.nio.ch.FileChannelImpl.map(FileChannelImpl.java:937)
    ... 8 more
```

```
Event: 1743.157 Thread 0x00007fc480375000 Exception <a 'java/lang/OutOfMemoryError': Map failed>
(0x00000000a1d750b0) thrown at [/HUDSON/workspace/8-2-build-linux-amd64/jdk8u161/10277/hotspot/src/share/vm
/prims/jni.cpp, line 735]
```

Решение:

1. Увеличить максимальное число областей виртуальной памяти

```
sysctl -w vm.max_map_count=262144
```

и объем виртуальной памяти, выделяемой процессу сервера, заменив в файле /usr/local/FlashphonerWebCallServer/bin/webcallserver строку

```
ulimit -n 20000
```

на строки

```
ulimit -n 20000  
ulimit -v 1000000000
```

2. Начиная со сборки 5.2.57, установить настройку

```
vod_mp4_container_isoparser_heap_datasource=true
```